Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 4 MIPS Throughput at 4 MHz
- High Endurance Non-volatile Memorie segments
  - 8K/16K Bytes of In-System Self-Programmable Flash Program Memory(ATmega8HVA/16HVA)
  - 256 Bytes EEPROM
  - 512 Bytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C /100 years at 25°C(1)
  - Programming Lock for Software Security
- Battery Management Features
  - One or Two Cells in Series
  - Over-current Protection (Charge and Discharge)
  - Short-circuit Protection (Discharge)
  - High Voltage Outputs to Drive N-Channel Charge/Discharge FETs
- Peripheral Features
  - Two configurable 8- or 16-bit Timers with Separate Prescaler, Optional Input Capture (IC), Compare Mode and CTC
  - SPI - Serial Programmable Interface
  - 12-bit Voltage ADC, Four External and One Internal ADC Inputs
  - High Resolution Coulomb Counter ADC for Current Measurements
  - Programmable Watchdog Timer
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI ports
  - Power-on Reset
  - On-chip Voltage Regulator with Short-circuit Monitoring Interface
  - External and Internal Interrupt Sources
  - Sleep Modes:
    Idle, ADC Noise Reduction, Power-save, and Power-off
- Additional Secure Authentication Features available only under NDA
- Packages
  - 36-pad LGA
  - 28-lead TSOP
- Operating Voltage: 1.8 - 9V
- Maximum Withstand Voltage (High-voltage pins): 28V
- Temperature Range: - 20°C to 85°C
- Speed Grade: 1-4 MHz

8-bit AVR® Microcontroller with 8K/16K Bytes In-System Programmable Flash

ATmega8HVA
ATmega16HVA

Preliminary

8024A–AVR–04/08

1. Pin Configurations

1.1 LGA

Figure 1-1. LGA - Pinout ATmega8HVA/16HVA

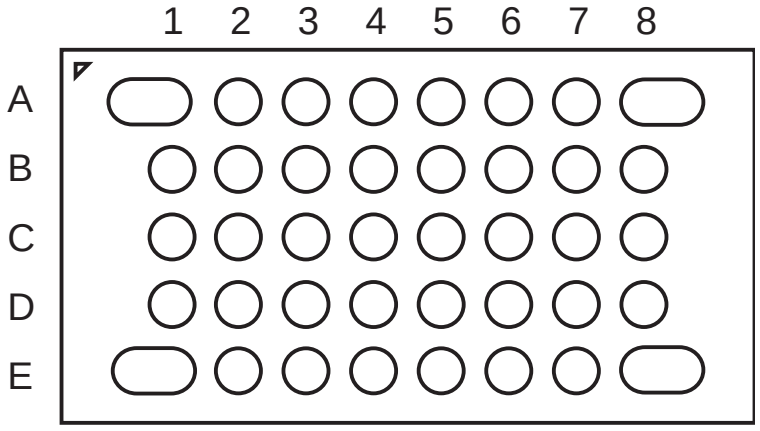
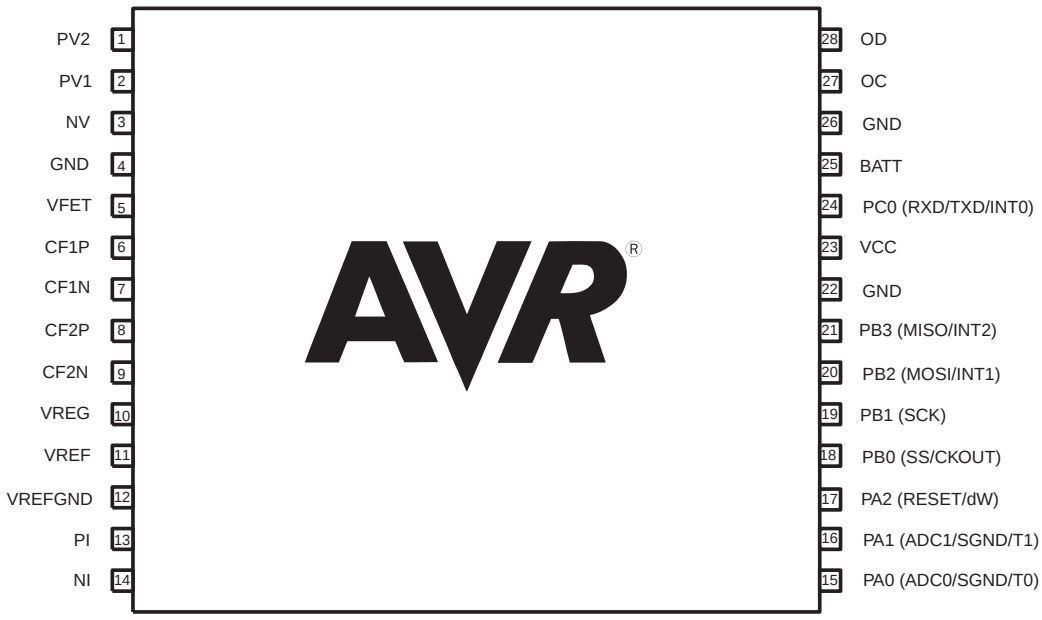


Figure 1-2. LGA - pinout ATmega8HVA/16HVA

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	DNC	PV2	PV1	NV	**GND**	OC	OD	DNC
B	CF2P	CF2N	VFET	CF1P	**GND**	PC0	DNC	**GND**
C	VREF	VREFGND	VREG	CF1N	VCC	**GND**	**GND**	BATT
D	PI	NI	**GND**	**GND**	GND	PB2	PB3	**GND**
E	DNC	DNC	PA1	PA0	PB1	PB0	RESET	DNC

1.2 TSOP

Figure 1-3. TSOP - pinout ATmega8HVA/16HVA



1.3 Pin Descriptions

1.3.1 VFET

Input to the internal voltage regulator.

1.3.2 VCC

Digital supply voltage. Normally connected to VREG.

1.3.3 VREG

Output from the internal voltage regulator.

1.3.4 CF1P/CF1N/CF2P/CF2N

CF1P/CF1N/CF2P/CF2N are the connection pins for connecting external fly capacitors to the step-up regulator.

1.3.5 VREF

Internal Voltage Reference for external decoupling.

1.3.6 VREFGND

Ground for decoupling of Internal Voltage Reference. Do not connect to GND or SGND on PCB.

1.3.7 GND

Ground

1.3.8 Port A (PA1..PA0)

Port A serves as a low-voltage 2-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega8HVA/16HVA as listed in ”Alternate Functions of Port A” on page 70.

1.3.9 Port B (PB3..PB0)

Port B is a low-voltage 4-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega8HVA/16HVA as listed in ”Alternate Functions of Port B” on page 71.

1.3.10 PC0

Port C serves the functions of various special features of the ATmega8HVA/16HVA as listed in ”Alternate Functions of Port C” on page 61.

1.3.11 OC

High voltage output to drive Charge FET.

1.3.12 OD

High voltage output to drive Discharge FET.

1.3.13 NI

NI is the filtered negative input from the current sense resistor.

1.3.14 PI

PI is the filtered positive input from the current sense resistor.

1.3.15 NV/PV1/PV2

NV, PV1, and PV2 are the inputs for battery cells 1 and 2.

1.3.16 BATT

Input for detecting when a charger is connected.

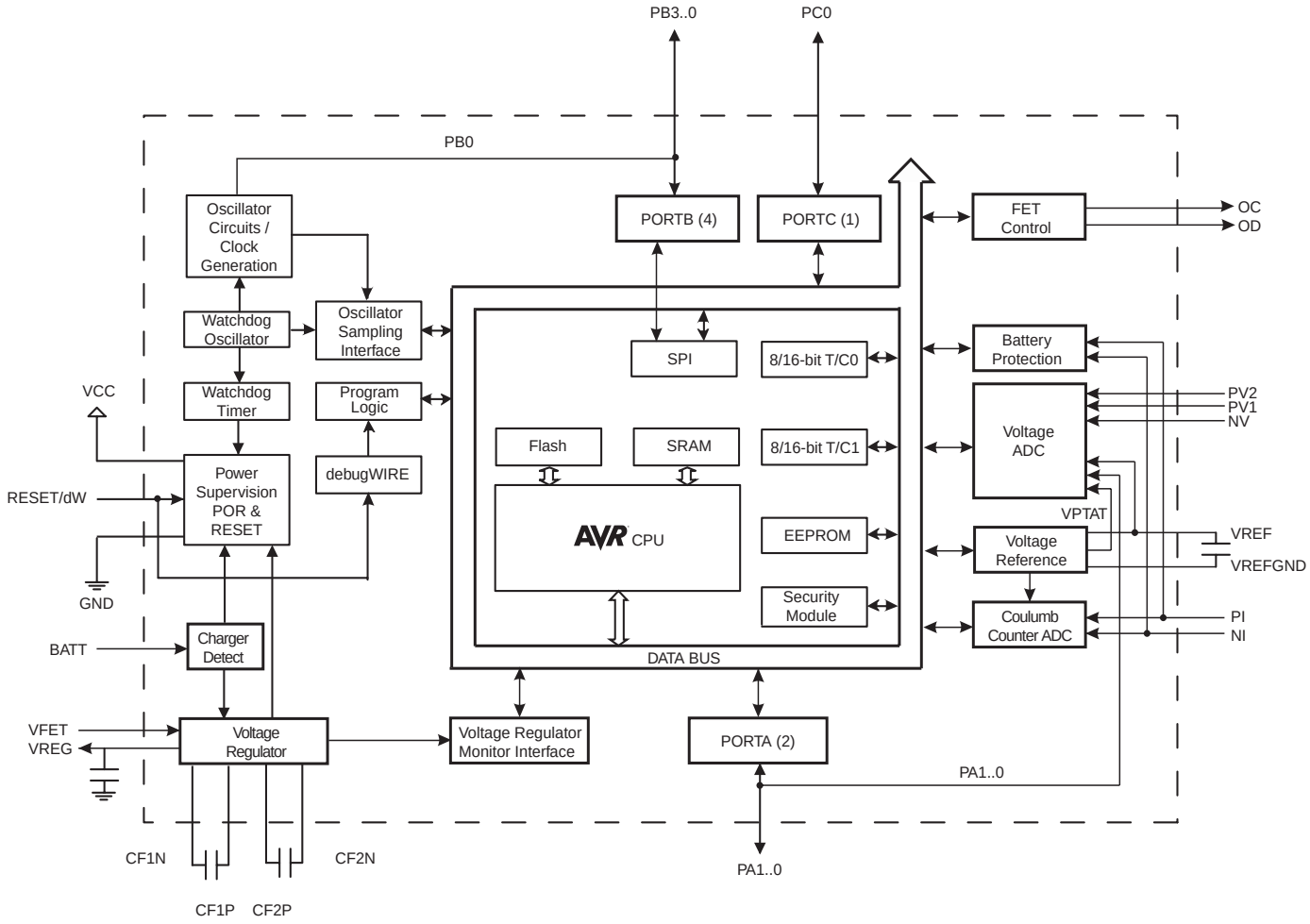
1.3.17 $\overline{\text{RESET}}$/dw

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 11 on page 38. Shorter pulses are not guaranteed to generate a reset. This pin is also used as debugWIRE communication pin.

2. Overview

The ATmega8HVA/16HVA is a monitoring and protection circuit for 1-cell and 2-cell Li-ion applications with focus on high security/authentication, accurate monitoring, low cost and high utilization of the cell energy. The device contains secure authentication features as well as autonomous battery protection during charging and discharging. The chip allows very accurate accumulated current measurements using an 18-bit ADC with a resolution of 0.84 µV. The feature set makes the ATmega8HVA/16HVA a key component in any system focusing on high security, battery protection, accurate monitoring, high system utilization and low cost.

Figure 2-1. Block Diagram



A combined step-up and linear voltage regulator ensures that the chip can operate with supply voltages as low as 1.8V for 1-cell applications. The regulator automatically switches to linear mode when the input voltage is sufficiently high, thereby ensuring a minimum power consumption at all times. For 2-cell applications, only linear regulation is enabled. The regulator capabilities, combined with an extremely low power consumption in the power saving modes, greatly enhances the cell energy utilization compared to existing solutions.

The chip utilizes Atmel's patented Deep Under-voltage Recovery (DUVR) mode that supports pre-charging of deeply discharged battery cells without using a separate Pre-charge FET.

The ATmega8HVA/16HVA contains a 12-bit ADC that can be used to measure the voltage of each cell individually. The ADC can also be used to monitor temperature, either on-chip temperature using the built-in temperature sensor, external temperature using thermistors connected to dedicated ADC inputs. The ATmega8HVA/16HVA contains a high-voltage tolerant, open-drain IO pin that supports serial communication. Programming can be done in-system using the 4 General Purpose IO ports that support SPI programming.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The MCU includes 8K/16K bytes of In-System Programmable Flash with Read-While-Write capabilities, 256 bytes EEPROM, 512 bytes SRAM, 32 general purpose working registers, 6 general purpose I/O lines, debugWIRE for On-chip debugging and SPI for In-system Programming, two flexible Timer/Counters with Input Capture and compare modes, internal and external interrupts, a 12-bit Sigma Delta ADC for voltage and temperature measurements, a high resolution Sigma Delta ADC for Coulomb Counting and instantaneous current measurements, Additional Secure Authentication Features, an authonomous Battery Protection module, a programmable Watchdog Timer with wake-up capabilities, and software selectable power saving modes.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two indepdent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The device is manufactured using Atmel’s high voltage high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System, through an SPI serial interface, by a conventional non-volatile memory programmer or by an On-chip Boot program running on the AVR core. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash, fuel gauging ADCs, dedicated battery protection circuitry, and a voltage regulator on a monolithic chip, the ATmega8HVA/16HVA is a powerful microcontroller that provides a highly flexible and cost effective solution for Li-ion Smart Battery applications.

The ATmega8HVA/16HVA AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, and On-chip Debugger.

The ATmega8HVA/16HVA is a low-power CMOS 8-bit microcontroller based on the AVR architecture. It is part of the AVR Smart Battery family that provides secure authentication, highly accurate monitoring and autonomous protection for Lithium-ion battery cells.

2.1 Comparison Between ATmega8HVA and ATmega16HVA

The ATmega8HVA and ATmega16HVA differ only in memory size and interrupt vector size. Table 2-1 summarizes the different configuration for the two devices.

Table 2-1. Configuration summary

Device	Flash	Interrupt vector size
ATmega8HVA	8K	1 Word
ATmega16HVA	16K	2 Word

3. Disclaimer

All Min, Typ and Max values contained in this datasheet are preliminary estimates based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Final values will be available after the device is characterized.

4. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

5. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

6. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

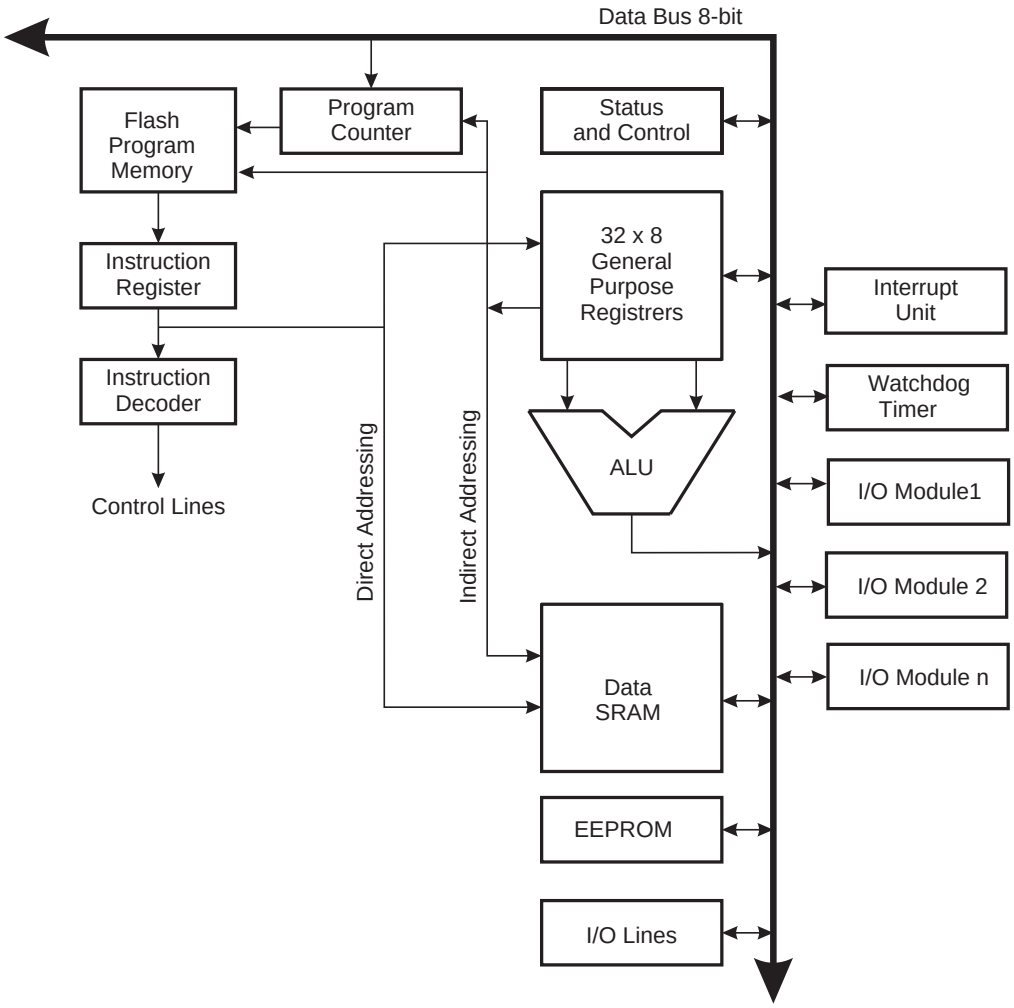
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

7. AVR CPU Core

7.1 Overview

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Figure 7-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typ-

ical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the ATmega8HVA/16HVA has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

7.2 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

7.3 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

7.3.1 SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, S = N $\oplus$ V**

The S-bit is always an exclusive or between the negative flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

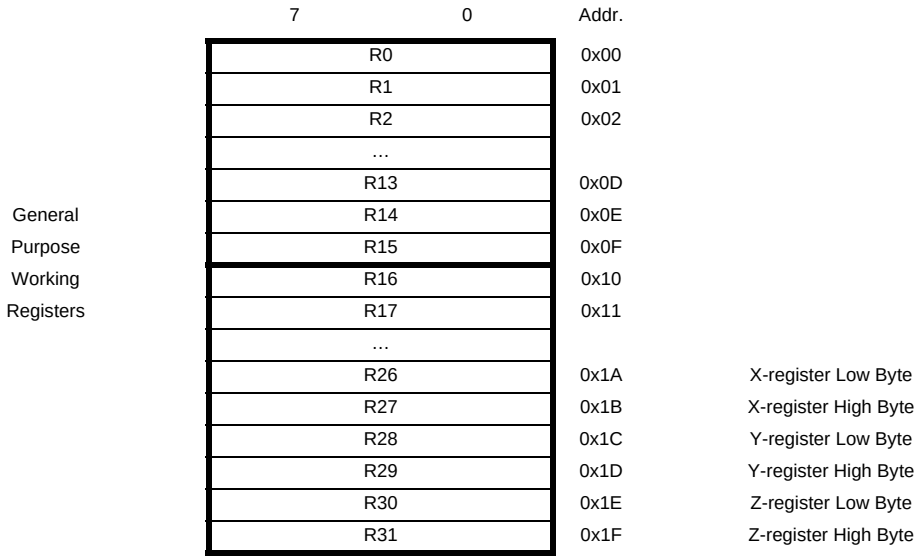
7.4 General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 7-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 7-2. AVR CPU General Purpose Working Registers



Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 7-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

7.4.1 The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 7-3 on page 12.

Figure 7-3. The X-, Y-, and Z-registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

7.5 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x100. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

7.5.1 SPH and SPL – Stack Pointer High and Stack Pointer Low

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

7.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 7-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 7-4. The Parallel Instruction Fetches and Instruction Executions

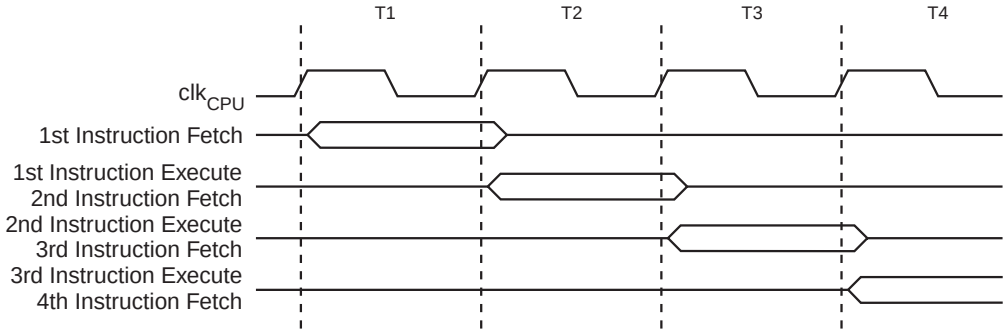
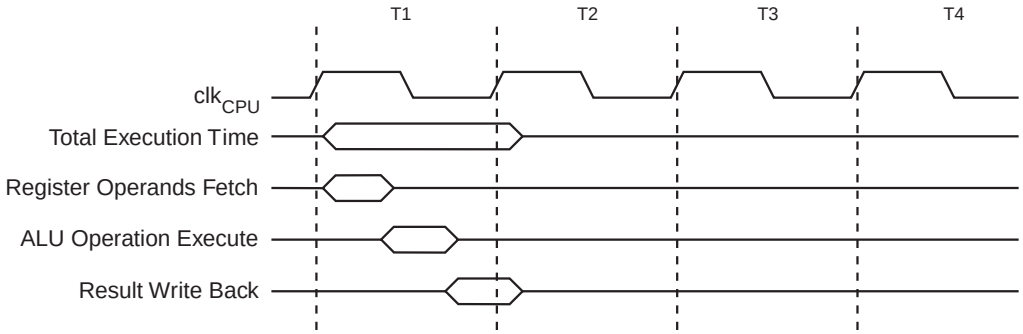


Figure 7-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7-5. Single Cycle ALU Operation



7.7 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in ”Interrupts” on page 52. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in  r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi EECR, EEMPE   ; start EEPROM write
sbi EECR, EEPE
out SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei  ; set Global Interrupt Enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set Global Interrupt Enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

7.7.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

8. AVR Memories

8.1 Overview

This section describes the different memories in the ATmega8HVA/16HVA. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega8HVA/16HVA features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

8.2 In-System Reprogrammable Flash Program Memory

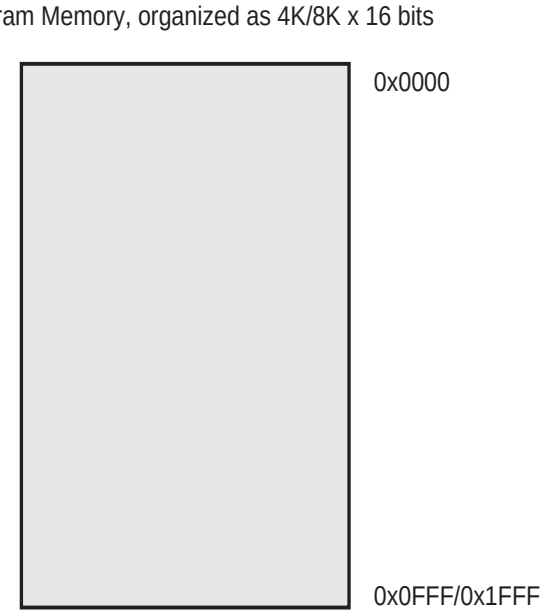
The ATmega8HVA/16HVA contains 8K/16K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K/8K x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8HVA/16HVA Program Counter (PC) is 12/13 bits wide, thus addressing the 4K/8K program memory locations. ”Memory Programming” on page 149 contains a detailed description on Flash data serial programming.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description).

Timing diagrams for instruction fetch and execution are presented in ”Instruction Execution Timing” on page 13.

Figure 8-1. Program Memory Map



8.3 SRAM Data Memory

Figure 8-2 on page 17 shows how the ATmega8HVA/16HVA SRAM Memory is organized.

The ATmega8HVA/16HVA is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For

the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 768 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

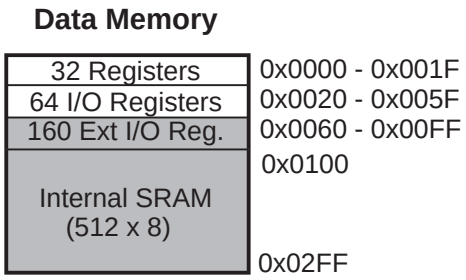
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 512 bytes of internal data SRAM in the ATmega8HVA/16HVA are all accessible through all these addressing modes. The Register File is described in ”General Purpose Register File” on page 11.

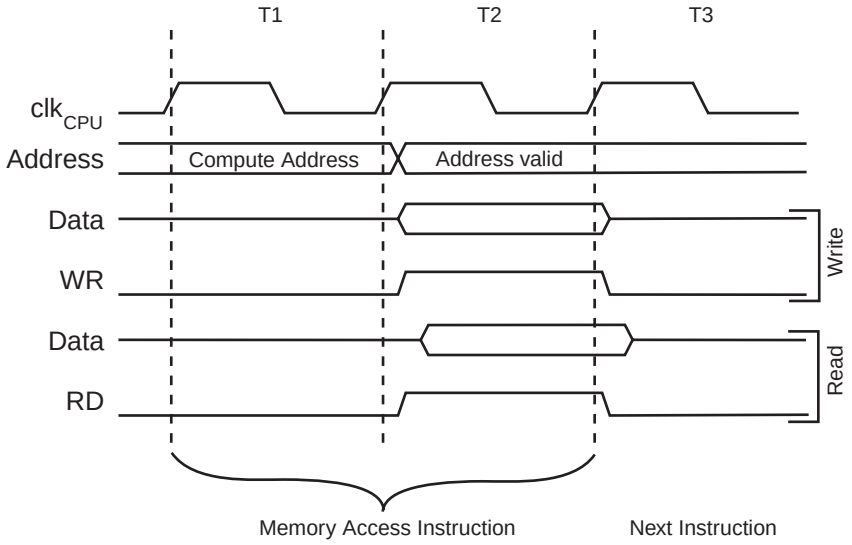
Figure 8-2. Data Memory Map



8.3.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 8-3.

Figure 8-3. On-chip Data SRAM Access Cycles



8.4 EEPROM Data Memory

The ATmega8HVA/16HVA contains 256 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of EEPROM programming, see page 151 and page 156 respectively.

8.4.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 8-1 on page 20. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

8.5 I/O Memory

The I/O space definition of the ATmega8HVA/16HVA is shown in "Register Summary" on page 175.

All ATmega8HVA/16HVA I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the

I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega8HVA/16HVA is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

8.5.1 General Purpose I/O Registers

The ATmega8HVA/16HVA contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

8.6 Register Description

8.6.1 EEAR – The EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	

- **Bits 7:0 – EEAR7:0: EEPROM Address**

The EEPROM Address Registers – EEAR specify the EEPROM address in the 256 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

8.6.2 EEDR – The EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – EEDR7:0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

8.6.3 EECR – The EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits**

The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 8-1. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 8-1. EEPROM Mode Bits

EEPM1	EEPM0	Typ Programming Time, f_{osc} = 4.0 MHz	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared.

- **Bit 2 – EEMPE: EEPROM Master Write Enable**

The EEMPE bit determines whether setting EEPE to one causes the EEPROM to be written. When EEMPE is set, setting EEPE within four clock cycles will write data to the EEPROM at the selected address If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

- **Bit 1 – EEPE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is not essential):

1. Wait until EEPE becomes zero.
2. Write new EEPROM address to EEAR (optional).
3. Write new EEPROM data to EEDR (optional).
4. Write a logical one to the EEMPE bit while writing a zero to EEPE in EECR.
5. Within four clock cycles after setting EEMPE, write a logical one to EEPE.

Caution:

An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted

EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

Caution:

A BOD reset during EEPROM write will invalidate the result of the ongoing operation.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses and the programming time will therefore depend on the calibrated oscillator frequency. Table 8-2 lists the typical programming time for EEPROM access from the CPU.

Table 8-2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ Programming Time, f_{osc} = 4.0 MHz
EEPROM write (from CPU)	13 600	3.4 ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEPE
  rjmp EEPROM_write
  ; Set up address (r17) in address register
  out  EEAR, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMPE
  sbi  EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi  EECR,EEPE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEPE
  rjmp EEPROM_read
  ; Set up address (r17) in address register
  out  EEAR, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

8.6.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	MSB							LSB	GPIOR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

8.6.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	MSB							LSB	GPIOR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

8.6.6 GPIOR0 – General Purpose I/O Register 0

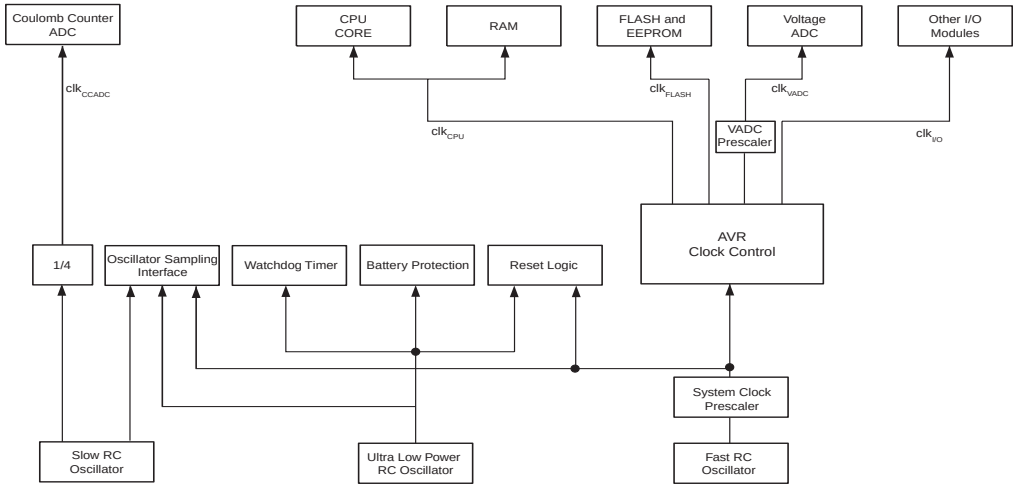
Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	MSB							LSB	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9. System Clock and Clock Options

9.1 Clock Systems and their Distribution

Figure 9-1 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in ”Power Management and Sleep Modes” on page 34. The clock systems are detailed below.

Figure 9-1. Clock Distribution



9.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

9.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

9.1.3 Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

9.1.4 Voltage ADC Clock – clk_{VADC}

The Voltage ADC is provided with a dedicated clock domain. The VADC clock is automatically prescaled relative to the System Clock Prescalers setting by the VADC Prescaler, giving a fixed VADC clock at 1 MHz.

9.1.5 Coulomb Counter ADC Clock - clk_{CCADC}

The Coulomb Counter ADC is provided with a dedicated clock domain. This allows operating the Coulomb Counter ADC in low power modes like Power-save for continuous current measurements.

9.1.6 Watchdog Timer and Battery Protection Clock

The Watchdog Timer and Battery Protection are provided with a dedicated clock domain. This allows operation in all modes except Power-off. It also allows very low power operation by utilizing an Ultra Low Power RC Oscillator dedicated to this purpose.

9.2 Clock Sources

The following section describe the clock sources available in the device. The clocks are input to the AVR clock generator, and routed to the appropriate modules.

9.3 Calibrated Fast RC Oscillator

The calibrated Fast RC Oscillator by default provides a 8.0 MHz clock to the system clock prescaler. The frequency is nominal value at 70°C. This clock will operate with no external components. With an accurate time reference and by using runtime calibration, this oscillator can be calibrated to an accuracy of +/- 1. % over the entire temperature range. During reset, hardware loads the calibration byte into the FOSCCAL Register and thereby automatically calibrates the Fast RC Oscillator. At 70°C, this calibration gives a frequency of 8 MHz ± 4%. The oscillator can be calibrated to any frequency in the range 7.3- 8.1 MHz by changing the FOSCCAL register. For more information on the pre-programmed calibration value, see the section ”Reading the Signature Row from Software” on page 144. Note that the frequency of the system clock is given by the ”System Clock Prescaler” on page 27.

Start-up time of the chip is referred to as a number of Prescaled Fast RC Oscillator cycles (CK) and a number of ULP oscillator cycles. The start-up time is selected by SUT fuses as defined in Table 9-1 on page 25.

Table 9-1. Start-up times according to SUT fuse selection.

SUT3..0	Start-up Time from Power-save	Additional Delay from Reset, Typical Values[2]
000	6 CK	14 CK + 4 ms
001	6 CK	14 CK + 8 ms
010	6 CK	14 CK + 16 ms
011	6 CK	14 CK + 32 ms
100	6 CK	14 CK + 64 ms
101	6 CK	14 CK + 128 ms
110	6 CK	14 CK + 256 ms
111[1]	6 CK	14 CK + 512 ms

Notes: 1. The device is shipped with this option selected.
2. The actual value of the added, selectable 4- 512 ms delay depends on the actual frequency of the “Ultra Low Power RC Oscillator”. See Table 9-2 on page 27 and ”Electrical Characteristics” on page 165

9.4 Slow RC Oscillator

The Slow RC Oscillator provides a 131 kHz clock (typical value, refer to section "Electrical Characteristics" on page 164 for details). This clock can be used as a timing reference for run-time calibration of the Fast RC Oscillator and for accurately determining the actual ULP Oscillator frequency, refer to ”OSI – Oscillator Sampling Interface” on page 28 for details. The Slow RC oscillator also provides the clock for the Coulomb Counter ADC.

The actual Slow RC Oscillator frequency depends on process variations and temperature, see ”Electrical Characteristics” on page 165.To provide a very good accuracy when used as a timing reference, the Slow RC Oscillator has prediction bytes stored in the signature address space, refer to section ”Reading the Signature Row from Software” on page 144 for details. The actual clock period of the Slow RC Oscillator in µs as a function of temperature is given by:

$$\text{Slow RC period} = \frac{\text{Slow RC period - Slow RC temp prediction word} \cdot \frac{(T - T_{HOT})}{64}}{1024}$$

where T is the die temperature in Kelvin and T_{HOT} is the calibration temperature stored in the signature row. The parameter "Slow RC period" holds information about the actual Slow RC oscillator period measured at Atmel production. This parameter can be read from the signature address space. Using the formula above, the actual Slow RC frequency can be found with an error of <1% over the temperature area from -10 °C to +70 °C.The die temperature can be found using the Voltage ADC, refer to section ”Voltage ADC – 5-channel General Purpose 12-bit Sigma-Delta ADC” on page 112 for details. For examples on Slow RC frequency Prediction, please refer to application note AVR351.

9.5 Ultra Low Power RC Oscillator

The Ultra Low Power RC Oscillator (ULP Oscillator) provides a 128 kHz clock (typical value). This oscillator provides the clock for the Watchdog Timer and Battery Protection modules. The actual ULP Oscillator frequency depends on process variations and temperature, see ”Electrical Characteristics” on page 165. The Oscillator is automatically enabled in all operational modes. It is also enabled during reset. There are two alternative methods for determining the actual clock period of the ULP Oscillator:

1. To determine the accurate clock period as a function of die temperature, if needed by the application, the Oscillator Sampling Interface should be used. Refer to section ”OSI – Oscillator Sampling Interface” on page 28 for details. For examples on ULP RC frequency calculations, please refer to application note AVR351.
2. To determine a fixed value for the actual clock period independent of the die temperature, for example to determine the best setting of the Battery Protection timing, use the byte ULP_RC_FRQ stored in the signature address space, refer to section ”Reading the Signature Row from Software” on page 144 for details.

9.6 CPU, I/O, Flash, and Voltage ADC Clock

The clock source for the CPU, I/O, Flash, and Voltage ADC is the calibrated Fast RC Oscillator.

9.7 Watchdog Timer, Battery Protection and Coulomb Counter ADC Clock

The clock source for the Watchdog Timer, Battery Protection and Coulomb Counter ADC (CC-ADC) is the Ultra Low Power RC Oscillator. The Oscillator is automatically enabled in all operational modes. It is also enabled during reset.

9.8 Clock Startup Sequence

When the CPU wakes up from Power-save, the CPU clock source is used to time the start-up, ensuring a stable clock before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the voltage regulator to reach a stable level before commencing normal operation. The Ultra Low Power RC Oscillator is used for timing this real-time part of the start-up time. Start-up times are determined by the SUT Fuses as shown in Table 9-1 on page 25. The number of Ultra Low Power RC Oscillator cycles used for each time-out is shown in Table 9-2.

Table 9-2. Number of Ultra Low Power RC Oscillator Cycles

Typ Time-out[1]	Number of Cycles
4 ms	512
8 ms	1K
16 ms	2K
32 ms	4K
64 ms	8K
128 ms	16K
256 ms	32K
512 ms	64K

Note: 1. The actual value depends on the actual clock period of the Ultra Low Power RC Oscillator, refer to "Ultra Low Power RC Oscillator" on page 26 for details.

9.9 Clock Output

The CPU clock divided by 2 can be output to the PB0 pin. The CPU can enable the clock output function by setting the CKOE bit in the MCU Control Register. The clock will not run in any sleep modes.

9.10 System Clock Prescaler

The ATmega8HVA/16HVA has a System Clock Prescaler, used to prescale the Calibrated Fast RC Oscillator. The system clock can be divided by setting the "CLKPR – Clock Prescale Register" on page 31, and this enables the user to decrease or increase the system clock frequency as the requirement for power consumption and processing power changes. This system clock will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{CPU} and clk_{FLASH} are divided by a factor as shown in Table 9-3 on page 32.

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, may be faster than the CPU's clock frequency. It is not possible to determine the state of the prescaler, and the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

9.11 VADC Clock Prescaler

The VADC clock will be automatically prescaled relative to the System Clock Prescaler settings, see "System Clock Prescaler" on page 27. Depending on the Clock Prescale Select bits, CLKPS1..0, the VADC clock, clk_{VADC}, will be prescaled by 4, 2 or 1 as shown in Table 9-4 on page 32.

9.12 OSI – Oscillator Sampling Interface

9.12.1 Features

- **Runtime selectable oscillator input (Slow RC or ULP RC Oscillator)**
- **7 bit prescaling of the selected oscillator**
- **Software read access to the phase of the prescaled clock**
- **Input capture trigger source for Timer/Counter0**

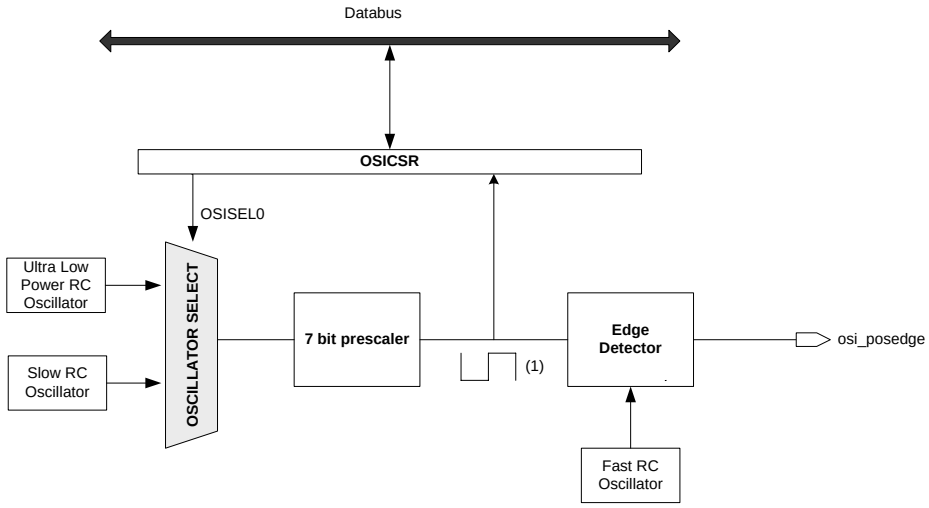
9.12.2 Overview

The Oscillator Sampling Interface (OSI) enables sampling of the Slow RC and Ultra Low Power RC (ULP) oscillators in ATmega8HVA/16HVA. OSI can be used to calibrate the Fast RC Oscillator runtime with high accuracy. OSI can also provide an accurate reference for compensating the ULP Oscillator frequency drift.

The prescaled oscillator phase can be continuously read by the CPU through the OSICSR register. In addition, the input capture function of Timer/Counter0 can be set up to trigger on the rising edge of the prescaled clock. This enables accurate measurements of the oscillator frequencies relative to the Fast RC Oscillator.

A simplified block diagram of the Oscillator Sampling Interface is shown in Figure 9-2 on page 29.

Figure 9-2. Oscillator Sampling Interface Block Diagram



Note: 1. One prescaled Slow RC/ULP oscillator period corresponds to 128 times the actual Slow RC/ULP oscillator period.

The osi_posedge signal pulses on each rising edge of the prescaled Slow RC/ ULP oscillator clock. This signal is not directly accessible by the CPU, but can be used to trigger the input capture function of Timer/Counter0. Using OSI in combination with the input capture function of Timer/Counter0 facilitates accurate measurement of the oscillator frequencies with a minimum of CPU calculation. Refer to "Timer/Counter(T/C0,T/C1)" on page 77 for details on how to enable the Input Capture function.

9.12.3 Usage

The Slow RC oscillator represents a highly predictable and accurate clock source over the entire temperature range and provides an excellent reference for calibrating the Fast RC oscillator runtime. Typically, runtime calibration is needed to provide an accurate Fast RC frequency for asynchronous serial communication in the complete temperature range. An accurate time reference is also needed to calculate accumulated charge during a CC-ADC measurement.

The Slow RC frequency at T_{HOT} (calibration temperature) and the Slow RC temperature coefficient are stored in the signature row. The value of T_{HOT} is also stored in the signature row. These characteristics can be used to calculate the actual Slow RC clock period at a given temperature with high precision. Refer to "Slow RC Oscillator" on page 26 for details.

By measuring the number of CPU cycles of one or more prescaled Slow RC clock periods, the actual Fast RC oscillator clock period can be determined. The Fast RC clock period can then be adjusted by writing to the FOSCCAL register. The new Fast RC clock period after calibration should be verified by repeating the measurement and repeating the calibration if necessary. The Fast RC clock period as a function of the Slow RC clock period is given by:

$$T_{FastRC} = T_{SlowRC} \cdot \frac{128 \cdot n}{\text{number of CPU cycles in n prescaled Slow RC periods}}$$

where n is the number of prescaled Slow RC periods that is used in the measurement. Using more prescaled Slow RC periods decreases the measurement error, but increases the time consumed for calibration. Note that the Slow RC Oscillator needs very short time to stabilize after

being enabled by the OSI module. Hence, the calibration algorithm may use the time between the first and second osi_posedge as time reference for calculations.

Another usage of OSI is determining the ULP frequency accurately. The ULP frequency at T_{HOT} and the ULP temperature coefficient are stored in the signature row, allowing the ULP frequency to be calculated directly. However, the ULP frequency is less predictable over temperature than the Slow RC oscillator frequency, therefore a more accurate result can be obtained by calculating the ratio between the Slow RC and ULP oscillators. This is done by sampling both the ULP and Slow RC oscillators and comparing the results. When the ratio is known, the actual ULP frequency can be determined with high accuracy. The ULP RC clock period as a function of the Slow RC clock period is given by:

$$T_{ULPRC} = T_{SlowRC} \cdot \frac{\text{number of CPU cycles in n prescaled ULP RC periods}}{\text{number of CPU cycles in n prescaled Slow RC periods}}$$

where n is the number of prescaled ULP RC and Slow RC periods that is used in the measurement. Using more prescaled ULP RC and Slow RC periods decreases the measurement error, but increases the time consumed for calibration. Note that the FOSCCAL register must be kept at a constant value during this operation to ensure accurate results.

These clock period calculations should be performed again when there is a significant change in die temperature since the previous calculation. The die temperature can be found using the Voltage ADC, refer to section ”Voltage ADC – 5-channel General Purpose 12-bit Sigma-Delta ADC” on page 112 for details.

9.13 Register Description

9.13.1 FOSCCAL – Fast RC Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
(0x66)	FCAL7	FCAL6	FCAL5	FCAL4	FCAL3	FCAL2	FCAL1	FCAL0	FOSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7:0 – FCAL7:0: Fast RC Oscillator Calibration Value**

The Fast RC Oscillator Calibration Register is used to trim the Fast RC Oscillator to remove process variations from the oscillator frequency. The factory-calibrated value is automatically written to this register during chip reset, giving an oscillator frequency of 8.0 MHz at 70°C. The application software can write this register to change the oscillator frequency. The oscillator can be run-time calibrated to any frequency in the range 7.3-8.1 MHz. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and Flash write accesses, and these write times will be affected accordingly. If the EEPROM or Flash are written, do not calibrate to more than 8.1 MHz. Otherwise, the EEPROM or Flash write may fail.

The FCAL[7:5] bits determine the range of operation for the oscillator. Setting these bits to 0b000 gives the lowest frequency range, setting this bit to 0b111 gives the highest frequency range. The frequency ranges are overlapping. A setting of for instance FOSCCAL = 0x1F gives a higher frequency than FOSCCAL = 0x20.

The FCAL[4:0] bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x1F gives the highest frequency in the range. Incrementing FCAL[4:0] by 1 will give a frequency increment of less than 1.5 % in the frequency range 7.3-8.1 MHz. With an accurate time reference, an oscillator accuracy of ±1% can be achieved after calibration. The frequency will drift with temperature, so run-time calibration will be required to maintain the accuracy. Refer to ”OSI – Oscillator Sampling Interface” on page 28 for details.

The default FOSCCAL value found in the signature row, is selected such that it is in the the lower half of a segment (see Figure 30-1 on page 174 for typical characteristics of the FAST RC oscillator). It is therefore sufficient to use the default segment and the one below to calibrate the frequency over the whole temperature range. To avoid a large frequency change when shifting between the two segments, a FOSC SEGMENT value is stored in the signature row. This is the first FOSCCAL value giving a lower frequency than the lowest value in the default segment, and should be used when calibrating the Fast RC oscillator.

9.13.2 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	-–	–	CKOE	PUD	–	–	–	–	MCUCR
Read/Write	R	R	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5 – CKOE: Clock Output**

When this bit is written to one, the CPU clock divided by 2 is output on the PB0 pin.

9.13.3 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
(0x61)	CLKPCE	–	–	–	–	–	CLKPS1	CLKPS0	CLKPR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	1	1	

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, or clear the CLKPCE bit.

- **Bit 1:0 – CLKPS1:0: Clock Prescaler Select Bit 1..0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 9-3 on page 32. Note that writing to the System Clock Prescaler Select bits will abort any ongoing VADC conversion.

Table 9-3. System Clock Prescaler Select

CLKPS1	CLKPS0	Clock Division Factor
0	0	Reserved(1)
0	1	2
1	0	4
1	1	8

Note: 1. Reserved values should not be written to CLKPS1..0.

Table 9-4. VADC Clock Prescaling(1)

CLKPS1	CLKPS0	VADC Division Factor
0	0	Reserved
0	1	4
1	0	2
1	1	1

Note: 1. When changing Prescaler value, the VADC Prescaler will automatically change frequency of the VADC clock and abort any ongoing conversion.

9.13.4 OSICSR – Oscillator Sampling Interface Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	–	–	–	OSISEL0	–	–	OSIST	OSIEN	OSICSR
Read/Write	R	R	R	R/W	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:5,3:2 – RES: Reserved bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bit 4 - OSISEL0: Oscillator Sampling Interface Select 0**

Table 9-5. OSISEL Bit Description

OSISEL0	Oscillator source
0	ULP Oscillator
1	Slow RC Oscillator

- **Bit 1 – OSIST: Oscillator Sampling Interface Status**

This bit continuously displays the phase of the prescaled clock. This bit can be polled by the CPU to determine the rising and falling edges of the prescaled clock.

- **Bit 0 – OSIEN: Oscillator Sampling Interface Enable**

Setting this bit enables the Oscillator Sampling Interface. When this bit is cleared, the Oscillator Sampling Interface is disabled.

Notes: 1. The prescaler is reset each time the OSICSR register is written, and hence each time a new oscillator source is selected.

10. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

10.1 Sleep Modes

Figure 9-1 on page 24 presents the different clock systems in the ATmega8HVA/16HVA, and their distribution. The figure is helpful in selecting an appropriate sleep mode. The different sleep modes and their wake up sources is summarized in Table 10-1, and Figure 10-1 on page 35 shows a sleep mode state diagram.

Table 10-1. Wake-up Sources for Sleep Modes

Mode	Wake-up sources									
	Wake-up on Regular Current	Battery Protection Interrupts	External Interrupts	WDT	EEPROM Ready	VREGMON	CC-ADC	V-ADC	Other I/O	Charger Detect(1)
Idle	X	X	X	X	X	X	X(2)	X	X	
ADC Noise Reduction	X	X	X	X	X	X	X(2)	X		
Power-save	X	X	X	X			X(2)			
Power-off										X

Notes: 1. Discharge FET must be switched off for Charger Detect to be active.
2. Instantaneous and Accumulate Conversion Complete wake-up only.

To enter any of the sleep modes, the SE bit in SMCR, see "SMCR – Sleep Mode Control Register" on page 39, must be written to logic one and a SLEEP instruction must be executed. The SM2..0 bits in the SMCR Register select which sleep mode will be activated by the SLEEP instruction. See Table 10-3 on page 39 for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file and SRAM are unaltered when the device wakes up from any sleep mode except Power-off. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 10-1. Sleep Mode State Diagram

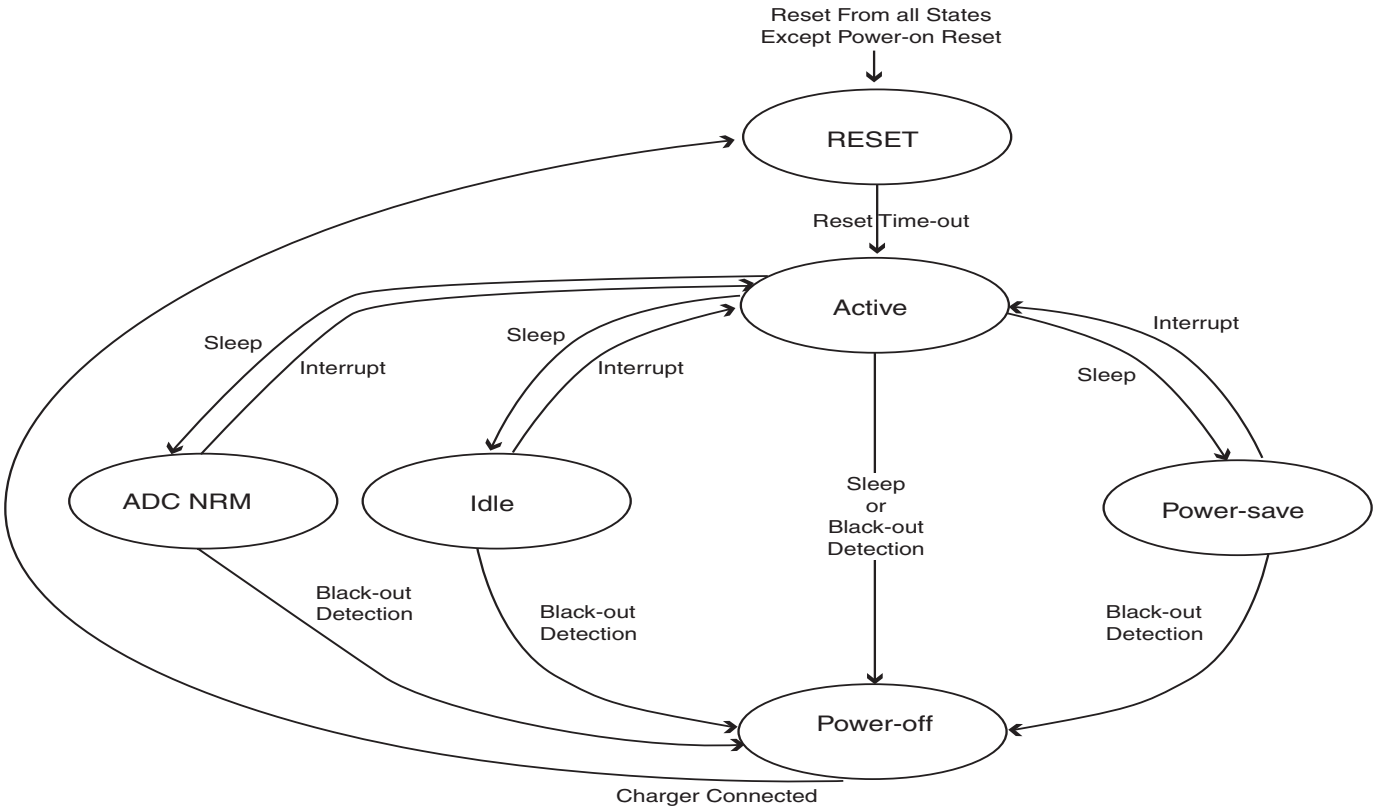


Table 10-2. Active modules in different Sleep Modes

Module	Mode				
	Active	Idle	ADC Noise Reduction	Power-save	Power-off
RCOSC_FAST	X	X	X	X(2)	
RCOSC_ULP	X	X	X	X	
RCOSC_SLOW	X(3)	X(3)	X(3)	X(3)	
CPU	X				
Flash	X				
Timer/Counter n	X	X			
SPI	X	X			
V-ADC	X	X	X		
CC-ADC	X	X(4)	X(4)	X(4)	
External Interrupts	X	X	X	X	
CBP	X	X	X	X	

Table 10-2. Active modules in different Sleep Modes (Continued)

Module	Mode				
	Active	Idle	ADC Noise Reduction	Power-save	Power-off
WDT	X	X	X	X	
VREG	X	X	X	X	
CHARGER_DETECT(1)	X	X	X	X	X
VREGMON	X	X	X		
OSI	X	X			

Notes:
1. Discharge FET must be switched off for Charger Detect to be enabled.
2. RCOSC_FAST runs in Power-save mode if DUVR mode is enabled. It also runs for approximately 125 ms after C-FET/D-FET has been enabled.
3. RCOSC_SLOW only runs if CC-ADC is enabled or when the oscillator is selected as input to the Oscillator sampling Interface and sampling is enabled.
4. Instantaneous and Accumulate Conversion Complete wake-up only.

10.2 Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing all peripheral functions to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run. Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow interrupt.

10.3 ADC Noise Reduction

When the SM2:0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the Voltage ADC (V-ADC), Watchdog Timer (WDT), Coulomb Counter (CC), Current Battery Protection (CBP), and the Ultra Low Power RC Oscillator (RCOSC_ULP) to continue operating. This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the Voltage ADC, enabling higher resolution measurements.

10.4 Power-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. In this mode, the internal Fast RC Oscillator (RCOSC_FAST) is stopped, while Watchdog Timer (WDT), Coulomb Counter (CC), Current Battery Protection (CBP), the Ultra Low Power RC Oscillator (RCOSC_ULP), and the Slow RC oscillator (RCOSC_SLOW) continue operating.

This mode will be the default mode when application software does not require operation of CPU, Flash or any of the peripheral units running at the Fast internal Oscillator (RCOSC_FAST).

If the current through the sense resistor is so small that the Coulomb Counter cannot measure it accurately, Regular Current detection should be enabled to reduce power consumption. The WDT keeps accurately track of the time so that battery self discharge can be calculated.

Note that if a level triggered interrupt is used for wake-up from Power-save mode, the changed level must be held for some time to wake up the MCU. Refer to ”External Interrupts” on page 56 for details.

When waking up from Power-save mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined in ”Clock Sources” on page 25.

10.5 Power-off Mode

When the SM2..0 bits are written to 100 and the SE bit is set, the SLEEP instruction makes the CPU shut down the Voltage Regulator, leaving only the Charger Detect Circuitry operational. To ensure that the MCU enters Power-off mode only when intended, the SLEEP instruction must be executed within 4 clock cycles after the SM2..0 bits are written. The MCU will reset when returning from Power-off mode.

Note: Before entering Power-off sleep mode, interrupts should be disabled by software. Otherwise interrupts may prevent the SLEEP instruction from being executed within the time limit.

10.6 Power Reduction Register

The Power Reduction Register (PRR), see ”PRR0 – Power Reduction Register 0” on page 39, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

10.7 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device’s functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

10.7.1 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes except Power-off. The Watchdog Timer current consumption is significant only in Power-save mode. Refer to ”Watchdog Timer” on page 46 for details on how to configure the Watchdog Timer.

10.7.2 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section ”Digital Input Enable and Sleep Modes” on page 67 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{REG}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{REG}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the Digital Input Disable Register. Refer to "DIDR0 – Digital Input Disable Register 0" on page 116 for details.

10.7.3 On-chip Debug System

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

10.7.4 Battery Protection

If one of the Battery Protection features is not needed by the application, this feature should be disabled, see "BPCR – Battery Protection Control Register" on page 127. The current consumption in the Battery Protection circuitry is only significant in Power-save mode. Disabling both FETs will automatically disable the Battery Protection module in order to save power. The bandgap reference should always be enabled whenever Battery Protection is enabled.

10.7.5 Voltage ADC

If enabled, the V-ADC will consume power independent of sleep mode. To save power, the V-ADC should be disabled when not used, and before entering Power-save sleep mode. See "Voltage ADC – 5-channel General Purpose 12-bit Sigma-Delta ADC" on page 112 for details on V-ADC operation.

10.7.6 Coloumb Counter

If enabled, the CC-ADC will consume power independent of sleep mode and keep the Slow RC oscillator running. To save power, the CC-ADC should be disabled when not used, or set in Regular Current detection mode. See "Coulomb Counter - Dedicated Fuel Gauging Sigma-delta ADC" on page 104 for details on CC-ADC operation.

10.7.7 Bandgap Voltage Reference

If enabled, the Bandgap reference will consume power independent of sleep mode. To save power, the Bandgap reference should be disabled when not used as reference for the Voltage ADC, the Coloumb Counter or Battery Protection. See "Voltage Reference and Temperature Sensor" on page 117 for details.

10.7.8 FET Driver

To minimize the power consumption in Power-save mode, the DUVR mode of the FET Driver should be disabled to make sure that the Fast RC Oscillator is stopped.

10.8 Register Description

10.8.1 SMCR – Sleep Mode Control Register

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA, and will always read as zero.

- **Bits 3:1 – SM2:0: Sleep Mode Select Bits 2, 1 and 0**

These bits select between the four available sleep modes as shown in Table 10-3.

Table 10-3. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Reserved
0	1	1	Power-save
1	0	0	Power-off
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

- **Bit 0 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

10.8.2 PRR0 – Power Reduction Register 0

Bit	7	6	5	4	3	2	1	0	
(0x64)	–	–	PRVRM	–	PRSPI	PRTIM1	PRTIM0	PRVADC	PRR0
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6, 4 - Res: Reserved bits**

These bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when PRR0 is written.

- **Bit 5 - PRVRM: Power Reduction Voltage Regulator Monitor**

Writing a logic one to this bit shuts down the Voltage Regulator Monitor interface by stopping the clock of the module.

- **Bit 3 - PRSPI: Power Reduction Serial Peripheral Interface**

Writing logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be reinitialized to ensure proper operation.

- **Bit 2 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 1 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 0 - PRVADC: Power Reduction V-ADC**

Writing a logic one to this bit shuts down the V-ADC. Before writing the PRVADC bit, make sure that the VADEN bit is cleared to minimize the power consumption.

Note: V-ADC control registers can be updated even if the PRVADC bit is set.

11. System Control and Reset

11.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. The circuit diagram in ”Reset Logic” on page 42 shows the reset logic. Table 12-2 on page 54 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

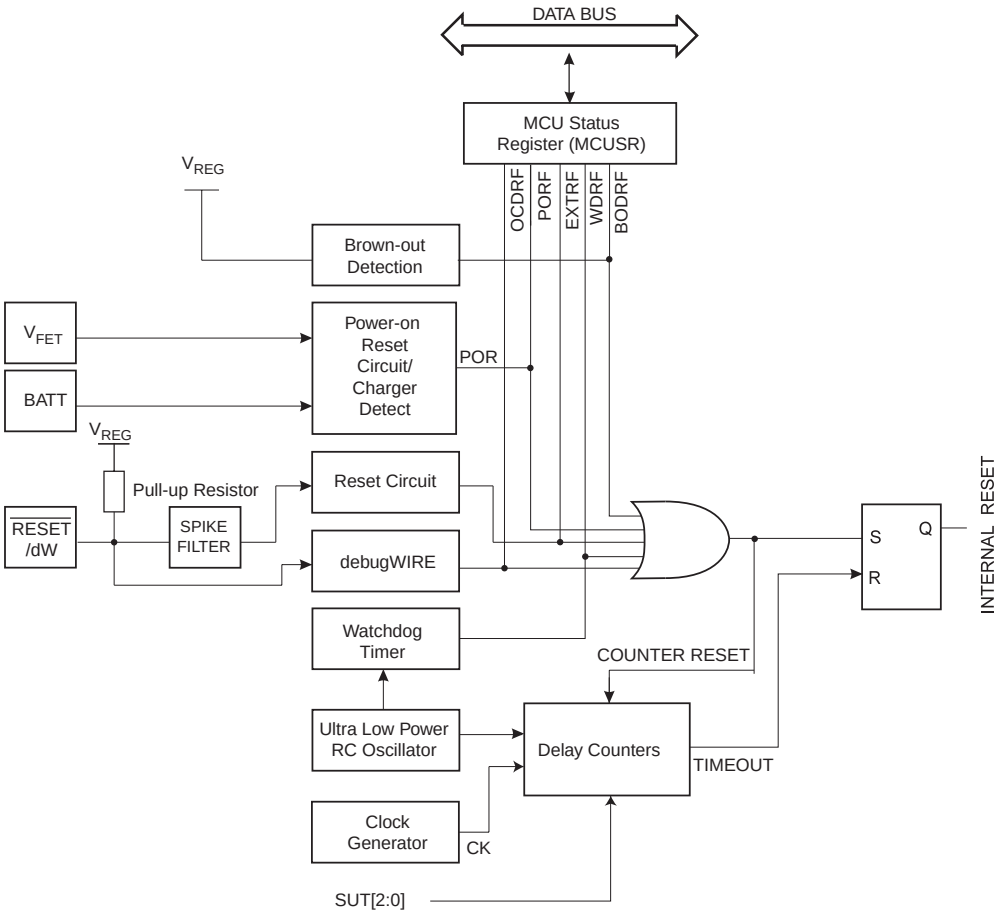
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the voltage regulator to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT Fuses. The different selections for the delay period are presented in ”Clock Sources” on page 25.

11.2 Reset Sources

The ATmega8HVA/16HVA has five sources of reset:

- The Power-on Reset module generates a Power-on Reset when the Voltage Regulator starts up.
- External Reset. The MCU is reset when a low level is present on the RESET pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when V_{REG} is below the Brown-out Reset Threshold, V_{BOT}. See “Brown-out Detection” on page 44.
- debugWIRE Reset. In On-chip Debug mode, the debugWIRE resets the MCU when giving the Reset command.

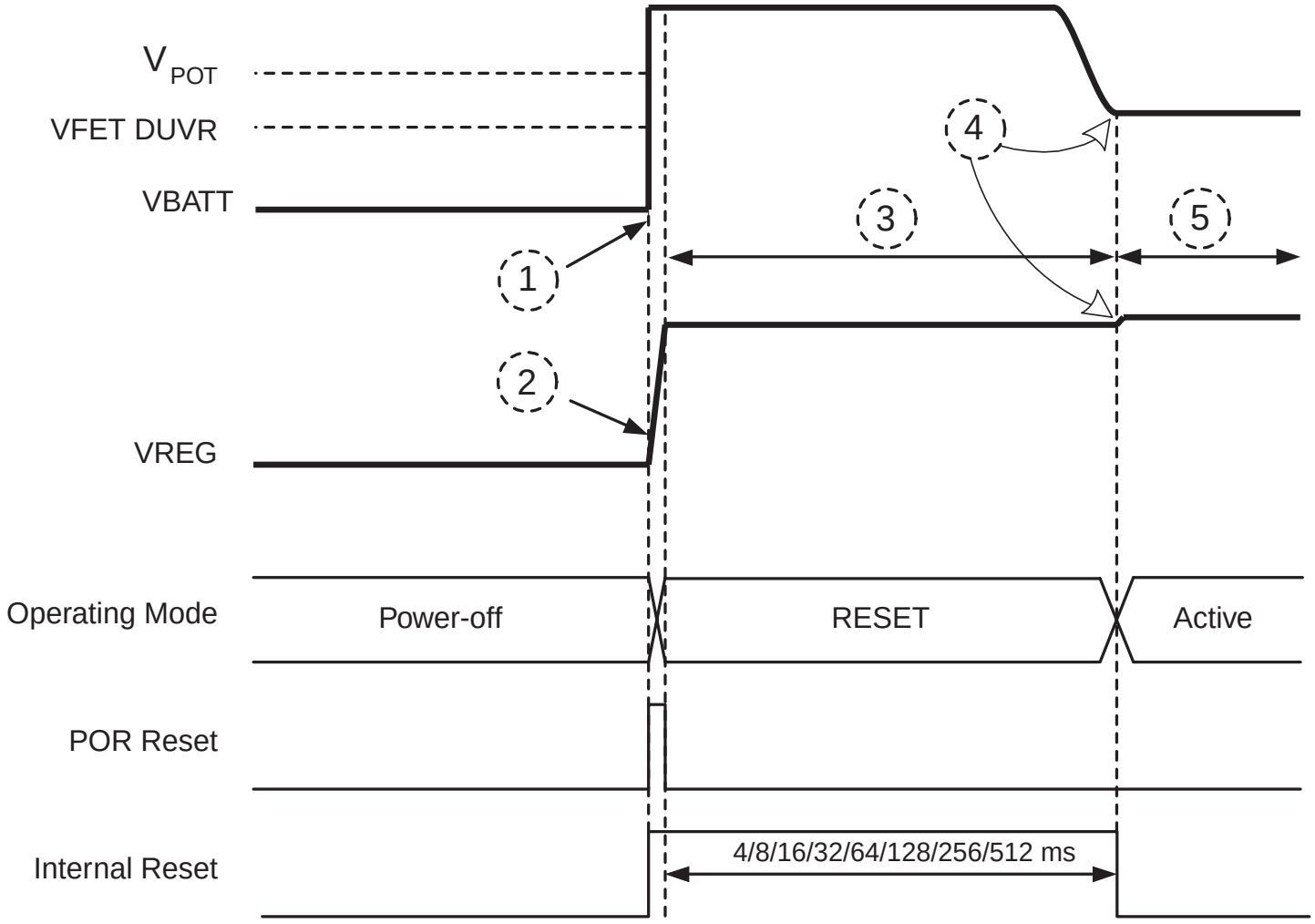
Figure 11-1. Reset Logic



11.2.1 Power-on Reset and Charger Connect

The Voltage Regulator will not start up until the Charger Detect module has enabled the Voltage Regulator. Before this happens the chip will be in Power-off mode and only the Charger Detect module is enabled. In order for the Charger Detect module to enable the Voltage Regulator, the voltage at the BATT pin must exceed the Power-On Threshold, V_{POT}. When the voltage at the BATT pin exceeds V_{POT}, the Voltage Regulator starts up and the chip enters RESET mode, refer to Figure 11-2 on page 43. When the Delay Counter times out, the chip will enter Active mode. See Figure 10-1 on page 35.

Figure 11-2. Normal Start-up Sequence in Power-off.



1. The charger voltage pulls the BATT pin above the Power-on Threshold Voltage (V_{POT}).
2. When V_{BATT} rises above V_{POT}, ATmega8HVA/16HVA turns on the Voltage Regulator and VREG starts to rise. The POR reset will go high while VREG is rising and initiate the internal reset state of the chip. The external FETs are initially switched off.
3. The internal reset is held high after POR reset goes low for a time given by t_{TOUT}, see ”System Control and Reset” on page 41. While the chip is in reset, VREF calibration registers will be reset to their default values. The VREG and BOD levels are both referenced to the VREF voltage. In reset all these voltage levels will therefore have default values. Both FETs are switched completely off in this state.
4. As soon as the internal reset goes low, the chip will start operating in DUVR mode (for details on DUVR mode, see ”DUVR – Deep Under-Voltage Recovery Mode operation” on page 137 and application note AVR354). In DUVR mode the FET driver controls the gate voltage of the Charge FET to get a voltage at the VFET pin given by the VFET level specified in Table 29-5 on page 170. This causes the BATT voltage to decrease. Note that DUVR mode will only regulate the VFET voltage as long as the cell voltage is lower than the VFET_DUVR level. For high cell voltages, DUVR mode will not have any impact. DUVR mode may be disabled by SW as soon as the chip enters ACTIVE mode.

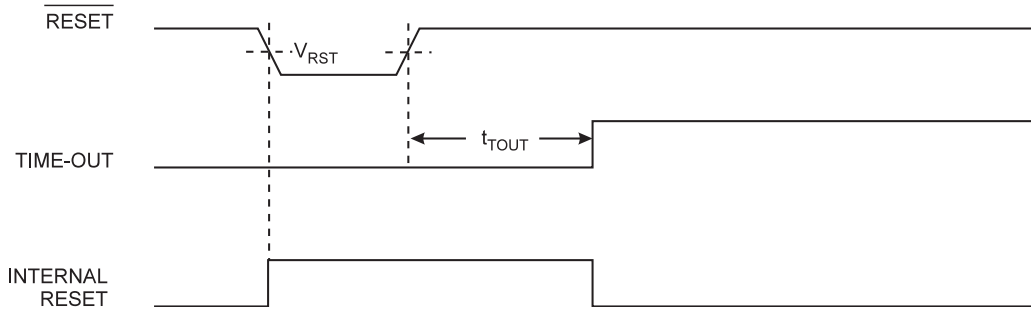
5. When the internal reset goes low, software starts up and loads the VREF calibration registers to get VREF = 1.100V. As the VREF voltage changes, VREG voltage and VFET DUVR voltage will rise proportionally to VREF.

Now the chip can operate normally, but writing to EEPROM in DUVR mode for single cell applications should be avoided.

11.2.2 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 29-6 on page 170) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period – t_{TOUT} – has expired.

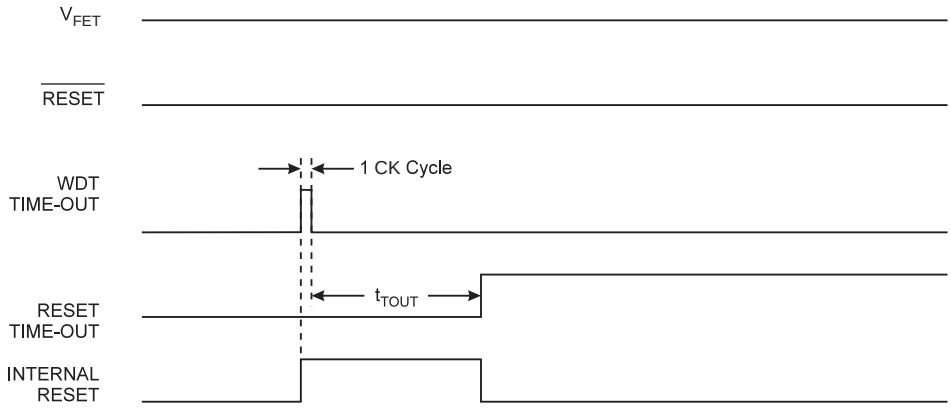
Figure 11-3. External Reset During Operation



11.2.3 Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 46 for details on operation of the Watchdog Timer.

Figure 11-4. Watchdog Reset During Operation



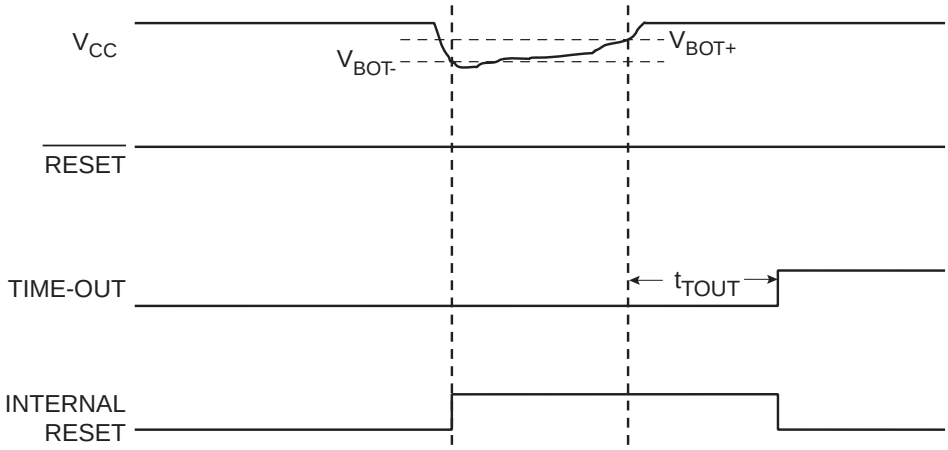
11.2.4 Brown-out Detection

ATmega8HVA/16HVA has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{REG} level during operation by comparing it to a fixed trigger level V_{BOT}. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD is automatically enabled in all modes of operation, except in Power-off mode.

When the BOD is enabled, and V_{REG} decreases to a value below the trigger level (V_{BOT-} in Figure 11-5), the Brown-out Reset is immediately activated. When V_{REG} increases above the trigger level (V_{BOT+} in Figure 11-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 11-5. Brown-out Reset During Operation



11.2.5 Black-out Detection

As an extra security feature, the chip will automatically enter Power-off if V_{REG} drops below V_{BLOT}. V_{BLOT} will always be well below the BOD level, V_{BOT}.

11.3 Watchdog Timer

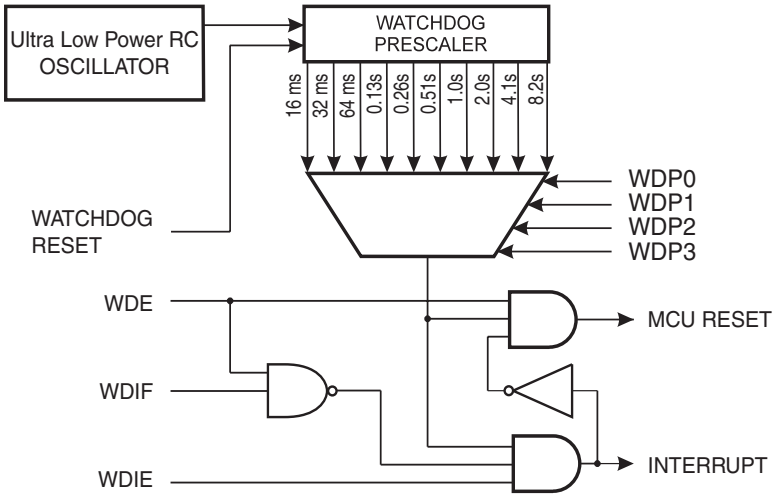
11.3.1 Features

- **Clocked from separate On-chip Oscillator**
- **3 Operating modes**
  - **Interrupt**
  - **System Reset**
  - **Interrupt and System Reset**
- **Selectable Time-out period from 16 ms to 8s**
- **Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode**

11.3.2 Overview

ATmega8HVA/16HVA has an Enhanced Watchdog Timer (WDT). The WDT counts cycles of the Ultra Low Power RC Oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

Figure 11-6. Watchdog Timer



In Interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In System Reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, Interrupt and System Reset mode, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and Watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example[1]

```
WDT_off:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Clear WDRF in MCUSR
  in    r16, MCUSR
  andi  r16, (0xff & (0<<WDRF))
  out   MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional time-out
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; Turn off WDT
  ldi   r16, (0<<WDE)
  out   WDTCSR, r16
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Clear WDRF in MCUSR */
  MCUSR &= ~(1<<WDRF);
  /* Write logical one to WDCE and WDE */
  /* Keep old prescaler setting to prevent unintentional time-out
*/
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

Note: 1. See “About Code Examples” on page 7.

Note: If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the Watchdog Timer will stay enabled. If the code is not set up to handle the Watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the Watchdog System Reset Flag (WDRF) and the WDE control bit in the initialisation routine, even if the Watchdog is not in use.

The following code example shows one assembly and one C function for changing the time-out value of the Watchdog Timer.

Assembly Code Example[1]

```
WDT_Prescaler_Change:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Start timed sequence
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; --  Got four cycles to set the new values from here -
  ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  out   WDTCSR, r16
  ; --  Finished setting new values, used 2 cycles -
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed  equence */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note: 1. See “About Code Examples” on page 7.

Note: The Watchdog Timer should be reset before any change of the WDP bits, since a change in the WDP bits can result in a time-out when switching to a shorter time-out period.

11.4 Register Description

11.4.1 MCUSR – MCU Status Register

The MCU Status Register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	OCDRF	WDRF	BODRF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **Bits 7:5 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA, and will always read as zero.

- **Bit 4 – OCDRF: OCD Reset Flag**

This bit is set if a debugWIRE Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BODRF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the reset flags.

11.4.2 WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- **Bit 7 - WDIF: Watchdog Interrupt Flag**

This bit is set when a time-out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

- **Bit 6 - WDIE: Watchdog Interrupt Enable**

When this bit is written to one and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if time-out in the Watchdog Timer occurs.

If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode). This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 11-1. Watchdog Timer Configuration

WDTON[1]	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

Note: 1. WDTON Fuse set to “0” means programmed, “1” means unprogrammed.

- **Bit 5, 2:0 - WDP3:0 : Watchdog Timer Prescaler 3, 2, 1 and 0**

The WDP3:0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 11-2.

- **Bit 4 - WDCE: Watchdog Change Enable**

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set.

Once written to one, hardware will clear WDCE after four clock cycles.

- **Bit 3 - WDE: Watchdog System Reset Enable**

WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

- **Bits 5, 2:0 – WDP3:0: Watchdog Timer Prescaler 3, 2, 1, and 0**

The WDP3:0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 11-2 on page 51.

.

Table 11-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out[1]
0	0	0	0	2K cycles	16 ms
0	0	0	1	4K cycles	32 ms
0	0	1	0	8K cycles	64 ms
0	0	1	1	16K cycles	0.13s
0	1	0	0	32K cycles	0.26s
0	1	0	1	64K cycles	0.51s
0	1	1	0	128K cycles	1.0s
0	1	1	1	256K cycles	2.0s
1	0	0	0	512K cycles	4.1s
1	0	0	1	1024K cycles	8.2s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Note: 1. The actual timeout value depends on the actual clock period of the Ultra Low Power RC Oscillator, refer to "Ultra Low Power RC Oscillator" on page 26 for details.

12. Interrupts

12.1 Overview

This section describes the specifics of the interrupt handling as performed in ATmega8HVA/16HVA. For a general explanation of the AVR interrupt handling, refer to ”Reset and Interrupt Handling” on page 13.

12.2 Interrupt Vectors in ATmega8HVA

.

Table 12-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and debugWIRE Reset
2	0x0001	BPINT	Battery Protection Interrupt
3	0x0002	VREGMON	Voltage Regulator Monitor Interrupt
4	0x0003	INT0	External Interrupt Request 0
5	0x0004	INT1	External Interrupt Request 1
6	0x0005	INT2	External Interrupt Request 2
7	0x0006	WDT	Watchdog Time-out Interrupt
8	0x0007	TIMER1 IC	Timer 1 input Capture
9	0x0008	TIMER1 COMPA	Timer 1 Compare Match A
10	0x0009	TIMER1 COMPB	Timer 1 Compare Match B
11	0x000A	TIMER1 OVF	Timer 1 Overflow
12	0x000B	TIMER0 IC	Timer 0 input Capture
13	0x000C	TIMER0 COMPA	Timer 0 Compare Match A
14	0x000D	TIMER0 COMPB	Timer 0 Compare Match B
15	0x000E	TIMER0 OVF	Timer 0 Overflow
16	0x000F	SPI, STC	SPI, Serial Transfer Complete
17	0x0010	VADC	Voltage ADC Conversion Complete
18	0x0011	CCADC CONV	CC-ADC Instantaneous Current Conversion Complete
19	0x0012	CCADC REG CUR	CC-ADC Regular Current
20	0x0013	CCADC ACC	CC-ADC Accumulate Current Conversion Complete
21	0x0014	EE READY	EEPROM Ready

If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega8HVA is:

```
Address   Labels   Code                    Comments
0x0000             rjmp    RESET           ; Reset Handler
0x0001             rjmp    BPINT           ; Battery Protection Interrupt Handler
0x0002             rjmp    VREGMON_INT     ; Voltage Regulator Monitor Interrupt Handler
0x0003             rjmp    EXT_INT0        ; External Interrupt Request 0 Handler
0x0004             rjmp    EXT_INT1        ; External Interrupt Request 1 Handler
0x0005             rjmp    EXT_INT2        ; External Interrupt Request 2 Handler
0x0006             rjmp    WDT             ; Watchdog Time-out Interrupt
0x0007             rjmp    TIM1_IC         ; Timer1 Input Capture Handler
0x0008             rjmp    TIM1_COMPA      ; Timer0 CompareA Handler
0x0009             rjmp    TIM1_COMPB      ; Timer0 CompareB Handler
0x000A             rjmp    TIM1_OVF        ; Timer1 Overflow Handler
0x000B             rjmp    TIM0_IC         ; Timer1 Input Capture Handler
0x000C             rjmp    TIM0_COMPA      ; Timer0 CompareA Handler
0x000D             rjmp    TIM0_COMPB      ; Timer0 CompareB Handler
0x000E             rjmp    TIM0_OVF        ; Timer0 Overflow Handler
0X000F             rjmp    SPI, STC        ; SPI, Serial Transfer Complete
0x0010             rjmp    VADC            ; Voltage ADC Conversion Complete Handler
0x0011             rjmp    CCADC_CONV      ; CC-ADC Instantaneous Current Conversion Complete
                                           Handler
0x0012             rjmp    CCADC_REC_CUR   ; CC-ADC Regular Current Handler
0x0013             rjmp    CCADC_ACC       ; CC-ADC Accumulate Current Conversion Complete
                                           Handler
0x0014             rjmp    EE_RDY          ; EEPROM Ready Handler
;
          RESET:   ldi     r16,            ; Main program start
                            high(RAMEND)
0x0015             out     SPH,r16         ; Set Stack Pointer to top of RAM
0x0016             ldi     r16,
                            low(RAMEND)
0x0017             out     SPL,r16
0x0018             sei                     ; Enable interrupts
0x0018             <instr  xxx
                   >
0x001A   ...       ...     ...
;
```

12.3 Interrupt Vectors in ATmega16HVA

.

Table 12-2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and debugWIRE Reset
2	0x0002	BPINT	Battery Protection Interrupt
3	0x0004	VREGMON	Voltage Regulator Monitor Interrupt
4	0x0006	INT0	External Interrupt Request 0
5	0x0008	INT1	External Interrupt Request 1
6	0x000A	INT2	External Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER1 IC	Timer 1 input Capture
9	0x0010	TIMER1 COMPA	Timer 1 Compare Match A
10	0x0012	TIMER1 COMPB	Timer 1 Compare Match B
11	0x0014	TIMER1 OVF	Timer 1 Overflow
12	0x0016	TIMER0 IC	Timer 0 input Capture
13	0x0018	TIMER0 COMPA	Timer 0 Compare Match A
14	0x001A	TIMER0 COMPB	Timer 0 Compare Match B
15	0x001C	TIMER0 OVF	Timer 0 Overflow
16	0x001E	SPI, STC	SPI, Serial Transfer Complete
17	0x0020	VADC	Voltage ADC Conversion Complete
18	0x0022	CCADC CONV	CC-ADC Instantaneous Current Conversion Complete
19	0x0024	CCADC REG CUR	CC-ADC Regular Current
20	0x0026	CCADC ACC	CC-ADC Accumulate Current Conversion Complete
21	0x0028	EE READY	EEPROM Ready

If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega16HVA is:

```
Address   Labels   Code                    Comments
0x0000             jmp     RESET           ; Reset Handler
0x0002             jmp     BPINT           ; Battery Protection Interrupt Handler
0x0004             jmp     VREGMON_INT     ; Voltage Regulator Monitor Interrupt Handler
0x0006             jmp     EXT_INT0        ; External Interrupt Request 0 Handler
0x0008             jmp     EXT_INT1        ; External Interrupt Request 1 Handler
0x000A             jmp     EXT_INT2        ; External Interrupt Request 2 Handler
0x000C             jmp     WDT             ; Watchdog Time-out Interrupt
0x000E             jmp     TIM1_IC         ; Timer1 Input Capture Handler
0x0010             jmp     TIM1_COMPA      ; Timer1 Compare A Handler
0x0012             jmp     TIM1_COMPB      ; Timer1 Compare B Handler
0x0014             jmp     TIM1_OVF        ; Timer1 Overflow Handler
0x0016             jmp     TIM0_IC         ; Timer0 Input Capture Handler
0x0018             jmp     TIM0_COMPA      ; Timer0 CompareA Handler
0x001A             jmp     TIM0_COMPB      ; Timer0 CompareB Handler
0x001C             jmp     TIM0_OVF        ; Timer0 Overflow Handler
0X001E             jmp     SPI, STC        ; SPI, Serial Transfer Complete
0x0020             jmp     VADC            ; Voltage ADC Conversion Complete Handler
0x0022             jmp     CCADC_CONV      ; CC-ADC Instantaneous Current Conversion Complete
                                           Handler
0x0024             jmp     CCADC_REC_CUR   ; CC-ADC Regular Current Handler
0x0026             jmp     CCADC_ACC       ; CC-ADC Accumulate Current Conversion Complete
                                           Handler
0x0028             jmp     EE_RDY          ; EEPROM Ready Handler
;
          RESET:   ldi     r16,            ; Main program start
                            high(RAMEND)
0x002A             out     SPH,r16         ; Set Stack Pointer to top of RAM
0x002B             ldi     r16,
                            low(RAMEND)
0x002C             out     SPL,r16
0x002D             sei                     ; Enable interrupts
0x002E             <instr  xxx
                   >
0x002F   ...       ...     ...
;
```

13. External Interrupts

13.1 Overview

The External Interrupts are triggered by the INT2:0 pins. Observe that, if enabled, the interrupts will trigger even if the INT2:0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The External Interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register – EICRA. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Interrupts are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-save mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the ULP Oscillator clock. The period of the ULP Oscillator is 7.8 µs (nominal) at 25°C. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT fuses as described in ”Clock Systems and their Distribution” on page 24. If the level is sampled twice by the ULP Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

13.2 Register Description

13.2.1 EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x69)	-	-	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7,6 – RES: Reserved Bits**

These bits are reserved in the ATmega8HVA/16HVA, and will always read as zero.

- **Bits 5:0 – ISC21, ISC20 - ISC01, ISC00: External Interrupt 2 - 0 Sense Control Bits**

The External Interrupts 2 - 0 are activated by the external pins INT2:0 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in Table 13-1. Edges on INT2..INT0 are registered asynchronously. Pulses on INT2:0 pins wider than the minimum pulse width given in Table 29-2 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low. When changing the ISCn bit, an interrupt can occur. Therefore, it is recommended to first disable INTn by clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISCn bit can be changed. Finally, the INTn interrupt flag should be cleared by writing a logical one to its Interrupt Flag bit (INTFn) in the EIFR Register before the interrupt is re-enabled.

Table 13-1. Interrupt Sense Control

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request.
1	0	The falling edge of INTn generates an interrupt request.
1	1	The rising edge of INTn generates an interrupt request.

Note: 1. n = 2, 1, or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

13.2.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	INT2	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – RES: Reserved Bits**

These bits are reserved bits ins the ATmega8HVA/16HVA, and will always read as zero.

- **Bits 2:0 – INT2 - INT0: External Interrupt Request 2:0 Enable**

When an INT2 – INT0 bit is written to one and the I-bit in the Status Register (SREG) is set (one), the corresponding external pin interrupt is enabled. The Interrupt Sense Control bits in the External Interrupt Control Register – EICRA – defines whether the external interrupt is activated on rising or falling edge or level sensed. Activity on any of these pins will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

13.2.3 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	INTF2	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – RES: Reserved Bits**

These bits are reserved bits ins the ATmega8HVA/16HVA, and will always read as zero.

- **Bits 2:0 – INTF2 - INTF0: External Interrupt Flags 2:0**

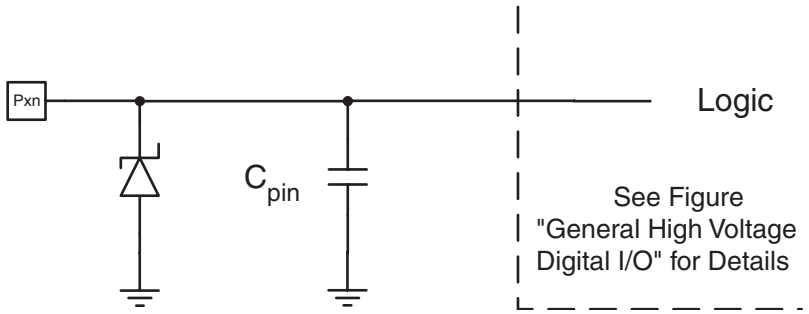
When an edge or logic change on the INT2:0 pin triggers an interrupt request, INTF2:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT2:0 in EIMSK, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. These flags are always cleared when INT2:0 are configured as level interrupt. Note that when entering sleep mode with the INT2:0 interrupts disabled, the input buffers on these pins will be disabled. This may cause a logic change in internal signals which will set the INTF2:0 flags. See ”Digital Input Enable and Sleep Modes” on page 67 for more information.

14. High Voltage I/O Ports

14.1 Overview

All high voltage AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the state of one port pin can be changed without unintentionally changing the state of any other pin with the SBI and CBI instructions. All high voltage I/O pins have protection Zener diodes to Ground as indicated in Figure 14-1. See ”Electrical Characteristics” on page 165 for a complete list of parameters.

Figure 14-1. High Voltage I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTC3 for bit number three in Port C, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in ”Register Description” on page 62.

One I/O Memory address location is allocated for each high voltage port, the Data Register – PORTx. The Data Register is read/write.

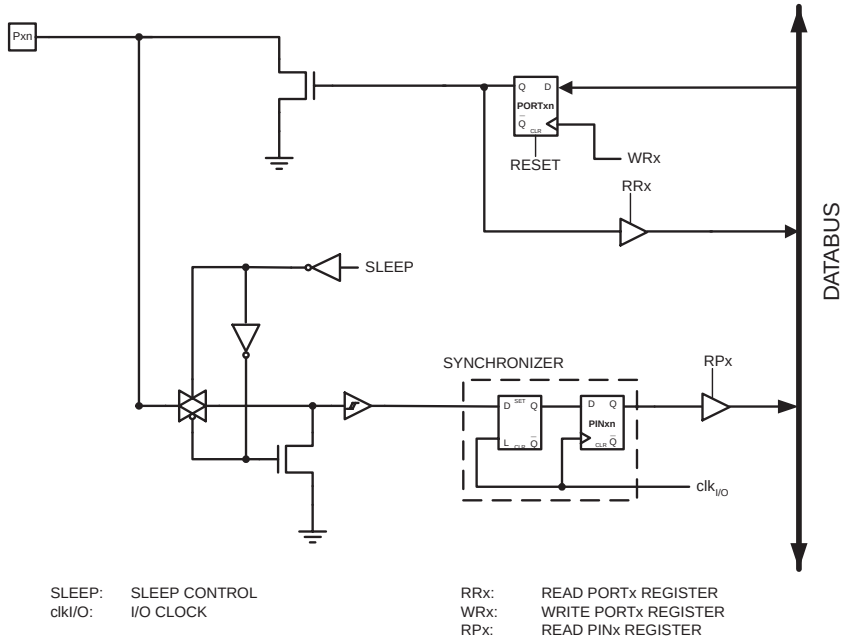
Using the I/O port as General Digital Output is described in ”High Voltage Ports as General Digital I/O” on page 59.

14.2 High Voltage Ports as General Digital I/O

14.3 Overview

The high voltage ports are high voltage tolerant open collector output ports. In addition they can be used as general digital inputs. Figure 14-2 shows a functional description of one output port pin, here generically called Pxn.

Figure 14-2. General High Voltage Digital I/O(1)



Note: 1. WRx, RRx and RPx are common to all pins within the same port. $clk_{I/O}$ and SLEEP are common to all ports.

14.3.1 Configuring the Pin

Each port pin consist of two register bits: PORTxn and PINxn. As shown in ”Register Description” on page 62, the PORTxn bits are accesed at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

If PORTxn is written logic one, the port pin is driven low (zero). If PORTxn is written logic zero, the port pin is tri-stated. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

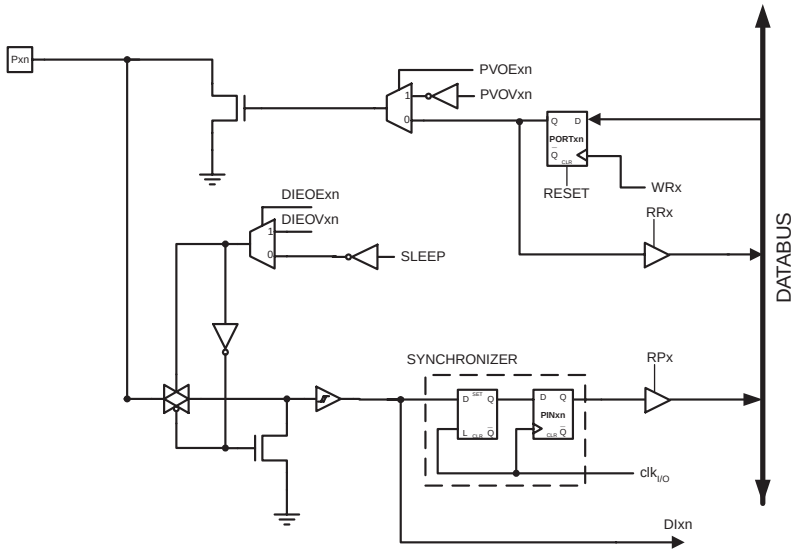
14.3.2 Reading the Pin

The port pin can be read through the PINxn Register bit. As shown in Figure 14-2, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay.

14.4 Alternate Port Functions

The High Voltage I/O has alternate port functions in addition to being general digital I/O. Figure 14-3 shows how the port pin control signals from the simplified Figure 14-2 on page 59 can be overridden by alternate functions.

Figure 14-3. High Voltage Digital I/O[1]



PVOExn: Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn: Pxn PORT VALUE OVERRIDE VALUE
DIEOExn: Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE
DIEOVxn: Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE

RRx: READ PORTx REGISTER
WRx: WRITE PORTx REGISTER
RPx: READ PINx REGISTER
$clk_{I/O}$: I/O CLOCK
DIxn: DIGITAL INPUT PIN n ON PORTx
SLEEP: SLEEP CONTROL

Note: 1. WRx, RRx and RPx are common to all pins within the same port. $clk_{I/O}$ and SLEEP are common to all ports. All other signals are unique for each pin.

Table 14-1 on page 61 summarizes the function of the overriding signals. The pin and port indexes from Figure 14-3 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 14-1. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.

14.4.1 Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 14-2.

Table 14-2. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC0	INT0/ ICP0 (External Interrupt 0 or Timer/Counter0 Input Capture Trigger)

The alternate pin configuration is as follows:

- **INT0 - Port C, Bit 0**

INT0: External Interrupt Source 0. The PC0 pin can serve as external interrupt source. INT0 can be used as an interrupt pin regardless of whether another special function is enabled or not.

Table 14-3 relates the alternate functions of Port C to the overriding signals shown in Figure 14-3 on page 60.

Table 14-3. Overriding Signals for Alternate Functions in PC0

Signal Name	PC0/INT0
PVOE	0
DIEOE	INT Enable
DIEOV	1
DI	INT0 INPUT

14.5 Register Description

14.5.1 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	–	–	–	–	–	–	–	PORTC0	PORTC
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.5.2 PINC – Port C Input Pins Address

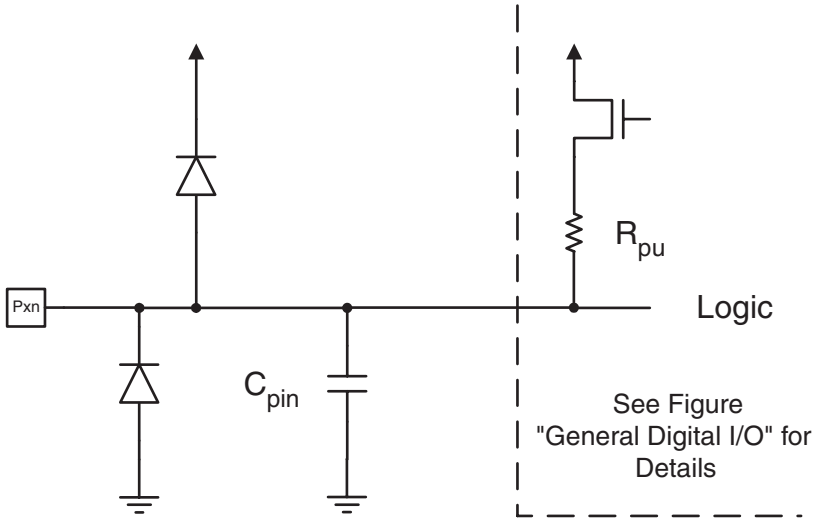
Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	–	–	–	–	–	–	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

15. Low Voltage I/O-Ports

15.1 Overview

All low voltage AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). All low voltage port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{REG} and Ground as indicated in Figure 15-1. Refer to ”Electrical Characteristics” on page 165 for a complete list of parameters.

Figure 15-1. Low Voltage I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in ”Register Description” on page 73.

Three I/O memory address locations are allocated for each low voltage port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all low voltage pins in all ports when set.

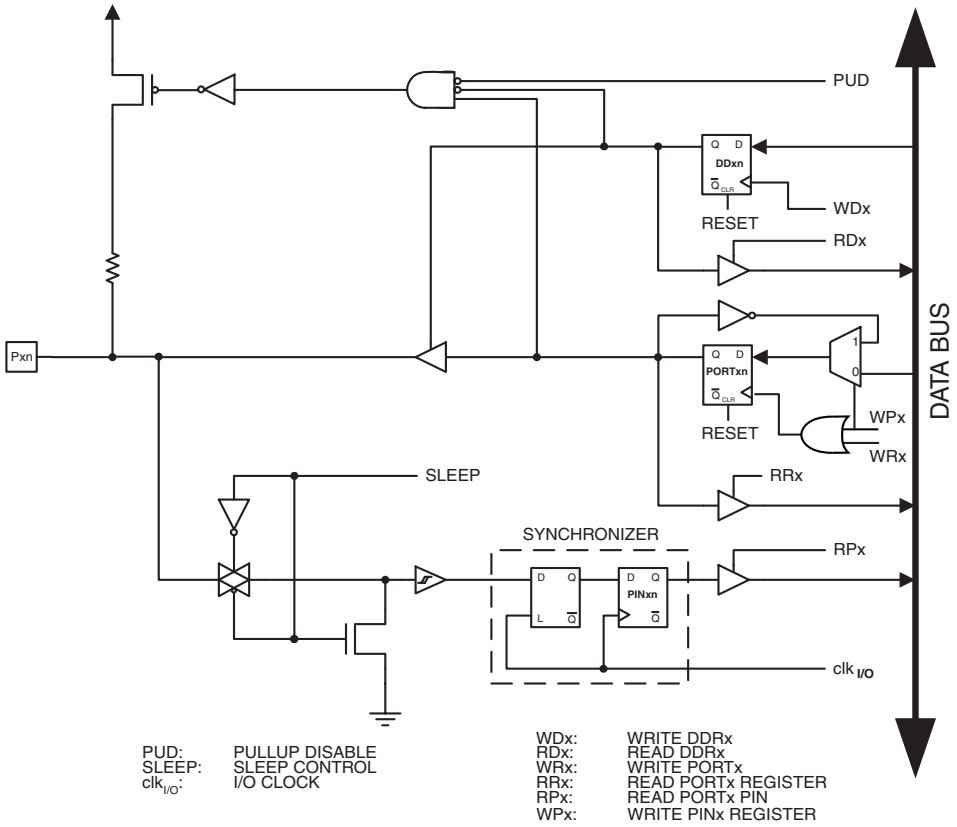
Using the I/O port as General Digital I/O is described in ”Low Voltage Ports as General Digital I/O” on page 64. Many low voltage port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in ”Alternate Port Functions” on page 68. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

15.2 Low Voltage Ports as General Digital I/O

The low voltage ports are bi-directional I/O ports with optional internal pull-ups. Figure 15-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 15-2. General Low Voltage Digital I/O(1)



Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

15.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in ”Register Description” on page 73, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

15.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

15.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 15-1 summarizes the control signals for the pin value.

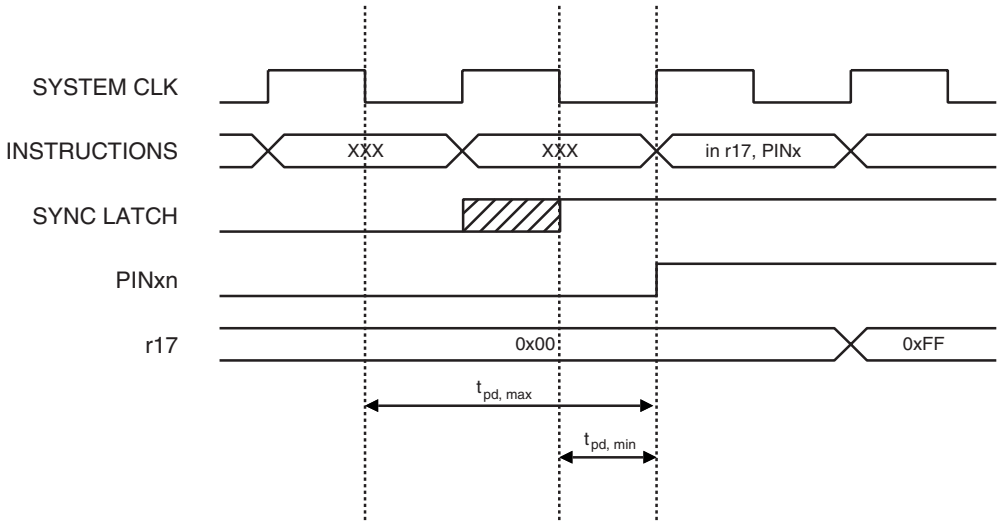
Table 15-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

15.2.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 15-2, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 15-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

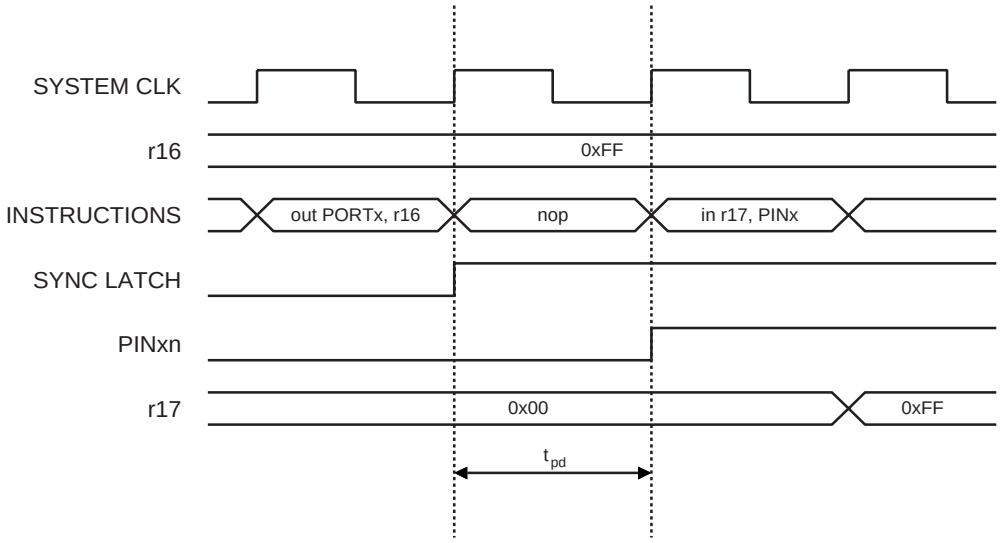
Figure 15-3. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 15-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 15-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi  r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out  PORTB,r16
out  DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

15.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 15-2 on page 64, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-save mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{REG}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 68.

If a logic high level ("one") is present on an asynchronous external interrupt pin configured as "Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin" while the external interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

15.2.6 Unconnected Pins

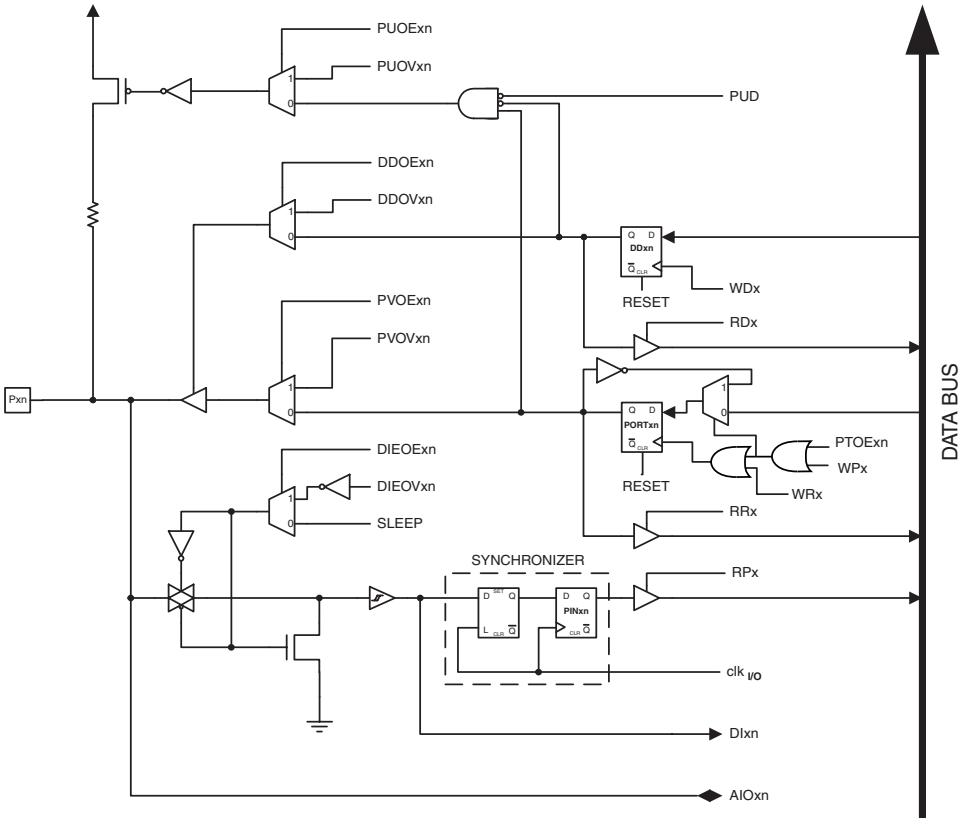
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

15.3 Alternate Port Functions

Many low voltage port pins have alternate functions in addition to being general digital I/Os. Figure 15-5 shows how the port pin control signals from the simplified Figure 15-2 on page 64 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 15-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE
SLEEP:	SLEEP CONTROL
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE
PUD:	PULLUP DISABLE
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WRx:	WRITE PORTx
RPx:	READ PORTx PIN
WPx:	WRITE PINx
$clk_{I/O}$:	I/O CLOCK
DIxn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 15-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 15-5 on page 68 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 15-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

15.3.1 Alternate Functions of Port A

The Port A pins with alternate functions are shown in Table 15-3.

Table 15-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA1	ADC1/ SGND/ T1 (ADC Input Channel 1, Signal Ground or Timer/Counter1 Clock Input)
PA0	ADC0/ SGND/ T0 (ADC Input Channel 0, Signal Ground or Timer/Counter0 Clock Input))

The alternate pin configuration is as follows:

- **ADC0/ SNGD/ T0 – Port A, Bit 0**

Voltage Analog to Digital Converter (Channel 0), Signal Ground for Voltage Analog to Digital Converter or Timer/Counter0 Counter Source.

- **ADC1/ SNGD/ T1 – Port A, Bit 1**

Voltage Analog to Digital Converter (Channel 1) , Signal Ground for Voltage Analog to Digital Converter or Timer/Counter1 Counter Source.

Table 15-4 relates the alternate functions of Port A to the overriding signals shown in Figure 15-5 on page 68.

Table 15-4. Overriding Signals for Alternate Functions in PA1..PA0

Signal Name	PA0/ADC0/SGND/T0	PA1/ADC1/SGND/T1
PUOE	0	0
PUOV	0	0
DDOE	0	0
DDOV	0	0
PVOE	0	0
PVOV	0	0
PTOE	–	–
DIEOE	PA0DID	PA1DID
DIEOV	0	0
DI	–	–
AIO	ADC0 INPUT/ SGND/ T0	ADC1 INPUT/ SGND/ T1

15.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 15-5.

Table 15-5. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB3	MISO/ INT2 (SPI Bus Master Input/Slave Output or External Interrupt 2 Input)
PB2	MOSI/ INT1 (SPI Bus Master Output/Slave Input or External Interrupt 1 Input)
PB1	SCK (SPI Bus Master clock Input)
PB0	$\overline{SS}$/ CKOUT (SPI Bus Master Slave select or Clock Output)

The alternate pin configuration is as follows:

- **MISO/INT2 - Port B, Bit 3**

MISO, Master Data input: Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB3. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB3. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB3 bit. When not operating in SPI mode, this pin can serve as an external interrupt source.

- **MOSI/INT1- Port B, Bit 2**

MOSI, SPI Master Data output: Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB2. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit. When not operating in SPI mode, this pin can serve as an external interrupt source.

- **SCK- Port B, Bit 1**

SCK, Master Clock output: Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB1. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB1. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB1 bit.

- **$\overline{SS}$/CKOUT- Port B, Bit 0**

$\overline{SS}$, Slave Select input: When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB0. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB0. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB0 bit. When not operating in SPI mode, this pin can serve as Clock Output, CPU Clock divided by 2. See ”Clock Output” on page 27.

Table 15-6. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/MISO	PB2/MOSI	PB1/SCK	PB0/$\overline{SS}$/CKOUT
PUOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$ • CKOE
PUOV	PORTB3 • $\overline{PUD}$	PORTB2 • $\overline{PUD}$	PORTB1 • $\overline{PUD}$	PORTB0 • $\overline{PUD}$
DDOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$ | CKOE
DDOV	0	0	0	CKOE
PVOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • MSTR	CKOE
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	CKOUT
PTOE	–	–	–	–
DIEOE	INT2 ENABLE	INT1 ENABLE	–	CKOE
DIEOV	INT2 ENABLE	INT1 ENABLE	–	0
DI	SPI MSTR INPUT INT2 INPUT	SPI SLAVE INPUT INT1 INPUT	SCK INPUT	SPI $\overline{SS}$
AIO	–	–	–	–

15.4 Register Description

15.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	–	CKOE	PUD	–	–	–	–	MCUCR
Read/Write	R	R	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See ”Configuring the Pin” on page 64 for more details about this feature.

15.4.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	-	-	-	-	-	-	PORTA1	PORTA0	PORTA
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.4.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	-	-	-	-	-	-	DDA1	DDA0	DDRA
Read/Write	R	R	R	R	R	R/	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.4.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	-	-	-	-	-	-	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

15.4.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	-	-	-	-	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.4.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	-	-	-	-	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.4.7 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	-	-	-	-	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

16. Timer/Counter0 and Timer/Counter1 Prescalers

16.1 Overview

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

16.1.1 Internal Clock Source

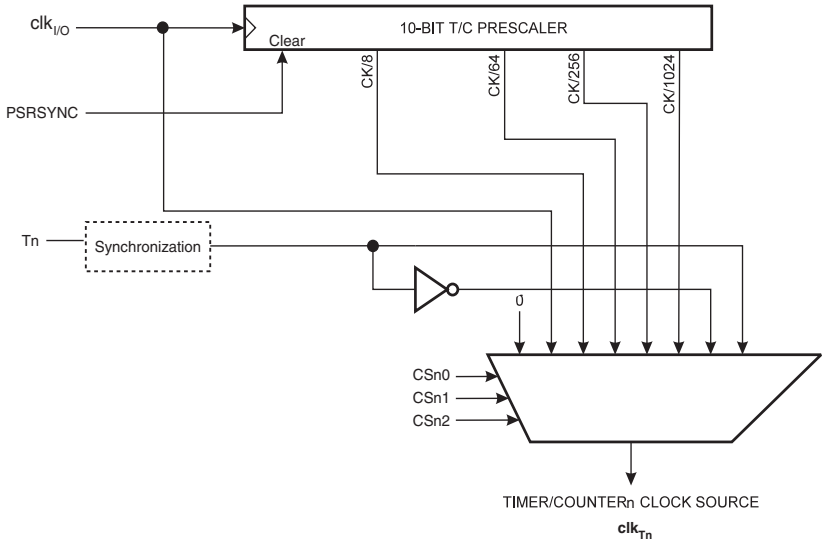
The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$.

16.1.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

Figure 16-1. Prescaler for Timer/Counter

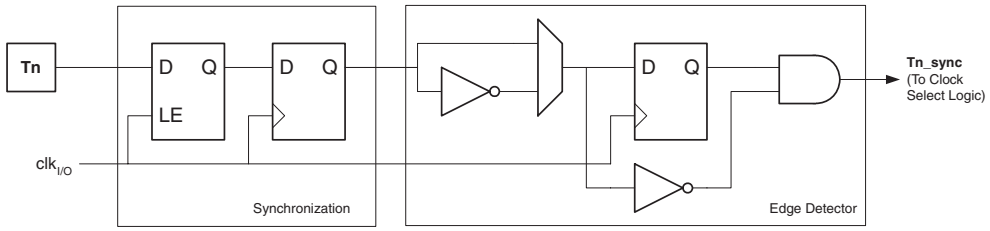


16.2 External Clock Source

An external clock source applied to the Tn pin can be used as Timer/Counter clock (clk_{Tn}). The Tn pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 16-2 shows a functional equivalent block diagram of the Tn synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{Tn} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects. See Table 16-1 on page 76 for details.

Figure 16-2. Tn Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the Tn pin to the counter is updated.

Enabling and disabling of the clock input must be done when Tn has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

16.3 Register Description

16.3.1 TCCRnB – Timer/Counter n Control Register B

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	CSn2	CSn1	CSn0	TCCRnB
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2, 1, 0 – CSn2, CSn1, CSn0: Clock Select n, Bit 2, 1, and 0**

The Clock Select n bits 2, 1, and 0 define the prescaling source of Timer n.

Table 16-1. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter n, transitions on the Tn pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

16.3.2 General Timer/Counter Control Register – GTCCR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	–	–	–	–	PSRSYNC	GTCCR
Read/Write	R/W	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRSYNC bit is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero the PSRSYNC bit is cleared by hardware, and the Timer/Counters start counting simultaneously.

- **Bit 0 – PSRSYNC: Prescaler Reset**

When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

17. Timer/Counter(T/C0,T/C1)

17.1 Features

- **Clear Timer on Compare Match (Auto Reload)**
- **Input Capture unit**
- **Four Independent Interrupt Sources (TOVn, OCFnA, OCFnB, ICFn)**
- **8-bit Mode with Two Independent Output Compare Units**
- **16-bit Mode with One Independent Output Compare Unit**

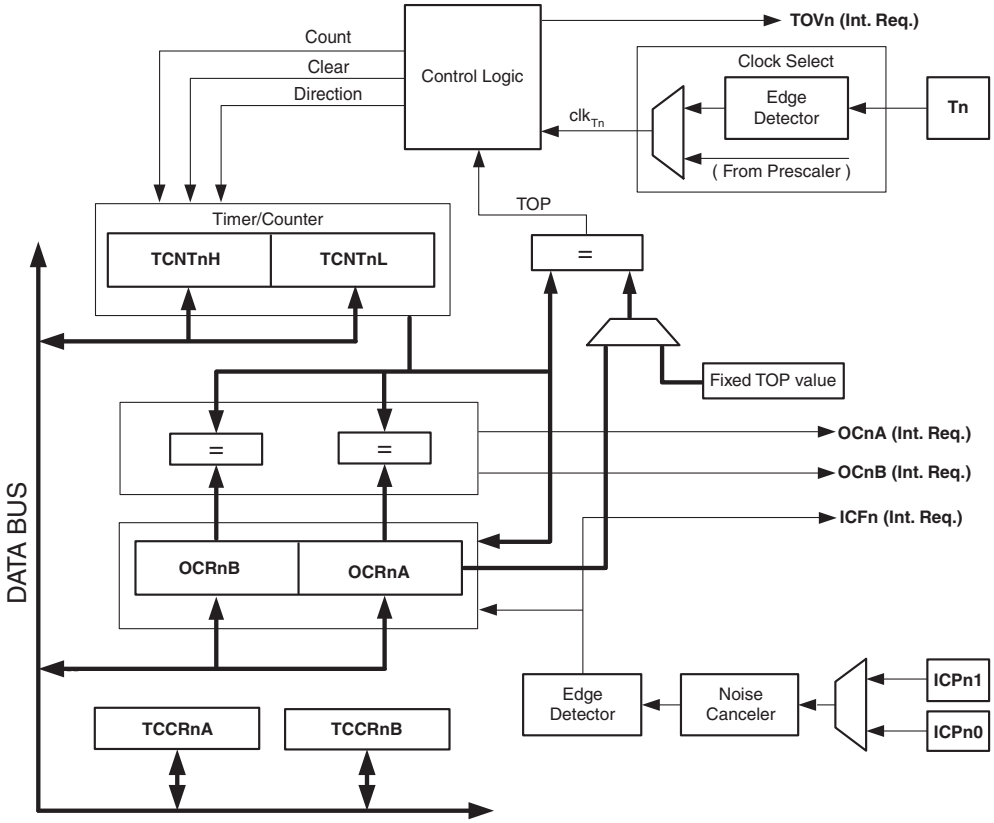
17.2 Overview

Timer/Counter n is a general purpose 8-/16-bit Timer/Counter module, with one/two Output Compare units and Input Capture functionality.

ATmega8HVA/16HVA has two Timer/Counters, Timer/Counter0 and Timer/Counter1. The functionality for both Timer/Counters is described below. Timer/Counter0 and Timer/Counter1 have different Timer/Counter registers, as shown in ”Register Summary” on page 175.

The Timer/Counter general operation is described in 8-/16-bit mode. A simplified block diagram of the 8-/16-bit Timer/Counter is shown in Figure 17-1. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the ”Register Description” on page 90.

Figure 17-1. 8-/16-bit Timer/Counter Block Diagram



17.2.1 Registers

The Timer/Counter Low Byte Register (TCNTnL) and Output Compare Registers (OCRnA and OCRnB) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in Figure 17-1 on page 77) signals are all visible in the Timer Interrupt Flag Register (TIFRn). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSKn). TIFRn and TIMSKn are not shown in the figure.

In 16-bit mode the Timer/Counter consists one more 8-bit register, the Timer/Counter High Byte Register (TCNTnH). Furthermore, there is only one Output Compare Unit in 16-bit mode as the two Output Compare Registers, OCRnA and OCRnB, are combined to one 16-bit Output Compare Register. OCRnA contains the low byte of the word and OCRnB contains the higher byte of the word. When accessing 16-bit registers, special procedures described in section "Accessing Registers in 16-bit Mode" on page 86 must be followed.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{Tn}).

17.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case "n" replaces the module number, e.g. Timer/Counter number. A lower case "x" replaces the unit, e.g. OCRnx and ICPnx describes OCRnB/A and ICP1/0x . However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0L for accessing Timer/Counter0 counter value and so on.

The definitions in Table 17-1 are also used extensively throughout the document.

Table 17-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255) in 8-bit mode or 0xFFFF (decimal 65535) in 16-bit mode.
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF/0xFFFF (MAX) or the value stored in the OCRnA Register.

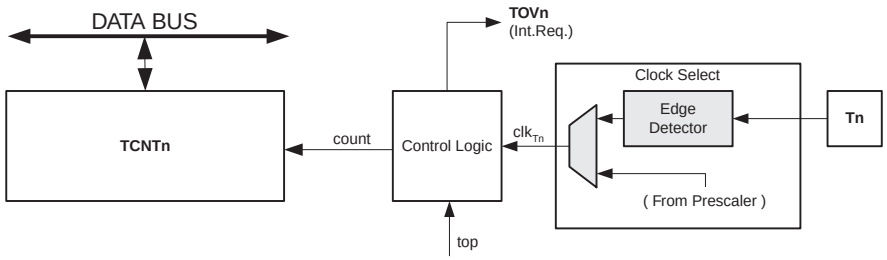
17.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source. The Clock Select logic is controlled by the Clock Select (CSn2:0) bits located in the Timer/Counter Control Register n B (TCCRnB), and controls which clock source and edge the Timer/Counter uses to increment its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{Tn}). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 74

17.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 17-2 on page 79 shows a block diagram of the counter and its surroundings.

Figure 17-2. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNTn by 1.

clk_{Tn} Timer/Counter clock, referred to as clk_{Tn} in the following.

top Signalize that TCNTn has reached maximum value.

The counter is incremented at each timer clock (clk_{Tn}) until it passes its TOP value and then restarts from BOTTOM. The counting sequence is determined by the setting of the WGMn0 bits located in the Timer/Counter Control Register (TCCRnA). For more details about counting sequences, see "Timer/Counter Timing Diagrams" on page 85. clk_{Tn} can be generated from an external or internal clock source, selected by the Clock Select bits (CSn2:0). When no clock source is selected (CSn2:0 = 0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, regardless of whether clk_{Tn} is present or not. A CPU write overrides (has priority over) all counter clear or count operations. The Timer/Counter Overflow Flag (TOVn) is set when the counter reaches the maximum value and it can be used for generating a CPU interrupt.

17.5 Modes of Operation

The mode of operation is defined by the Timer/Counter Width (TCWn), Input Capture Enable (ICENn) and the Waveform Generation Mode (WGMn0)bits in ”TCCRnA – Timer/Counter n Control Register A” on page 90. Table 17-2 on page 80 shows the different Modes of Operation.

Table 17-2. Modes of Operation

Mode	ICENn	TCWn	WGMn0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal 8-bit Mode	0xFF	Immediate	MAX (0xFF)
1	0	0	1	8-bit CTC	OCRnA	Immediate	MAX (0xFF)
2	0	1	0	16-bit Mode	0xFFFF	Immediate	MAX (0xFFFF)
3	0	1	1	16-bit CTC	OCRnB, OCRnA	Immediate	MAX (0xFFFF)
4	1	0	0	8-bit Input Capture mode	0xFF	–	MAX (0xFF)
5	1	1	0	16-bit Input Capture mode	0xFFFF	–	MAX (0xFFFF)

17.5.1 Normal 8-bit Mode

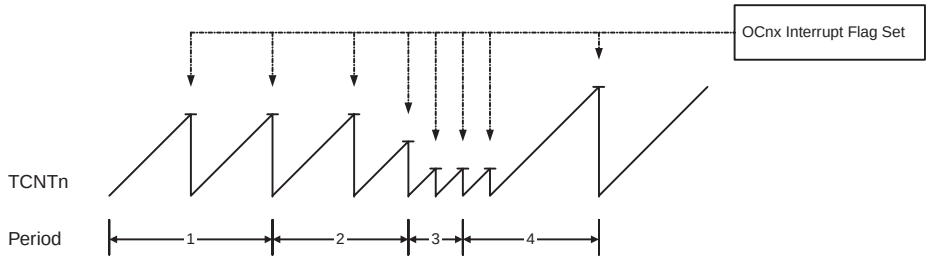
In the normal mode, the counter (TCNTnL) is incrementing until it overruns when it passes its maximum 8-bit value (MAX = 0xFF) and then restarts from the bottom (0x00), see Table 17-2 on page 80 for bit settings. The Overflow Flag (TOVn) will be set in the same timer clock cycle as the TCNTnL becomes zero. The TOVn Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal 8-bit mode, a new counter value can be written anytime. The Output Compare Unit can be used to generate interrupts at some given time.

17.5.2 Clear Timer on Compare Match (CTC) 8-bit Mode

In Clear Timer on Compare or CTC mode, the OCRnA Register is used to manipulate the counter resolution, see Table 17-2 on page 80 for bit settings. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches the OCRnA. The OCRnA defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 17-3 on page 81. The counter value (TCNTn) increases until a Compare Match occurs between TCNTn and OCRnA, and then counter (TCNTn) is cleared.

Figure 17-3. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCFnA Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care. If the new value written to OCRnA is lower than the current value of TCNTn, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the Compare Match can occur. As for the Normal mode of operation, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

17.5.3 16-bit Mode

In 16-bit mode, the counter (TCNTnH/L) is a incrementing until it overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the bottom (0x0000), see Table 17-2 on page 80 for bit settings. The Overflow Flag (TOVn) will be set in the same timer clock cycle as the TCNTnH/L becomes zero. The TOVn Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime. The Output Compare Unit can be used to generate interrupts at some given time.

17.5.4 Clear Timer on Compare Match (CTC) 16-bit Mode

In Clear Timer on Compare 16-bit mode, OCRnB/A Registers are used to manipulate the counter resolution, see Table 17-2 on page 80 for bit settings. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches OCRnB/A, where OCRnB represents the eight most significant bits and OCRnA represents the eight least significant bits. OCRnB/A defines the top value of the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

An interrupt can be generated each time the counter reaches the TOP value by using the OCFnA flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close the BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCRnB/A is lower than the current value of TCNTn, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before Compare Match can occur. As for the 16-bit Mode, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

17.5.5 8-bit Input Capture Mode

The Timer/Counter can be used in a 8-bit Input Capture mode, see Table 17-2 on page 80 for bit settings. For full description, see "Input Capture Unit" on page 82.

17.5.6 16-bit Input Capture Mode

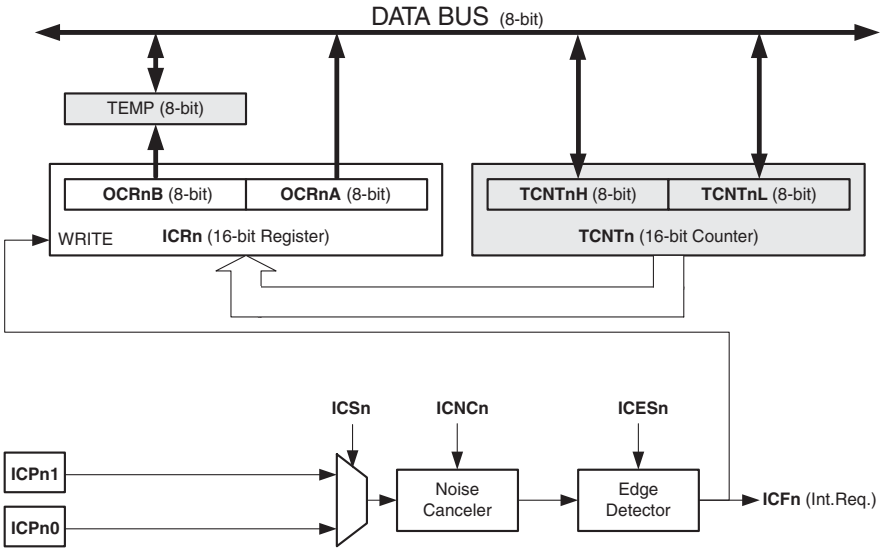
The Timer/Counter can also be used in a 16-bit Input Capture mode, see Table 17-2 on page 80 for bit settings. For full description, see "Input Capture Unit" on page 82.

17.6 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicates an event, or multiple events. For Timer/Counter0, the events can be applied via the PC0 pin (ICP01), or alternatively via the osi_posedge pin on the Oscillator Sampling Interface (ICP00). For Timer/Counter1, the events can be applied by the Battery Protection Interrupt (ICP10) or alternatively by the Voltage Regulator Interrupt (ICP11). The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 17-4 on page 82. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 17-4. Input Capture Unit Block Diagram



The Output Compare Register OCRnA is a dual-purpose register that is also used as an 8-bit Input Capture Register ICRn. In 16-bit Input Capture mode the Output Compare Register OCRnB serves as the high byte of the Input Capture Register ICRn. In 8-bit Input Capture mode the Output Compare Register OCRnB is free to be used as a normal Output Compare Register, but in 16-bit Input Capture mode the Output Compare Unit cannot be used as there are no free Output Compare Register(s). Even though the Input Capture register is called ICRn in this section, it is referring to the Output Compare Register(s). For more information on how to access the 16-bit registers refer to "Accessing Registers in 16-bit Mode" on page 86.

When a change of the logic level (an event) occurs on the *Input Capture pin* (ICPx), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the value of the counter (TCNTn) is written to the *Input Capture Register* (ICRn). The *Input Capture Flag* (ICFn) is set at the same system clock as the TCNTn value is copied into Input Capture Register. If enabled (TICIEn=1), the Input Capture Flag generates an Input Capture interrupt. The ICFn flag is automatically cleared when the interrupt is executed. Alternatively the ICFn flag can be cleared by software by writing a logical one to its I/O bit location.

17.6.1 Input Capture Trigger Source

The default trigger source for the Input Capture unit is the I/O port PC0 in Timer/Counter0 and the Battery Protection Interrupt in Timer/Counter1. Alternatively can the osi_posedge pin on the Oscillator Sampling Interface in Timer/Counter0 and Voltage Regulator Interrupt in Timer/Counter1 be used as trigger sources. The osi_posedge pin in Timer/Counter0 Control Register A (TCCR0A) and the Voltage Regulator Interrupt bit in the Timer/Counter1 Control Register A (TCCR1A) is selected as trigger sources by setting the Input Capture Select bits respectively to 00 and 11. Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both Input Capture inputs are sampled using the same technique. The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. An Input Capture on Timer/Counter0 can also be triggered by software by controlling the port of the PC0 pin.

17.6.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the Input Capture Noise Canceler (ICNCn) bit in *Timer/Counter Control Register n B* (TCCRnB). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICRn Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

17.6.3 Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICRn Register before the next event occurs, the ICRn will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture interrupt, the ICRn Register should be read as early in the interrupt handler routine as possible. The maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Measurement of an external signal duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICRn Register has been read. After a change of the edge, the Input Capture Flag (ICFn) must be

cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the trigger edge change is not required.

Table 17-3. Timer/Counter0 Input Capture Source (ICS)

ICS0	Source
0	ICP00: osi_posedge pin from OSI module(1)
1	ICP01: Port PC0

Note: 1. See "OSI – Oscillator Sampling Interface" on page 28 for details.

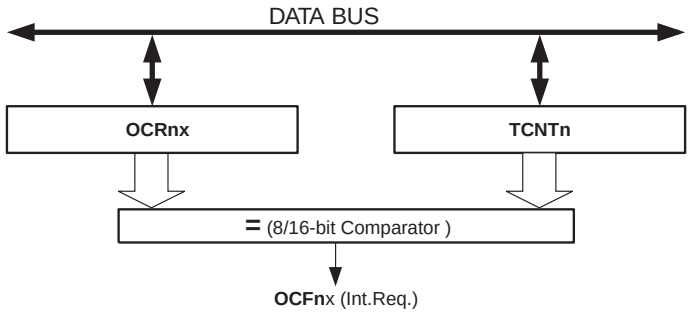
Table 17-4. Timer/Counter1 Input Capture Source (ICS)

ICS1	Source
0	ICP10: Battery Protection Interrupt
1	ICP11: Voltage Regulator Interrupt

17.7 Output Compare Unit

The comparator continuously compares the Timer/Counter (TCNTn) with the Output Compare Registers (OCRnA and OCRnB), and whenever the Timer/Counter equals to the Output Compare Registers, the comparator signals a match. A match will set the Output Compare Flag at the next timer clock cycle. In 8-bit mode the match can set either the Output Compare Flag OCFnA or OCFnB, but in 16-bit mode the match can set only the Output Compare Flag OCFnA as there is only one Output Compare Unit. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. Figure 17-5 on page 84 shows a block diagram of the Output Compare unit.

Figure 17-5. Output Compare Unit, Block Diagram



17.7.1 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNTnH/L Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCRnB/A to be initialized to the same value as TCNTn without triggering an interrupt when the Timer/Counter clock is enabled.

17.7.2 Using the Output Compare Unit

Since writing TCNTnH/L will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNTnH/L when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNTnH/L equals the OCRnB/A value, the Compare Match will be missed.

17.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 17-6 on page 85 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 17-6. Timer/Counter Timing Diagram, no Prescaling

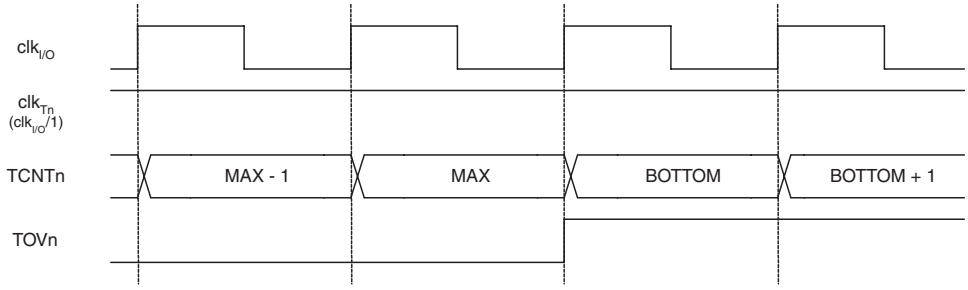


Figure 17-7 on page 85 shows the same timing data, but with the prescaler enabled.

Figure 17-7. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

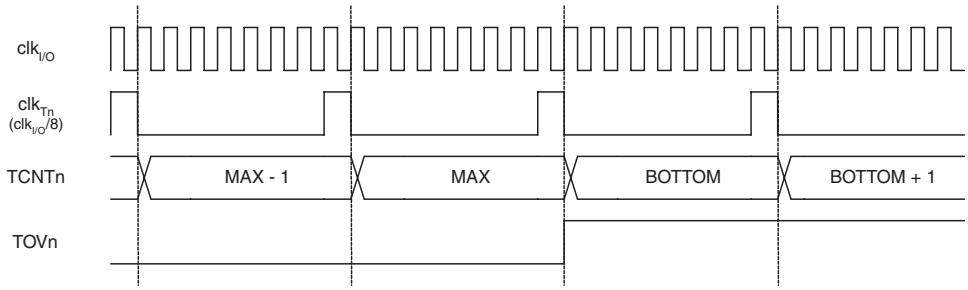
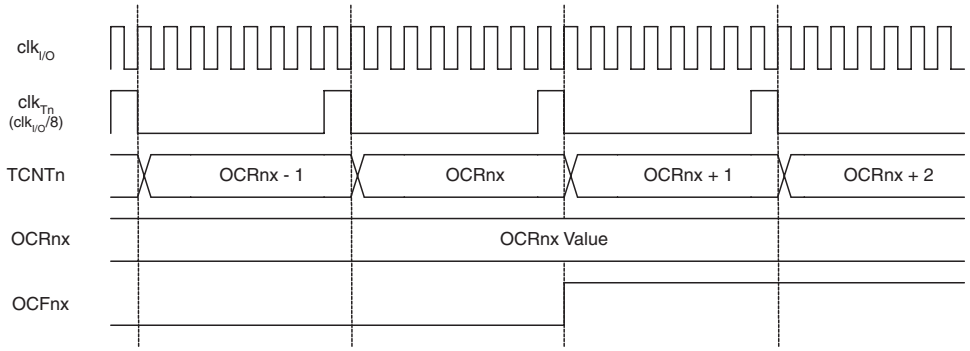


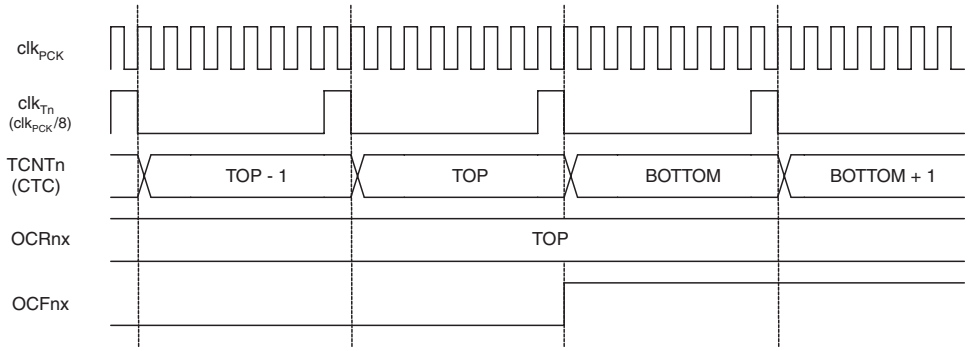
Figure 17-8 on page 86 shows the setting of OCFnA and OCFnB in Normal mode.

Figure 17-8. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk\_I/O}/8$)



shows the setting of OCFnA and the clearing of TCNTn in CTC mode.

Figure 17-9. Timer/Counter Timing Diagram, CTC mode, with Prescaler ($f_{clk\_I/O}/8$)



17.9 Accessing Registers in 16-bit Mode

In 16-bit mode (the TCWn bit is set to one) the TCNTnH/L and OCRnB/A are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. The 16-bit Timer/Counter has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written, are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

There is one exception in the temporary register usage. In the Output Compare mode the 16-bit Output Compare Register OCRnB/A is read without the temporary register, because the Output Compare Register contains a fixed value that is only changed by CPU access. However, in 16-bit Input Capture mode the ICRn register formed by the OCRnA and OCRnB registers must be accessed with the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCRnB/A registers.

Assembly Code Example

```
...
; Set TCNTn to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNTnH,r17
out TCNTnL,r16
; Read TCNTn into r17:r16
in  r16,TCNTnL
in  r17,TCNTnH
...
```

C Code Example

```
unsigned int i;
...
/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;
/* Read TCNTn into i */
i = TCNTn;
...
```

Note: 1. See “About Code Examples” on page 7.

The assembly code example returns the TCNTnH/L value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNTn register contents. Reading any of the OCRn register can be done by using the same principle.

Assembly Code Example

```
TIMn_ReadTCNTn:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNTn into r17:r16
  in  r16,TCNTnL
  in  r17,TCNTnH
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example

```
unsigned int TIMn_ReadTCNTn( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Read TCNTn into i */
  i = TCNTn;
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Note: 1. See “About Code Examples” on page 7.

The assembly code example returns the TCNTnH/L value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNTnH/L register contents. Writing any of the OCRnB/A registers can be done by using the same principle.

Assembly Code Example

```
TIMn_WriteTCNTn:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNTn to r17:r16
  out TCNTnH,r17
  out TCNTnL,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example

```
void TIMn_WriteTCNTn( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNTn to i */
  TCNTn = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

Note: See “About Code Examples” on page 7.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNTnH/L.

17.9.1 Reusing the temporary high byte register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

17.10 Register Description

17.10.1 TCCRnA – Timer/Counter n Control Register A

Bit	7	6	5	4	3	2	1	0	
	TCWn	ICENn	ICNCn	ICESn	ICSn	–	–	WGMn0	TCCRnA
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– TCWn: Timer/Counter Width**

When this bit is written to one 16-bit mode is selected. The Timer/Counter width is set to 16-bits and the Output Compare Registers OCRnA and OCRnB are combined to form one 16-bit Output Compare Register. Because the 16-bit registers TCNTnH/L and OCRnB/A are accessed by the AVR CPU via the 8-bit data bus, special procedures must be followed. These procedures are described in section ”Accessing Registers in 16-bit Mode” on page 86.

- **Bit 6– ICENn: Input Capture Mode Enable**

The Input Capture Mode is enabled when this bit is written to one.

- **Bit 5 – ICNCn: Input Capture Noise Canceler**

Setting this bit activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture Source is filtered. The filter function requires four successive equal valued samples of the Input Capture Source for changing its output. The Input Capture is therefore delayed by four System Clock cycles when the noise canceler is enabled.

- **Bit 4 – ICESn: Input Capture Edge Select**

This bit selects which edge on the Input Capture Source that is used to trigger a capture event. When the ICESn bit is written to zero, a falling (negative) edge is used as trigger, and when the ICESn bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICESn setting, the counter value is copied into the Input Capture Register. The event will also set the Input Capture Flag (ICFn), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

- **Bit 3 - ICSn: Input Capture Select**

When written to a logic one, this bit selects the alternate Input Capture Source as trigger for the Timer/Counter input capture function. To trigger the Timer/Counter Input Capture interrupt, the ICIEn bit in the Timer Interrupt Mask Register (TIMSK) must be set. See Table 17-3 on page 84 and Table 17-4 on page 84.

- **Bits 2:0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bit 0 – WGMn0: Waveform Generation Mode**

This bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, see Figure 17-6 on page 85. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter) and Clear Timer on Compare Match (CTC) mode (see ”Timer/Counter Timing Diagrams” on page 85).

17.10.2 TCNTnL – Timer/Counter n Register Low Byte

Bit	7	6	5	4	3	2	1	0	
	TCNTnL[7:0]								TCNTnL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register TCNTnL gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNTnL Register blocks (disables) the Compare Match on the following timer clock. Modifying the counter (TCNTnL) while the counter is running, introduces a risk of missing a Compare Match between TCNTnL and the OCRnx Registers. In 16-bit mode the TCNTnL register contains the lower part of the 16-bit Timer/Counter n Register.

17.10.3 TCNTnH – Timer/Counter n Register High Byte

Bit	7	6	5	4	3	2	1	0	
	TCNTnH[7:0]								TCNTnH
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

When 16-bit mode is selected (the TCWn bit is set to one) the Timer/Counter Register TCNTnH combined to the Timer/Counter Register TCNTnL gives direct access, both for read and write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See ”Accessing Registers in 16-bit Mode” on page 86. In 8-bit mode, this register is accessable for both reading and writing, but will not be updated by the counter.

17.10.4 OCRnA – Timer/Counter n Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
	OCRnA[7:0]								OCRnA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNTnL). A match can be used to generate an Output Compare interrupt.

In 16-bit mode the OCRnA register contains the low byte of the 16-bit Output Compare Register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See ”Accessing Registers in 16-bit Mode” on page 86.

Note that the OCRnA is not writable in Input Capture mode.

17.10.5 OCRnB – Timer/Counter n Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
	OCRnB[7:0]								OCRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNTnL in 8-bit mode and TCNTnH in 16-bit mode). A match can be used to generate an Output Compare interrupt.

In 16-bit mode the OCRnB register contains the high byte of the 16-bit Output Compare Register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing Registers in 16-bit Mode" on page 86.

Note that the OCRnB is not writable in Input Capture mode.

17.10.6 TIMSKn – Timer/Counter n Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ICIEn	OCIEnB	OCIEnA	TOIEn	TIMSKn
Read/Write	R	R	R	R	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3 – ICIEn: Timer/Counter n Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter n Input Capture interrupt is enabled. The corresponding Interrupt Vector (See Section "12." on page 52.) is executed when the ICFn flag, located in TIFRn, is set.

- **Bit 2 – OCIEnB: Timer/Counter n Output Compare Match B Interrupt Enable**

When the OCIEnB bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCFnB bit is set in the "TIFRn – Timer/Counter n Interrupt Flag Register" on page 93.

- **Bit 1 – OCIEnA: Timer/Counter n Output Compare Match A Interrupt Enable**

When the OCIEnA bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter n Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter n occurs, i.e., when the OCFnA bit is set in the "TIFRn – Timer/Counter n Interrupt Flag Register" on page 93.

- **Bit 0 – TOIEn: Timer/Counter n Overflow Interrupt Enable**

When the TOIEn bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter n Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter n occurs, i.e., when the TOVn bit is set in the "TIFRn – Timer/Counter n Interrupt Flag Register" on page 93.

17.10.7 TIFRn – Timer/Counter n Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ICFn	OCFnB	OCFnA	TOVn	TIFRn
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 3 – ICFn: Timer/Counter n Input Capture Flag**

This flag is set when a capture event occurs, according to the setting of ICENn, ICESn and ICSn bits in the TCCRnA Register.

ICFn is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICFn can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCFnB: Output Compare Flag n B**

The OCFnB bit is set when a Compare Match occurs between the Timer/Counter and the data in OCRnB – Output Compare Register n B. OCFnB is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCFnB is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIEnB (Timer/Counter Compare B Match Interrupt Enable), and OCFnB are set, the Timer/Counter Compare Match Interrupt is executed.

The OCFnB is not set in 16-bit Output Compare mode when the Output Compare Register OCRnB is used as the high byte of the 16-bit Output Compare Register or in 16-bit Input Capture mode when the Output Compare Register OCRnB is used as the high byte of the Input Capture Register.

- **Bit 1– OCFnA: Output Compare Flag n A**

The OCFnA bit is set when a Compare Match occurs between the Timer/Counter n and the data in OCRnA – Output Compare Register n. OCFnA is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCFnA is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIEnA (Timer/Counter n Compare Match Interrupt Enable), and OCFnA are set, the Timer/Counter n Compare Match Interrupt is executed.

The OCFnA is also set in 16-bit mode when a Compare Match occurs between the Timer/Counter n and 16-bit data in OCRnB/A. The OCFnA is not set in Input Capture mode when the Output Compare Register OCRnA is used as an Input Capture Register.

- **Bit 0 – TOVn: Timer/Counter n Overflow Flag**

The bit TOVn is set when an overflow occurs in Timer/Counter n. TOVn is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOVn is cleared by writing a logic one to the flag. When the SREG I-bit, TOIEn (Timer/Counter n Overflow Interrupt Enable), and TOVn are set, the Timer/Counter n Overflow interrupt is executed.

18. SPI – Serial Peripheral Interface

18.1 Features

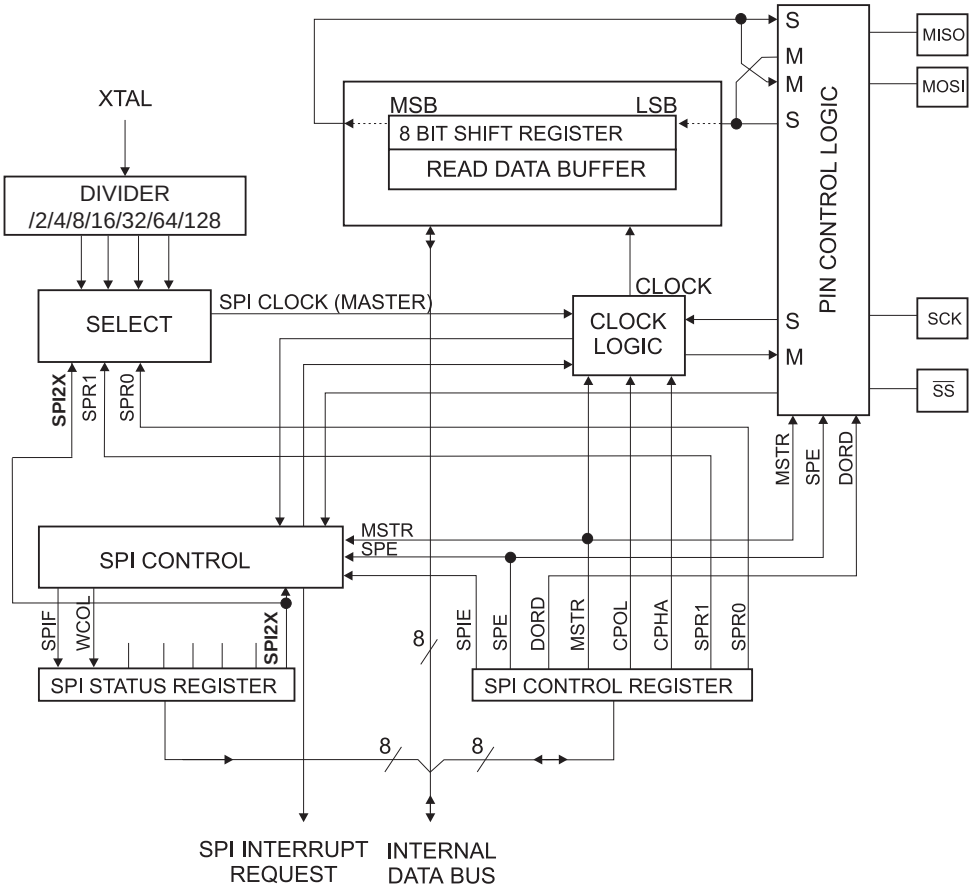
- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Protection Flag**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

18.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega8HVA/16HVA and peripheral devices or between several AVR devices.

The PRSPI bit in ”PRR0 – Power Reduction Register 0” on page 39 must be written to zero to enable SPI module.

Figure 18-1. SPI Block Diagram[1]



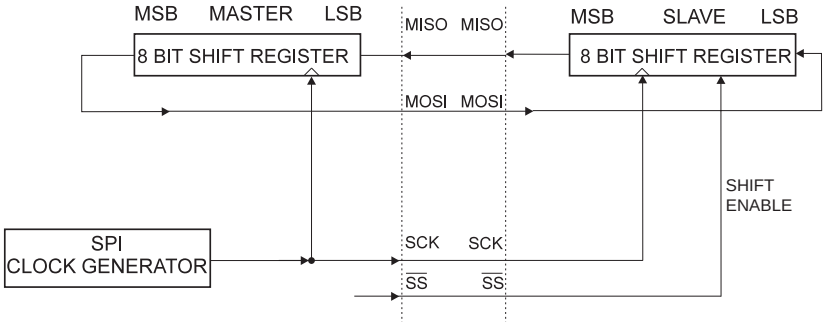
Note: 1. Refer to ”Alternate Port Functions” on page 68 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 18-2. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 18-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{osc}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 18-1 on page 96. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 68.

Table 18-1. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See "Alternate Functions of Port B" on page 71 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.

Assembly Code Example[1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi  r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out  DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi  r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out  SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out  SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis SPSR,SPIF
  rjmp Wait_Transmit
  ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

Note: 1. See “About Code Examples” on page 7.

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example[1]

```
SPI_SlaveInit:
  ; Set MISO output, all others input
  ldi   r17,(1<<DD_MISO)
  out   DDR_SPI,r17
  ; Enable SPI
  ldi   r17,(1<<SPE)
  out   SPCR,r17
  ret

SPI_SlaveReceive:
  ; Wait for reception complete
  sbis SPSR,SPIF
  rjmp SPI_SlaveReceive
  ; Read received data and return
  in    r16,SPDR
  ret
```

C Code Example[1]

```
void SPI_SlaveInit(void)
{
  /* Set MISO output, all others input */
  DDR_SPI = (1<<DD_MISO);
  /* Enable SPI */
  SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)))
    ;
  /* Return Data Register */
  return SPDR;
}
```

Note: 1. See “About Code Examples” on page 7.

18.3 $\overline{SS}$ Pin Functionality

18.3.1 Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

18.3.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

18.4 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 18-3 and Figure 18-4 on page 100. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 18-3 on page 101 and Table 18-4 on page 101, as done in Table 18-2.

Table 18-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing eDge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

Figure 18-3. SPI Transfer Format with CPHA = 0

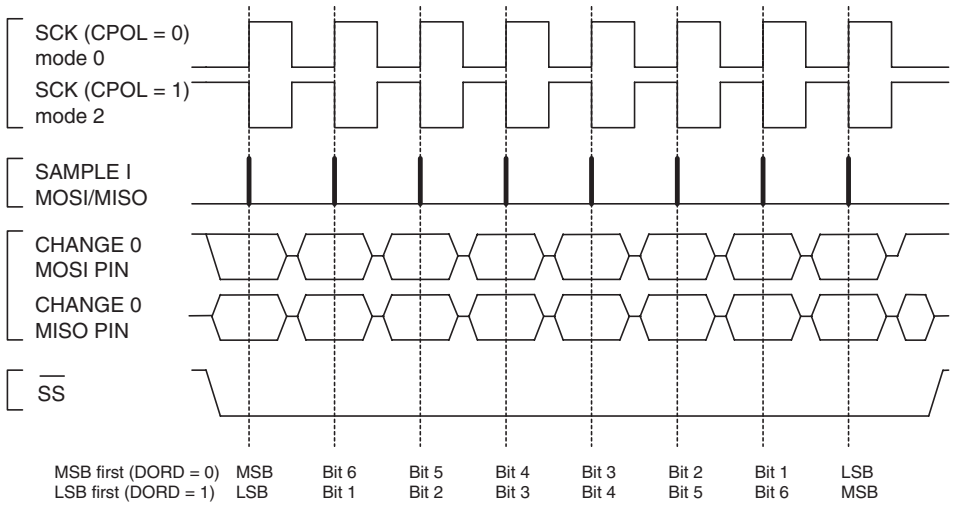
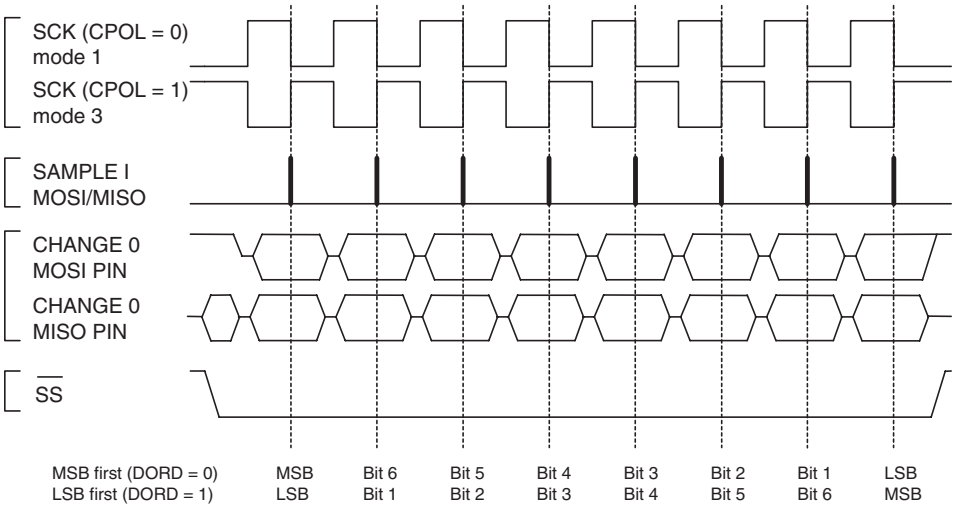


Figure 18-4. SPI Transfer Format with CPHA = 1



18.5 Register Description

18.5.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 18-3 and Figure 18-4 for an example. The CPOL functionality is summarized below:

Table 18-3. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 18-3 and Figure 18-4 for an example. The CPOL functionality is summarized below:

Table 18-4. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 18-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

18.5.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write Collision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5:1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 18-5 on page 102). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega8HVA/16HVA is also used for program memory and EEPROM downloading or uploading. See Table 27.6 on page 151 for serial programming and verification.

18.5.3 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

19. Coulomb Counter - Dedicated Fuel Gauging Sigma-delta ADC

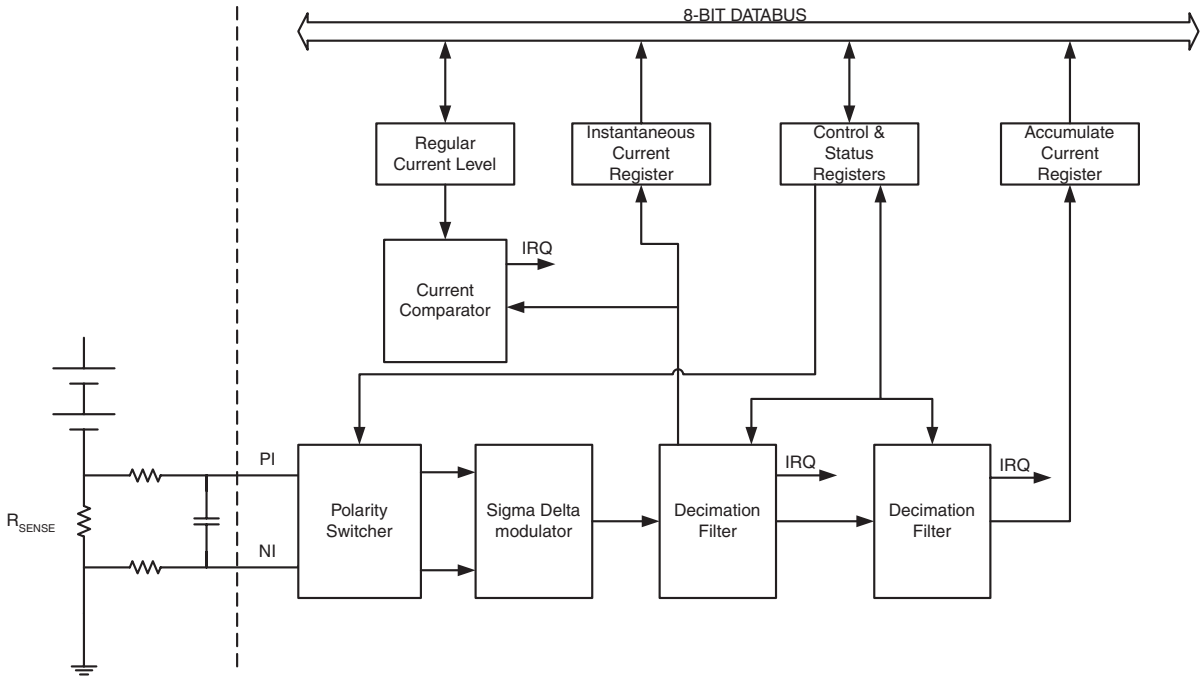
19.1 Features

- **Sampled System Coulomb Counter**
- **Low Power Sigma-Delta ADC Optimized for Coulomb Counting**
- **Instantaneous Current Output with 3.9 ms Conversion Time**
  - **13 bit Resolution (including sign bit)**
  - **Interrupt on Instantaneous Current Conversion Complete**
- **Accumulate Current Output**
  - **Programmable Conversion Time: 125/250/500/1000 ms**
  - **18-bit Resolution (including sign bit)**
  - **Interrupt on Accumulation Current Conversion Complete**
- **Regular Current Detection Mode**
  - **Programmable Sampling Interval: 250/500/1000/2000 ms**
- **Input Voltage Range ± 100mV**
  - **Allowing Measurement of ± 10A @ 10 mΩ**
- **Offset canceling by input polarity switching**

19.2 Overview

ATmega8HVA/16HVA features a dedicated Sigma-Delta ADC (CC-ADC) optimized for Coulomb Counting. By sampling the voltage across an external sense resistor R_{SENSE}, the CC-ADC is used to track the flow of current going into and out of the battery cells.

Figure 19-1. Coulomb Counter Block Diagram



In normal conversion mode two different output values are provided, Instantaneous Current and Accumulate Current. The Instantaneous Current Output has a short conversion time at the cost

of lower resolution. The Accumulate Current Output provides a highly accurate current measurement for Coulomb Counting.

The CC-ADC also provides a special Regular Current detection mode. This allows ultra-low power operation in Power-save mode when small charge or discharge currents are flowing.

For offset cancellation the polarity of the input signal could be switched run time. Using this feature correctly will remove the internal CC-ADC offset. See application note AVR352.

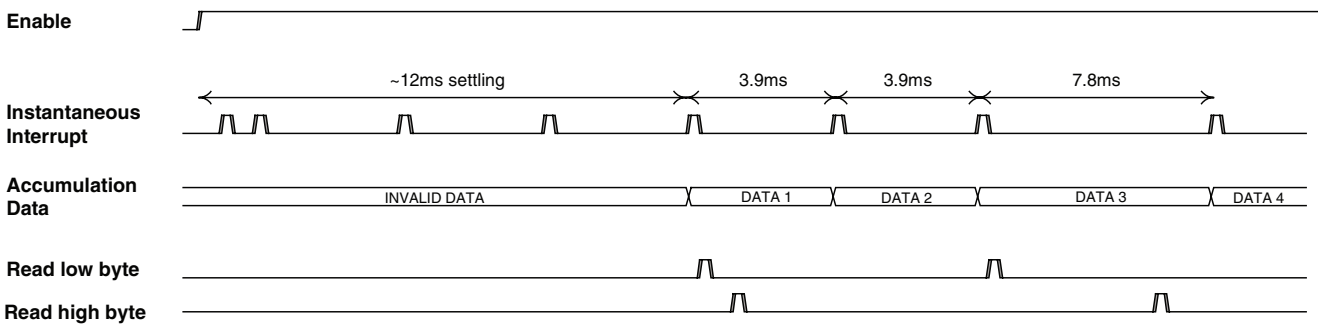
19.3 Normal Operation

When enabled the CC-ADC continuously measures the voltage over the external sense resistor R_{SENSE}. Running in normal conversion mode, two data conversion output is provided.

- Instantaneous Conversion Result
- Accumulation Conversion Result

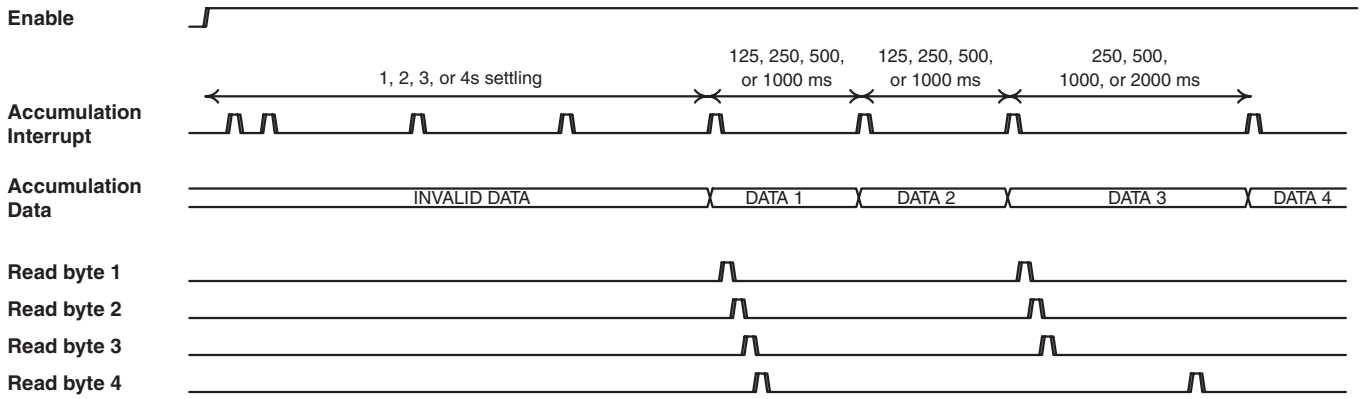
The Instantaneous Current conversion time is fixed to 3.9 ms (typical value) allowing the output value to closely follow the input. After each Instantaneous Current conversion an interrupt is generated if the interrupt is enabled. Data from conversion will be updated in the Instantaneous Current registers CADICL and CADICH at the same time as the interrupt is given. To avoid losing conversion data, both the low and high byte must be read within a 3.9 ms timing window after the corresponding interrupt is given. When the low byte register is read, updating of the Instantaneous Current registers and interrupts will be stopped until the high byte is read. Figure 19-2 shows an Instantaneous Current conversion diagram, where DATA4 will be lost because DATA3 reading is not completed within the limited period.

Figure 19-2. Instantaneous Current Conversions



The Accumulate Current output is a high-resolution, high accuracy output with programmable conversion time selected by the CADAS bits in CADCSRA. The converted value is an accurate measurement of the average current flow during one conversion period. The CC-ADC generates an interrupt each time a new Accumulate Current conversion has finished if the interrupt is enabled. Data from conversion will be updated in the Accumulation Current registers CADAC0, CADAC1, CADAC2 and CADAC3 at the same time as the interrupt is given. To avoid losing conversion data, all bytes must be read within the selected conversion period. When the lower byte registers are read, updating of the Accumulation Current registers and interrupts will be stopped until the highest byte is read. Table 19-3 shows an Accumulation Current conversion diagram, where DATA4 will be lost because DATA3 reading is not completed within the limited period.

Figure 19-3. Accumulation Current Conversions



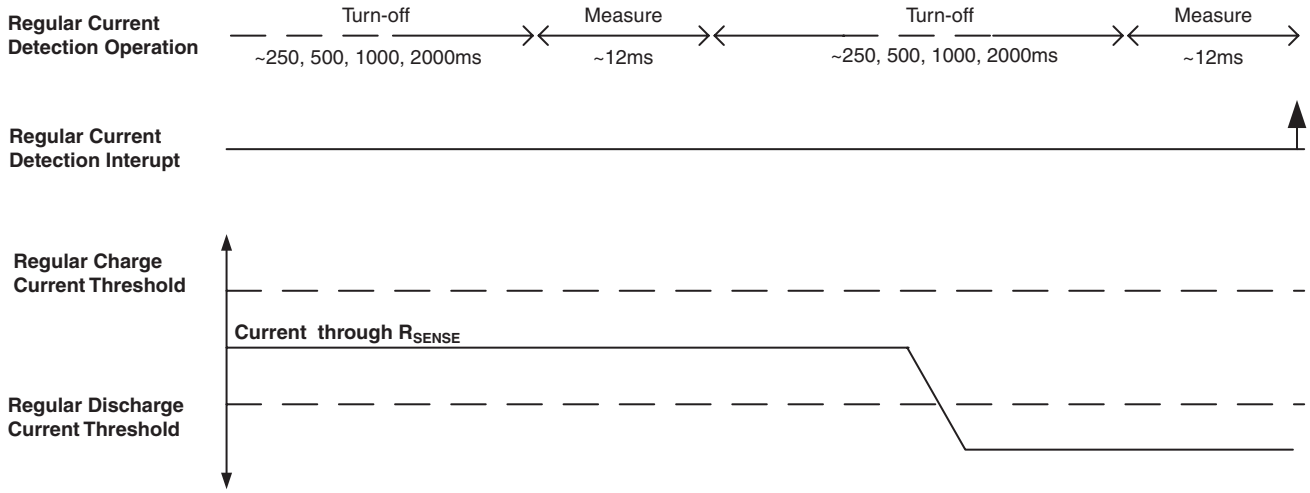
19.4 Regular Current Detection Operation

By setting the CADSE bit in CADCSRA the CC-ADC will enter a special Regular Current Detection Sampling Mode.

In this mode the CC-ADC will do one Instantaneous Current Conversion on regular sampling intervals while updating of the Accumulation Current Register is automatically disabled. The sampling interval is controlled by writing to the CADSI bits in CADCSRA.

Each time a conversion is completed the result is compared with Regular Charge/Discharge Threshold levels specified in the CADRC register. If interrupt is enabled and the voltage is above/below the specified limit a Regular Current Detection Interrupt will be issued. Figure 19-4 illustrates the Regular Current Detection Mode

Figure 19-4. Regular Current Detection Mode (CADSE=1)



The Regular Current Detection has a separate Interrupt and by setting the CADRCIE bit, this interrupt is enabled. Note that this Regular Current Detection interrupt cannot wake-up the CPU from sleep mode. To be able to use the Regular Current Detection function in sleep modes, the

Instantaneous Current Interrupt should be enabled as wake-up source by setting the CADICIE bit. The device will then wake-up from sleep after each single IC measurement. To check if Regular Current Detection has occurred the Regular Current Detection flag, CADRCIF, should be read.

19.5 Offset Canceling by Polarity Switching

The CC-ADC offers Polarity Switching for internal offset canceling. By switching the polarity of the sampled input signal at selected time intervals, the internal voltage offset of the CC-ADC will cancel at the output. This feature prevents the CC-ADC from accumulating an offset error over time.

19.6 Configuration and Usage

While the CC-ADC is converting, the CPU can enter sleep mode and wait for an interrupt. After adding the conversion data for the Coulomb Counting, the CPU can go back to sleep again. This reduces the CPU workload, and allows more time spent in low power modes, reducing power consumption.

To use the CC-ADC, the bandgap voltage reference must be enabled. See ”Voltage Reference and Temperature Sensor” on page 117.

The CC-ADC will not consume power when CADEN is cleared. It is therefore recommended to switch off the CC-ADC whenever the Coulomb Counter or Regular Current Detection functions are not used. The CC-ADC is automatically disabled in Power-off mode.

After the CC-ADC is enabled by setting the CADEN bit, the first three Instantaneous conversions do not contain useful data and should be ignored. This also applies after clearing the CADSE bit, or after changing the CADPOL or CADVSE bits.

The conversion times and sampling intervals are controlled by the Slow RC Oscillator, and will depend on its actual frequency. To obtain accurate coulomb counting results, the actual conversion time must be calculated. Refer to ”Slow RC Oscillator” on page 26 for details.

19.7 Register Description

19.7.1 CADCSRA - CC-ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0xE4)	CADEN	CADPOL	CADUB	CADAS1	CADAS0	CADSI1	CADSI0	CADSE	CADCSRA
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - CADEN: CC-ADC Enable**

When the CADEN bit is cleared (zero), the CC-ADC is disabled, and any ongoing conversions will be terminated. When the CADEN bit is set (one), the CC-ADC will continuously measure the voltage drop over the external sense resistor R_{SENSE}. In Power-off, the CC-ADC is always disabled. Note that the bandgap voltage reference must be enabled, see ”Voltage Reference and Temperature Sensor” on page 117.

- **Bit 6 - CADPOL: CC-ADC Polarity**

The CADPOL bit is used to change input sampling polarity in the Sigma Delta Modulator. Writing this bit to one, the polarity will be negative. When the bit is zero, the polarity will be positive.

- **Bit 5 - CADUB: CC-ADC Update Busy**

The CC-ADC operates in a different clock domain than the CPU. Whenever a new value is written to CADCSRA or CADRC this value must be synchronized to the CCADC clock domain. Subsequent writes to these registers will be blocked during this synchronization. Synchronization of one of the registers will block updating of all the others. The CADUB bit will be read as one while any of these registers is being synchronized, and will be read as zero when neither register is being synchronized.

- **Bits 4:3: CADAS[1:0]: CC-ADC Accumulate Current Select**

The CADAS bits select the conversion time for the Accumulate Current output as shown in Table 19-1.

Table 19-1. CC-ADC Accumulate Current Conversion Time

CADAS[1:0]	CC-ADC Accumulate Current Conversion Time(1)	Number of CC-ADC Clock Cycles
00	125 ms	4096
01	250 ms	8192
10	500 ms	16384
11	1s	32768

Note: 1. The actual value depends on the actual frequency of the Slow RC oscillator, see "Slow RC Oscillator" on page 26.

- **Bits 2:1: CADSI[1:0]: CC-ADC Current Sampling Interval**

The CADSI bits determine the current sampling interval for the Regular Current detection as shown in Table 19-2.

Table 19-2. CC-ADC Regular Current Sampling Interval

CADSI[1:0]	CC-ADC Regular Current Sampling Interval(1)(2)	Number of CC-ADC Clock Cycles
00	250 ms (+ sampling time)	8192 (+ sampling time)
01	500 ms (+ sampling time)	16384 (+ sampling time)
10	1s (+ sampling time)	32768 (+ sampling time)
11	2s (+ sampling time)	65536 (+ sampling time)

Notes: 1. The actual value depends on the actual frequency of the Slow RC oscillator, see "Slow RC Oscillator" on page 26.
2. Sampling time ~ 12 ms.

- **Bit 0 - CADSE: CC-ADC Sampling Enable**

When the CADSE bit is written to one, the ongoing CC-ADC conversion is aborted, and the CCADC enters Regular Current detection mode.

19.7.2 CADCSRB - CC-ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0xE5)	–	CADACIE	CADRCIE	CADICIE	–	CADACIF	CADRCIF	CADICIF	CADCSRB
Read/Write	R	R/W	R	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7 - Res: Reserved**

This bit is reserved bit and will always read as zero.

- **Bit 6 - CADACIE: CC-ADC Accumulate Current Interrupt Enable**

When the CADACIE bit is set (one), and the I-bit in the Status Register is set (one), the CC-ADC Accumulate Current Interrupt is enabled.

- **Bit 5 - CADRCIE: CC-ADC Regular Current Interrupt Enable**

When the CADRCIE bit is set (one), and the I-bit in the Status Register is set (one), the CC-ADC Regular Current Interrupt is enabled.

- **Bit 4 - CADICIE: CC-ADC Instantaneous Current Interrupt Enable**

When the CADICIE bit is set (one), and the I-bit in the Status Register is set (one), the CC-ADC Instantaneous Current Interrupt is enabled.

- **Bits 3 - Res: Reserved**

This bit is reserved bit and will always read as zero.

- **Bit 2 - CADACIF: CC-ADC Accumulate Current Interrupt Flag**

The CADACIF bit is set (one) after the Accumulate Current conversion has completed. The CCADC Accumulate Current Interrupt is executed if the CADACIE bit and the I-bit in SREG are set (one). CADACIF is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, CADACIF is cleared by writing a logic one to the flag.

- **Bit 1 - CADRCIF: CC-ADC Regular Current Interrupt Flag**

The CADRCIF bit is set (one) when the absolute value of the result of the last CC-ADC conversion is greater than, or equal to, the compare values set by the CC-ADC Regular Current Level Register.

- **Bit 0 - CADICIF: CC-ADC Instantaneous Current Interrupt Flag**

The CADICIF bit is set (one) when a CC-ADC Instantaneous Current conversion is completed. The CC-ADC Instantaneous Current Interrupt is executed if the CADICIE bit and the I-bit in SREG are set (one). CADICIF is cleared by hardware when executing the corresponding Interrupt Handling vector. Alternatively, CADICIF is cleared by writing a logic one to the flag.

19.7.3 CADICH and CADICL - CC-ADC Instantaneous Current

Bit	15	14	13	12	11	10	9	8	
	7	6	5	4	3	2	1	0	
(0xE9)	CADIC[15:8]								CADICH
(0xE8)	CADIC[7:0]								CADICL
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When a CC-ADC Instantaneous Current conversion is complete, the result is found in these two registers. CADIC[15:0] represents the converted result in 2's complement format. CADIC[12:0] are the 13-bit ADC result (including sign), while CADIC[15:13] are the sign extension bits.

When CADICL is read, the CC-ADC Instantaneous Current register is not updated until CADCH is read. Reading the registers in the sequence CADICL, CADICH will ensure that consistent values are read. When a conversion is completed, both registers must be read before the next conversion is completed, otherwise data will be lost.

19.7.4 CADAC3, CADADC2, CADAC1, CADAC0 - CC-ADC Accumulate Current

Bit	31	30	29	28	27	26	25	24	
	23	22	21	20	19	18	17	16	
	15	14	13	12	11	10	9	8	
	7	6	5	4	3	2	1	0	
(0xE3)	CADAC[31:24]								CADAC3
(0xE2)	CADAC[23:16]								CADAC2
(0xE1)	CADAC[15:8]								CADAC1
(0xE0)	CADAC[7:0]								CADAC0
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The CADAC3, CADAC2, CADAC1 and CADAC0 Registers contain the Accumulate Current measurements in 2's complement format. CADAC[17:0] are the 18-bit ADC result (including sign), while CADAC[31:18] are the sign extension bits.

When CADAC0 is read, the CC-ADC Accumulate Current register is not updated until CADAC3 is read. Reading the registers in the sequence CADAC0, CADAC1, CADAC2, CADAC3 will ensure that consistent values are read. When a conversion is completed, all four registers must be read before the next conversion is completed, otherwise data will be lost.

19.7.5 CADRC- CC-ADC Regular Current

Bit	7	6	5	4	3	2	1	0	
(0xE6)	CADRC[7:0]								CADRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The CC-ADC Regular Current Register determines the threshold level for the Regular Current detection. When the result of a CC-ADC Instantaneous Current conversion has an absolute value greater than, or equal to, the Regular Current level, the CC-ADC Regular Current Interrupt Flag is set.

The value in this register defines the eight least significant bits of the Regular Current level. The most significant bits of the Regular Current level are always zero. The programmable range for the Regular Current level is given in Table 19-3.

Table 19-3. Programmable Range for the Regular Current Level

		Minimum	Maximum	Step size
Voltage (µV)		0	6848	26.9
Current (mA)	R_{SENSE} = 1 mΩ	0	6848	26.9
	R_{SENSE} = 5 mΩ	0	1370	5.4
	R_{SENSE} = 10 mΩ	0	685	2.7

The CC-ADC Regular Current Register does not affect the setting of the CC-ADC Conversion Complete Interrupt Flag.

20. Voltage ADC – 5-channel General Purpose 12-bit Sigma-Delta ADC

20.1 Features

- **12-bit Resolution**
- **519µs Conversion Time @ 1 MHz clk_{VADC}**
- **Two Differential Input Channels for Cell Voltage Measurements**
- **Three Single Ended Input Channels**
- **0.2x Pre-scaling of Cell Voltages**
- **Interrupt on V-ADC Conversion Complete**

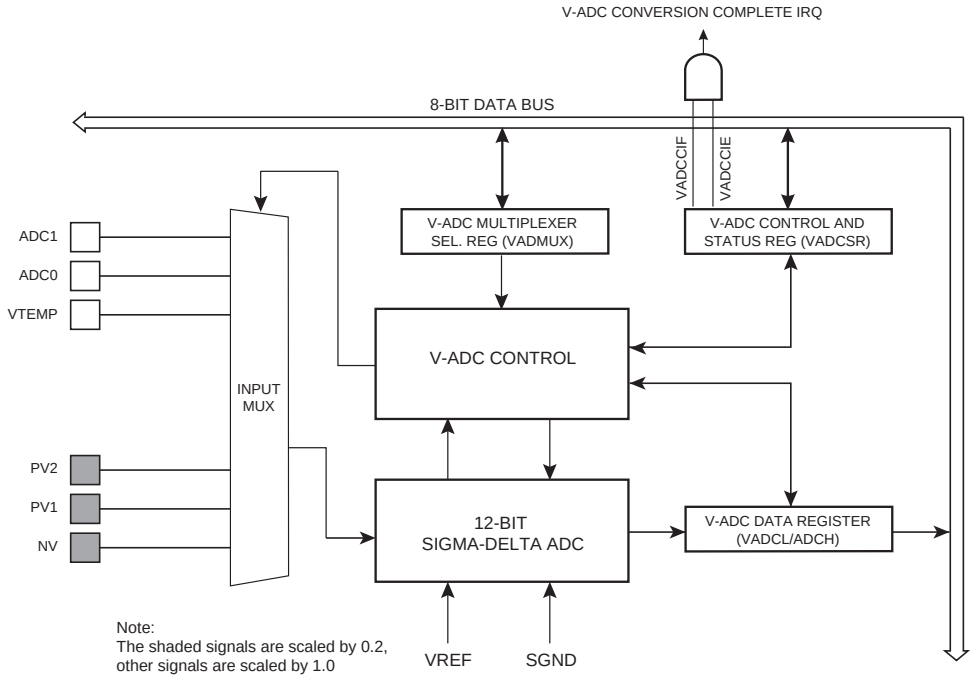
20.2 Overview

The ATmega8HVA/16HVA features a 12-bit Sigma-Delta ADC.

The Voltage ADC (V-ADC) is connected to five different sources through the Input Multiplexer. There are two differential channels for Cell Voltage measurements. These channels are scaled 0.2x to comply with the Full Scale range of the V-ADC. In addition there are three single ended channels referenced to SGND. One channel is for measuring the internal temperature sensor VPTAT and two channels for measuring the voltage at ADC0 and ADC1.

When the V-ADC is not used, power consumption can be minimized by writing the PRVADC bit in PRR0 to one. See "PRR0 – Power Reduction Register 0" on page 39 for details on how to use the PRVADC bit.

Figure 20-1. Voltage ADC Block Schematic

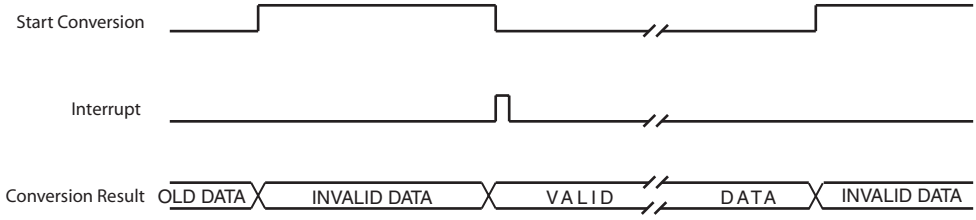


20.3 Operation

To enable V-ADC conversions, the V-ADC Enable bit, VADEN, in V-ADC Control and Status Register – VADCSR must be set. If this bit is cleared, the V-ADC will be switched off, and any ongoing conversions will be terminated. The V-ADC is automatically disabled in Power-save and

Power-off mode. Note that the bandgap voltage reference must be enabled and disabled separately, see ”BGCCR – Bandgap Calibration C Register” on page 118.

Figure 20-2. Voltage ADC Conversion Diagram



To perform a V-ADC conversion, the analog input channel must first be selected by writing to the VADMUX register. When a logical one is written to the V-ADC Start Conversion bit VADSC, a conversion of the selected channel will start. The VADSC bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. When a conversion is in progress, the V-ADC Data Register - VADCL and VADCH will be invalid. If the System Clock Prescaler setting is changed during a V-ADC conversion, the conversion will be aborted. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change. When a conversion is finished the V-ADC Conversion Complete Interrupt Flag – VADCCIF is set. One 12-bit conversion takes 519 µs to complete from the start bit is set to the interrupt flag is set. The V-ADC Data Register - VADCL and VADCH will be valid until a new conversion is started. To ensure that correct data is read, both high and low byte data registers should be read before starting a new conversion.

20.3.1 Configuring PA1 and PA0 for V-ADC operation

When one of the single ended channels ADC0 or ADC1 is used as analog input to the VADC, either PA0 or PA1 are used as signal ground (SGND). When ADC0/1 is selected as input channel, PA1/0 is automatically switched to SGND.

The use of PA1 and PA0 as SGND is efficient for the thermistor configuration shown in ”Operating Circuit” on page 162. Both termistors, RT1 and RT2, are connected through a common divider resistor, R1, to PA0 and PA1 respectively.

Both PA0 and PA1 have very high input impedance when used as ADC inputs, which makes it possible to connect two thermistors in the configuration, shown in ”Operating Circuit” on page 162. However, input impedance is limited and if high accuracy is required, only one thermistor should be connected between PA0 and PA1. If two termistors are connected, the configuration is as follows:

- When measuring RT1, PA1 should be used as input channel and PA0 is automatically switched to SGND.
- When measuring RT2, PA0 should be used as input channel and PA1 is automatically switched to SGND.

20.4 Register Description

20.4.1 VADMUX – V-ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	–	–	–	–	VADMUX3	VADMUX2	VADMUX1	VADMUX0	VADMUX
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4 – RES: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bit 3:0 – VADMUX3:0: V-ADC Channel Selection Bits**

The VADMUX bits determine the V-ADC channel selection. See Table 20-1 on page 114.

Table 20-1. VADMUX channel selection

VADMUX3:0	Channel Selected	Scale
0000	RESERVED	–
0001	CELL 1	0.2
0010	CELL 2	0.2
0011	RESERVED	–
0100	VTEMP(1)	1.0
0101	ADC0	1.0
0110	ADC1	1.0
0111...1111	RESERVED	–

Note: 1. VTEMP must be measured in Active mode to get the highest accuracy when using calibration value stored in signature row.

20.4.2 VADCSR – V-ADC Control and Status Register

Bit	7	6	5	4	3	2	1	0	
(0x7A)	–	–	–	–	VADEN	VADSC	VADCCIF	VADCCIE	VADCSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4 – RES: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA and will always read as zero.

- **Bit 3 – VADEN: V-ADC Enable**

Writing this bit to one enables V-ADC conversion. By writing it to zero, the V-ADC is turned off. Turning the V-ADC off while a conversion is in progress will terminate this conversion. Note that the bandgap voltage reference must be enabled separately, see “BGCCR – Bandgap Calibration C Register” on page 118.

- **Bit 2 – VADSC: Voltage ADC Start Conversion**

Write this bit to one to start a new conversion of the selected channel.

VADSC will read as one as long as the conversion is not finished. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect. VADSC will automatically be cleared when the VADEN bit is written to zero.

- **Bit 1 – VADCCIF: V-ADC Conversion Complete Interrupt Flag**

This bit is set when a V-ADC conversion completes and the data registers are updated. The V-ADC Conversion Complete Interrupt is executed if the VADCCIE bit and the I-bit in SREG are set. VADCCIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, VADCCIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on VADCSR, a pending interrupt can be lost.

- **Bit 0 – VADCCIE: V-ADC Conversion Complete Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the V-ADC Conversion Complete Interrupt is activated.

20.4.3 VADCL and VADCH – V-ADC Data Register

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	VADC[11:8]				VADCH
(0x78)	VADC[7:0]								VADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When a V-ADC conversion is complete, the result is found in these two registers. To ensure that correct data is read, both high and low byte data registers should be read before starting a new conversion.

- **VADC11:0: V-ADC Conversion Result**

These bits represent the result from the conversion.

To obtain the best absolute accuracy for the cell voltage measurements, gain and offset compensation is required. Factory calibration values are stored in the device signature row, refer to section "Reading the Signature Row from Software" on page 144 for details. The cell voltage in mV is given by:

$$Cell_n[\text{mV}] = \frac{(\text{VADCH/L} - \text{VADC Cell}_n\text{Offset}) \cdot \text{VADC Cell}_n\text{ Gain Calibration Word}}{16384}$$

The voltage on the ADCn is given by:

$$\text{ADCn[mV]} = \frac{1}{10} \cdot \frac{(\text{VADCH/L} - \text{VADC ADCn Offset}) \cdot \text{VADC ADCn Gain Calibration Word}}{16384}$$

When performing a VTEMP conversion, the result must be adjusted by the factory calibration value stored in the signature row, refer to section "Reading the Signature Row from Software" on page 144 for details. The absolute temperature in Kelvin is given by:

$$T(K) = \frac{VADCH/L \cdot VPTAT\ CAL}{16384}$$

20.4.4 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x7E)	–	–	–	–	–	–	PA1DID	PA0DID	DIDR0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when DIDR0 is written.

- **Bit 1:0 – PA1DID:PA0DID: Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding Port A pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the PA1:0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

21. Voltage Reference and Temperature Sensor

21.1 Features

- **Accurate Voltage Reference of 1.100V**
- **Internal Temperature Sensor**
- **Possibility for Runtime Compensation of Temperature Drift in Both Voltage Reference and On-chip Oscillators**
- **External Decoupling for Optimum Noise Performance**
- **Low Power Consumption**

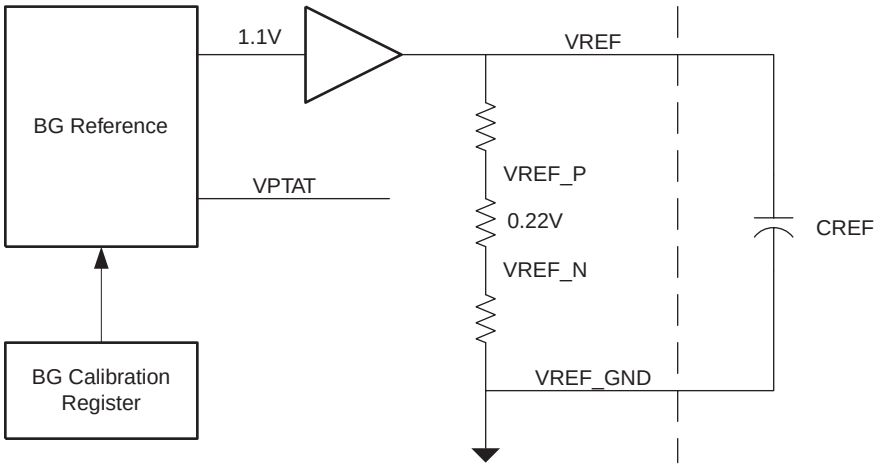
21.2 Overview

A low power band-gap reference provides ATmega8HVA/16HVA with an accurate On-chip voltage reference V_{REF} of 1.100V. This reference voltage is used as reference for the On-chip Voltage Regulator, the V-ADC and the CC-ADC. The reference to the ADCs uses a buffer with external decoupling capacitor to enable excellent noise performance with minimum power consumption. The reference voltage V_{REF_P}/V_{REF_N} to the CC-ADC is scaled to match the full scale requirement at the current sense input pins. This configuration also enables concurrent operation of both V-ADC and CC-ADC.

To guarantee ultra low temperature drift after factory calibration, ATmega8HVA/16HVA features a two-step calibration algorithm. The first step is performed at T_{HOT}°C and the second at room temperature. By default, Atmel factory calibration is performed at T_{HOT}°C, and the result is stored in the signature row. The value of T_{HOT} can also be found in the signature row. See ”Reading the Signature Row from Software” on page 144 for details. The customer can easily implement the second calibration step in their test flow. This requires an accurate input voltage and a stable room temperature. Temperature drift after this calibration is guaranteed by design and characterization to be less than 90 ppm/°C from -10°C to 70°C. The BG Calibration C Register can also be altered runtime to implement temperature compensation in software. Very high accuracy for any temperature inside the temperature range can thus be achieved at the cost of extra calibration steps.

ATmega8HVA/16HVA has an On-chip temperature sensor for monitoring the die temperature. A voltage Proportional-To-Absolute-Temperature, V_{PTAT}, is generated in the voltage reference circuit and connected to the multiplexer at the V-ADC input. This temperature sensor can be used for runtime compensation of temperature drift in both the voltage reference and the On-chip Oscillator. To get the absolute temperature in degrees Kelvin, the measured V_{PTAT} voltage must be scaled with the VPTAT factory calibration value stored in the signature row. See Section “26.2.5” on page 144. for details.

Figure 21-1. Reference Circuitry



21.3 Register Description

21.3.1 BGCCR – Bandgap Calibration C Register

Bit	7	6	5	4	3	2	1	0	
(0xD0)	BGD	–	BGCC5	BGCC4	BGCC3	BGCC2	BGCC1	BGCC0	BGCCR
Read/Write	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - BGD: Bandgap Disable**

Setting the BGD bit to one will disable the bandgap voltage reference. This bit must be cleared (zero) before enabling CC-ADC, V-ADC or Battery Protection, and must remain unset (zero) while either of these modules are enabled.

- **Bit 6 – Res: Reserved Bit**

This bit is reserved for future use.

- **Bit 5:0 – BGCC5:0: BG Calibration of PTAT Current**

These bits are used for trimming of the nominal value of the bandgap reference voltage. These bits are binary coded. Minimum VREF: 000000, maximum VREF: 111111. Step size approximately 2 mV.

Updating the BGCC bits will affect both the regulator output voltage and the BOD detection level. The BOD will react quickly to the new detection level, while the voltage regulator will adjust the output voltage more slowly due to the external reservoir capacitor. Therefore, if the value is increased more than a certain step size, the new BOD level may rise above the regulator output voltage, and a false BOD reset will occur. It is recommended that the BGCC bits are updated with a step size of 1. To allow the voltage regulator to reach the new level between each step, a delay of 20 µs should be added between each update of the BGCC values .

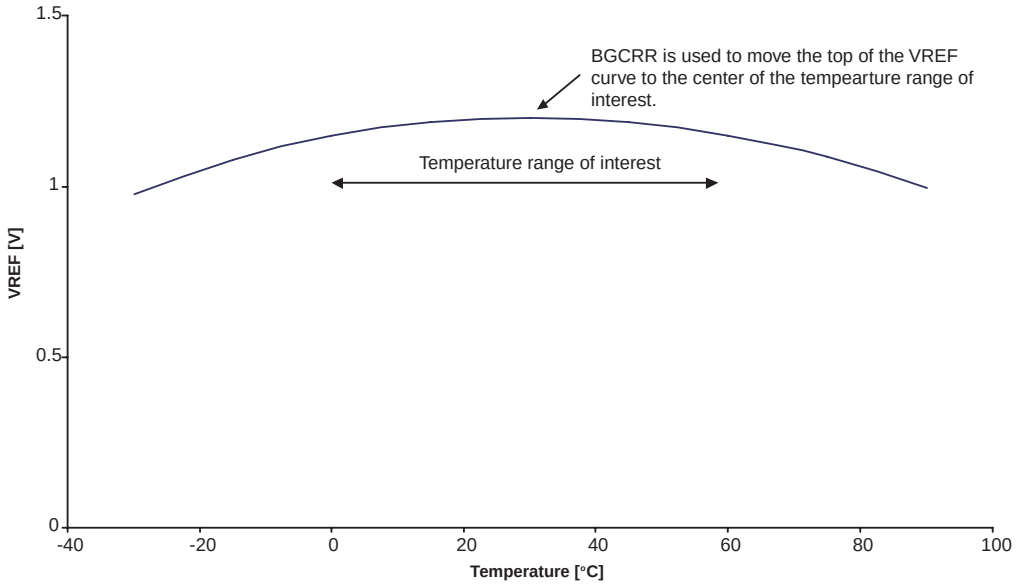
21.3.2 BGCRR – Bandgap Calibration R Register

Bit	7	6	5	4	3	2	1	0	
(0xD1)	BGCR7	BGCR6	BGCR5	BGCR4	BGCR3	BGCR2	BGCR1	BGCR0	BGCRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	1	1	1	1	

- **Bit 7:0 – BGCR7:0: BG Calibration of Resistor ladder**

These bits are used for temperature gradient adjustment of the bandgap reference. Figure 21-2 illustrates VREF as a function of temperature. VREF has a positive temperature coefficient at low temperatures and negative temperature coefficient at high temperatures. Depending on the process variations, the top of the VREF curve may be located at higher or lower temperatures. To minimize the temperature drift in the temperature range of interest, BGCRR is used to adjust the top of the curve towards the centre of the temperature range of interest. The BGCRR bits are thermometer coded, resulting in 9 possible settings: 00000000, 00000001, 00000011, 00000111, … , 11111111. The value 00000000 shifts the top of the VREF curve to the highest possible temperature, and the value 11111111 shifts the top of the VREF curve to the lowest possible temperature.

Figure 21-2. Illustration of VREF as a function of temperature.



22. Voltage Regulator

22.1 Features

- **3.3V fixed output voltage**
- **Automatic selection of Step-up or Linear Regulation depending on VFET voltage.**
- **Fixed Linear Regulation mode can be selected for 2-cell applications**
- **Battery Pack Short mode allowing large voltage drop at VFET without pulling VREG low.**

22.2 Overview

The Voltage Regulator is a Combined Step-up and Linear Voltage Regulator. This allows the same Voltage Regulator module to be used efficiently for a large range of input voltages.

A built in Charge-pump with external capacitors is combined with a linear regulator to keep a constant output voltage for input voltages in the range 1.8 - 9.0V.

Figure 22-1 on page 121 shows the Voltage Regulator block diagram with external components for combined Step-up and Linear mode. Figure 22-2 on page 121 shows the regulated voltage VREG as a function of the input voltage VFET for 1-cell operation. When the VFET is sufficiently high, the regulator switches automatically to linear operation. When VFET drops below a certain level the regulator automatically switch back to step-up regulation. The different reset sources during initialisation and shut down is also shown.

Figure 22-3 on page 122 shows the Voltage Regulator block diagram with external components for Linear mode only, intended for 2-cell applications. In Linear mode only, the input voltage range is 3.6 - 9.0V. In this case, no external fly capacitors are needed, and CF1N should be grounded. Figure 22-4 on page 122 illustrates this operation.

In case of battery pack shortening, the voltage at the input of the regulator will drop quickly. If it drops below minimum operating voltage, the voltage regulator can no longer supply internal or external circuitry. However, the output voltage will not be pulled down by this incident, and the external CREG capacitor can supply the circuitry for a time given by the size of the capacitor and the total current consumption during the same period. VREG must stay above the Brown-Out Threshold to avoid BOD reset. If a battery pack short occurs when VREG is equal to 3.3V and the BOD level is 2.9V, the chip can continue operation for a time given by:

$$t = \frac{c\Delta v}{I_{AVG}} = \frac{CREG \cdot 0.4V}{I_{AVG}}$$

where I_{AVG} represents the average current drawn from CREG. For CREG = 2.2 µF and I_{AVG} = 100 µA, this time equals 8.8 ms. The Voltage Regulator Monitor will detect if a short-circuit has occured, allowing SW to minimize I_{AVG}.

When charging deeply over-discharged cells, the FET Driver will be operated in Deep Under-Voltage Recovery (DUVR) mode. See “FET Driver” on page 136. In this mode a suitable voltage drop is developed across the Charge FET to ensure proper operating voltage at the VFET pin. This will ensure normal operation of the chip during 0-volt charging without setting the charger in quick-charge mode before the cell has reached a safe cell voltage.

Figure 22-1. Voltage regulator block diagram, combined Step-up and Linear mode

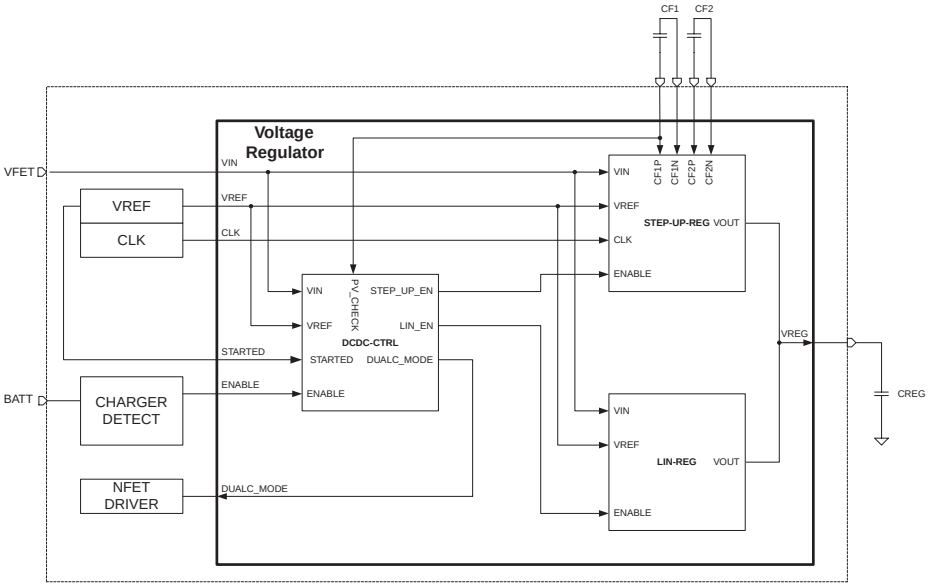


Figure 22-2. Voltage regulator operation and reset signals as a function of rising and falling input voltage for 1-cell operation.

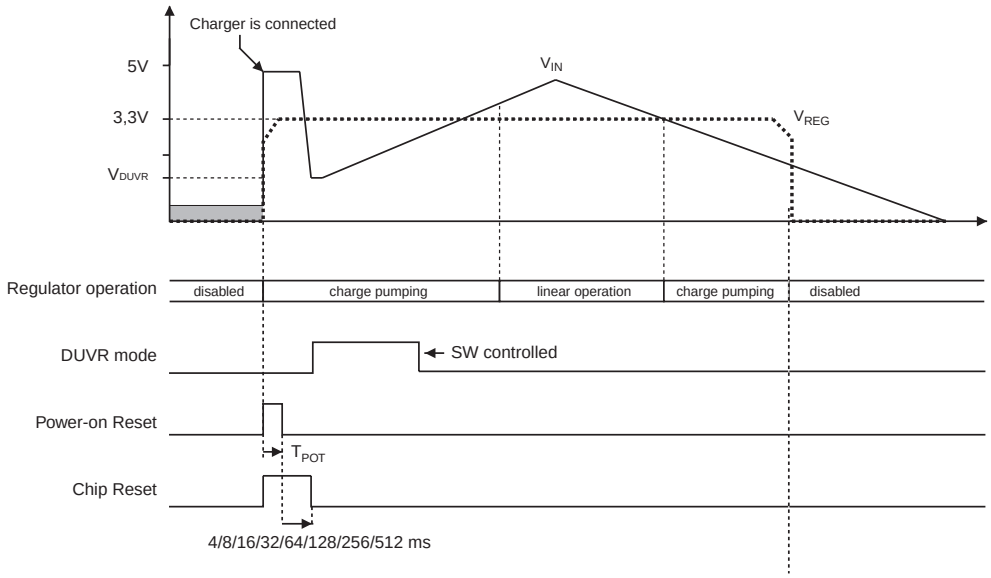


Figure 22-3. Voltage Regulator block diagram, Linear mode only

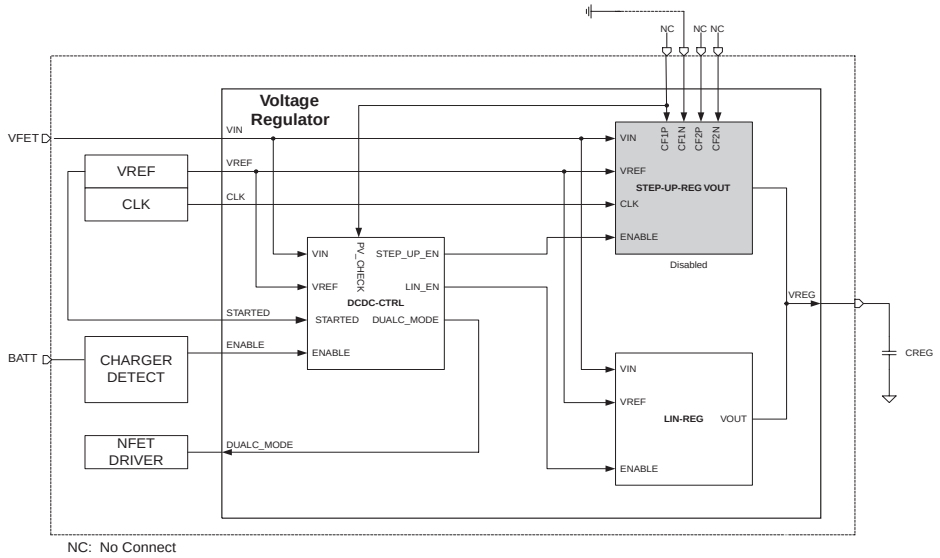
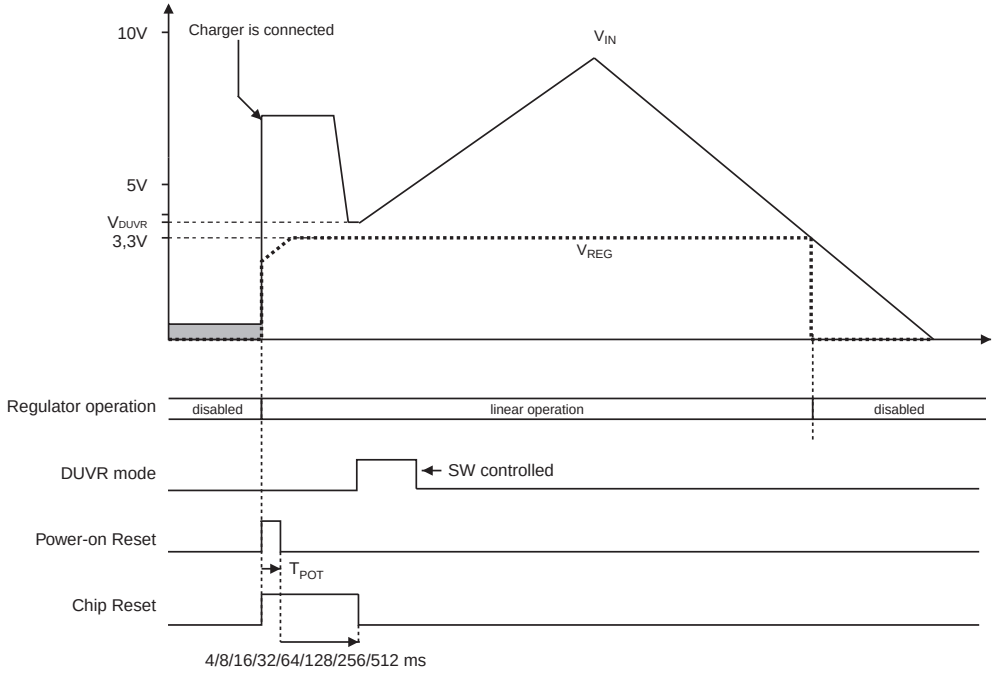


Figure 22-4. Voltage regulator operation and reset signals as a function of rising and falling input voltage for 2-cell operation.



22.3 Voltage Regulator Monitor

This module monitors the operating state of the Voltage Regulator. If the voltage at VFET drops below the Regulator Short-circuit Level (RSCL), see ”Electrical Characteristics” on page 165, the Voltage Regulator enters the Battery Pack Short mode. In this mode, VFET is disconnected from VREG to avoid a quick drop in the voltage regulator output. When the voltage regulator enters this mode, the chip will be completely powered by the external reservoir capacitor (CREG). This allows the chip to operate a certain time without entering BOD reset, even if the VFET voltage is too low for the voltage regulator to operate.

An interrupt is issued when the regulator enters Battery Pack Short mode, if the ROCWIE bit in ROCR Register is set. This allows actions to be taken to reduce power consumption and hence prolonging the time that CREG can be used to power the chip. In a typical short-circuit situation, VFET will drop as a consequence of high current consumption, and recover as soon as the Battery Protection module has disabled the FETs. Hence CREG should be dimensioned so that the chip can sustain operation without entering BOD reset, until the FETs are disabled either by HW or SW. To minimize power consumption when the Voltage Regulator enters the Battery Pack Short mode, the chip should enter Power-save sleep mode as soon as possible after the ROCWIF interrupt is detected. The Watchdog Timer should be configured to wake up the CPU after a time that is considered safe, see appnote AVR132 for use of enhanced Watchdog Timer. Software should then check the status of the ROC flag. If the ROCS flag is cleared, normal operation may be resumed.

22.4 Register Description

22.4.1 ROCR – Regulator Operating Condition Register

Bit	7	6	5	4	3	2	1	0	
(0xC8)	**ROCS**	-	-	-	-	-	**ROCWIF**	**ROCWIE**	**ROCR**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ROCS: ROC Status**

This bit is set when the Voltage Regulator operates in the Battery Pack Short mode, and cleared otherwise.

- **Bit 6:2 – Res: Reserved Bits**

These bits are reserved bits and will always read as zero.

- **Bit 1 – ROCWIF: ROC Warning Interrupt Flag**

The ROCWIF Flag is set within the ROCW reaction time when the Voltage Regulator enters the Battery Pack Short mode. The flag is cleared by writing a logic one to it or by hardware, by executing the corresponding interrupt handling vector.

- **Bit 0 – ROCWIE: ROC Warning Interrupt Enable**

The ROCWIE bit enables interrupt caused by the Regulator Operating Condition Warning interrupt flag.

23. Battery Protection

23.1 Features

- **Short-circuit Protection**
- **Discharge Over-current Protection**
- **Charge Over-current Protection**
- **Discharge High-current Protection**
- **Charge High-current Protection**
- **Programmable and Lockable Detection Levels and Reaction Times**
- **Autonomous Operation Independent of CPU**

23.2 Overview

The Current Battery Protection circuitry (CBP) monitors the charge and discharge current and disables C-FET and D-FET if a Short-circuit, Over-current or High-current condition is detected. There are five different programmable detection levels: Short-circuit Detection Level, Discharge Over-current Detection Level, Charge Over-current Detection Level, Discharge High-current Detection Level, Charge High-current Detection Level. There are three different programmable delays for activating Current Battery Protection: Short-circuit Reaction Time, Over-current Reaction Time and High-current Reaction Time. After Current Battery Protection has been activated, the application software must re-enable the FETs. The Battery Protection hardware provides a hold-off time of 1 second before software can re-enable the discharge FET. This provides safety in case the application software should unintentionally re-enable the discharge FET too early.

The activation of a protection also issues an interrupt to the CPU. The battery protection interrupts can be individually enabled and disabled by the CPU.

The effect of the various battery protection types is given in Table 23-1.

Table 23-1. Effect of Battery Protection Types

Battery Protection Type	Interrupt Requests	C-FET	D-FET	MCU
Short-circuit Protection	Entry	Disabled	Disabled	Operational
Discharge Over-current Protection	Entry	Disabled	Disabled	Operational
Charge Over-current Protection	Entry	Disabled	Disabled	Operational
Discharge High-current Protection	Entry	Disabled	Disabled	Operational
Charge High-current Protection	Entry	Disabled	Disabled	Operational

In order to reduce power consumption, Short-circuit, Discharge High-current and Discharge Over-current Protection are automatically deactivated when the D-FET is disabled. The Charge Over-current and Charge High-current Protection are disabled when the C-FET is disabled. Note however that Charge Over-current Protection and Charge High-current Protection are never automatically disabled when the chip is operated in DUVR mode.

The Current Battery Protection (CBP) monitors the cell current by sampling the shunt resistor voltage at the PI/NI input pins. A differential operational amplifier amplifies the voltage with a suitable gain. The output from the operational amplifier is compared to an accurate, programmable On-chip voltage reference by an Analog Comparator. If the shunt resistor voltage is above the Detection level for a time longer than the corresponding Protection Reaction Time, the chip activates Current Protection. A sampled system clocked by the internal ULP Oscillator is used for Short-circuit, Over-current, and High-current Protection. This ensures a reliable clock source, offset cancellation and low power consumption.

23.3 Short-circuit Protection

The Short-circuit detection is provided to enable a fast response time to very large discharge currents. If the voltage at the PI/NI pins is above the Short-circuit Detection Level for a period longer than Short-circuit Reaction Time, the Short-circuit Protection is activated.

When the Short-circuit Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the D-FET and C-FET are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the D-FET is re-enabled before the cause of the short-circuit condition is removed, the Short-circuit Protection will be activated again.

23.4 Discharge Over-current Protection

If the voltage at the PI/NI pins is above the Discharge Over-current Detection level for a time longer than Over-current Protection Reaction Time, the chip activates Discharge Over-current Protection.

When the Discharge Over-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the D-FET is re-enabled while the loading of the battery still is too large, the Discharge Over-current Protection will be activated again.

23.5 Charge Over-current Protection

If the voltage at the PI/NI pins is above the Charge Over-current Detection level for a time longer than Over-current Protection Reaction Time, the chip activates Charge Over-current Protection.

When the Charge Over-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the C-FET is re-enabled and the charger continues to supply too high currents, the Charge Over-current Protection will be activated again.

23.6 Discharge High-current Protection

If the voltage at the PI/NI pins is above the Discharge High-current Detection level for a time longer than High-current Protection Reaction Time, the chip activates Discharge High-current Protection.

When the Discharge High-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the D-FET is re-enabled while the loading of the battery still is too large, the Discharge High-current Protection will be activated again.

23.7 Charge High-current Protection

If the voltage at the PI/NI pins is above the Charge High-current Detection level for a time longer than High-current Protection Reaction Time, the chip activates Charge High-current Protection.

When the Charge High-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the C-FET is re-enabled and the charger continues to supply too high currents, the Charge High-current Protection will be activated again.

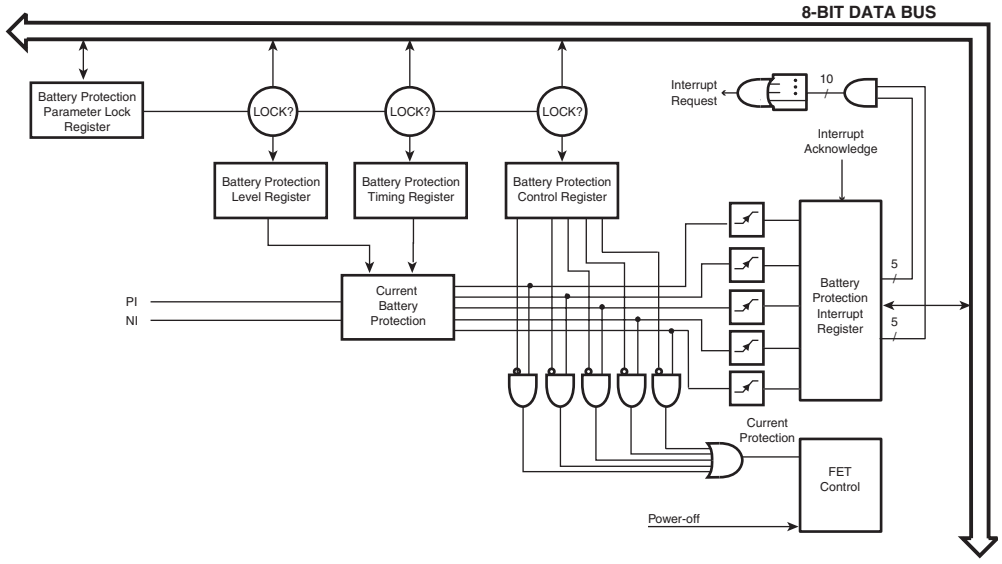
The Short-circuit, Over-current and High-current Protection parameters are programmable to adapt to different types of batteries. The parameters are set by writing to I/O Registers. The Parameter Registers can be locked after the initial configuration, prohibiting any further updates until the next Hardware Reset.

Refer to ”Register Description for Battery Protection” on page 125 for register descriptions.

23.8 Battery Protection CPU Interface

The Battery Protection CPU Interface is illustrated in Figure 22-1.

Figure 23-1. Battery Protection CPU Interface



Each protection has an Interrupt Flag. Each Flag can be read and cleared by the CPU, and each flag has an individual interrupt enable. All enabled flags are combined into a single battery pro-

tection interrupt request to the CPU. This interrupt can wake up the CPU from any operation mode, except Power-off. The interrupt flags are cleared by writing a logic ‘1’ to their bit locations from the CPU.

Note that there are neither flags nor status bits indicating that the chip has entered the Power Off mode. This is because the CPU is powered down in this mode. The CPU will, however be able to detect that it came from a Power-off situation by monitoring CPU reset flags when it resumes operation.

23.9 Register Description

23.9.1 BPPLR – Battery Protection Parameter Lock Register

Bit	7	6	5	4	3	2	1	0	
(0xFE)	–	–	–	–	–	–	BPPLE	BPPL	BPPLR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 1 – BPPLE: Battery Protection Parameter Lock Enable**

- **Bit 0 – BPPL: Battery Protection Parameter Lock**

The BPCR, BPHCTR, BPOCTR, BPSCTR, BPDHCD, BPCHCD, BPDOCD, BPCOCD and BPSCD Battery Protection registers can be locked from any further software updates. Once locked, these registers cannot be accessed until the next hardware reset. This provides a safe method for protecting the registers from unintentional modification by software runaway. It is recommended that software sets these registers shortly after reset, and then protect the registers from further updates.

To lock these registers, the following algorithm must be followed:

1. In the same operation, write a logic one to BPPLE and BPPL.
2. Within the next four clock cycles, in the same operation, write a logic zero to BPPLE and a logic one to BPPL.

23.9.2 BPCR – Battery Protection Control Register

Bit	7	6	5	4	3	2	1	0	
(0xFD)	–	–	–	SCD	DOCD	COCD	DHCD	CHCD	BPCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 5 – Res: Reserved Bits**

This bit are reserved and will always read as one.

- **Bit 4 – SCD: Short Circuit Protection Disabled**

When the SCD bit is set, the Short-circuit Protection is disabled. The Short-circuit Detection will be disabled, and any Short-circuit condition will be ignored.

- **Bit 3 – DOCD: Discharge Over-current Protection Disabled**

When the DOCD bit is set, the Discharge Over-current Protection is disabled. The Discharge Over-current Detection will be disabled, and any Discharge Over-current condition will be ignored.

- **Bit 2 – COCD: Charge Over-current Protection Disable**

When the COCD bit is set, the Charge Over-current Protection is disabled. The Charge Over-current Detection will be disabled, and any Charge Over-current condition will be ignored.

- **Bit 1 – DHCD: Discharge High-current Protection Disabled**

When the DHCD bit is set, the Discharge High-current Protection is disabled. The Discharge High-current Detection will be disabled, and any Discharge High-current condition will be ignored.

- **Bit 0 – CHCD: Charge High-current Protection Disable**

When the CHCD bit is set, the Charge High-current Protection is disabled. The Charge High-current Detection will be disabled, and any Charge High-current condition will be ignored.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPCR register is written. Any writing to the BPCR register during this period will be ignored.

23.9.3 BPSCTR – Battery Protection Short-circuit Timing Register

Bit	7	6	5	4	3	2	1	0	
(0xFA)	–	SCPT[6:0]							BPSCTR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	1	0	0	0	0	

- **Bit 7 – Res: Reserved Bits**

This bit is reserved and will always read as zero.

- **Bit 6:0 – SCPT6:0: Short-circuit Protection Timing**

These bits control the delay of the Short-circuit Protection. The Short-circuit Timing can be set with a step size of 62.5 µs as shown in Table 23-2 on page 128.

Table 23-2. Short-circuit Protection Reaction Time. SCPT[6:0] with corresponding Short-circuit Delay Time.

Short-circuit Protection Reaction Time[1]	
SCPT[6:0][2]	**Typ**
0x00	(15.5 - 70.5 µs) + T_d[3]
0x01	(15.5 - 70.5 µs) + T_d[3]
0x02	(78.0 - 133.0 µs) + T_d[3]
0x03	(140.5 - 195.5 µs) + T_d[3]

Table 23-2. Short-circuit Protection Reaction Time. SCPT[6:0] with corresponding Short-circuit Delay Time.

Short-circuit Protection Reaction Time[1]	
...	...
0x7E	(7.83 - 7.88 ms) + T_d[3]
0x7F	(7.89 - 7.95 ms) + T_d[3]

Notes:
1. The actual value depends on the actual frequency of the "Ultra Low Power RC Oscillator" on page 26. See "Electrical Characteristics" on page 165.
2. Initial value: SCPT[0x10](1ms).
3. An additional delay T_d can be expected after enabling the Discharge FET due to initialization of the protection circuit. With nomial ULP frequency this delay is maximum 86 µs.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPSCTR register is written. Any writing to the BPSCTR register during this period will be ignored.

23.9.4 BPOCTR – Battery Protection Over-current Timing Register

Bit	7	6	5	4	3	2	1	0	
(0xFB)	–	–	OCPT[5:0]						BPOCTR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	1	0	

- **Bit 7:6 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 5:0 – OCPT5:0: Over-current Protection Timing**

These bits control the delay of the Over-circuit Protection. The Over-current Timing can be set with a step size of 0.5 ms as shown in Table 23-3 on page 129.

Table 23-3. Over-current Protection Reaction Time. OCPT[5:0] with corresponding Over-current Delay Time.

Over-current Protection Reaction Time[1]	
OCPT[5:0]	**Typ**
0x00	(0.0 - 0.5 ms) + T_d[3]
0x01	(0.0 - 0.5 ms) + T_d[3]
0x02[2]	(0.5 - 1.0 ms) + T_d[3]
0x03	(1.0 - 1.5 ms) + T_d[3]
...	...
0x3E	(30.5 - 31.0 ms) + T_d[3]
0x3F	(31.0 - 31.5 ms) + T_d[3]

Notes:
1. The actual value depends on the actual frequency of the "Ultra Low Power RC Oscillator" on page 26. See "Electrical Characteristics" on page 165.
2. Initial value.
3. An additional delay T_d can be expected after enabling the corresponding FET. This is related to the initialization of the protection circuitry. For the Discharge Over-Current protection, this

applies when enabling the Discharge FET. For Charge Over-Current protection, this applies when enabling the Charge FET. With nominal ULP frequency this delay is maximum 0.1 ms.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPOCTR register is written. Any writing to the BPOCTR register during this period will be ignored.

23.9.5 BPHCTR – Battery Protection High-current Timing Register

Bit	7	6	5	4	3	2	1	0	
(0xFC)	–	–	HCPT[5:0]						BPHCTR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	1	

- **Bit 7:6 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 5:0 – HCPT5:0: High-current Protection Timing**

These bits control the delay of the High-circuit Protection. The High-current Timing can be set with a step size of 2 ms as shown in Table 23-4 on page 130.

Table 23-4. High-current Protection Reaction Time. HCPT[5:0] with corresponding High-current Delay Time.

High-current Protection Reaction Time(1)	
HCPT[5:0]	**Typ**
0x00	(0 - 2 ms) + T_d(3)
0x01(2)	(0 - 2 ms) + T_d(3)
0x02	(2 - 4 ms) + T_d(3)
0x03	(4 - 6 ms) + T_d(3)
...	...
0x3E	(122 - 124 ms) + T_d(3)
0x3F	(124 - 126 ms) + T_d(3)

Notes:
1. The actual value depends on the actual frequency of the "Ultra Low Power RC Oscillator" on page 26. See "Electrical Characteristics" on page 165.
2. Initial value.
3. An additional delay T_d can be expected after enabling the corresponding FET. This is related to the initialization of the protection circuitry. For the Discharge High-Current protection, this applies when enabling the Discharge FET. For Charge High-Current protection, this applies when enabling the Charge FET. With nominal ULP frequency this delay is maximum 0.2 ms.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPHCTR register is written. Any writing to the BPHCTR register during this period will be ignored.

23.9.6 BPSCD – Battery Protection Short-circuit Detection Level Register

Bit	7	6	5	4	3	2	1	0	
(0xF5)	SCDL[7:0]								BPSCD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	1	1	

- **Bits 7:0 – SCDL7:0: Short-circuit Detection Level**

These bits sets the R_{SENSE} voltage level for detection of Short-circuit in the discharge direction, as defined in Table 23-5 on page 132.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPSCD register is written. Any writing to the BPSCD register during this period will be ignored.

23.9.7 BPDOCD – Battery Protection Discharge-Over-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
(0xF6)	DOCDL[7:0]								BPDOCD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	1	1	

- **Bits 7:0 – DOCDL7:0: Discharge Over-current Detection Level**

These bits sets the R_{SENSE} voltage level for detection of Discharge Over-current, as defined in Table 23-5 on page 132.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPDOCD register is written. Any writing to the BPDOCD register during this period will be ignored.

23.9.8 BPCOCD – Battery Protection Charge-Over-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
(0xF7)	COCDL[7:0]								BPCOCD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	1	1	

- **Bits 7:0 –COCDL7:0: Charge Over-current Detection Level**

These bits sets the R_{SENSE} voltage level for detection of Charge Over-current, as defined in Table 23-5 on page 132.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPCOCD register is written. Any writing to the BPCOCD register during this period will be ignored.

23.9.9 BPDHCD – Battery Protection Discharge-High-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
(0xF8)	DHCDL[7:0]								BPDHCD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	1	1	

- **Bits 7:0 – DHCDL7:0: Discharge High-current Detection Level**

These bits sets the R_{SENSE} voltage level for detection of Discharge High-current, as defined in Table 23-5 on page 132.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPDHCD register is written. Any writing to the BPDHCD register during this period will be ignored.

23.9.10 BPCHCD – Battery Protection Charge-High-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
(0xF9)	CHCDL[7:0]								BPCHCD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	1	1	

- **Bits 7:0 –CHCDL7:0: Charge High-current Detection Level**

These bits sets the R_{SENSE} voltage level for detection of Charge High-current, as defined in Table 23-5 on page 132.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BPCHCD register is written. Any writing to the BPCHCD register during this period will be ignored.

Table 23-5. DL[7:0] with corresponding R_{SENSE} Current for all Current Detection Levels (R_{SENSE} = 10 mΩ, VREF = 1.100 ± 0.005V, T_A = -10°C to 70°C)

Current Protection Detection Level			
DL[7:0]	**Min.**	**Typ.**	**Max.**
0xF3		2.0A	
0xF4		2.5A	
0xF5		3.0A	
0xF6		3.5A	
0xF7		4.0A	
0xF8		4.5A	
0xF9		5.0A	
0xFA		5.5A	
0xFB		6.0A	
0xFC		6.5A	
0xFD		7.0A	

Table 23-5. DL[7:0] with corresponding R_{SENSE} Current for all Current Detection Levels (R_{SENSE} = 10 mΩ, VREF = 1.100 ± 0.005V, T_A = -10°C to 70°C) (Continued)

Current Protection Detection Level			
0xFE		7.5A	
0xDD		8.0A	
0xDE		8.5A	
0xDF		9.0A	
0xBD		9.5A	
0xBE		10.0A	
0x9D		11.0A	
0x9E		12.0A	
0x7C		13.0A	
0x7D		14.0A	
0x7E		15.0A	
0x7F		16.0A	
0x5C		17.0A	
0x5D		18.0A	
0x5E		19.0A	
All other values	Reserved		

23.9.11 BPIMSK – Battery Protection Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0xF2)	-	-	-	SCIE	DOCIE	COCIE	DHCIE	CHCIE	BPIMSK
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:5 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 4 – SCIE: Short-circuit Protection Activated Interrupt**

The SCIE bit enables interrupt caused by the Short-circuit Protection Activated Interrupt.

- **Bit 3 – DOCIE: Discharge Over-current Protection Activated Interrupt**

The DOCIE bit enables interrupt caused by the Discharge Over-current Protection Activated Interrupt.

- **Bit 2 – COCIE: Charge Over-current Protection Activated Interrupt**

The COCIE bit enables interrupt caused by the Charge Over-current Protection Activated Interrupt.

- **Bit 1 - DHCIE : Discharger High-current Protection Activated Interrupt**

The DHCIE bit enables interrupt caused by the Discharge High-current Protection Activated Interrupt.

- **Bit 0 - CHCIE : Charger High-current Protection Activated Interrupt**

The CHCIE bit enables interrupt caused by the Charge High-current Protection Activated Interrupt.

23.9.12 BPIFR – Battery Protection Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
(0xF3)	-	-	-	SCIF	DOCIF	COCIF	DHCIF	CHCIF	BPIFR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:5 – Res: Reserved Bit**

These bits are reserved and will always read as zero.

- **Bit 4 – SCIF: Short-circuit Protection Activated Interrupt**

Once Short-circuit violation is detected, SCIF becomes set. The flag is cleared by writing a logic one to it.

- **Bit 3 – DOCIF: Discharge Over-current Protection Activated Interrupt**

Once Disharge Over-current violation is detected, DOCIF becomes set. The flag is cleared by writing a logic one to it.

- **Bit 2 – COCIF: Charge Over-current Protection Activated Interrupt**

Once Charge Over-current violation is detected, COCIF becomes set. The flag is cleared by writing a logic one to it.

- **Bit 1 – DHCIF: Disharge High-current Protection Activated Interrupt**

Once Discharge High-current violation is detected, DHCIF becomes set. The flag is cleared by writing a logic one to it.

- **Bit 0 – CHCIF: Charge High-current Protection Activated Interrupt**

Once Charge High-current violation is detected, CHCIF becomes set. The flag is cleared by writing a logic one to it.

24. FET Control

24.1 Overview

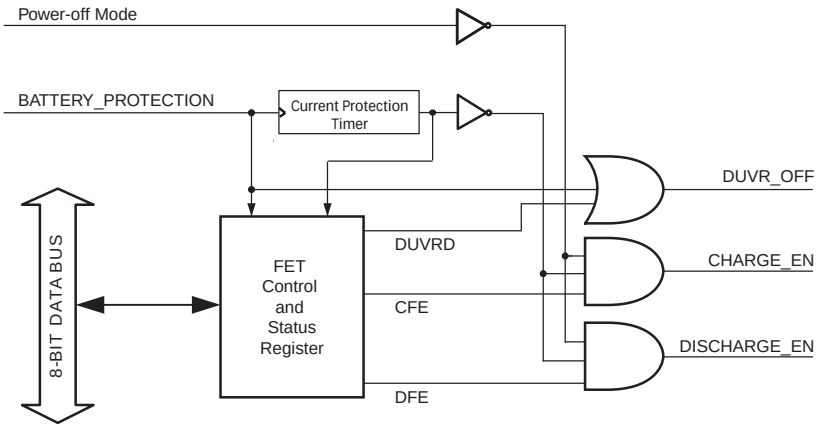
The FET control is used to enable and disable the Charge FET and Discharge FET. Normally, the FETs are enabled and disabled by SW writing to the FET Control and Status Register (FCSR). However, the autonomous Battery Protection circuitry will if necessary override SW settings to protect the battery cells from too high Charge- or Discharge currents. Note that the CPU is never allowed to enable a FET that is disabled by the battery protection circuitry. The FET control is shown in Figure 24-1.

If Current Protection is activated by the Battery Protection circuitry both the Charge-FET and Discharge FET will be disabled by hardware. When the protection disappears the Current Protection Timer will ensure a hold-off time of 1 second before software can re-enable the external FETs.

If C-FET is disabled and D-FET enabled, discharge current will run through the body-drain diode of the C-FET and vice versa. To avoid the potential heat problem from this situation, software must ensure that D-FET is not disabled when a charge current is flowing, and that C-FET is not disabled when a discharge current is flowing.

If charging deeply over-discharged cells, the FET driver must be operated in the Deep Undervoltage Recovery mode. When the cell voltage raises to an acceptable level, Deep Under-voltage Recovery mode should be disabled by software by setting the FCSR (DUVRD bit). To avoid that C-FET is opened while current protection is active, DUVR mode is automatically disabled by hardware, in this case.

Figure 24-1. FET Control Block Diagram



24.1.1 FETs disabled during reset

During reset, both FETs will be disabled immediately and the chip will exit from DUVR mode. It is important to notice that a reset will lead to an immediate disabling of the FETs regardless of the Battery Protection parameter settings. A BOD reset may occur as a result of a short-circuit condition. Depending on the selected Battery Protection Timing, actual current consumption and dimensioning of CREG, a BOD reset may occur before the Battery Protection delay timing has expired, causing the FETs to be disabled.

24.2 FET Driver

24.2.1 Features

- **Charge-pump for generating suitable gate drive for N-Channel FET switch on high side**
- **Deep Under-voltage Recovery mode that allows normal operation while charging a Deeply Over-discharged battery from 0-volt**

24.2.2 Overview

The ATmega8HVA/16HVA includes a FET Driver. The FET Driver is designed for driving N-channel FETs used as high side switch in 1- or 2-Cell Li-Ion battery pack. A block diagram of the FET driver is shown in Figure 24-2 on page 136.

When charging deeply over-discharged cells, the FET Driver will be operated in Deep Under-Voltage Recovery (DUVR) mode. In DUVR mode the FET Driver regulates the voltage at the VFET pin to typically 2.0 Volts in 1-Cell applications and typically 4.0 Volts in 2-Cell applications. This is done by operating the Charge FET at a point where the drain-source voltage is equal to the voltage difference between the cell voltage and the required VFET operating voltage. As the cell voltage increases, the drain-source voltage of the Charge FET will decrease until the Charge FET is completely on. See Table 29-5 on page 170 for details.

In normal operation (DUVRD = 1), the Charge FET/Discharge FET is switched on by pumping OC/OD sufficiently above the VFET supply voltage. To turn off the Charge FET/Discharge FET, OC/OD is pulled quickly low. See Figure 24-3 on page 137 and Table 29-5 on page 170 for details.

Figure 24-2. FET Driver block diagram.

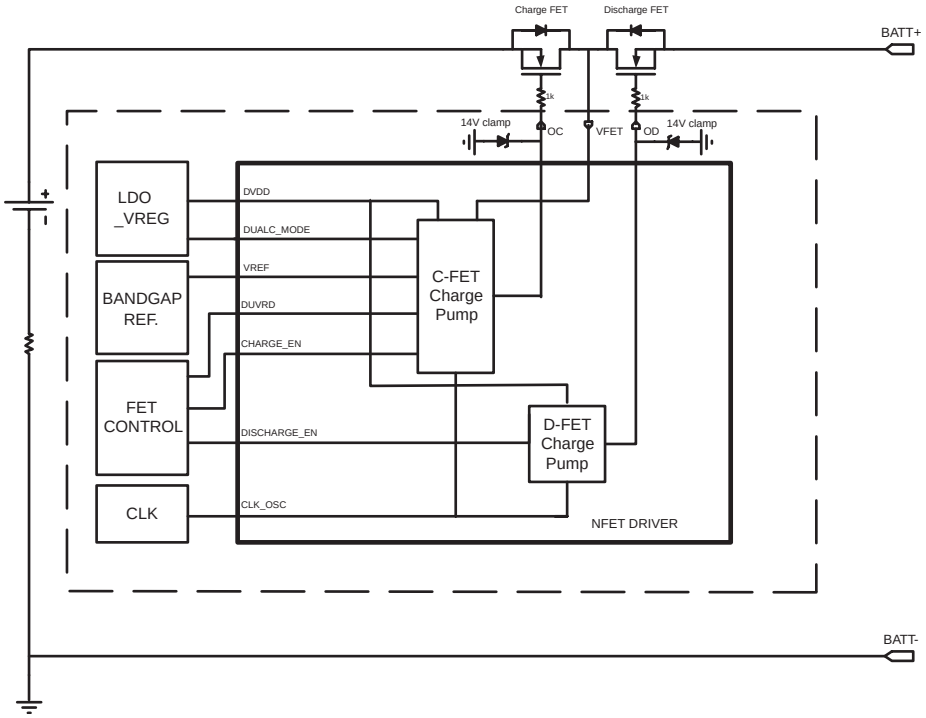
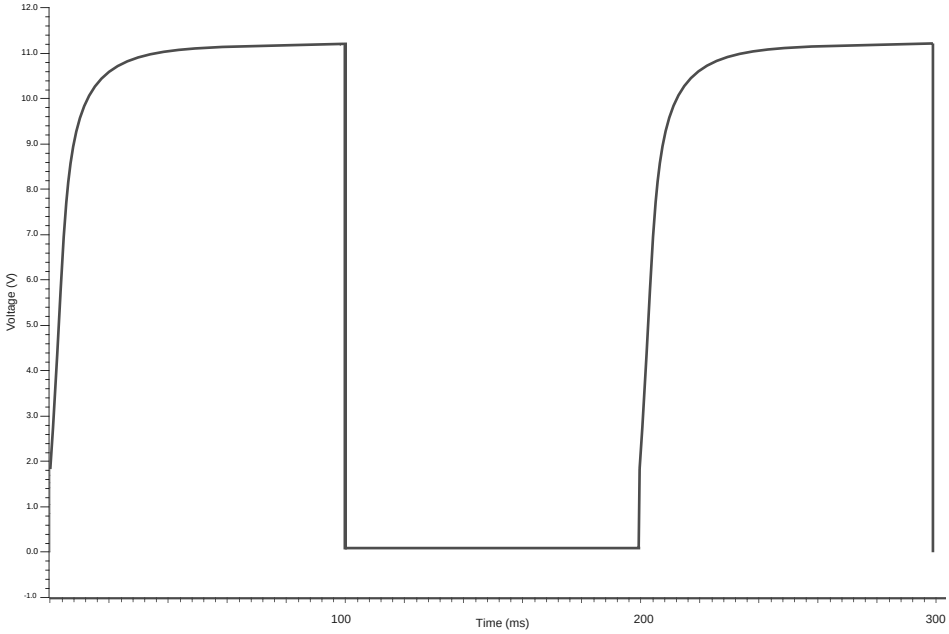


Figure 24-3. Switching NFET on and off during NORMAL operation



24.3 DUVR – Deep Under-Voltage Recovery Mode operation

The purpose of DUVR mode is to control the Charge FET so that the VFET voltage is above the minimum operating voltage while charging cells below minimum operating voltage. This is useful when the cell has been discharged below the minimum operating voltage of the chip. In DUVR mode the Charge FET is switched partly on to provide a suitable voltage drop between the cell voltage and the VFET terminal. As the cell voltage increases, the voltage drop across the Charge FET will gradually decrease until the Charge FET is switched completely on. This means that for high cell voltages, DUVR mode operation is equivalent to normal enabling of the Charge FET (CFE=1).

ATmega8HVA/16HVA should operate in DUVR mode until software detects that the cell has recovered from Deep Under-Voltage condition. When the cell has recovered from Deep Under-Voltage condition, software should first set CFE=1. This is safe now since the cell voltage is above minimum operating voltage. After that software should disable DUVR mode by setting DUVRD = 1.

If both DUVRD and CFE bit is set before the cell voltage is above minimum operating voltage, the VFET voltage will drop and the chip will enter BOD reset and switch off both the Charge- and Discharge FET. Switching off the FET's will cause the VFET voltage to rise again so that the chip restarts from BOD reset, with DUVRD = 0 and CFE = 0 (default values). To avoid this, software must always check the cell voltage by V-ADC measurements before setting CFE=1.

DUVR mode is default enabled after reset. However, while the chip is in reset state, DUVR mode is disabled. This is a safety feature that ensures that the Charge FET will not be switched on until the Charge Over-current Protection is operating. This implies that the DUVR mode will be disabled from the time that a charger is connected until the selected start-up time expired. During this period, the VFET voltage will be higher than the normal VFET Level in DUVR mode.

For more details about DUVR mode, refer to application note AVR354.

24.4 Register Description

24.4.1 FCSR – FET Control and Status Register

Bit	7	6	5	4	3	2	1	0	
(0xF0)	–	–	–	–	DUVRD	CPS	DFE	CFE	FCSR
Read/Write	R	R	R	R	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA, and will always read as zero.

- **Bit 3 – DUVRD: Deep Under-voltage Recovery Disabled**

When the DUVRD is cleared (zero), the FET Driver will be forced to operate in Deep Under-voltage Recovery DUVR mode. See ”DUVR – Deep Under-Voltage Recovery Mode operation” on page 137 for details. To avoid that the FET driver tries to switch on the C-FET during current protection or during internal reset, the DUVRD bit is overridden to one by hardware in these cases. When this bit is set (one), Deep Under-voltage Recovery mode of the FET Driver will be disabled.

- **Bit 2 – CPS: Current Protection Status**

The CPS bit shows the status of the Current Protection. This bit is set (one) when a Current Protection is active, and cleared (zero) otherwise.

- **Bit 1 – DFE: Discharge FET Enable**

When the DFE bit is cleared (zero), the Discharge FET will be disabled regardless of the state of the Battery Protection circuitry. When this bit is set (one), the Discharge FET is enabled. This bit will automatically be cleared by the CBP circuitry when Current Protection is activated. When this bit is cleared, Short-circuit, Discharge High-current and Discharge Over-current are disabled regardless of the settings in the BPCR Register.

- **Bit 0 – CFE: Charge FET Enable**

When the CFE bit is cleared (zero), the Charge FET will be disabled regardless of the state of the Battery Protection circuitry. When this bit is set (one), the Charge FET is enabled. This bit will automatically be cleared by the CBP circuitry when Current Protection is activated. When this bit is cleared and the DUVRD bit is set, Charge High-current Protection and Charge Over-current Protection are disabled regardless of the settings in the BPCR Register. When the DUVRD bit is cleared, the charge FET will be enabled by DUVR mode regardless of the CFE status.

Note: Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the FCSR register is written. Any writing to the FCSR register during this period will be ignored.

25. debugWIRE On-chip Debug System

25.1 Features

- **Complete Program Flow Control**
- **Emulates All On-chip Functions, Both Digital and Analog, except RESET Pin**
- **Real-time Operation**
- **Symbolic Debugging Support (Both at C and Assembler Source Level, or for Other HLLs)**
- **Unlimited Number of Program Break Points (Using Software Break Points)**
- **Non-intrusive Operation**
- **Electrical Characteristics Identical to Real Device**
- **Automatic Configuration System**
- **High-Speed Operation**
- **Programming of Non-volatile Memories**

25.2 Overview

The debugWIRE On-chip debug system uses a One-wire, bi-directional interface to control the program flow, execute AVR instructions in the CPU and to program the different non-volatile memories.

25.3 Physical Interface

When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 25-1. The debugWIRE Setup

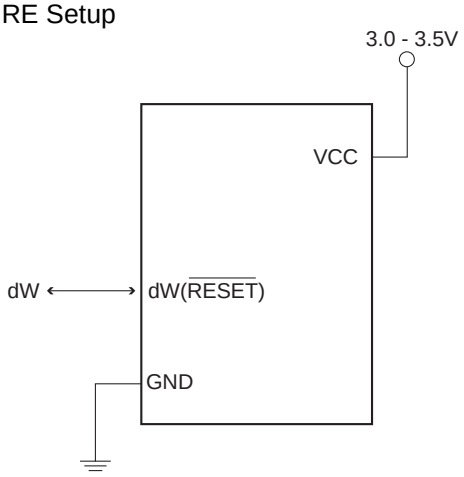


Figure 25-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the OSCSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

25.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio® will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

25.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

25.6 Register Description

The following section describes the registers used with the debugWire.

25.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
0x31 (0x51)	DWDR[7:0]								DWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

26. Self-Programming the Flash

26.1 Overview

The device provides a Self-Programming mechanism for downloading and uploading program code by the MCU itself. The Self-Programming can use any available data interface and associated protocol to read code and write (program) that code into the Program memory.

The Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

Alternative 1, fill the buffer before a Page Erase

- Fill temporary page buffer
- Perform a Page Erase
- Perform a Page Write

Alternative 2, fill the buffer after Page Erase

- Perform a Page Erase
- Fill temporary page buffer
- Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page.

26.1.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “00000011” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- The CPU is halted during the Page Erase operation.

26.1.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a Page Write operation or by writing the CTPB bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

26.1.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- The CPU is halted during the Page Write operation.

26.2 Addressing the Flash During Self-Programming

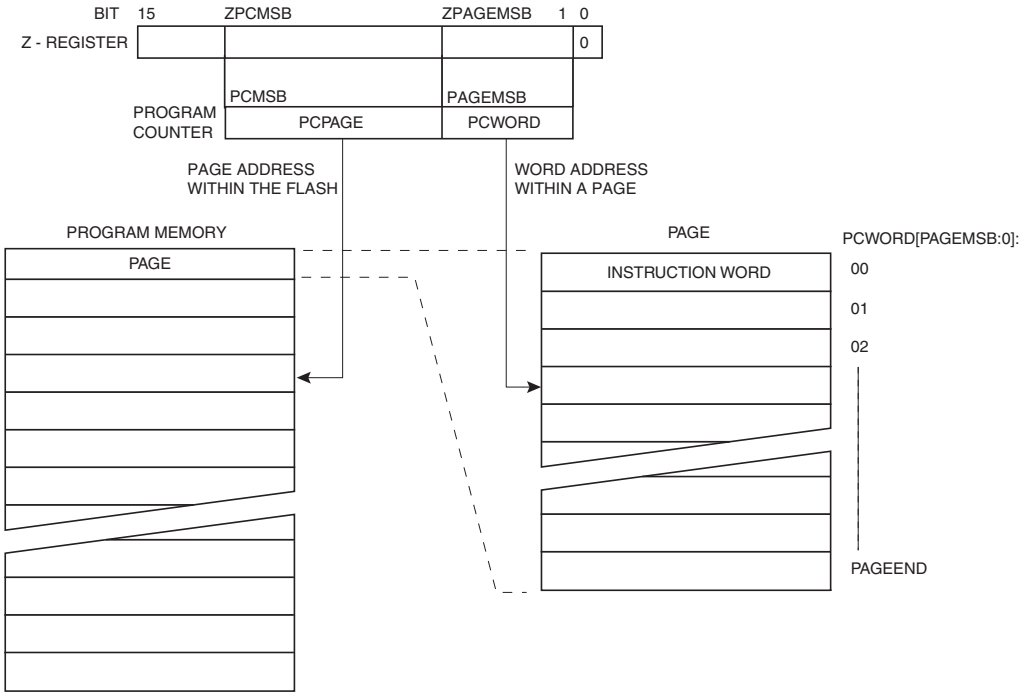
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 27-6 on page 151), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 26-1. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 26-1. Addressing the Flash During SPM[1]



Note: 1. The different variables used in Figure 26-1 are listed in Table 27-6 on page 151.

26.2.1 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

26.2.2 Setting the Lock Bits from Software

To set the Lock Bits, write the desired data to R0. If bits 1..0 in R0 are cleared (zero), the corresponding Lock bit will be programmed if an SPM instruction is executed within four cycles after RFLB and SPMEN are set in SPMCSR. The Z-pointer is don't care during this operation, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the Lock bits). For future compatibility it is also recommended to set bit 7..2 in R0 to "1" when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

Bit	7	6	5	4	3	2	1	0
R0	1	1	1	1	1	1	LB2	LB1

See Table 27-1 on page 149 and Table 27-2 on page 149 for how the different settings of the Lock bits affect the Flash access.

26.2.3 Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear 6 cycles after writing to SPMCSR if no SPM instruction is executed within four CPU cycles. SPMCSR is locked for further writing until it is auto-cleared. The LPM instruction must be executed within three CPU cycles after writing SPMCSR. When RFLB and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

The algorithm for reading the Fuse Low byte is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown below. Refer to Table 27-4 on page 150 for a detailed description and mapping of the Fuse Low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

26.2.4 Preventing Flash Corruption

During periods of low V_{CC}, the Flash program can be corrupted because the supply voltage is too low for the CPU and the Flash to operate properly. These issues are the same as for board level systems using the Flash, and the same design solutions should be applied.

A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
2. Keep the AVR core in Power-save sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

26.2.5 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 26-1 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear 6 cycles after writing to SPMCSR, which is locked for further writing during these cycles. The LPM instruction must be executed within 3 CPU cycles after writing SPMCSR. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Table 26-1. Signature Row Addressing.

Signature Byte Description	Z-Pointer Address
Device ID 0, Manufacture ID	00H
Device ID 1, Flash Size	02H
Device ID 2, Device	04H
FOSCCAL(1)	01H
FOSC SEGMENT(2)	03H
Reserved	05H
SLOW RC Period L	06H
SLOW RC Period H(3)	07H
SLOW RC Temp Prediction L	08H
SLOW RC Temp Prediction H(4)	09H
ULP RC FRQ(6)	0AH
SLOW RC FRQ(5)	0BH
Reserved	0CH:0EH
BGCCR Calibration Byte @ 25°C	0FH
Reserved	10H
BGCRR Calibration Byte @ 25°C	11H

Table 26-1. Signature Row Addressing. (Continued)

Signature Byte Description	Z-Pointer Address
Reserved	12H
BGCCR Calibration Byte @ HOT(7)	13H
V-ADC RAW Cell1 L	14H
V-ADC RAW Cell1 H(8)	15H
V-ADC RAW ADC0 L	16H
V-ADC RAW ADC0 H(8)	17H
VPTAT CAL L	18H
VPTAT CAL H(15)	19H
V-ADC Cell1 Calibration Word L	1AH
V-ADC Cell1 Calibration Word H(9)	1BH
V-ADC Cell2 Calibration Word L	1CH
V-ADC Cell2 Calibration Word H(10)	1DH
V-ADC Cell1 Offset(11)	1EH
V-ADC Cell2 Offset(11)	1FH
V-ADC ADC0 Gain Calibration Word L	20H
V-ADC ADC0 Gain Calibration Word H(12)	21H
V-ADC ADC1 Gain Calibration Word L	22H
V-ADC ADC1 Gain Calibration Word H(13)	23H
V-ADC ADC0 Offset(14)	24H
V-ADC ADC1 Offset(14)	25H
Reserved	26H:2FH
T_{HOT}(16)	30H

Notes:
1. Default FOSCCAL value after reset.
2. FOSCCAL setting used to smooth the transition from one segment to the next when calibrating the Fast RC oscillator.
3. 8 prescaled Slow RC periods in µs using the Oscillator Sampling Interface (@T_{HOT}°C).
4. Characterized Slow RC oscillator frequency temperature drift prediction value.
5. Slow RC oscillator frequency in kHz (@T_{HOT}°C).
6. ULP RC oscillator frequency in kHz (@ T_{HOT}°C).
7. Calibration value found in BGCCR in the first step of VREF calibration (BGCRR = 0x0F), (@ T_{HOT}°C).
8. Calibration word used for the second step of the VREF calibration.
9. Calibration word used to compensate for gain error in V-ADC Cell input 1.
10. Calibration word used to compensate for gain error in V-ADC Cell input 2.
11. Calibration byte used to compensate for offset in V-ADC Cells.
12. Calibration word used to compensate for gain error in ADC0.
13. Calibration word used to compensate for gain error in ADC1.
14. Calibration byte used to compensate for offset in ADC0 and ADC1.
15. Calibration word used to calculate the absolute temperature in Kelvin from a VTEMP conversion.
16. Hot tamperature used for factory calibration in °C.

All other addresses are reserved for future use.

26.2.6 Programming Time for Flash when Using SPM

The Fast RC Oscillator is used to time Flash accesses. Table 26-2 shows the typical programming time for Flash accesses from the CPU.

Table 26-2. SPM Programming Time, f_{OSC} = 8.0 MHz[1]

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms

Note: 1. Minimum and maximum programming times is per individual operation.

Table 26-3. Explanation of different variables used in Figure 26-1 and the mapping to the Z-pointer, ATmega8HVA.

Variable		Corresponding Z-value	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires six bits PC [5:0]).
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[11:6]	Z12:Z7	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[5:0]	Z6:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

Table 26-4. Explanation of different variables used in Figure 26-1 and the mapping to the Z-pointer, ATmega16HVA.

Variable		Corresponding Z-value	Description
PCMSB	12		Most significant bit in the Program Counter. (The Program Counter is 13 bits PC[12:0])

Table 26-4. Explanation of different variables used in Figure 26-1 and the mapping to the Z-pointer, ATmega16HVA.

Variable		Corresponding Z-value	Description
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires six bits PC [5:0]).
ZPCMSB		Z13	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[12:6]	Z13:Z7	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[5:0]	Z6:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

26.3 Register Description

26.3.1 SPMCSR – Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Program memory operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	–	–	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use.

For compatibility with future devices, these bits must be written to zero when SPMCSR is written.

- **Bit 5 – SIGRD: Signature Row Read**

If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. See "Reading the Signature Row from Software" on page 144 for details.

An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect. This operation is reserved for future use and should not be used.

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

- **Bit 3 – RFLB: Read Fuse and Lock Bits**

An LPM instruction within three cycles after RFLB and SPMEN are set in the SPMCSR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See "EEPROM Write Prevents Writing to SPMCSR" on page 143 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a Page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either CTPB, RFLB, PGWRT, or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During Page Erase and Page Write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “100001”,“010001”, “001001”, “000101”, “000011” or “000001” in the lower six bits will have no effect.

27. Memory Programming

27.1 Program And Data Memory Lock Bits

The ATmega8HVA/16HVA provides two Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 27-2. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 27-1. Lock Bit Byte

Lock Bit Byte	Bit No	Description	Default Value[1]
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed

Table 27-2. Lock Bit Protection Modes[1][2]

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	The Fuse bits are locked, and further programming of the Flash and EEPROM is disabled in Programming mode.[1]
3	0	0	The Fuse bits are locked, and further programming and verification of the Flash and EEPROM is disabled in Programming mode.[1]

Notes: 1. Program the Fuse bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed.

27.2 Fuse Bits

The ATmega8HVA/16HVA has two Fuse bytes. Table 27-4 and Table 27-3 describe briefly the functionality of all the fuses and how they are mapped into the Fuse byte. Note that the fuses are read as logical zero, “0”, if they are programmed.

27.2.1 High Byte

Table 27-3. Fuse High Byte

Bit No	Fuse High Byte	Description	Default Value
7:2	–	–	1 (unprogrammed)
1	OSCSEL1	Oscillator Select 1	0 (programmed)
0	OSCSEL0	Oscillator Select 0	1 (unprogrammed)

Note: 1. The default OSCSEL1:0 setting should not be changed. OSCSEL1:0 = '00' is reserved for test purposes. Other values are reserved for future use.

27.2.2 Low Byte

Table 27-4. Fuse Low Byte

Bit No	Fuse Low Byte	Description	Default Value
7	WDTON(3)	Watchdog Timer always on	1 (unprogrammed)
6	EESAVE	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
5	SPIEN(2)	Enable SPI Programming Interface	0 (programmed, SPI prog. enabled)
4	DWEN	Enable debugWIRE	1 (unprogrammed)
3	SELFPRGEN	Self Programming enable	1 (unprogrammed)
2	SUT2(1)	Select start-up time	1 (unprogrammed)
1	SUT1(1)	Select start-up time	1 (unprogrammed)
0	SUT0(1)	Select start-up time	1 (unprogrammed)

Notes: 1. See Table 9-1 on page 25 for details about start-up time.
2. The SPIEN Fuse is not accessible in SPI programming mode.
3. See "WDTCSR – Watchdog Timer Control Register" on page 49 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

27.2.3 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

27.3 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both Programming mode, also when the device is locked. The three bytes reside in a separate address space. The signature bytes of ATmega8HVA/16HVA is given in Table 27-5.

Table 27-5. Device ID

	Signature Bytes Address		
Part	**0x000**	**0x001**	**0x002**
ATmega8HVA	0x1E	0x93	0x10
ATmega16HVA	0x1E	0x94	0x0C

27.4 Calibration Bytes

The ATmega8HVA/16HVA has calibration bytes for the RC Oscillators, internal voltage reference, internal temperature reference and each differential cell voltage input. These bytes reside in the signature address space. See ”Reading the Signature Row from Software” on page 144 for details.

27.5 Page Size

Table 27-6. No. of Words in a Page and No. of Pages in the Flash, ATmega8HVA/16HVA

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega8HVA	4K words (8K bytes)	64 words	PC[5:0]	64	PC[11:6]	11
ATmega16HVA	8K words (16K bytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 27-7. No. of Words in a Page and No. of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7

27.6 Serial Programming

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 27-8 on page 152, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 27-1. Serial Programming and Verify.

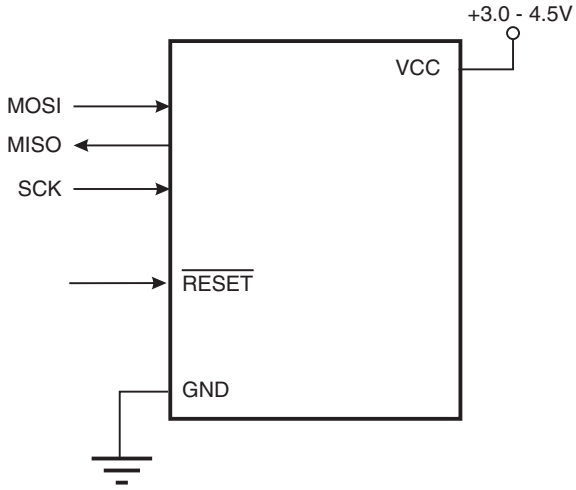


Table 27-8. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
SCK	PB1	I	Serial Clock
MOSI	PB2	I	Serial Data in
MISO	PB3	O	Serial Data out

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on OSCSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2.2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz

High: > 2.2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz

27.6.1 Serial Programming Algorithm

When writing serial data to the ATmega8HVA/16HVA, data is clocked on the rising edge of SCK.

When reading data from the ATmega8HVA/16HVA, data is clocked on the falling edge of SCK. See "Serial Programming" on page 172 for timing details.

To program and verify the ATmega8HVA/16HVA in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 27-10 on page 154):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.

3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the Load Program memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program memory Page is stored by loading the Write Program memory Page instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 27-9.) Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 27-9.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM Memory Page is stored by loading the Write EEPROM Memory Page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 27-7 on page 151). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

Table 27-9. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	4.0 ms
t_{WD_ERASE}	4.0 ms
t_{WD_FUSE}	4.5 ms

27.6.2 Serial Programming Instruction set

Table 27-10 on page 154 and Figure 27-2 on page 155 describes the Instruction set.

Table 27-10. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{BSY}$	$F0	$00	$00	data byte out
Load Instructions				
Load Extended Address byte(1)	$4D	$00	Extended adr	$00
Load Program Memory Page, High byte	$48	adr MSB	adr LSB	high data byte in
Load Program Memory Page, Low byte	$40	adr MSB	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	adr MSB	adr LSB	data byte in
Read Instructions				
Read Program Memory, High byte	$28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	$A0	adr MSB	adr LSB	data byte out
Read Lock bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	adr LSB	data byte out
Read Fuse bits	$50	$00	$00	data byte out
Read Fuse High bits	$58	$08	$00	data byte out
Read Extended Fuse Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions(6)				
Write Program Memory Page	$4C	adr MSB	adr LSB	$00
Write EEPROM Memory	$C0	adr MSB	adr LSB	data byte in
Write EEPROM Memory Page (page access)	$C2	adr MSB	adr LSB	$00
Write Lock bits	$AC	$E0	$00	data byte in
Write Fuse bits	$AC	$A0	$00	data byte in
Write Fuse High bits	$AC	$A8	$00	data byte in
Write Extended Fuse Bits	$AC	$A4	$00	data byte in

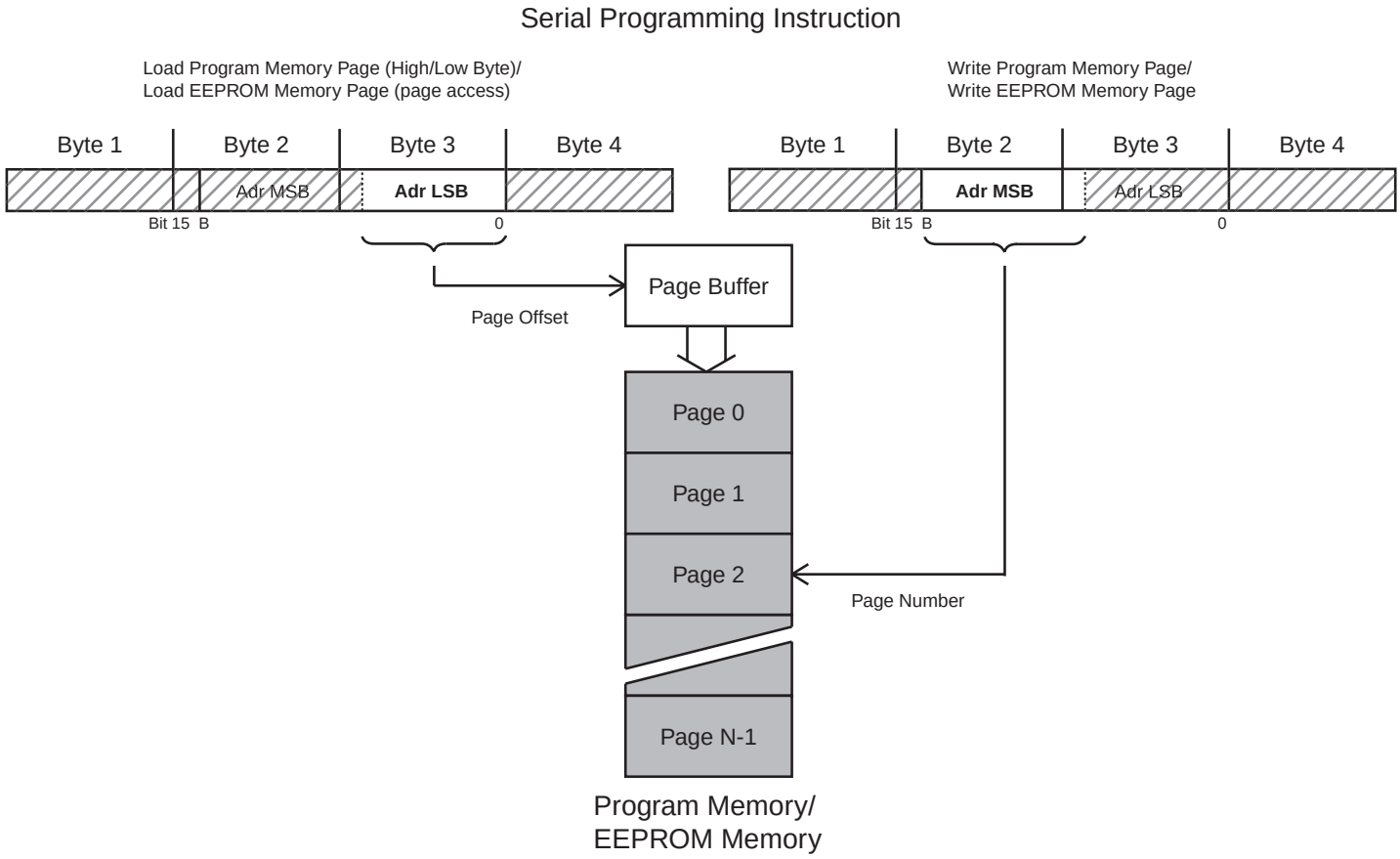
Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. Instructions accessing program memory use word address. This address may be random within the page range.
7. See htt://www.atmel.com/avr for Application Notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 27-2 on page 155.

Figure 27-2. Serial Programming Instruction example



27.7 High-voltage Serial Programming

This section describes how to program and verify Flash Program memory, EEPROM Data memory, Lock bits and Fuse bits in the ATmega8HVA/16HVA.

Figure 27-3. High-voltage Serial Programming

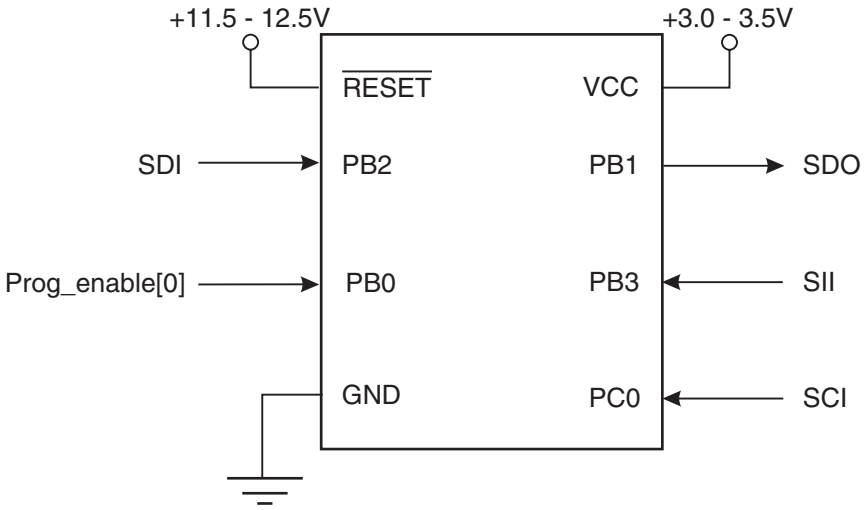


Table 27-11. Pin Name Mapping

Signal Name in High-voltage Serial Programming Mode	Pin Name	I/O	Function
SDO	PB1	O	Serial Data Output
SDI	PB2	I	Serial Data Input
SII	PB3	I	Serial Instruction Input
SCI	PC0	I	Serial Clock Input (min. $2/f_{ck}$ period)

Table 27-12. Pin Values Used to Enter Programming Mode

Pin Name	Symbol	Value
PB0	Prog_enable[0]	0
PB1	Prog_enable[1]	0
PB2	Prog_enable[2]	0
PB3	Prog_enable[3]	0

27.8 High-voltage Serial Programming Algorithm

To program and verify the ATmega8HVA/16HVA in the High-voltage Serial Programming mode, the following sequence is recommended (See instruction formats in Table 27-14):

27.8.1 Enter High-voltage Serial Programming Mode

The following algorithm puts the device in Serial (High-voltage) Programming mode:

1. Set Prog_enable pins listed in Table 27-12 on page 156 to “0000”, RESET pin to 0V and V_{CC} to 0V.
2. Apply 3.0 - 3.5V between V_{CC} and GND. Ensure that V_{CC} reaches at least 1.8V within the next 20 µs.
3. Wait 20 - 60 µs, and apply V_{HRST} - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least t_{HVRST} after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Release Prog_enable[1] pin to avoid drive contention on the Prog_enable[1]/SDO pin.
6. Wait at least 300 µs before giving any serial instructions on SDI/SII.

If the rise time of the V_{CC} is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set Prog_enable pins listed in Table 27-12 on page 156 to “0000”, RESET pin to 0V and V_{CC} to 0V.
2. Apply 3.0 - 3.5V between V_{CC} and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9 - 1.1V, apply V_{HRST} - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least t_{HVRST} after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Release Prog_enable[1] pin to avoid drive contention on the Prog_enable[1]/SDO pin.
6. Wait until V_{CC} actually reaches 3.0 - 3.5V before giving any serial instructions on SDI/SII.

Table 27-13. High-voltage Reset Characteristics

Supply Voltage	RESET Pin High-voltage Threshold	Minimum High-voltage Period for Latching Prog_enable
V_{CC}	V_{HVRST}	t_{HVRST}
3.0V	11.5V	10 µs
3.5V	11.5V	10 µs

27.8.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address High byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

27.8.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM[1] memories plus Lock bits. The Lock bits are not reset until the Program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are re-programmed.

Note: 1. The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

1. Load command “Chip Erase” (see Table 27-14).
2. Wait after Instr.3 until SDO goes high for the “Chip Erase” cycle to finish.
3. Load Command “No Operation”.

27.8.4 Programming the Flash

The Flash is organized in pages, see Table 27-10 on page 154. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

1. Load Command “Write Flash” (see Table 27-14).
2. Load Flash Page Buffer.
3. Load Flash High Address and Program Page. Wait after Instr. 3 until SDO goes high for the “Page Programming” cycle to finish.
4. Repeat 2 through 3 until the entire Flash is programmed or until all data has been programmed.
5. End Page Programming by Loading Command “No Operation”.

When writing or reading serial data to the ATmega8HVA/16HVA, data is clocked on the rising edge of the serial clock, see Figure 27-5, Figure 29-4 and ”High-voltage Serial Programming” on page 173 for details.

Figure 27-4. Addressing the Flash which is Organized in Pages

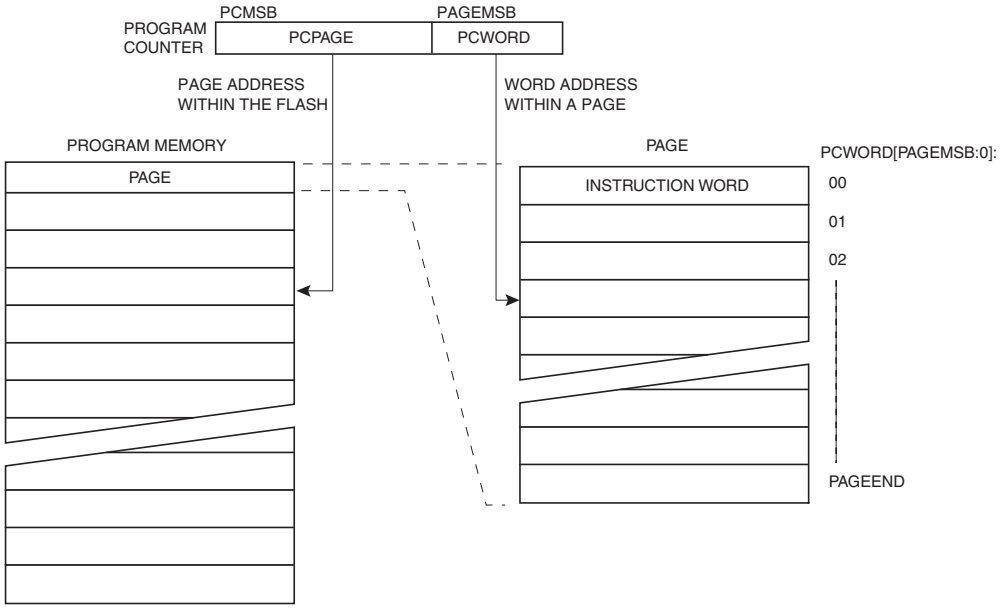
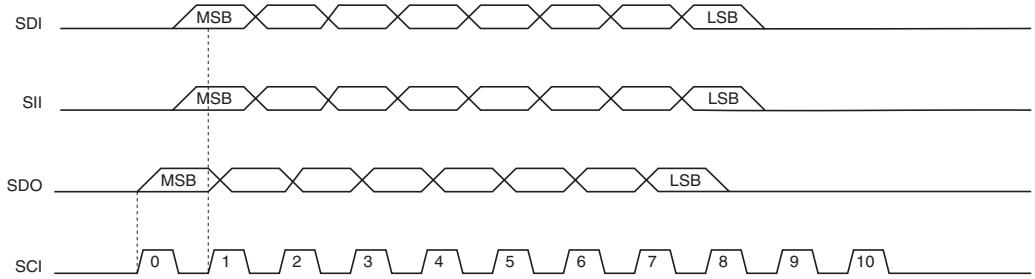


Figure 27-5. High-voltage Serial Programming Waveforms



27.8.5 Programming the EEPROM

The EEPROM is organized in pages, see ”High-voltage Serial Programming” on page 173. When programming the EEPROM, the data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM Data memory is as follows (refer to Table 27-14 on page 160):

1. Load Command “Write EEPROM”.
2. Load EEPROM Page Buffer.
3. Program EEPROM Page. Wait after Instr. 2 until SDO goes high for the “Page Programming” cycle to finish.
4. Repeat 2 through 3 until the entire EEPROM is programmed or until all data has been programmed.
5. End Page Programming by Loading Command “No Operation”.

27.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to Table 27-14 on page 160):

1. Load Command "Read Flash".
2. Read Flash Low and High Bytes. The contents at the selected address are available at serial output SDO.

27.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Table 27-14 on page 160):

1. Load Command “Read EEPROM”.
2. Read EEPROM Byte. The contents at the selected address are available at serial output SDO.

27.8.8 Programming and Reading the Fuse and Lock Bits

The algorithms for programming and reading the Fuse Low/High bits and Lock bits are shown in Table 27-14 on page 160.

27.8.9 Reading the Signature Bytes and Calibration Byte

The algorithms for reading the Signature bytes and Calibration byte are shown in Table 27-14 on page 160.

27.8.10 Power-off sequence

Exit Programming mode by powering the device down, or by bringing RESET pin to 0V.

Table 27-14. High-voltage Serial Programming Instruction Set for ATmega8HVA/16HVA

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Chip Erase	SDI SII SDO	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		
Load “Write Flash” Command	SDI SII SDO	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				
Load Flash Page Buffer	SDI SII SDO	0_ **bbbb_bbbb** _00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_**dddd_dddd**_00 0_0011_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1101_00 x_xxxx_xxxx_xx	
	SDI SII SDO	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx				
Load Flash High Address and Program Page	SDI SII SDO	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		
Load “Read Flash” Command	SDI SII SDO	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx				
Read Flash Low and High Bytes	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	
	SDI SII SDO	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx			
Load “Write EEPROM” Command	SDI SII SDO	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx				
Load EEPROM Page Buffer	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	
Program EEPROM Page	SDI SII SDO	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx			
Write EEPROM Byte	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	
	SDI SII SDO	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx				
Load “Read EEPROM” Command	SDI SII SDO	0_0000_0011_00 0_0100_1100_00 x_xxxx_xxxx_xx				

Table 27-14. High-voltage Serial Programming Instruction Set for ATmega8HVA/16HVA (Continued)

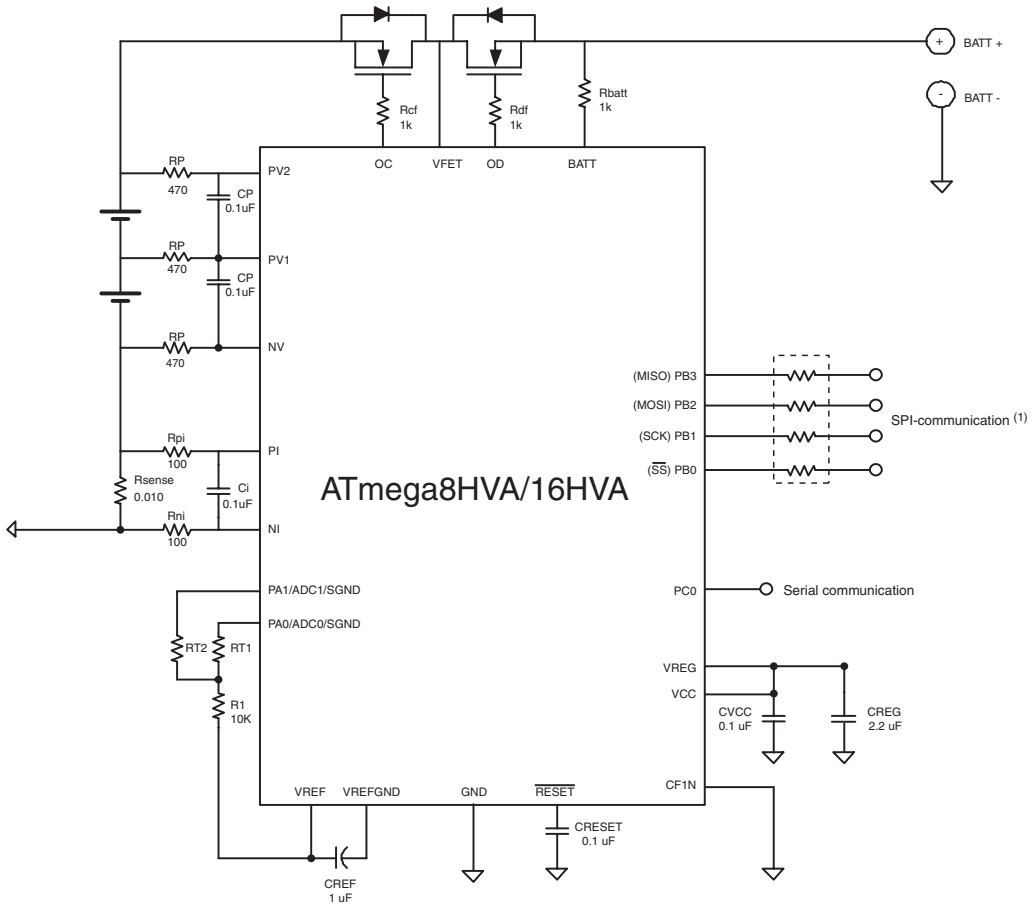
Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Read EEPROM Byte	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**0_00	
Write Fuse High Byte	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**hhhh_hhhh**_00 0_0010_1100_11 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write “0” to program the Fuse Bits.
Write Fuse Low Byte	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**llll_llll**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write “0” to program the Fuse bit.
Write Lock Bit Byte	SDI SII SDO	0_0010_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_cccc_cccc_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write “0” to program the Lock Bit.
Read Fuse High Byte	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_0 h_hhhh_hhhx_xx		Reading “0” means the Fuse bit is programmed.
Read Fuse Low Byte	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **l_llll_lll**x_xx		Reading “0” means the Fuse bit is programmed.
Read Lock Bit Byte	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 c_cccc_cccx_xx		Reading “0” means the Lock bit is programmed.
Read Signature Row Low Byte	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeats Instr 2 4 for each signature low byte address.
Read Signature Row High Byte	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx	Repeats Instr 2 4 for each signature high byte address.
Load “No Operation” Command	SDI SII SDO	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				

Note: 1. **a** = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits, x = don’t care, **c** = Lock Bit Byte, **l** = fuse low byte, **h** = fuse high byte.

Notes: 1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.

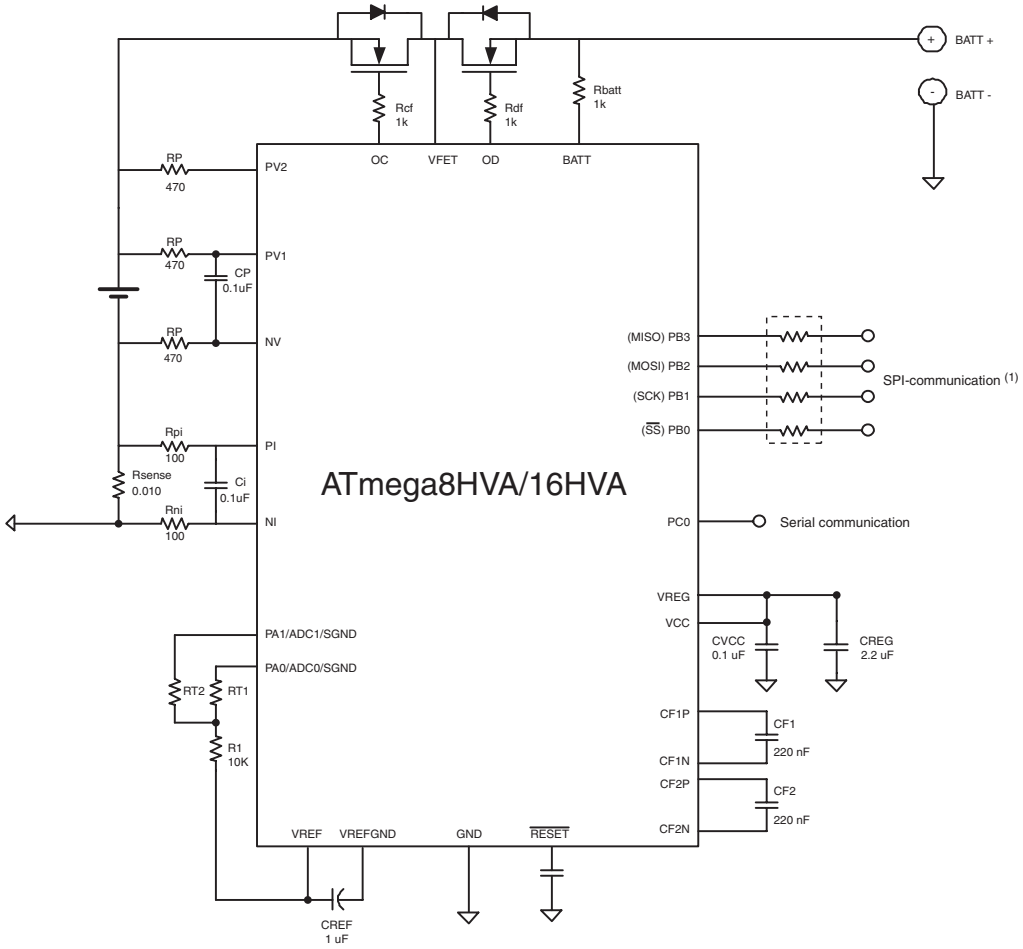
28. Operating Circuit

Figure 28-1. Operating Circuit Diagram, 2-cell.



Notes:
1. The series resistors on the SPI lines are required for In-System Programming and On-chip Debug support. The value of the series resistor depends on the application. A value of 10k will ensure that programming and debugging operates correctly, but it must be determined by the end user that this does not affect the normal operation of the SPI interface.
2. PA1 should be connected to SGND when measuring $V(RT_2)$.
   PA0 should be connected to SGND when measuring $V(RT_1)$.
3. It is recommended to connect CF1P, CF2N, and CF2P to GND.

Figure 28-2. Operating Circuit Diagram, 1-cell



Notes:
1. The series resistors on the SPI lines are required for In-System Programming and On-chip Debug support. The value of the series resistor depends on the application. A value of 10k will ensure that programming and debugging operates correctly, but it must be determined by the end user that this does not affect the normal operation of the SPI interface.
2. PA1 should be connected to SNGD when measuring $V(RT_2)$.
   PA0 should be connected to SNGD when measuring $V(RT_1)$.

Table 28-1. Recommended values for external devices

Symbol	Use	Parameter	Min	Typ	Max	Unit
R1	Pull-up resistor for thermistors	R	8	10	12	kΩ
RT1/RT2	NTC Thermistor	R@25°C	8	10	12	kΩ
		B-constant	3000		4000	N/A
R_S	Source impedance when using PA1..PA0 as V-ADC inputs	R	0	3	7	kΩ
		Worst-case Gain-error due to R_S	0	1	2	%
CREF	VREF decoupling	C	1	2.2	22	µF
CREG	VREG charge-storage capacitor	C	1.1(1)	2.2	22	µF
CVCC	VCC decoupling capacitor	C		0.1		µF
CF1	Fly capacitors	C		220		nF
R_{CF}/R_{DF}		R		1		kΩ
R_{BATT}		R		1		kΩ
RP	Cell-input LP-filter resistors	R	10	500	1000	Ω
CP	Cell-input LP-filter capacitors	C	0.01	0.1	0.5	µF
RP*CP	Cell-input LP-filter time-constants	τ	6.5	25	100	µs
R_{PI}	Current sense LP-filter resistors	R	10	100	500	Ω
R_{NI}						
Ci	Current sense LP-filter capacitor	C	0.01	0.1	0.4	µF
$(R_{PI}+R_{NI})*Ci$	Current sense LP-filter time-constant	τ		10	20	µs
Rsense	Coulomb Counter sense resisotor	R		10		mΩ

Notes: 1. This is the absolute minimum capacitance required to ensure stable operation of the voltage regulator.

29. Electrical Characteristics

29.1 Absolute Maximum Ratings*

Operating Temperature	-20°C to +85°C
Storage Temperature	-65°C to +150°C
Voltage on PA0 - PA1, PI, and NI with respect to Ground	-0.5V to V_{REG} +0.5V
Voltage on PB0 - PB3 with respect to Ground	-0.5V to VCC +0.5V
Voltage on PC0 with respect to Ground	-0.5V to + 6.0V
Voltage on VFET with respect to Ground	-0.5V to + 18V
Voltage on OD, OC, BATT, and $\overline{\text{RESET}}$ with respect to Ground	-0.5V to + 13V
Voltage on NV, PV1, and PV2 with respect to Ground	-0.5V to VFET + 1.0V
Maximum Operating Voltage on VREG and VCC	4.5V
Maximum Operating Voltage on VFET	9V

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

29.2 DC Characteristics

Table 29-1. Electrical Characteristics[1] (T_A = -10°C to 70°C unless otherwise specified)

	Parameter	Condition	Min	Typ	Max	Unit
Supply Current	Active	4.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits set.		2.5		mA
		1.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits set.		800		µA
	Idle	4.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits set.		550		µA
		1.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits set.		270		µA
	ADC Noise Reduction	4.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits except PRVADC set. VADC enabled.		580		µA
		1.0 MHz, 4V ≤ V_{FET} ≤ 8.4V, All PRR bits except PRVADC set. VADC enabled.		380		µA
	Power-save	Only WDT Enabled, DUVR mode disabled, V_{FET} =4V		25		µA
		WDT, CC-ADC, OC, OD and Battery Protection Enabled, DUVR mode disabled, V_{FET} = 8.4V		110		µA
		WDT, CC-ADC, OC, OD and Battery Protection Enabled, DUVR mode disabled, V_{FET} = 3V		240		µA
	Power-off	V_{FET} ≤ 6V		10	1000	nA
		6V ≤ V_{FET} ≤ 8.4V		2	10	µA

Table 29-1. Electrical Characteristics[1] (T_A = -10°C to 70°C unless otherwise specified) (Continued)

	Parameter	Condition		Min	Typ	Max	Unit
Voltage Regulator[2]	Voltage Regulator Operating Voltage	Combined Step-up and Linear mode		1.8		9	V
		Linear mode only		3.6		9	
	Regulated Output Voltage[3] (V_{REG})	V_{FET} = 1.8V, I_{load} = 2 mA		2.9		3.4	
		V_{FET} = 2.0V, I_{load} = 5 mA		3.0		3.4	
		V_{FET} = 2.4V, I_{load} = 8.5 mA		3.1		3.4	
		V_{FET} = 3.0V, I_{load} = 10 mA		3.1		3.4	
		V_{FET} = 3.8V, I_{load} = 10 mA		3.1		3.4	
		V_{FET} = 5.5V, I_{load} = 10 mA		3.1		3.4	
		V_{FET} = 9.0V, I_{load} = 10 mA		3.1		3.4	
	V_{FET} Linear/Step-up switching level[5]	Linear-> Step-up	I_{load} = 2 mA		3.5		
		Step-up-> Linear			3.6		
	Voltage Regulator Short-circuit Level (RSCL)	Combined Step-up and Linear mode			1.7		
		Linear mode only			3.5		
	Ripple, Step-up mode[5]	IOUT = 1 mA, CREG = 2.2 µF, ESR = 0.1 Ω	VFET = 3.0V		5		mV
		IOUT = 10 mA, CREG = 2.2 µF, ESR = 0.1 Ω	VFET = 3.0V		30		
VREF	Reference Voltage				1.100		V
	Ref. Voltage Accuracy	After calibration, at calibration temperature			± 0.1	± 0.2	%
	Temperature Drift[3][5]					90	ppm/ °C
V-ADC	Conversion Time	clk_{VADC} = 1 MHz			519		µs
	Effective Resolution				12		Bits
	Gain ADC0/1 Un-scaled Inputs				263		µV/LSB
	Gain Cell Inputs (x 0.2)				1.43		mV/LSB
	INL				1	3	LSB
	Input Voltage range ADC0, ADC1, VTEMP			0		1	V
	Input Voltage range CELL1[9]			1.5		5	
	Input Voltage range CELL2[9]	PV1 >= 1.5V		0		5	
	Offset[8]				6		LSB
	Error ADC0/1 Inputs[3][7]	0.1V < V_{ADC} < 0.9V				± 0.5	%
	Error Cell Inputs[3][7]	V_{CELL} = 4V / T_A = -10° - 70°C				± 0.5	%

Table 29-1. Electrical Characteristics[1] (T_A = -10°C to 70°C unless otherwise specified) (Continued)

	Parameter	Condition	Min	Typ	Max	Unit
Coulomb Counter	Reference Voltage			± 110		mV
	Conversion Time and Resolution[5]	26.9 µV Resolution		3.9		ms
		0.84 µV Resolution		1000		ms
	INL[5]				4	LSB
	CC-ADC Offset[5][6]			2.5	± 15	LSB
	Gain Error[3]	-100 mV < $V_{PI - NI}$ < 100 mV		± 0.1	± 1	%
Temperature Sensor	V_{PTAT}, Voltage Proportional to Absolute Temperature			0.67		mV/K
	Absolute Accuracy[4]	Measured in Active mode		± 2	± 5	K
Slow RC Oscillator[10]	Frequency		91	131	171	kHz
	Frequency drift over temperature[5]			1.5		%
	Slow RC Frequency predicion error[5]				1	%
Ultra Low Power RC Oscillator[10]	Frequency		89	128	167	kHz
	Frequency drift over temperature[5]			6		%

Notes:
1. All DC Characteristics contained in this data sheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.
2. Voltage Regulator performance is based on 220 nF fly capacitors and 2.2 µF smooth capacitor.
3. After VREF calibration at a second temperature. By default the first calibration is performed at temperature T_{HOT} in Atmel factory test. The value of T_{HOT} is stored in the signature row. The second calibration step can easily be implemented in a standard test flow at room temperature.
4. The measured V_{PTAT} voltage must be scaled with the calibration value stored in the VPTAT Calibration Register to get the absolute temperature. The specified value represents target accuracy after Atmel factory calibration. Accuracy can be further improved by doing a system calibration measurement at a well-known temperature.
5. This value is not tested in production.
6. After software offset compensation, using the polarity switching (CADPOL) feature.
7. After scaling of VADC raw data using Gain and Offset Calibration values stored in Signature Row.
8. Actual offset for each channel stored in signature row can be used to remove this offset error.
9. If the cell input needs to be measured when PV1 is below 1.5V, Atmel can provide data that facilitates less accurate measurements in this range.
10. Actual frequency measured at Atmel factory stored in signature row.

29.3 External Interrupt Characteristics

Table 29-2. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

29.4 General I/O Lines characteristics

Table 29-3. [1] T_A = -10°C to 70°C, V_{CC} = 3.3V

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, Except RESET pin		-0.5		$0.3V_{CC}$ [2]	V
V_{IL1}	Input Low Voltage, RESET pin				$0.3V_{CC}$ [2]	
V_{IH}	Input High Voltage, Except RESET pin		$0.6V_{CC}$ [3]		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage, RESET pin		$0.9V_{CC}$ [3]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 5mA			0.5	V
V_{OH}	Output High Voltage	I_{OH} = 2 mA	2.3			V
I_{IL}	Input Leakage Current I/O Pin	Pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	Pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor		20		50	kΩ

Notes:
1. Applicable for all except PC0.
2. “Max” means the highest value where the pin is guaranteed to be read as low
3. “Min” means the lowest value where the pin is guaranteed to be read as high
4. Although each I/O port can sink more than the test conditions (5 mA at V_{CC} = 3.3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all IOL should not exceed 20 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (2 mA at V_{CC} = 3.3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all IOH should not exceed 2 mA.

Table 29-4. PC0 Characteristics (T_A = -10°C to 70°C unless otherwise specified)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.8 [1]	V
V_{IH}	Input High-voltage		2.1 [2]	5.5	V
V_{OL}	Output Low-voltage	350 µA sink current	0	0.4	V
t_r [3]	Rise Time			300	ns
t_{of} [3]	Output Fall Time from V_{IHmin} to V_{ILmax}	C_b < 400 pF [4]		250	ns
t_{SP} [3]	Spikes Suppressed by Input Filter		0	50	ns
I_i [3]	Input Current	$0.1V_{BUS} < V_i < 0.9V_{BUS}$	-5	5	µA
C_i [3]	Capacitance			10	pF

Notes:
1. “Max” means the highest value where the pin is guaranteed to be read as low
2. “Min” means the lowest value where the pin is guaranteed to be read as high
3. This value is not tested in production.
4. C_b = capacitance of one bus line in pF

29.5 FET Driver Characteristics

Table 29-5. FET Driver Outputs specification[1] (T_A = -10°C to 70°C unless otherwise specified)

Parameter	Condition	Min.	Typ.	Max.	Units
VFET DC level[2]	1 cell DUVR operation, VREF = 1.100V	1.9	2.0	2.1	V
	2 cell DUVR operation, VREF = 1.100V	3.8	4.0	4.2	V
VFET ripple[2]	1 cell DUVR operation		±0.1		V
	2 cell DUVR operation		±0.1		V
OC, OD clamping voltage			14.0		V
OC, OD	Normal ON operation	VFET + 2.5	VFET + 4	VFET + 6.5	V
OC, OD	Normal OFF operation		0.0	0.1	V
Risetime[2][3] (OC, OD, 0 - 90 %)	Normal ON operation		1	2	ms
Falltime[2][3] (OC, OD, 100 - 10 %)	Normal OFF operation		5	10	µs

Notes:
1. All DC Characteristics contained in this data sheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.
2. These numbers assume the use of one external N-channel FET of model TPCS8210. If other FETs are used, the numbers may deviate somewhat. The equivalent capacitive loads at OC and OD are around 1.2 nF. Rise and fall times scale approximately proportional to the capacitive loading
3. Not tested in production.

29.6 Power-on and Reset Characteristics

Table 29-6. Reset Characteristics(T_A = -10°C to 70°C unless otherwise specified)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Threshold Voltage[1]	VFET = 8.4V	2.75	3.65	4.1	V
		VFET = 4.2V	2.75	3.5	3.95	
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin			900		ns
V_{BOT}	Brown-Out Detection (BOD) Trigger Level			2.9		V
V_{HYST}	BOD Level Hysteresis			100		mV
V_{BLOT}	Power-off Threshold Voltage			2.4		V

Note:
1. The voltage at the Pack + terminal will be slightly higher than V_{POT} when the chip is enabled. This is because of an internal Pull-down current on the BATT pin in the range 50 - 110 uA and the R_{BATT} resistor connected between the Pack + terminal and the BATT pin. R_{BATT} = 1k gives a voltage drop 0.05 - 0.11V.

29.7 SPI Timing Characteristics

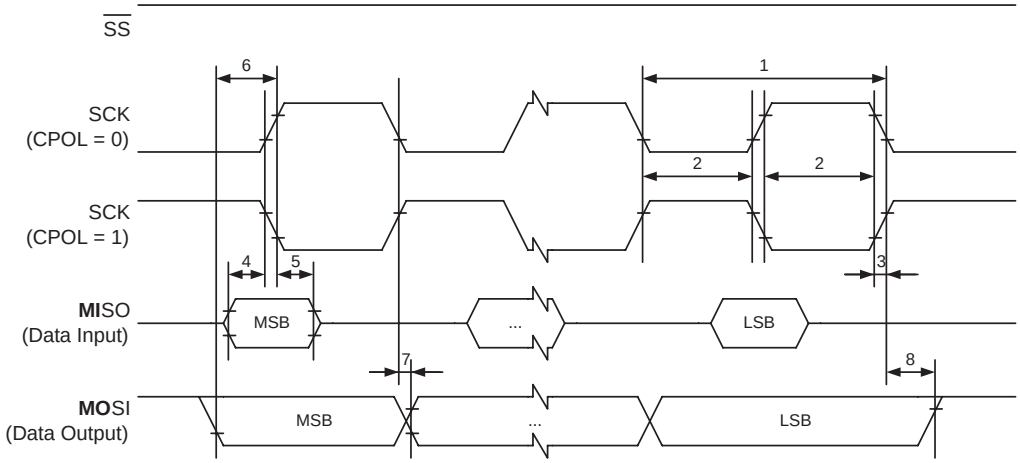
See Figure 29-1 on page 171 and Figure on page 172 for details.

Table 29-7. SPI Timing Parameters

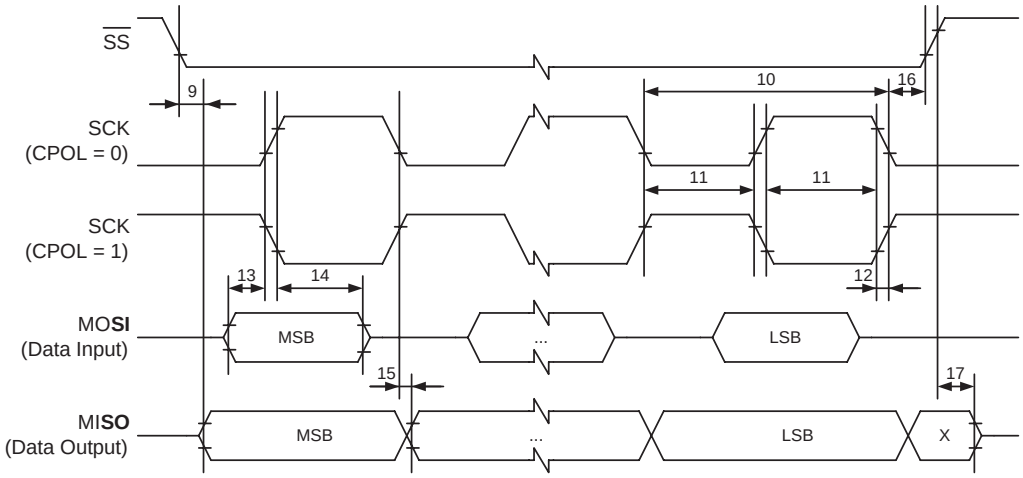
	Description	Mode	Min	Typ	Max	Units
1	SCK period	Master		See Figure		ns
2	SCK high/low	Master		50% duty		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t_{sck}		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	$\overline{SS}$ low to out	Slave		15		
10	SCK period	Slave	4 • t_{ck} + 40 ns			
11	SCK high/low(1)	Slave	2 • t_{ck} + 20 ns			
12	Rise/Fall time	Slave		1.6		µs
13	Setup	Slave	10			ns
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	$\overline{SS}$ low to SCK	Slave	20			

Note: 1. Refer to "Serial Programming" on page 151 for serial programming requirements.

Figure 29-1. SPI Interface Timing Requirements (Master Mode)



SPI Interface Timing Requirements (Slave Mode)



29.8 Programming Characteristics

29.8.1 Serial Programming

Figure 29-2. Serial Programming Timing

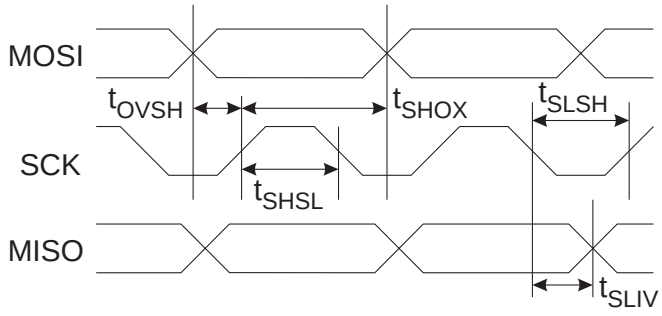


Figure 29-3. Serial Programming Waveforms

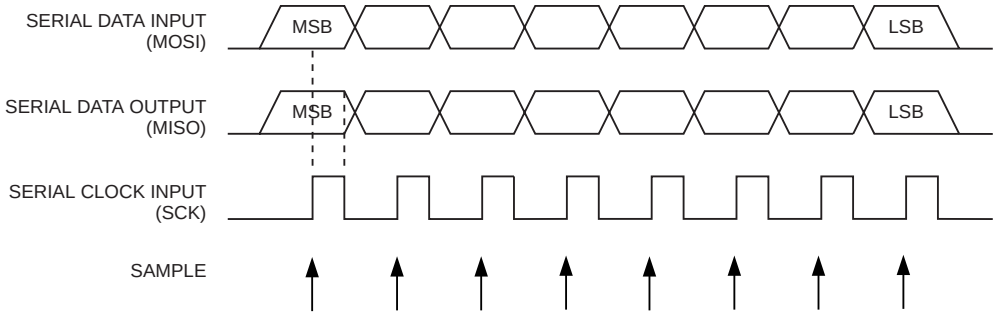


Table 29-8. Serial Programming Characteristics, T_A = -10°C to 70°C, V_{CC} = 3.0 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (ATmega8HVA/16HVA)	0		4	MHz
t_{CLCL}	Oscillator Period (ATmega8HVA/16HVA)	250			ns
t_{SHSL}	SCK Pulse Width High	2.2 t_{CLCL} (1)			
t_{SLSH}	SCK Pulse Width Low	2.2 t_{CLCL} (1)			
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			
t_{SLIV}	SCK Low to MISO Valid		15		ns

Note: 1. 2.2 t_{CLCL} for f_{ck} < 12 MHz, 3 t_{CLCL} for f_{ck} >= 12 MHz

29.8.2 High-voltage Serial Programming

Figure 29-4. High-voltage Serial Programming Timing

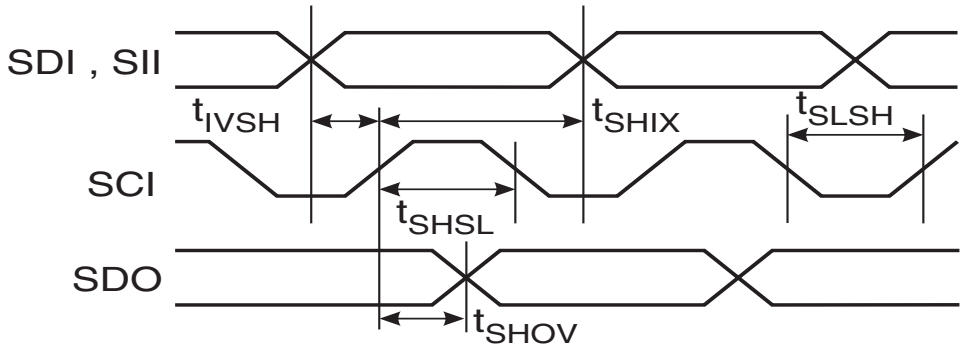


Table 29-9. High-voltage Serial Programming Characteristics T_A = 25°C ± 10%, V_{CC} = 3.3V ± 10% (Unless otherwise noted)

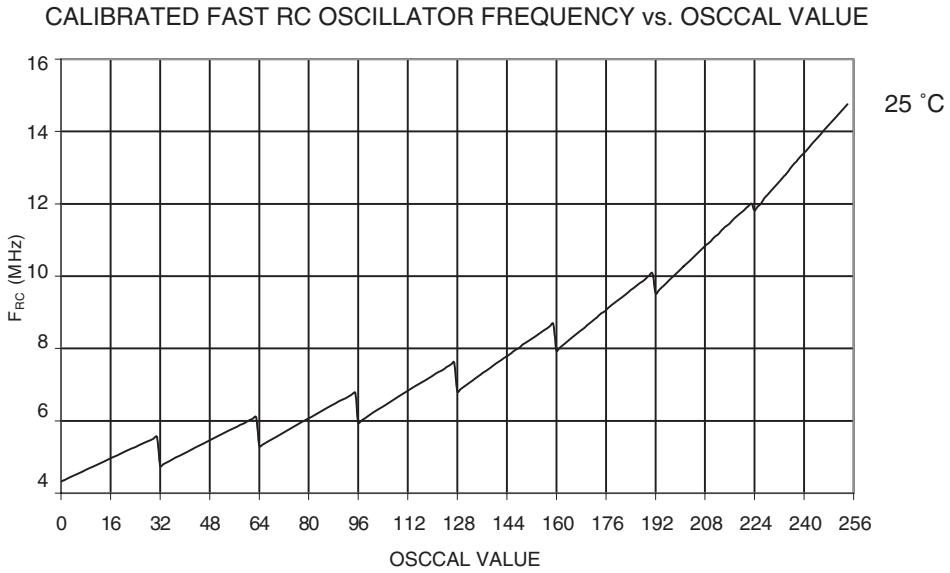
Symbol	Parameter	Min	Typ	Max	Units
t_{SHSL}	SCI (PC0) Pulse Width High	$1/f_{ck}$			ns
t_{SLSH}	SCI (PC0) Pulse Width Low	$1/f_{ck}$			ns
t_{IVSH}	SDI (PB2), SII (PB3) Valid to SCI (PC0) High	50			ns
t_{SHIX}	SDI (PB2), SII (PB3) Hold after SCI (PC0) High	50			ns
t_{SHOV}	SCI (PC0) High to SDO (PB1) Valid		16		ns
t_{WLWH_PFB}	Wait after Instr. 3 for Write Fuse Bits		2.5		ms

30. Typical Characteristics – Preliminary Data

All Typical Characteristics contained in this data sheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These figures are preliminary and will be updated after characterization of actual silicon.

These figures are not tested during manufacturing, and are added for illustration purpose only.

Figure 30-1. Fast RC Oscillator frequency vs. OSCCAL value.



31. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	BPPLR	–	–	–	–	–	–	BPPLE	BPPL	127
(0xFD)	BPCR	–	–	–	SCD	DOCD	COCD	DHCD	CHCD	127
(0xFC)	BPHCTR	–	–	HCPT[5:0]						130
(0xFB)	BPOCTR	–	–	OCPT[5:0]						129
(0xFA)	BPSCTR	–	SCPT[6:0]							128
(0xF9)	BPCHCD	CHCDL[7:0]								132
(0xF8)	BPDHCD	DHCDL[7:0]								132
(0xF7)	BPCOCD	COCDL[7:0]								131
(0xF6)	BPDOCD	DOCDL[7:0]								131
(0xF5)	BPSCD	SCDL[7:0]								131
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	BPIFR	–	–	–	SCIF	DOCIF	COCIF	DHCIF	CHCIF	134
(0xF2)	BPIMSK	–	–	–	SCIE	DOCIE	COCIE	DHCIE	CHCIE	133
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	FCSR	–	–	–	–	DUVRD	CPS	DFE	CFE	138
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	CADICH	CADIC[15:8]								110
(0xE8)	CADICL	CADIC[7:0]								110
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	CADRC	CADRC[7:0]								111
(0xE5)	CADCSRB	–	CADACIE	–	CADICIE	–	CADACIF	CADRCIF	CADICIF	109
(0xE4)	CADCSRA	CADEN	CADPOL	CADUB	CADAS[1:0]		CADSI[1:0]		CADSE	107
(0xE3)	CADAC3	CADAC[31:24]								110
(0xE2)	CADAC2	CADAC[23:16]								110
(0xE1)	CADAC1	CADAC[15:8]								110
(0xE0)	CADAC0	CADAC[7:0]								110
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	BGCRR	BGCR[7:0]								119
(0xD0)	BGCCR	BGD	–	BGCC[5:0]						118
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	ROCR	ROCS	–	–	–	–	–	ROCWIF	ROCWIE	123
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	Reserved	–	–	–	–	–	–	–	–	
(0xC5)	Reserved	–	–	–	–	–	–	–	–	
(0xC4)	Reserved	–	–	–	–	–	–	–	–	
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	Reserved	–	–	–	–	–	–	–	–	
(0xC1)	Reserved	–	–	–	–	–	–	–	–	
(0xC0)	Reserved	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	Reserved	–	–	–	–	–	–	–	–	
(0xBC)	Reserved	–	–	–	–	–	–	–	–	
(0xBB)	Reserved	–	–	–	–	–	–	–	–	
(0xBA)	Reserved	–	–	–	–	–	–	–	–	
(0xB9)	Reserved	–	–	–	–	–	–	–	–	
(0xB8)	Reserved	–	–	–	–	–	–	–	–	
(0xB7)	Reserved	–		–	–	–	–	–	–	
(0xB6)	Reserved	–	–	–	–	–	–	–	–	
(0xB5)	Reserved	–	–	–	–	–	–	–	–	
(0xB4)	Reserved	–	–	–	–	–	–	–	–	
(0xB3)	Reserved	–	–	–	–	–	–	–	–	
(0xB2)	Reserved	–	–	–	–	–	–	–	–	
(0xB1)	Reserved	–	–	–	–	–	–	–	–	
(0xB0)	Reserved	–	–	–	–	–	–	–	–	
(0xAF)	Reserved	–	–	–	–	–	–	–	–	
(0xAE)	Reserved	–	–	–	–	–	–	–	–	
(0xAD)	Reserved	–	–	–	–	–	–	–	–	
(0xAC)	Reserved	–	–	–	–	–	–	–	–	
(0xAB)	Reserved	–	–	–	–	–	–	–	–	
(0xAA)	Reserved	–	–	–	–	–	–	–	–	
(0xA9)	Reserved	–	–	–	–	–	–	–	–	
(0xA8)	Reserved	–	–	–	–	–	–	–	–	
(0xA7)	Reserved	–	–	–	–	–	–	–	–	
(0xA6)	Reserved	–	–	–	–	–	–	–	–	
(0xA5)	Reserved	–	–	–	–	–	–	–	–	
(0xA4)	Reserved	–	–	–	–	–	–	–	–	
(0xA3)	Reserved	–	–	–	–	–	–	–	–	
(0xA2)	Reserved	–	–	–	–	–	–	–	–	
(0xA1)	Reserved	–	–	–	–	–	–	–	–	
(0xA0)	Reserved	–	–	–	–	–	–	–	–	
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	Reserved	–	–	–	–	–	–	–	–	
(0x96)	Reserved	–	–	–	–	–	–	–	–	
(0x95)	Reserved	–	–	–	–	–	–	–	–	
(0x94)	Reserved	–	–	–	–	–	–	–	–	
(0x93)	Reserved	–	–	–	–	–	–	–	–	
(0x92)	Reserved	–	–	–	–	–	–	–	–	
(0x91)	Reserved	–	–	–	–	–	–	–	–	
(0x90)	Reserved	–	–	–	–	–	–	–	–	
(0x8F)	Reserved	–	–	–	–	–	–	–	–	
(0x8E)	Reserved	–	–	–	–	–	–	–	–	
(0x8D)	Reserved	–	–	–	–	–	–	–	–	
(0x8C)	Reserved	–	–	–	–	–	–	–	–	
(0x8B)	Reserved	–	–	–	–	–	–	–	–	
(0x8A)	Reserved	–	–	–	–	–	–	–	–	
(0x89)	OCR1B	Timer/Counter1 – Output Compare Register B								92
(0x88)	OCR1A	Timer/Counter1 – Output Compare Register A								91
(0x87)	Reserved	–	–	–	–	–	–	–	–	
(0x86)	Reserved	–	–	–	–	–	–	–	–	
(0x85)	TCNT1H	Timer/Counter1 (8 Bit) High Byte								91
(0x84)	TCNT1L	Timer/Counter1 (8 Bit) Low Byte								91
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	Reserved	–	–	–	–	–	–	–	–	
(0x81)	TCCR1B	–	–	–	–	–	CS12	CS11	CS10	76
(0x80)	TCCR1A	TCW1	ICEN1	ICNC1	ICES1	ICS1	–	–	WGM10	90
(0x7F)	Reserved	–	–	–	–	–	–	–	–	
(0x7E)	DIDR0	–	–	–	–	–	–	PA1DID	PA0DID	116

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	VADMUX	–	–	–	–	VADMUX[3:0]				114
(0x7B)	Reserved	–	–	–	–	–	–	–	–	
(0x7A)	VADCSR	–	–	–	–	VADEN	VADSC	VADCCIF	VADCCIE	114
(0x79)	VADCH	–	–	–	–	VADC Data Register High byte				115
(0x78)	VADCL	VADC Data Register Low byte								115
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	Reserved	–	–	–	–	–	–	–	–	
(0x6F)	TIMSK1	–	–	–	–	ICIE1	OCIE1B	OCIE1A	TOIE1	92
(0x6E)	TIMSK0	–	–	–	–	ICIE0	OCIE0B	OCIE0A	TOIE0	92
(0x6D)	Reserved	–	–	–	–	–	–	–	–	
(0x6C)	Reserved	–	–	–	–	–	–	–	–	
(0x6B)	Reserved	–	–	–	–	–	–	–	–	
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	56
(0x68)	Reserved	–	–	–	–	–	–	–	–	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	FOSCCAL	Fast Oscillator Calibration Register								30
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR0	–	–	PRVRM	–	PRSPI	PRTIM1	PRTIM0	PRVADC	39
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	–	–	CLKPS1	CLKPS0	31
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	49
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	12
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	–	–	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	147
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	–	CKOE	PUD	–	–	–	–	73/31
0x34 (0x54)	MCUSR	–	–	–	OCDRF	WDRF	BODRF	EXTRF	PORF	49
0x33 (0x53)	SMCR	–	–	–	–	SM[2:0]			SE	39
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	DWDR	debugWIRE Data Register								140
0x30 (0x50)	Reserved	–	–	–	–	–	–	–	–	
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPI Data Register								103
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	102
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	101
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								23
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								23
0x29 (0x49)	OCR0B	Timer/Counter0 Output Compare Register B								92
0x28 (0x48)	OCR0A	Timer/Counter0 Output Compare Register A								91
0x27 (0x47)	TCNT0H	Timer/Counter0 (8 Bit) High Byte								91
0x26 (0x46)	TCNT0L	Timer/Counter0 (8 Bit) Low Byte								91
0x25 (0x45)	TCCR0B	–	–	–	–	–	CS02	CS01	CS00	76
0x24 (0x44)	TCCR0A	TCW0	ICEN0	ICNC0	ICES0	ICS0	–	–	WGM00	90
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	–	PSRSYNC	
0x22 (0x42)	Reserved	–	–	–	–	–	–	–	–	
0x21 (0x41)	EEAR	EEPROM Address Register Low Byte								19
0x20 (0x40)	EEDR	EEPROM Data Register								19
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	19
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								23
0x1D (0x3D)	EIMSK	–	–	–	–	–	INT2	INT1	INT0	57
0x1C (0x3C)	EIFR	–	–	–	–	–	INTF2	INTF1	INTF0	57

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	Reserved	–	–	–	–	–	–	–	–	
0x1A (0x3A)	Reserved	–	–	–	–	–	–	–	–	
0x19 (0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	OSICSR	–	–	–	OSISEL0	–	–	OSIST	OSIEN	32
0x16 (0x36)	TIFR1	–	–	–	–	ICF1	OCF1B	OCF1A	TOV1	93
0x15 (0x35)	TIFR0	–	–	–	–	ICF0	OCF0B	OCF0A	TOV0	93
0x14 (0x34)	Reserved	–	–	–	–	–	–	–	–	
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E (0x2E)	Reserved	–	–	–	–	–	–	–	–	
0x0D (0x2D)	Reserved	–	–	–	–	–	–	–	–	
0x0C (0x2C)	Reserved	–	–	–	–	–	–	–	–	
0x0B (0x2B)	Reserved	–	–	–	–	–	–	–	–	
0x0A (0x2A)	Reserved	–	–	–	–	–	–	–	–	
0x09 (0x29)	Reserved	–	–	–	–	–	–	–	–	
0x08 (0x28)	PORTC	–	–	–	–	–	–	–	PORTC0	62
0x07 (0x27)	Reserved	–	–	–	–	–	–	–	–	
0x06 (0x26)	PINC	–	–	–	–	–	–	–	PINC0	62
0x05 (0x25)	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0	73
0x04 (0x24)	DDRB	–	–	–	–	DDB3	DDB2	DDB1	DDB0	73
0x03 (0x23)	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0	73
0x02 (0x22)	PORTA	–	–	–	–	–	–	PORTA1	PORTA0	73
0x01 (0x21)	DDRA	–	–	–	–	–	–	DDA1	DDA0	73
0x00 (0x20)	PINA	–	–	–	–	–	–	PINA1	PINA0	73

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The ATmega8HVA/16HVA is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from $60 - $FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

32. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP[1]	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL[1]	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2

32. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1

32. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Note: 1. These instructions are only available in ATmega16HVA.

33. Ordering Information

33.1 ATmega8HVA

Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
1 - 4	1.8 - 9.0V	ATmega8HVA-4CKU ATmega8HVA-4TU	36CK1 28T	-20 to +85°C

Notes: 1. Pb-free packaging, complies with the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
36CK1	36-pad, (6.50 x 3.50 x 0.85 mm Body, 0.60 mm Pitch), Land Grid Array (LGA) Package.
28T	28-lead (8 x 13.4 mm) Plastic Thin Small Outline Package, Type I (TSOP)

33.2 ATmega16HVA

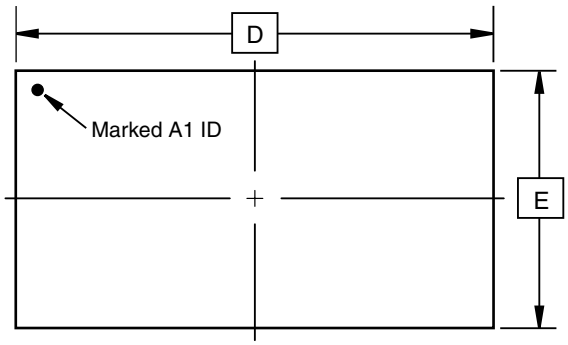
Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
1 - 4	1.8 - 9.0V	ATmega16HVA-4CKU ATmega16HVA-4TU	36CK1 28T	-20 to +85°C

Notes: 1. Pb-free packaging, complies with the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

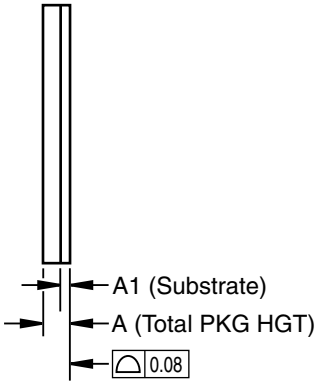
Package Type	
36CK1	36-pad, (6.50 x 3.50 x 0.85 mm Body, 0.60 mm Pitch), Land Grid Array (LGA) Package.
28T	28-lead (8 x 13.4 mm) Plastic Thin Small Outline Package, Type I (TSOP)

34. Packaging Information

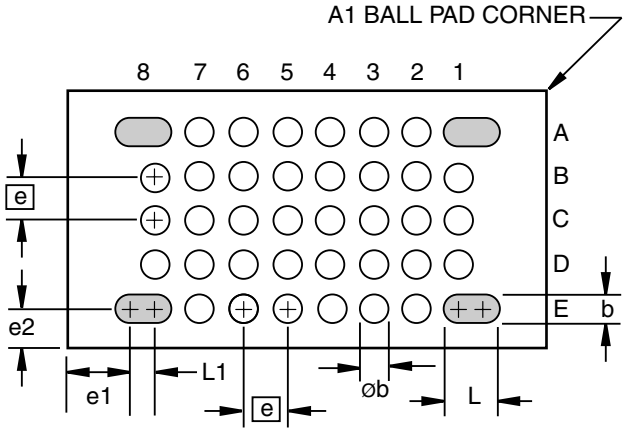
34.1 36CK1



Top View



Side View



Bottom View

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
D	6.40	6.50	6.60	
E	3.40	3.50	3.60	
A	0.59	0.66	0.73	
A1	0.17	0.21	0.25	
L		0.70 REF		2
L1		0.35 REF		
b		0.35 REF		2
øb	0.32	0.35	0.38	2
e		0.60 TYP		
e1		0.80 REF		
e2		0.55 REF		

Notes: 1. This drawing is for general information only.

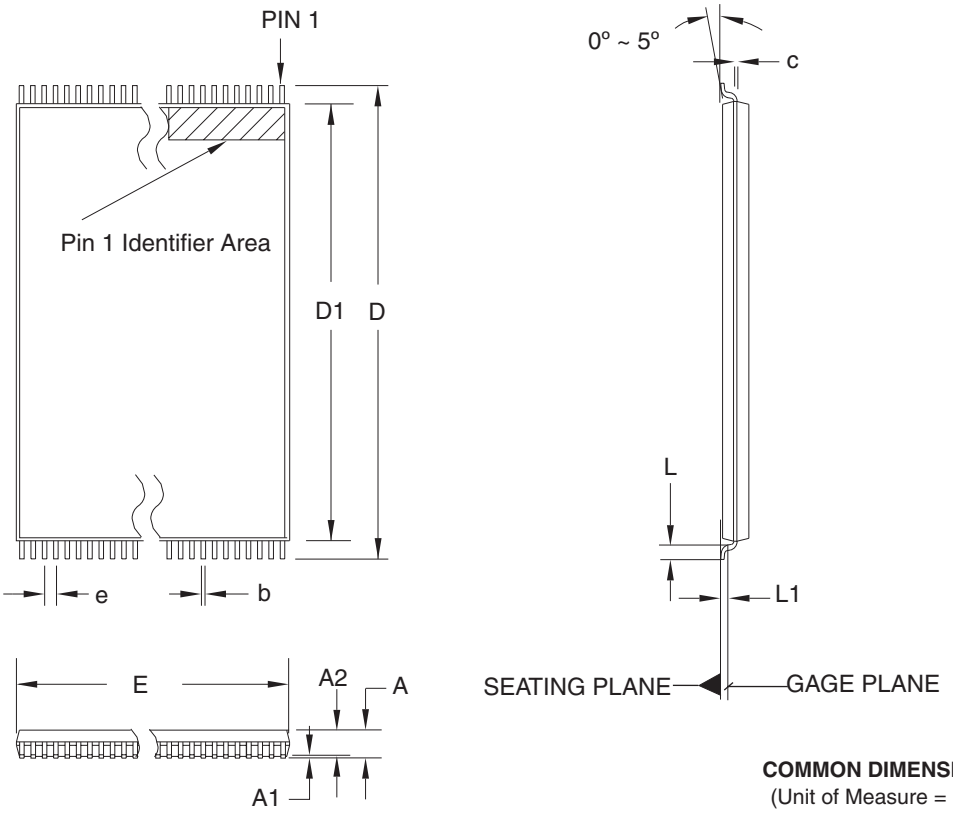
2. Metal pad dimensions.

3. ⬭ = > Dummy pad.

3/15/07

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**36CK1**, 36-Pad, 6.50 x 3.50 x 0.73 mm Body, 0.60 mm Pitch, Land Grid Array (LGA) Package	36CK1	D

34.2 28T



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.90	1.00	1.05	
D	13.20	13.40	13.60	
D1	11.70	11.80	11.90	Note 2
E	7.90	8.00	8.10	Note 2
L	0.50	0.60	0.70	
L1	0.25 BASIC			
b	0.17	0.22	0.27	
c	0.10	–	0.21	
e	0.55 BASIC			

Notes:
1. This package conforms to JEDEC reference MO-183.
2. Dimensions D1 and E do not include mold protrusion. Allowable protrusion on E is 0.15 mm per side and on D1 is 0.25 mm per side.
3. Lead coplanarity is 0.10 mm maximum.

12/06/02

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**28T**, 28-lead (8 x 13.4 mm) Plastic Thin Small Outline Package, Type I (TSOP)	28T	C

35. Errata

35.1 ATmega8HVA

35.1.1 Rev. A

No known errata.

35.2 ATmega16HVA

35.2.1 Rev. A

No known errata.

36. Datasheet Revision History

36.1 Rev. 8024A – 04/08

1. Initial revision

Table of Contents

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature