Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 124 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
- Nonvolatile Program and Data Memories
  - 4K/8K Bytes of In-System Self-Programmable Flash (ATmega4HVD/8HVD)
  - 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 Cell Batteries
  - Short-circuit Protection (Discharge)
  - Over-current Protection (Charge and Discharge)
  - External Protection Input
  - High Voltage Outputs to Drive N-Channel Charge/Discharge FETs
  - Operation with 1 FET or 2 FETs supported
    Charge FET is optional
  - Battery authentication features (Available only under NDA)
- Peripheral Features
  - Two 8/16-bit Timer/Counters with Separate Prescaler and two output compare units
  - 10-bit ADC with One External Input
  - Two High-voltage open-drain I/O pins
  - Programmable Watchdog Timer
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable
  - Power-on Reset
  - On-chip Voltage Reference with built-in Temperature Sensor
  - On-chip Voltage Regulator
  - External and Internal Interrupt Sources
  - Sleep Modes:
    Idle, ADC Noise Reduction, Power-save, and Power-off
- Package
  - 18-pad DRDFN/ MLF
- Operating Voltage (VFET): 2.1 - 6.0V
- Operating Voltage (V_{CC}):2.0 - 2.4V
- Maximum Withstand Voltage (VFET): 12V
- Maximum Withstand Voltage (High-voltage pins): 5V
- Temperature Range: -20°C to 85°C
- Speed Grade: 1 - 4 MHz

8-bit AVR® Microcontroller with 4K/8K Bytes In-System Programmable Flash

ATmega4HVD
ATmega8HVD

Preliminary

8052B–AVR–09/08

1. Pin Configurations

Figure 1-1. Dual Row DFN/ MLF-pinout ATmega4HVD/8HVD.

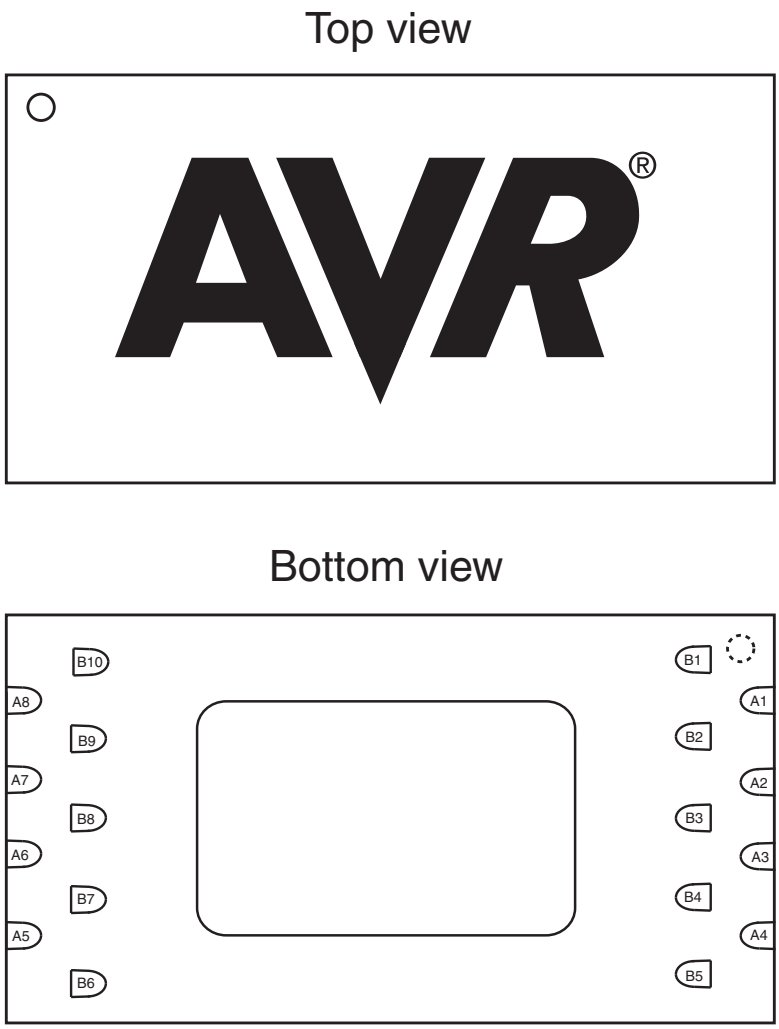


Table 1-1. Dual Row DFN/ MLF-pinout ATmega4HVD/8HVD.

	1	2	3	4	5	6	7	8	9	10
A	DNC	BATT	GND	PV1	PB1 (SCK/ SGND/T0)	DNC	VCC	PC1 (MOSI/INT1/ EXT_PROT)	-	-
B	OD	OC	VFET	VREG	NI	PB0 (ADC0)	PB2 (MISO/CKOUT/T1)	GND	PC0 (INT0/ICP0/XTAL)	$\overline{\text{RESET}}$

1.1 Pin Descriptions

1.1.1 VFET

Input to the internal voltage regulator.

1.1.2 VCC

Pin for connection of external decoupling capacitor. VCC is internally connected to the voltage regulator output VREG.

1.1.3 VREG

Output from the internal voltage regulator. Internally connected to VCC.

1.1.4 GND

Ground

1.1.5 Port B (PB2:PB0)

Port B is a low-voltage 3-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 ATmega4HVD/8HVD.

1.1.6 Port C (PC1:PC0)

Port C is a High-voltage open-drain 2 bit bi-directional I/O port. Port C also serves the functions of various special features of the ATmega4HVD/8HVD.

1.1.7 OC

High voltage output to drive Charge FET (optional).

1.1.8 OD

High voltage output to drive Discharge FET.

1.1.9 NI

Negative input from the battery protection resistor.

1.1.10 PV1

Input from battery cell to ADC.

1.1.11 BATT

Input for detecting when a charger is connected.

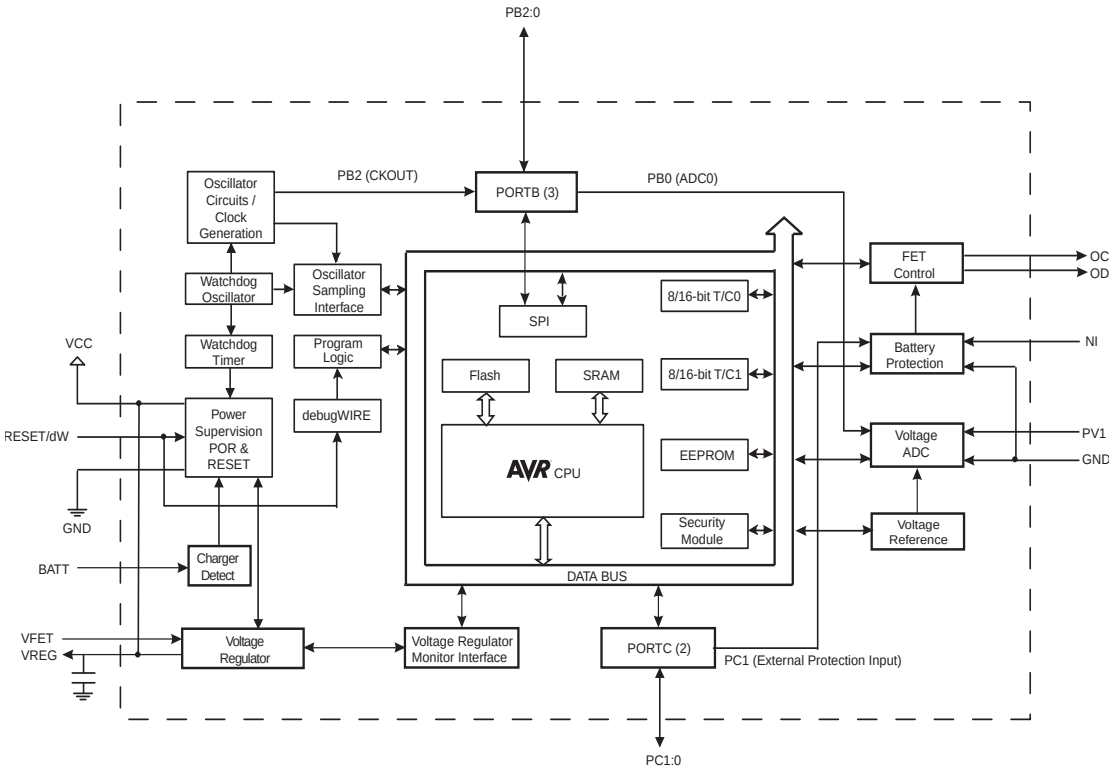
1.1.12 $\overline{\text{RESET}}$/dw

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. Shorter pulses are not guaranteed to generate a reset. This pin is also used as debugWIRE communication pin.

2. Overview

The ATmega4HVD/8HVD is a monitoring and protection circuit for 1-cell Li-ion applications with focus on high security/authentication, 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 External Protection Input can be used to implement other battery protection mechanisms using external components, e.g. protection against chargers with too high charge voltage can be easily implemented with a few low cost passive components. The feature set makes the ATmega4HVD/8HVD a key component in any system focusing on high security, battery protection, high system utilization and low cost.

Figure 2-1. Block Diagram



An integrated, low-dropout linear regulator that can handle input voltages as low as 2.1V, ensures that the stored energy can be fully exploited. The regulator capabilities, combined with a extremely low power consumption in the power saving modes, greatly enhances the cell energy utilization compared to existing solutions.

The chip utilizes Atmel's Deep Under-voltage Recovery (DUVR) mode that supports pre-charging of deeply discharged battery cells without using a separate Pre-charge FET. An enhanced start-up scheme allows the chip to operate correctly even with only Discharge FET connected. This makes it possible to further reduce system cost for applications that do not require Charge Over-current protection.

The ATmega4HVD/8HVD contains a 10-bit ADC for cell voltage measurements. The ADC is also used to monitor the on-chip temperature. Temperature is measured by the integrated Voltage Reference, which contains a built-in temperature sensor. ATmega4HVD/8HVD con-

tains 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 MCU includes 4K/8K bytes of In-System Programmable Flash with Self-programming capabilities, 256 bytes EEPROM, 512 bytes SRAM, 32 general purpose working registers, 4 general purpose I/O lines, debugWIRE for On-chip debugging and SPI for In-system Programming, two flexible Timer/Counters with Input Capture, internal and external interrupts, a 10-bit ADC for measuring the cell voltage and on-chip temperature, 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 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 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, by a conventional non-volatile memory programmer or by an On-chip Boot program running on the AVR core.

The ATmega4HVD/8HVD 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 ATmega4HVD/8HVD 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.

3. Resources

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

4. 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.

5. 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.

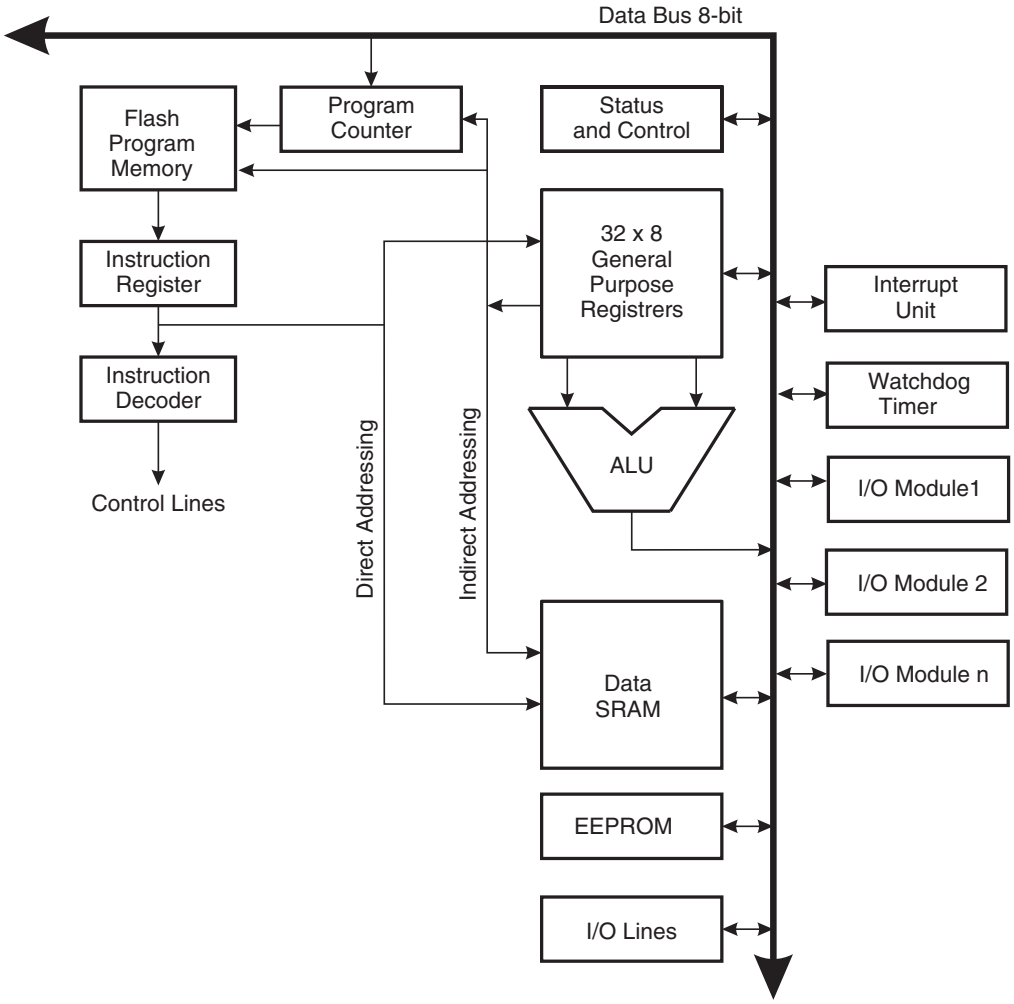
For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instru0ctions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

6. AVR CPU Core

6.1 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 6-1. 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 typical 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 ATmega4HVD/8HVD has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

6.2 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.

6.3 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.

6.3.1 SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
	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.

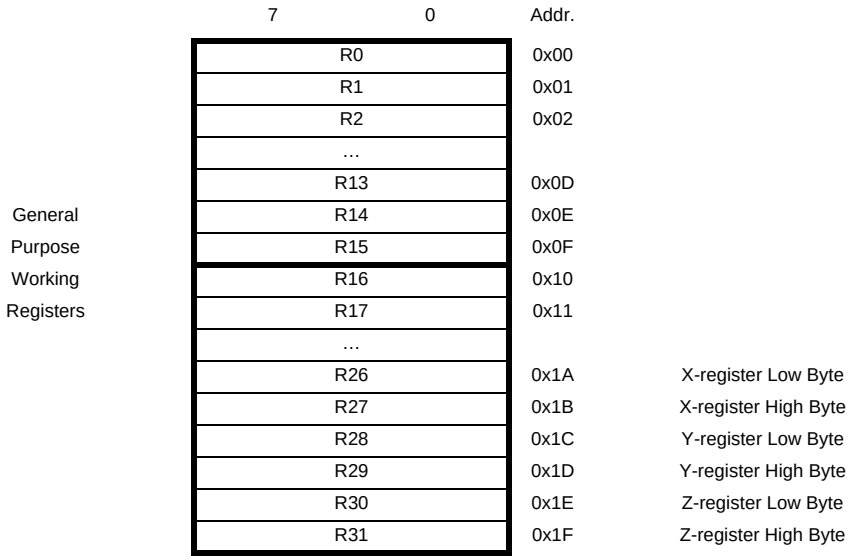
6.4 General Purpose Register File

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 6-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 6-2. AVR CPU General Purpose Working Registers



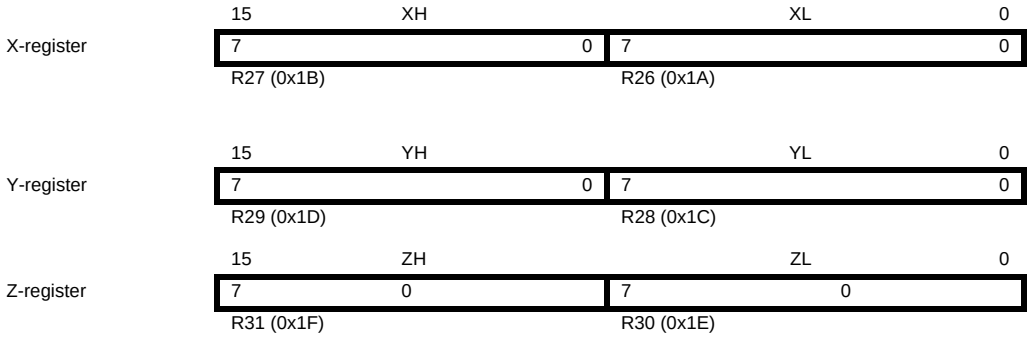
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 6-2, 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.

6.4.1 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 6-3.

Figure 6-3. 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).

6.5 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.

6.5.1 SPH and SPL – Stack pointer High and Low Register

Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	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	1	0	
	1	1	1	1	1	1	1	1	

6.6 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 6-4 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 6-4. The Parallel Instruction Fetches and Instruction Executions

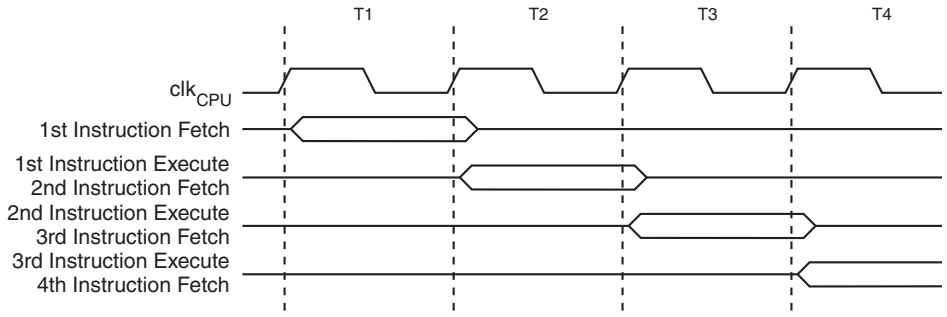
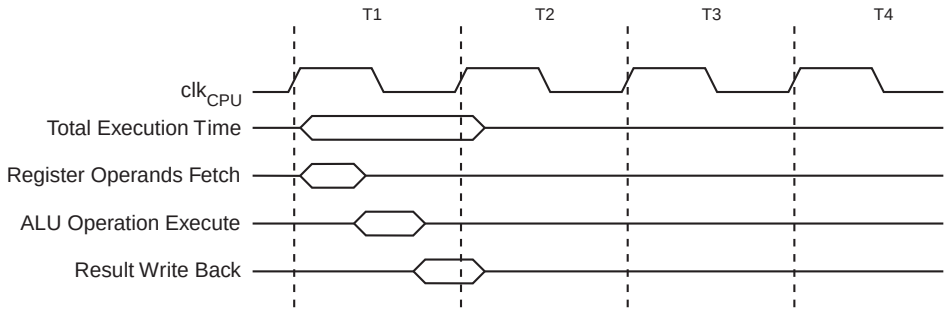


Figure 6-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 6-5. Single Cycle ALU Operation



6.7 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 51. 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, EEMWE   ; start EEPROM write
sbi EECR, EEWE
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<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
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) */
```

6.7.1 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.

7. AVR Memories

7.1 Overview

This section describes the different memories in the ATmega4HVD/8HVD. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega4HVD/8HVD features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

7.2 In-System Reprogrammable Flash Program Memory

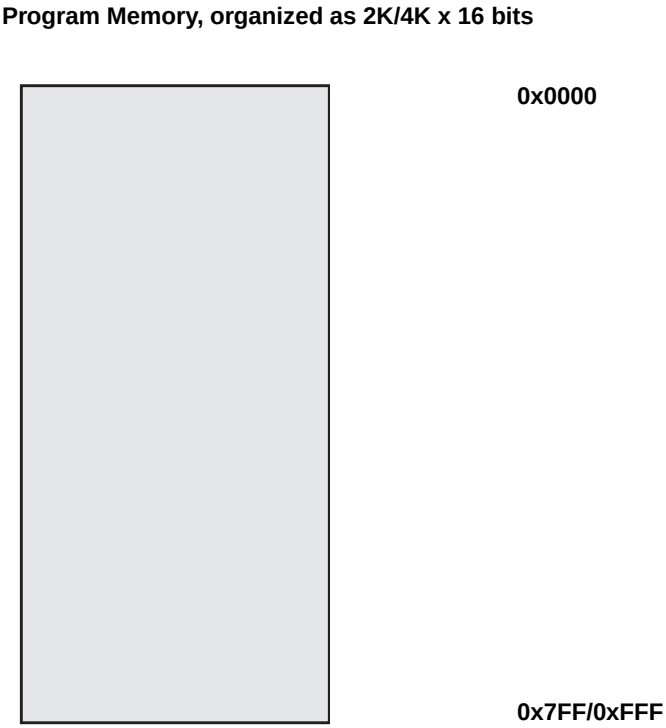
The ATmega4HVD/8HVD contains 4/8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 2K/4K x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega4HVD/8HVD Program Counter (PC) is 11/12 bits wide, thus addressing the 2K/4K program memory locations. ”Memory Programming” on page 129 contains a detailed description on Flash data serial downloading.

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 11.

Figure 7-1. Program Memory Map



7.3 SRAM Data Memory

Figure 7-2 shows how the ATmega4HVD/8HVD SRAM Memory is organized.

The ATmega4HVD/8HVD 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 512 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.

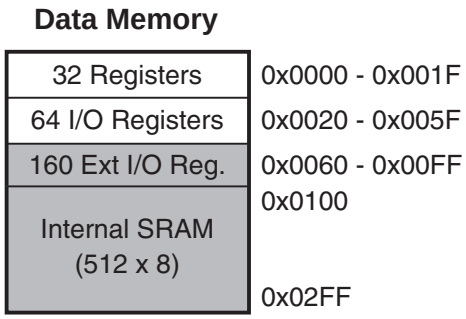
The direct addressing reaches the entire data space.

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 ATmega4HVD/8HVD are all accessible through all these addressing modes. The Register File is described in ”General Purpose Register File” on page 9.

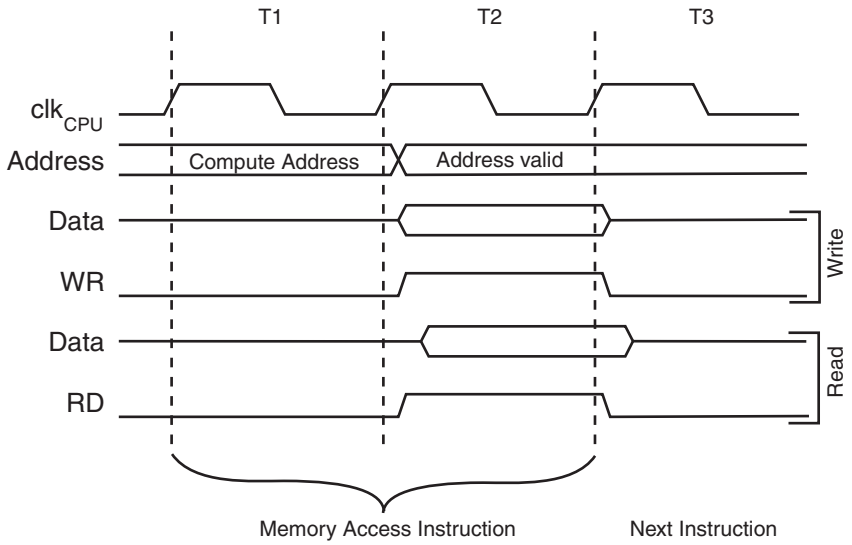
Figure 7-2. Data Memory Map



7.3.1 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 1.

Figure 1. On-chip Data SRAM Access Cycles



7.4 EEPROM Data Memory

The ATmega4HVD/8HVD 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 Serial and Parallel data downloading to the EEPROM, see page 131 and page 131 respectively.

7.4.1 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 7-1. 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.

7.5 I/O Memory

The I/O space definition of the ATmega4HVD/8HVD is shown in ”Register Summary” on page 151.

All ATmega4HVD/8HVD 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 instruc-

tions. 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 ATmega4HVD/8HVD 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, unlike most other AVRs, 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.

The ATmega4HVD/8HVD 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.

7.6 Register Description

7.6.1 EEARL – The EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
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 – EEARL 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 EEARL is undefined. A proper value must be written before the EEPROM may be accessed.

7.6.2 EEDR – The EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
	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 EEARL Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEARL.

7.6.3 EECR – The EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	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 ATmega4HVD/8HVD 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 7-1. 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 7-1. 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 3 and 4 is not essential):

1. Wait until EEPE becomes zero.
2. Write new EEPROM address to EEARL (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 timeout. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEARL 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.

Note that a BLOD reset will abort any ongoing write operation and invalidate the result.

- **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 EEARL 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 EEARL Register.

The Calibrated Fast RC Oscillator is used to time the EEPROM access and the programing time will therefore depend on the calibrated oscillator frequency. Table 7-2 lists the typical programming time for EEPROM access from the CPU.

Table 7-2. 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,EEWE
  rjmp EEPROM_write
  ; Set up address (r17) in address register
  out  EEARL, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

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

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,EEWE
  rjmp EEPROM_read
  ; Set up address (r17) in address register
  out  EEARL, 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<<EEWE))
    ;
  /* Set up address register */
  EEARL = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

7.6.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
	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	

7.6.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
	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	

7.6.6 GPIOR0 –General Purpose I/O Register 0

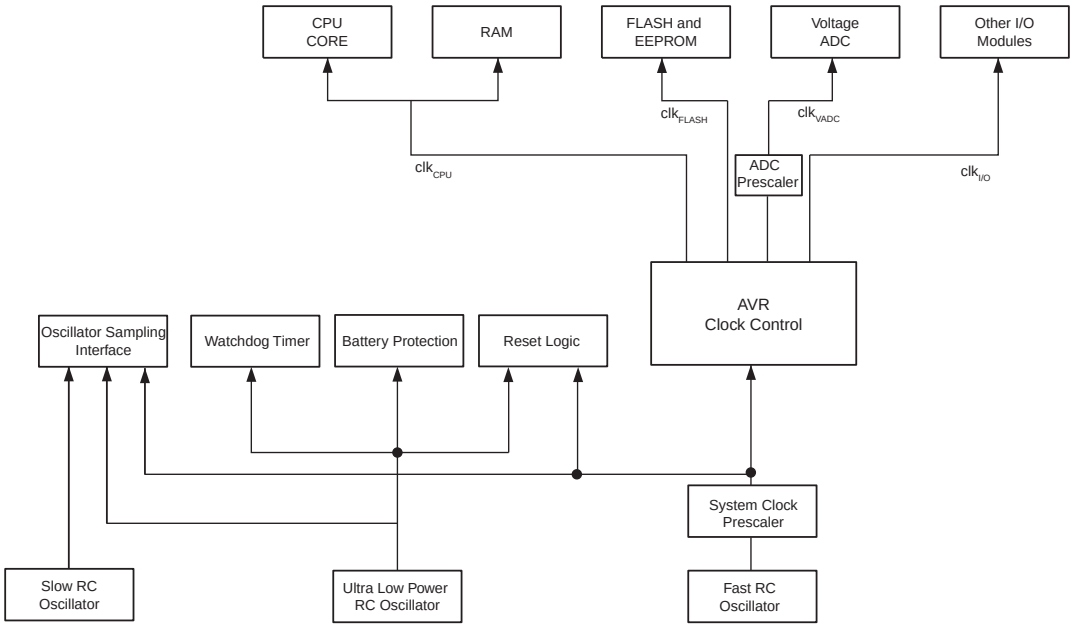
Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

8. System Clock and Clock Options

8.1 Clock Systems and their Distribution

Figure 8-1 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 32. The clock systems are detailed below.

Figure 8-1. Clock Distribution



8.1.1 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.

8.1.2 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.

8.1.3 Flash Clock – clk_{FLASH}

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

8.1.4 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. The ADC is automatically prescaled according to the System Clock Prescaler's setting to provide a fixed clock frequency to the

ADC. The dedicated ADC clock allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

8.1.5 Watchdog Timer and Battery Protection Clock

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

8.2 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.

8.3 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 85°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 85°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 124. Note that the frequency of the system clock is given by the ”System Clock Prescaler” on page 25.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 8-1 on page 23.

Table 8-1. Start-up times for the calibrated Fast RC Oscillator clock 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” on page 24. See Table 8-2 on page 25 and ”Electrical Characteristics” on page 142

8.4 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 27 for details.

To provide good accuracy when used as a timing reference, the Slow RC Oscillator has calibration bytes stored in the signature address space, refer to section ”Reading the Signature Row from Software” on page 124 for details. The actual clock period of the Slow RC Oscillator in µs as a function of temperature is given by:

$$\text{Clock period} = \frac{\text{Slow RC Word}}{1024} \cdot (\text{1-Slow RC Temp Prediction}(T - T_{HOT}))$$

where T is the die temperature in Kelvin and T_{HOT} is the calibration temperature stored in the signature row. The die temperature can be found using the ADC, refer to ”ADC - Analog-to-Digital Converter” on page 90 for details.

8.5 Ultra Low Power RC Oscillator

The Ultra Low Power RC Oscillator (ULP Oscillator) provides a 128 kHz clock (typical value, refer to section ”Electrical Characteristics” on page 142). There are two alternative methods for determining the actual clock period of the ULP Oscillator:

1. To determine the actual clock period as a function of die temperature, the Oscillator Sampling Interface should be used. Refer to section ”OSI – Oscillator Sampling Interface” on page 27 for details.
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 calibration byte ULP_RC_FRQ stored in the signature address space, refer to section ”Reading the Signature Row from Software” on page 124 for details.

8.6 CPU, I/O, Flash, and ADC Clock

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

8.7 Watchdog Timer and Battery Protection

The clock source for the Watchdog Timer and Battery Protection is the Ultra Low Power RC Oscillator. The Oscillator is automatically enabled in all operational modes. It is also enabled during reset.

8.8 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

8-1 on page 23. The number of Ultra Low Power RC Oscillator cycles used for each time-out is shown in Table 8-2.

Table 8-2. 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 24 for details.

8.9 Clock Output

The CPU clock divided by 2 can be output to the PB2 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.

8.10 System Clock Prescaler

The ATmega4HVD/8HVD 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 29, 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 8-4 on page 30.

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.

8.11 ADC Clock Prescaler

The ADC clock will be automatically prescaled relative to the System Clock Prescaler settings, see ”System Clock Prescaler” on page 25. Depending on the Clock Prescale Select bits, CLKPS1..0, the ADC clock, clk_{ADC}, will be prescaled by 24, 12 or 6 as shown in Table 8-3 on page 26.

Table 8-3. ADC Clock Prescaling[1]

CLKPS1	CLKPS0	ADC Division Factor
0	0	Reserved
0	1	24
1	0	12
1	1	6

Note: 1. When changing Prescaler value, the ADC Prescaler will automatically change frequency of the ADC. The result of the ongoing conversion will be invalid.

8.12 OSI – Oscillator Sampling Interface

8.12.1 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**

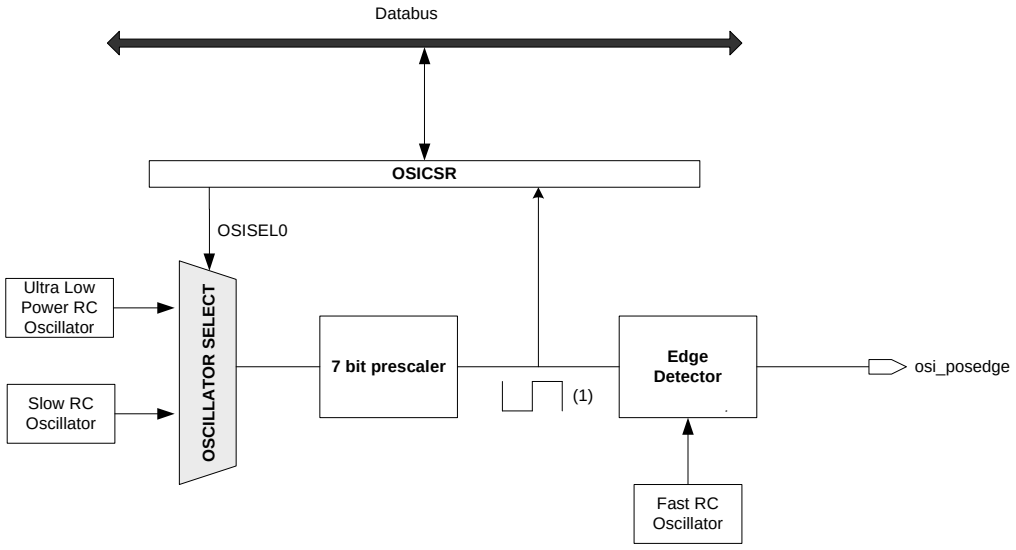
8.12.2 Overview

The Oscillator Sampling Interface (OSI) enables sampling of the Slow RC and Ultra Low Power RC (ULP) oscillators in ATmega4HVD/8HVD. 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 8-2.

Figure 8-2. 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 74 for details on how to enable the Input Capture function.

8.12.3 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.

The Slow RC frequency at 70°C and the Slow RC temperature coefficient are 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 24 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 70°C 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 ADC, refer to section TBD for details.

8.13 Register Description

8.13.1 FOSCCAL – Fast RC Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
	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 85°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 27 for details.

8.13.2 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	-–	–	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 PB2 pin.

8.13.3 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
	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 8-4 on page 30. Note that writing to the System Clock Prescaler Select bits will abort any ongoing ADC conversion.

Table 8-4. 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

8.13.4 OSICSR – Oscillator Sampling Interface Control and Status Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	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 ATmega4HVD/8HVD and will always read as zero.

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

Table 8-5. 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.
2. Enabling the OSI module and selecting Slow RC Oscillator as input source is the only way to enable the Slow RC Oscillator. The Slow RC Oscillator will not run in any other modes.

9. 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.

9.1 Sleep Modes

Figure 8-1 on page 22 presents the different clock systems in the ATmega4HVD/8HVD, and their distribution. The figure is helpful in selecting an appropriate sleep mode. The different sleep modes and their wake-up sources are summarized in Table 9-1, and Figure 9-1 on page 33 shows a sleep mode state diagram.

Table 9-1. Wake-up Sources for Sleep Modes

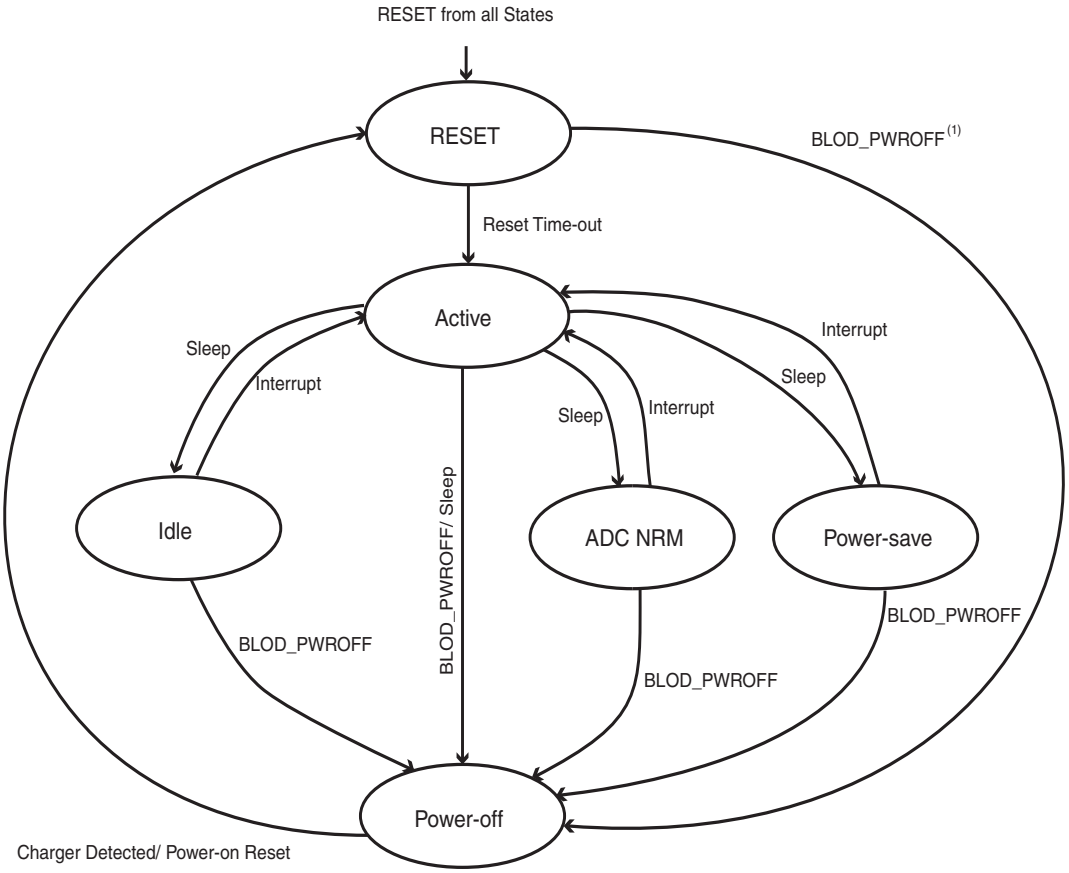
Mode	Wake-up sources							
	Battery Protection Interrupts	External Interrupts	WDT	EEPROM Ready	VREGMON	ADC	Other I/O	Charger Detect(1)
Idle	X	X	X	X	X	X	X	
ADC Noise Reduction	X	X	X	X	X	X		
Power-save	X	X	X					
Power-off								X

Note: 1. Discharge FET must be switched off for Charger Detect to be active.

To enter any of the available sleep modes, the SE bit in SMCR 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 (Idle, ADC NRM, Power-save or Power-off) will be activated by the SLEEP instruction. See Table 9-3 on page 36 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. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 9-1. Sleep Mode State Diagram



Note: 1. For details on BLOD Power-off refer to "Black-out Detection" on page 40.

Table 9-2. Active modules in different Sleep Modes

	Mode				
Module	**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)			
CPU	X				
Flash	X				
Timer/ Counter n	X	X			
ADC	X	X	X		
External Interrupts	X	X	X	X	
CBP	X	X	X	X	
WDT	X	X	X	X	

Table 9-2. Active modules in different Sleep Modes (Continued)

Module	Mode				
	Active	Idle	ADC Noise Reduction	Power-save	Power-off
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 128 ms after C-FET/D-FET has been enabled.
3. Runs only when OSI is enabled and RCOSC_SLOW is selected as source for OSI.

9.2 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.

9.3 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 ADC, Watchdog Timer (WDT), 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 ADC, enabling higher resolution measurements.

9.4 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), Current Battery Protection (CBP) and the Ultra Low Power RC Oscillator (RCOSC_ULP) 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).

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 Interrupt” on page 53 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 23.

9.5 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.

Notes:
1. 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.
2. Before entering power-off mode, make sure that no EEPROM write sequence is ongoing. Any ongoing write operation will be aborted when Power-off sleep mode is entered.

9.6 Power Reduction Register

The Power Reduction Register (PRR), see ”PRR0 – Power Reduction Register 0” on page 37, 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.

9.7 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.

9.7.1 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 45 for details on how to configure the Watchdog Timer.

9.7.2 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 65 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 VCC/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 VCC/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 98 for details.

9.7.3 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.

9.7.4 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 109. 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.

9.7.5 ADC

If enabled, the ADC will consume power independent of sleep mode. To save power, the ADC should be disabled when not used, and before entering Power-save sleep mode. See ”ADC - Analog-to-Digital Converter” on page 90 for details on ADC operation. When PB0 is used as ADC0, the digital input buffer of this pin should be disabled by setting the PB0DID bit in the DIDR0 register.

9.7.6 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.

9.8 Register Description

9.8.1 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	
	–	–	–	–	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 ATmega4HVD/8HVD, 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 9-3.

Table 9-3. 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

Table 9-3. Sleep Mode Select (Continued)

SM2	SM1	SM0	Sleep Mode
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.

9.8.2 PRR0 – Power Reduction Register 0

Bit	7	6	5	4	3	2	1	0	
	-	-	PRVRM	-	PRSPI	PRTIM1	PRTIM0	PRADC	PRR0
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 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 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 4 - Res: Reserved bits**

This bit is reserved for future use. For compatibility with future devices, this bit must be written to zero when PRR0 is written.

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

Writing a logic one to this bit shuts down the Serial Periperal Interface by stopping the clock to the module.

- **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 - PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. Before writing the PRADC bit, make sure that the ADEN bit is cleared to minimize the power consumption.

Note: ADC control registers can be updated even if the PRADC bit is set.

10. System Control and Reset

10.1 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 Figure 10-1 on page 39 shows the reset logic. ”System and Reset Characteristics” on page 144 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.

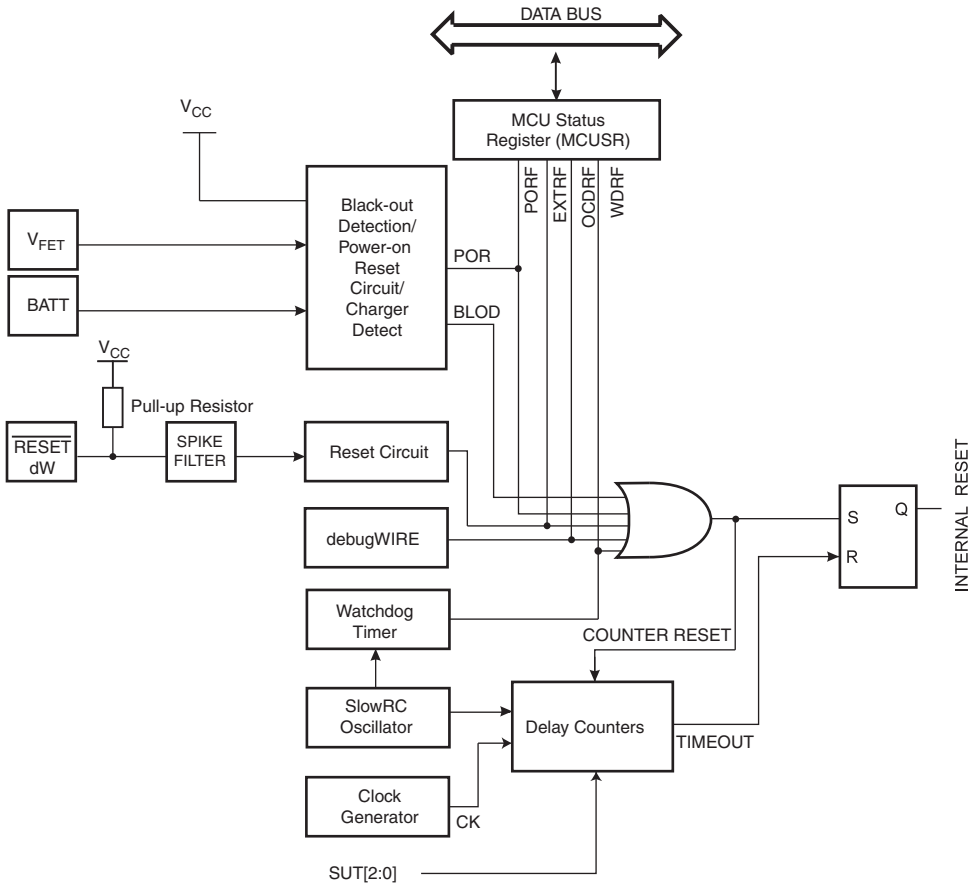
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the voltage regulator to reach a stable level before normal operation starts. The timeout period of the delay counter is defined by the user through the SUT Fuses. The different selections for the delay period are presented in ”Clock Sources” on page 23.

10.2 Reset Sources

The ATmega4HVD/8HVD has these reset sources:

- 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.
- Black-out Reset. The MCU is reset when V_{REG} is below the Black-out Reset Threshold, V_{BLOT}. See “Black-out Detection” on page 40.
- debugWIRE. In On-chip Debug mode, the debugWIRE resets the MCU when giving the Reset command.

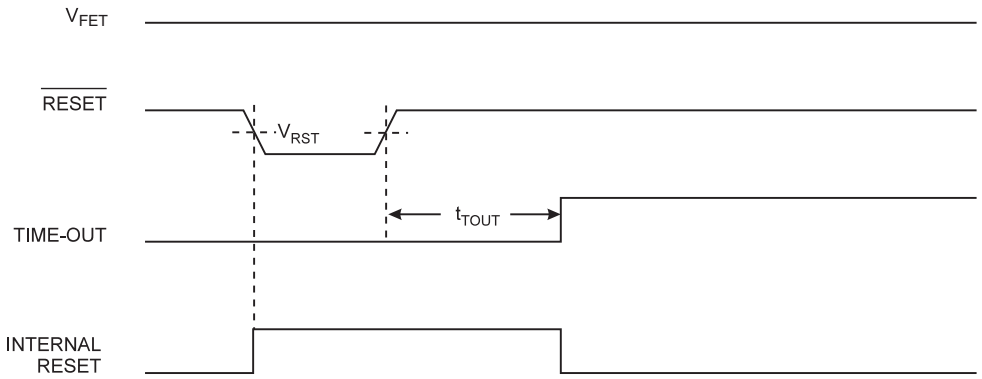
Figure 10-1. Reset Logic



10.3 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 "System and Reset Characteristics" on page 144) 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 timeout period, t_{TOUT}, has expired.

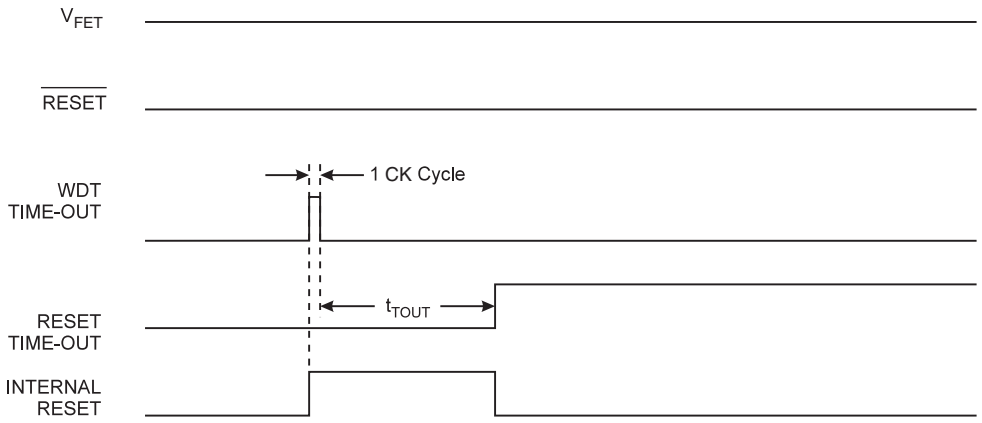
Figure 10-2. External Reset During Operation



10.4 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 timeout period t_{TOUT}. Refer to page 38 for details on operation of the Watchdog Timer.

Figure 10-3. Watchdog Reset During Operation



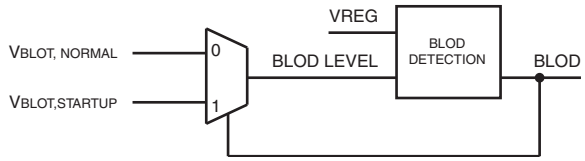
10.5 Black-out Detection

ATmega4HVD/8HVD has an on-chip Black-out Detection (BLOD) circuit for monitoring the VREG level during operation by comparing it to a trigger level defined by hardware.

The ATmega4HVD/8HVD has two detection levels and two application areas for BLOD, see "System and Reset Characteristics" on page 144. One detection level ($V_{BLOT, START-UP}$) is used to ensure that the voltage on VFET is sufficient to operate the voltage regulator within its specifications when the chip starts up. The other detection level is used during normal operation ($V_{BLOT, NORMAL}$) to determine if VREG drops below a voltage where correct operation cannot be

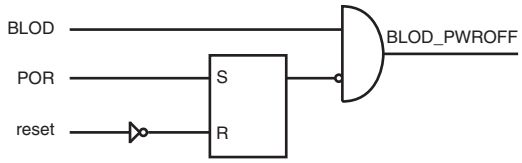
guaranteed and the chip should be forced into Power-off mode. The algorithm used for switching between the two V_{BLOT} levels is illustrated Figure 10-4 on page 41. As long as BLOD is set, the $V_{BLOT, START-UP}$ level will always be selected.

Figure 10-4. BLOD levels switching



Notice that during the Power-On Reset start-up sequence, a Black-out detection will only generate a normal reset. The chip will not enter Power-off in this case. This is illustrated in Figure 10-5 on page 41. See TBD for details on Power-on Reset and start-up sequence.

Figure 10-5. BLOD detection with POR



In normal operation, when V_{REG} decreases to a value below the trigger level, the Black-out Reset is immediately activated. After a fixed delay of $T_{BLODTOUT}$ the chip will enter Power-off mode, see Figure 10-6 on page 41 and "System and Reset Characteristics" on page 144. Any ongoing operations, including EEPROM write sequences that were started while V_{REG} was above V_{BLOD}, will be aborted. The result of an ongoing EEPROM write operation will be invalid. A charger must be connected to start up the chip from Power-off.

The BLOD circuit will only detect a drop in V_{REG} if the voltage stays below the trigger level for longer than t_{BLOD} given in "System and Reset Characteristics" on page 144.

Figure 10-6. Black-out Reset During Operation

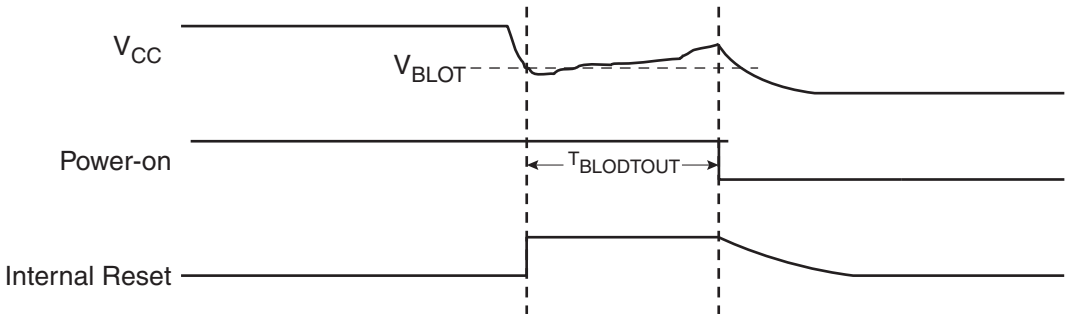
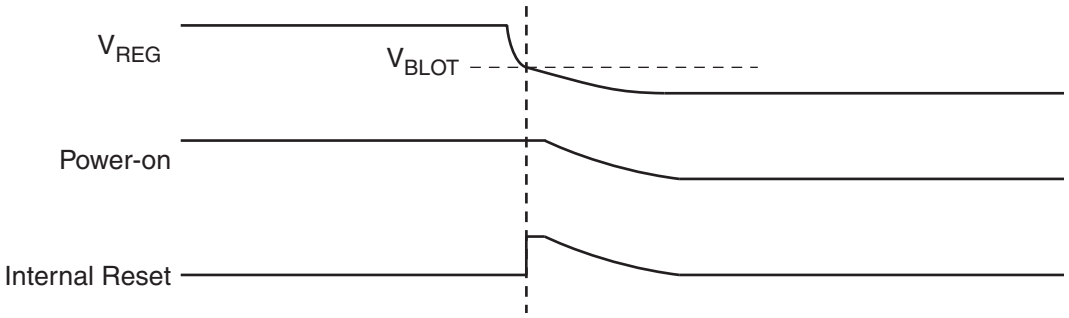


Figure 10-7. Black-out Reset with high current consumption at V_{REG}



10.6 ATmega4HVD/8HVD Start-up Sequence

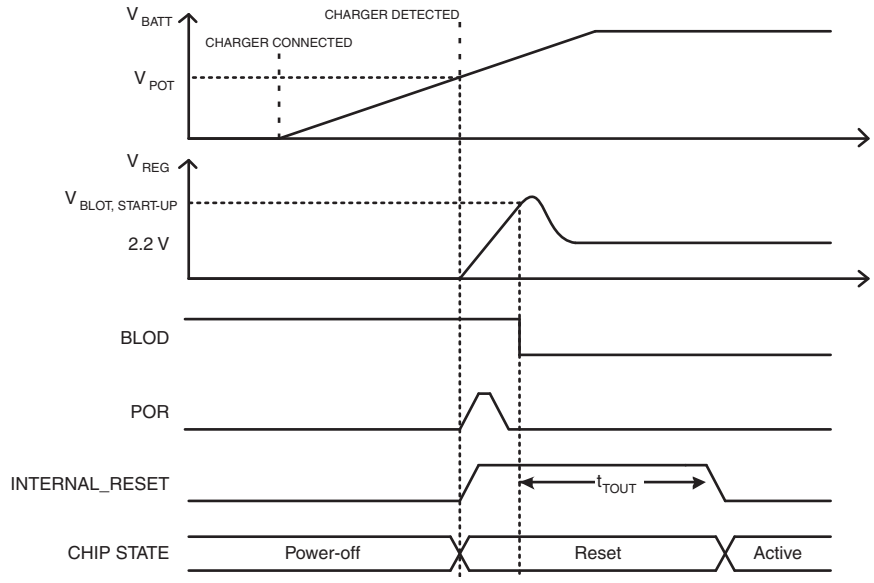
The Voltage Regulator will not start until it is enabled by the Charger Detect module. Before this happens the chip will be in Power-off mode and only the Charger Detect module is enabled. In order for the Charger Detect module to enable the Voltage Regulator, the V_{BATT} voltage must exceed the Power-On Threshold, V_{POT}. When the voltage at V_{BATT} exceeds V_{POT}, the Voltage Regulator starts up and a Power-On Reset forces the chip into RESET state.

10.6.1 Start-up with one FET connected

During start-up with only one FET (D-FET), cell charging through the body diode of the Discharge FET will start immediately when a charger is connected, even if the chip has not started yet. The voltage on the BATT pin equals the cell voltage (VFET) plus a diode drop during the complete start-up sequence, and will hence increase as the cell is being charged.

A typical start-up sequence for the application is illustrated in Figure 10-8 on page 42.

Figure 10-8. Powering up ATmega4HVD/8HVD (1-FET example)



When V_{BATT} reaches the power-on threshold V_{POT}, the voltage regulator starts up. This causes VCC to increase rapidly while CREG is being charged. At the same time the digital part of the

chip is powered and an internal Power-on Reset (POR) is generated. During the initial start-up when a valid reference for the voltage regulator is missing, VCC is driven as close as possible to VFET. Voltage regulation will only start when VCC has reached $V_{BLOT, start-up}$, which represents the voltage level that guarantees proper start-up conditions for the voltage regulator.

Even though the voltage regulator has started up and the digital part is powered, the chip is kept in RESET state until VCC exceeds $V_{BLOT, start-up}$. Once the condition VCC > $V_{BLOT, start-up}$ is met, BLOD will be released and the Reset Delay counter starts counting. VREG will now be regulated to its nominal value. The Reset Delay counter makes sure that the chip is continuously kept in RESET state internally for a time corresponding to the start-up time selected by the SUT fuses (time indicated as t_{TOUT} in Figure 10-8 on page 42). Note that as a consequence of this start-up scheme, the start-up period must be measured from BLOD is released, and not from the time that a charger is connected. The time from a charger is detected until BLOD is released may be significant in the one-FET application if the V_{POT} level is at its minimum and BLOD level is at its maximum.

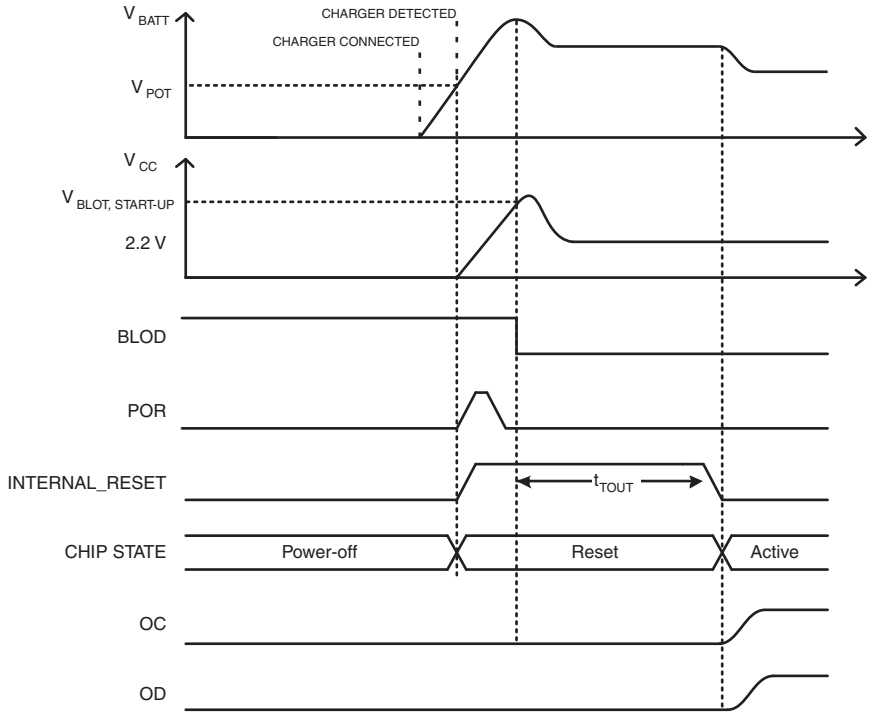
During start-up VCC may exceed its nominal value of operation for a short time period. This overshoot will typically occur when $V_{BLOT, START-UP}$ is at its maximum value since the regulator output follows VFET until the regulator enters normal mode. However, this situation will only occur during start-up while the chip is kept in RESET state, and will not occur in normal operation.

10.6.2 Start-up with two FETs connected

For two FET applications, the VFET node is high-impedant as long as C-FET is switched off. This means that the VFET voltage and hence also the BATT voltages increase rapidly once a charger is connected. The charger will be detected almost immediately and the voltage regulator is switched on.

A typical start-up sequence for the application is illustrated in Figure 10-9 on page 44.

Figure 10-9. Powering up ATmega4HVD/8HVD (2-FET example)



During the initial start-up when a valid reference for the voltage regulator is missing, VCC is driven as close as possible to VFET. Voltage regulation will only start when VCC has reached VBLOT, start-up, which represents the voltage level that guarantees proper start-up conditions for the voltage regulator.

Even though the voltage regulator has started up and the digital part is powered, the chip is kept in the RESET state and cell charging is disabled until VCC exceeds $V_{BLOT, start-up}$. The time from a charger is connected until this happens is very short in the two- FET application (actual timing depends on the CREG value). Once the condition VCC > $V_{BLOT, start-up\ is}$ met, BLOD is released and the Reset Delay counter starts counting. For details, see ”DUVR – Deep Under-Voltage Recovery Mode operation” on page 117.

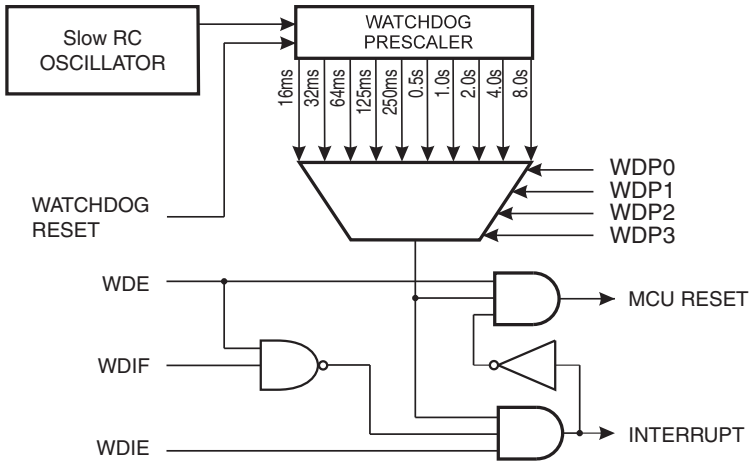
VCC will now be regulated to its nominal value. The Reset Delay counter makes sure that the chip is continuously kept in RESET state internally for a time corresponding to the start-up time selected by the SUT fuses (time indicated as t_{TOUT} in Figure 10-9 on page 44).

10.7 Watchdog Timer

10.7.1 Features

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

Figure 10-10. Watchdog Timer



ATmega4HVD/8HVD has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of the Slow RC Oscillator that runs at 131 kHz (typical value, see "Electrical Characteristics" on page 142). The WDT gives an interrupt or a system reset when the counter reaches a given timeout value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the timeout value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

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 timeout 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 timeout
  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 timeout */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

Note: 1. See "About Code Examples" on page 5.

Note: If the Watchdog is accidentally enabled, for example by a runaway pointer or Black-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 timeout 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 timeout 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(timeout) 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(timeout) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note: 1. See "About Code Examples" on page 5.

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 timeout when switching to a shorter timeout period.

10.8 Register Description

10.8.1 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	
	–	–	–	OCDRF	WDRF	–	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	(1)	0	(1)	(1)	

- **Bits 7:4, 2 – Res: Reserved Bits**

These bits are reserved, and will always read as zero.

- **Bit 4 – OCDRF: OCD Reset Flag**

This bit is set if a debugWIRE Reset occurs. This 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 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.

10.8.2 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 timeout 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 Timeout 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 timeout in the Watchdog Timer occurs.

If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first timeout 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 timeout, a System Reset will be applied.

Table 10-1. Watchdog Timer Configuration

WDTON[1]	WDE	WDIE	Mode	Action on Timeout
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 10-2.

- **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 and WDE must be written to one. Within the next four clock cycles, write the WDE and WDP bits as desired, and the WDCE bit cleared.

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

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

When the WDE bit is written to logic one, the Watchdog Timer is enabled, and if the WDE bit is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one, refer to the WDCE bit description.

If the WDTON fuse is programmed, it is not possible to disable the Watchdog Timer. Furthermore, 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 10-2.

Table 10-2. Watchdog Timer Prescale Select (Typical Timeout at V_{CC} = 2.2V)

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Timeout			Units
					Min.	Typ.	Max.	
0	0	0	0	2K cycles	12	16	23	ms
0	0	0	1	4K cycles	24	31	45	ms
0	0	1	0	8K cycles	48	63	90	ms
0	0	1	1	16K cycles	0.096	0.125	0.180	s
0	1	0	0	32K cycles	0.19	0.25	0.36	s
0	1	0	1	64K cycles	0.4	0.5	0.7	s
0	1	1	0	128K cycles	0.8	1.0	1.4	s
0	1	1	1	256K cycles	1.5	2.0	2.9	s
1	0	0	0	512K cycles	3.1	4.0	5.8	s
1	0	0	1	1024K cycles	6.1	8.0	11.5	s
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					

11. Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega4HVD/8HVD. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 11.

11.1 Interrupt Vectors in ATmega4HVD/8HVD

Table 1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Black-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	WDT	Watchdog Time-out Interrupt
7	0x0006	TIMER1 IC	Timer 1 Input Capture
8	0x0007	TIMER1 COMPA	Timer 1 Compare Match A
9	0x0008	TIMER1 COMPB	Timer 1 Compare Match B
10	0x0009	TIMER1 OF	Timer 1 Overflow
11	0x000A	TIMER0 IC	Timer 0 Input Capture
12	0x000B	TIMER0 COMPA	Timer 0 Compare Match A
13	0x000C	TIMER0 COMPB	Timer 0 Compare Match B
14	0x000D	TIMER0 OF	Timer 0 Overflow
15	0x000E	ADC COMPLETE	ADC Conversion Complete
16	0x000F	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 the locations of the Interrupt Vectors.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega4HVD/8HVD 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       WDT                      ; Watchdog Time-out Interrupt
0x0006               rjmp       TIM1_IC                  ; Timer1 Input Capture Handler
0x0007               rjmp       TIM1_COMPA               ; Timer1 Compare A Handler
0x0008               rjmp       TIM1_COMPB               ; Timer1 Compare B Handler
0x0009               rjmp       TIM1_OVF                 ; Timer1 Overflow Handler
0x000A               rjmp       TIM0_IC                  ; Timer0 Input Capture Handler
0x000B               rjmp       TIM0_COMPA               ; Timer0 Compare A Handler
0x000C               rjmp       TIM0_COMPB               ; Timer0 Compare B Handler
0x000D               rjmp       TIM0_OVF                 ; Timer0 Overflow Handler
0x000E               rjmp       ADC                      ; ADC Conversion Complete Handler
0x000F               rjmp       EE_READY                 ; EEPROM Ready
;
0x000F     RESET:    ldi        r16, high(RAMEND)        ; Main program start
0x0010               out        SPH,r16                  ; Set Stack Pointer to top of RAM
0x0010               ldi        r16, low(RAMEND)
0x0012               out        SPL,r16
0x0013               sei                                 ; Enable interrupts
0x0014               <instr>    xxx
0x0015     ...       ...        ...
;
```

12. External Interrupt

The External Interrupts are triggered by the INT1:0 pins. Observe that, if enabled, the interrupt will trigger even if the INT1:0 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 ”EICRA – External Interrupt Control Register A” on page 53. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. A interrupt is detected asynchronously. This implies that the interrupt 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 22. If the level is sampled twice by the Slow RC 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.

12.1 Register Description

12.1.1 EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	RR/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 in the ATmega4HVD/8HVD, and will always read as zero.

- **Bits 3:0 – ISC11, ISC10 - ISC01, ISC00: External Interrupt 1 - 0 Sense Control Bits**

The External Interrupts 1- 0 are activated by the external pins INT1:0 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 interrupt is defined in Table 12-1 on page 54. Edges on INT1:0 are registered asynchronously. Pulses on the INT1:0 pins wider than the minimum pulse width given in ”External Interrupts Characteristics” on page 144 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 INTnby clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISCn bit can

be changed. Finally, the INTn interrupt flags 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 12-1. 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 = 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.

12.1.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	INT1	INT0	EIMSK
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 bits in the ATmega4HVD/8HVD, and will always read as zero.

- **Bit 1:0 – INT1:0: External Interrupt Request 1:0 Enable**

When the INT1 - 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 this pin will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

12.1.3 EIFR – External Interrupt Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – RES: Reserved Bits**

These bits are reserved bits ins the ATmega4HVD/8HVD, and will always read as zero.

- **Bits 1:0 – INTF1:0: External Interrupt Flag 1:0**

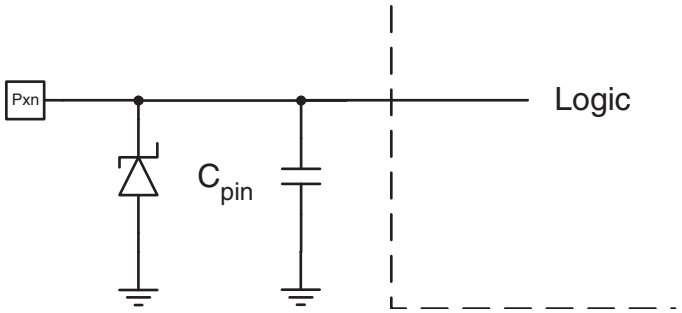
When an edge or logic change on the INT1:0 pin triggers an interrupt request, INTF1:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT1:0 in EIMSK, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the

interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1:0 are configured as level interrupt. Note that when entering sleep mode with the INT1:0 interrupt disabled, the input buffer on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF1:0 flag. See ”Digital Input Enable and Sleep Modes” on page 65 for more information.

13. High Voltage I/O Ports

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 13-1. See ”Electrical Characteristics” on page 142 for a complete list of parameters.

Figure 13-1. High Voltage I/O Pin Equivalent Schematic



Note: 1. See Figure 13-2 on page 57 for details.

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 60.

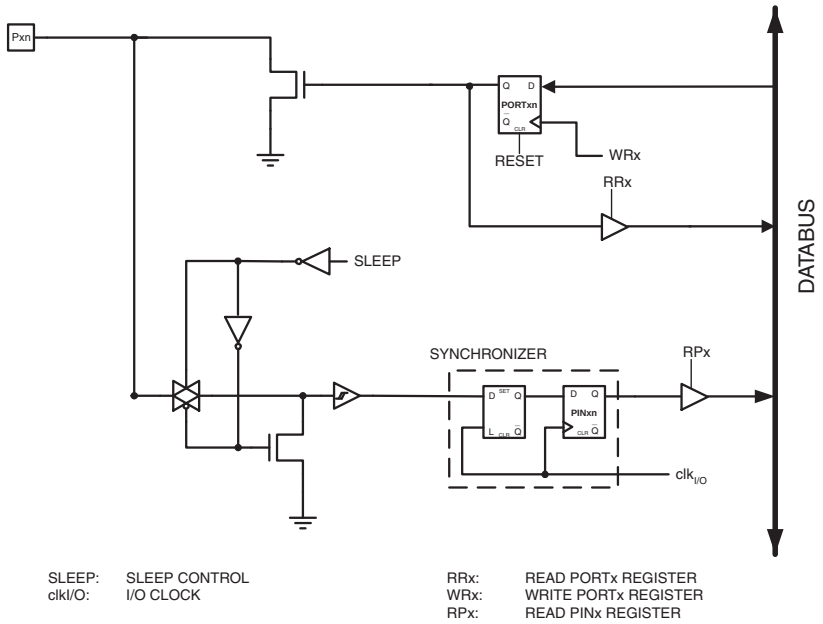
One I/O Memory address location is allocated for each high voltage port, the Data Register – PORTx. The Data Register is read/write.

Using the I/O port as General Digital Output is described in ”High Voltage Ports as General Digital Outputs” on page 57.

13.1 High Voltage Ports as General Digital Outputs

The high voltage ports are high voltage tolerant open collector output ports. Figure 13-2 shows a functional description of one output port pin, here generically called Pxn.

Figure 13-2. 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.

13.1.1 Configuring the Pin

Each port pin consist of two register bits: PORTxn and PINxn. As shown in ”Register Description” on page 60, 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.

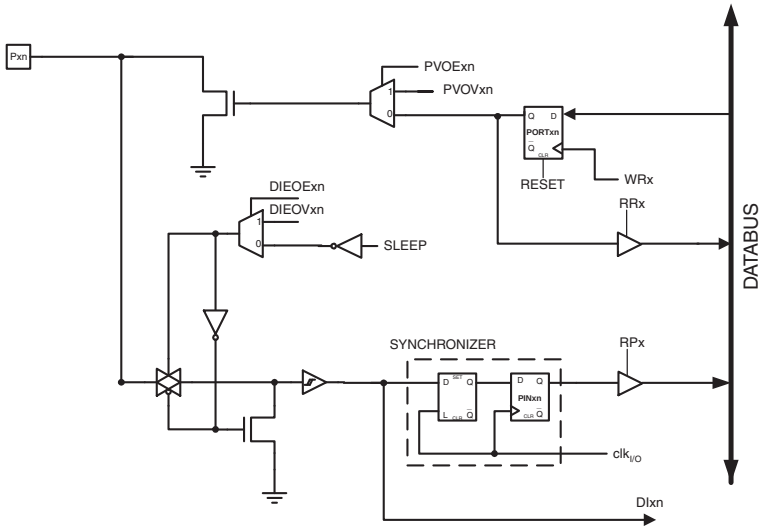
13.1.2 Reading the Pin

The port pin can be read throughthe PINxn Register bit. As shown in Figure 13-2, 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.

13.2 Alternate Port Functions

The High Voltage I/O has an alternate port function in addition to being general digital I/O. Figure 13-3 shows how the port pin control signals from the simplified Figure 13-2 can be overridden by alternate functions.

Figure 13-3. 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 13-1 summarizes the function of the overriding signals. The pin and port indexes from Figure 13-3 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 13-1. 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.

13.2.1 Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 13-2.

Table 13-2. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC0	INT0/ICP0/XTAL (External Interrupt 0, Timer/Counter 0 input Capture Trigger or External Clock)
PC1	MOSI/INT1/EXT_PROT (SPI BUS Serial Data Input, External Interrupt 1, External Protection Input)

The alternate pin configuration is as follows:

- **INT0/ICP0/XTAL - Port C, Bit 0**

INT0, External Interrupt 0: When INT0 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 "EICRA – External Interrupt Control Register A" on page 53 - 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.

XTAL, External Clock: When the CKSEL fuse is programmed, PC0 is used as clock source instead of the Internal RC oscillator (For test purposes only).

- **MOSI/INT1/EXT_PROT - Port C, Bit 1**

MOSI, Slave Data Input pin for SPI Programming.

INT1, External Interrupt 1: When INT1 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 "EICRA – External Interrupt Control Register A" on page 53 - 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.

EXT_PROT, External Protection Input: When the EPID bit in the BPCR Register is cleared, the External Protection Input functionality is enabled. Note that this port overriding is default enabled.

Table 13-3 relates the alternate functions of Port C to the overriding signals shown in Figure 13-3 on page 58.

Table 13-3. Overriding Signals for Alternate Functions in PC1:0

Signal Name	PC1/MOSI/INT1/EXT_PROT	PC0/INT0/ICP0/XTAL
PVOE	0	0
PVOV	0	0
DIEOE	INT Enable + $\overline{\text{EPID}}$	INT Enable + $\overline{\text{CKSEL}}$
DIEOV	1	1
DI	INT1/EXT_PROT	INT0/ICP0/XTAL INPUT

13.3 Register Description

13.3.1 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	PORTC1	PORTC0	PORTC
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.3.2 PINC – Port C Input Pins Address

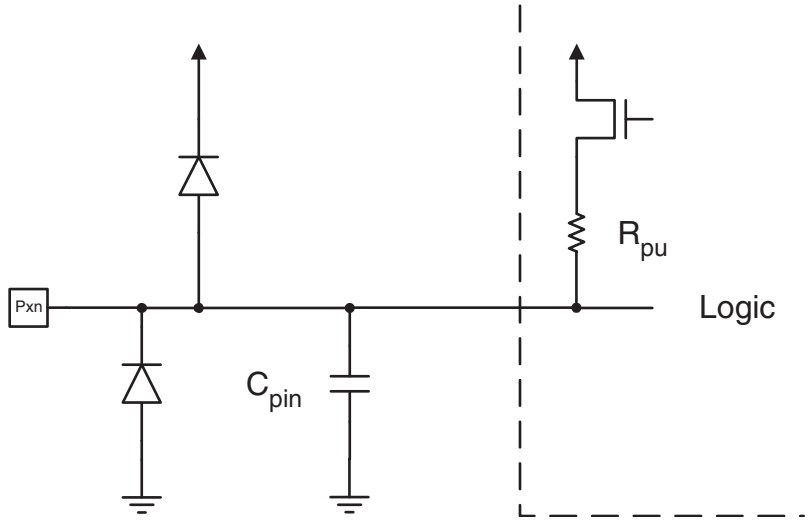
Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14. Low Voltage I/O-Ports

14.1 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 VCC and Ground as indicated in Figure 14-1. Refer to ”Electrical Characteristics” on page 142 for a complete list of parameters.

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



Note: See Figure 14-2 on page 62 for details.

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 70.

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.

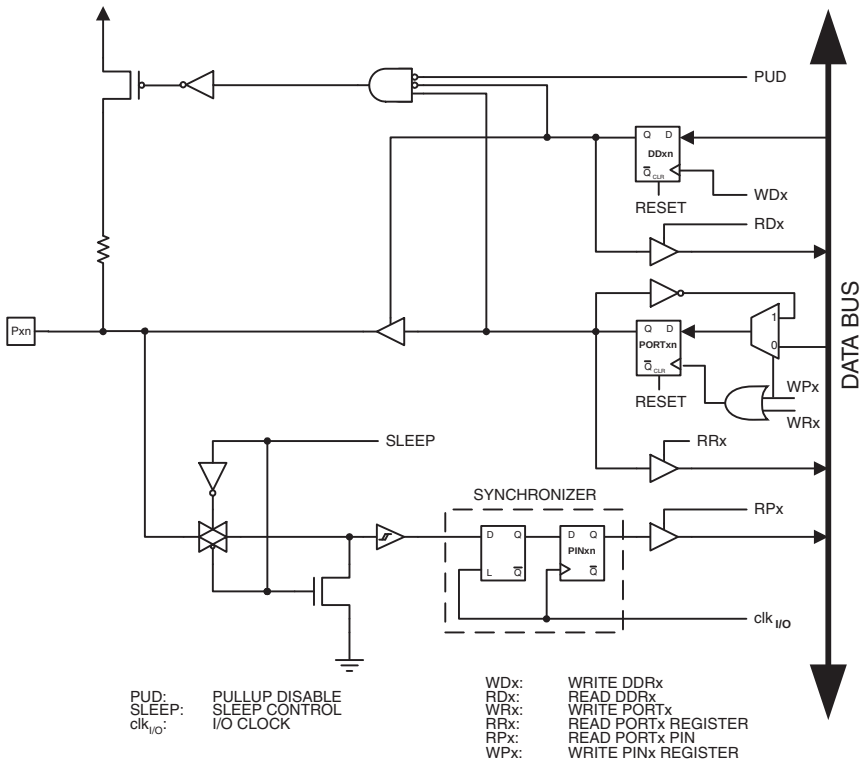
Using the I/O port as General Digital I/O is described in ”Low Voltage Ports as General Digital I/O” on page 62. Many low voltage port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in ”Alternate Port Functions” on page 66. 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.

14.2 Low Voltage Ports as General Digital I/O

The low voltage ports are bi-directional I/O ports with optional internal pull-ups. Figure 14-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 14-2. 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.

14.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in ”Register Description” on page 70, 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).

14.2.2 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.

14.2.3 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 14-1 summarizes the control signals for the pin value.

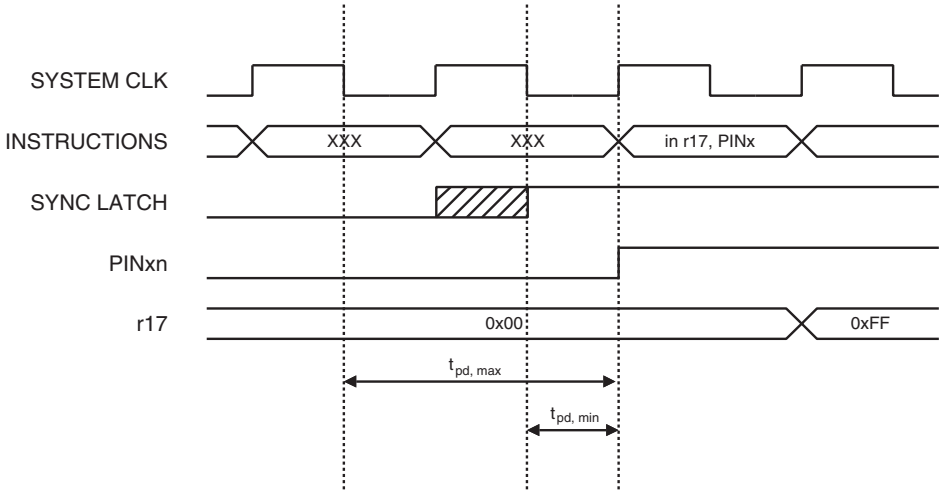
Table 14-1. Port Pin Configurations

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)

14.2.4 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 14-2 on page 62, 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 14-3 on page 64 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.

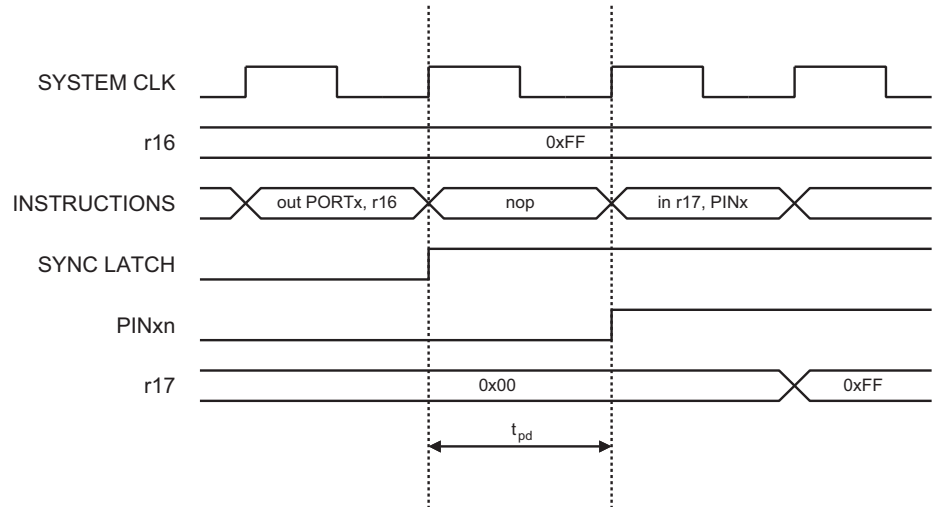
Figure 14-3. Synchronization when Reading an Externally Applied Pin value



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 14-4. 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 14-4. 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.

14.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 14-2 on page 62, 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 VCC/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 66.

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.

14.2.6 Unconnected Pins

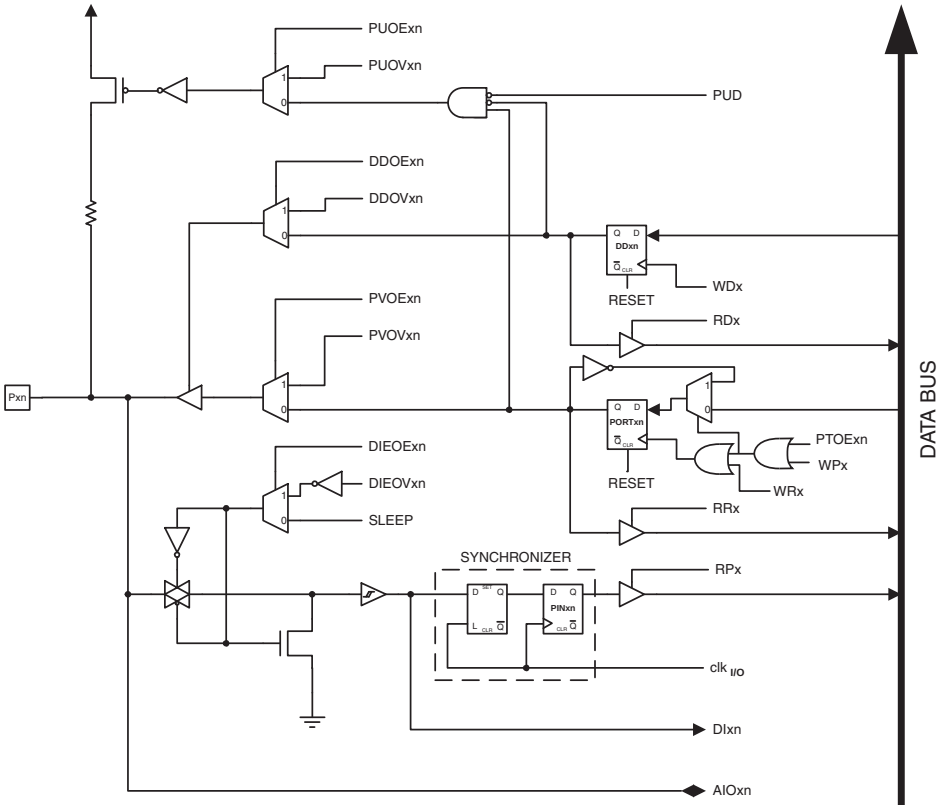
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

14.3 Alternate Port Functions

Many low voltage port pins have alternate functions in addition to being general digital I/Os. Figure 14-5 on page 67 shows how the port pin control signals from the simplified Figure 14-2 on page 62 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 14-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	$clk_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	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 14-2 on page 68 summarizes the function of the overriding signals. The pin and port indexes from Figure 14-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 14-2. 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.

14.3.1 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 14-3.

Table 14-3. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB2	MISO/CKOUT/T1 (SPI Bus Serial DataOutput, Clock Output, Timer/Counter Clock Input)
PB1	SCK/SGND/T0 (SPI Bus Master Clock input, GND for ADC0 measurements, Timer/Counter 0 Clock Input)
PB0	ADC0 (ADC Input Channel 0)

The alternate pin configuration is as follows:

- **MISO/CKOUT/T1 - Port B, Bit 2**

MISO : Slave Data Output pin for SPI programming

When not operating in programming mode, this pin can serve as Clock Output, CPU clock divided by 2. When not operating in programming mode or as clock output, the pin can be used as clock input for Time/Counter1

- **SCK/SGND/T0 - Port B, Bit 1**

SCK : Clock Input pin for SPI programming

When not operating in programming mode, this pin can serve as ground reference for ADC0 channel by configuring the pin as output low. When not used as SGND, the pin can be used as clock input for Time/Counter0.

- **ADC0 - Port B, Bit 0**

Analog to Digital Converter, channel 0.

.

Table 14-4. Overriding Signals for Alternate Functions in PB2:0

Signal Name	PB2/MISO/CKOUT/T1	PB1/SCK/SGND/T0	PB0/ADC0

14.4 Register Description

14.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	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 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 62 for more details about this feature.

14.4.2 PORTB – Port B Data Register

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

14.4.3 DDRB – Port B Data Direction Register

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

14.4.4 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	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	

15. Timer/Counter0 and Timer/Counter1 Prescalers

15.1 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.

15.2 Internal Clock Source

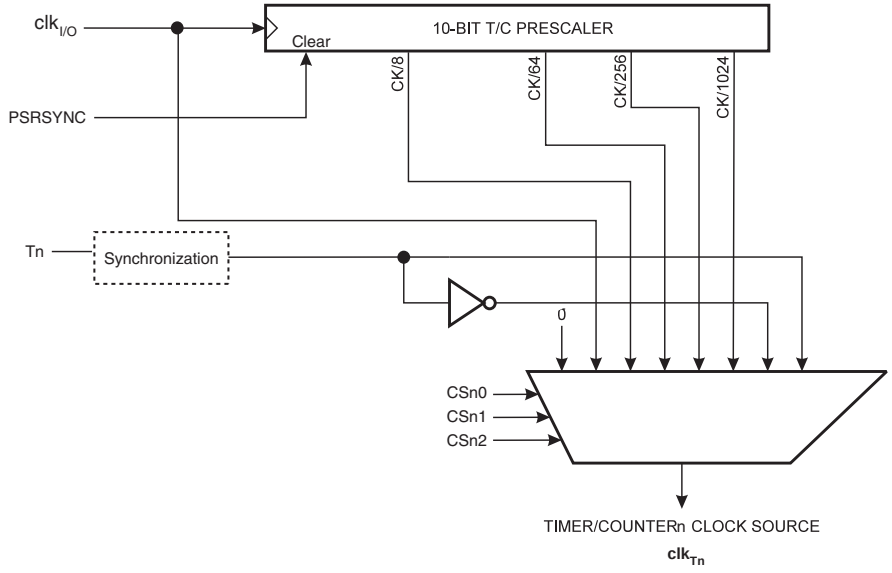
The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source.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

15.3 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 15-1. Prescaler for Timer/Counter

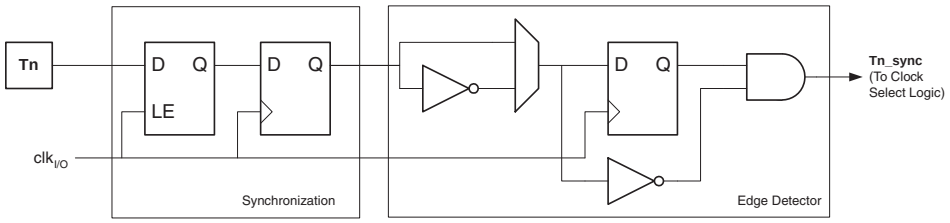


15.4 External Clock Source

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 15-2 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 15-1 on page 73 for details.

Figure 15-2. 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.

Note: The synchronization logic on the input pins (Tn) is shown in Figure 15-2.

15.5 Register Description

15.5.1 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 Select0, Bit 2, 1, and 0**

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

Table 15-1. 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.

15.5.2 GTCCR – General Timer/Counter Control Register

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.

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

16.1 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**

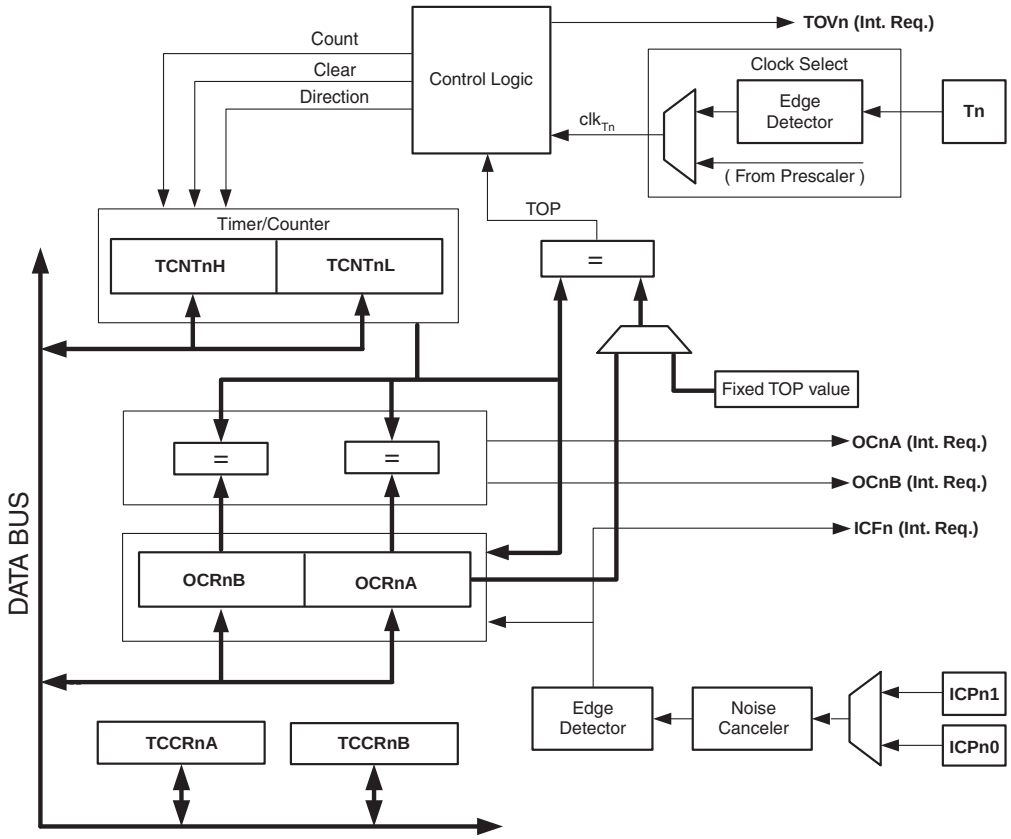
16.2 Overview

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

ATmega4HVD/8HVD 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 151.

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 16-1. 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 86.

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



16.2.1 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 16-1 on page 74) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK 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 82 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}).

16.2.2 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 OCRnA/B 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 16-1 are also used extensively throughout the document.

Table 16-1. 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.

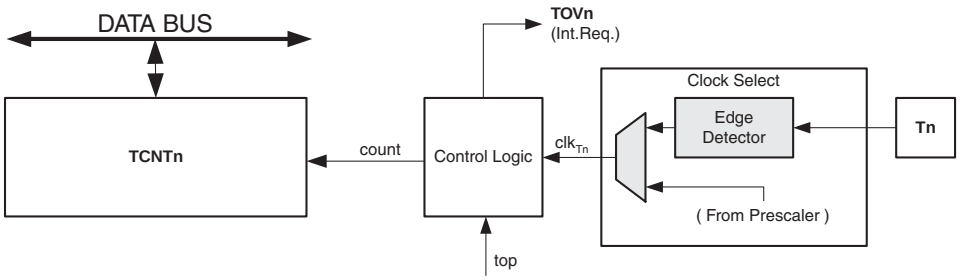
16.3 Timer/Counter Clock Sources

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 71

16.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 16-2 on page 76 shows a block diagram of the counter and its surroundings.

Figure 16-2. 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 81. 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.

16.5 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 86. Table 16-2 on page 76 shows the different Modes of Operation.

Table 16-2. 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)

16.5.1 Normal 8-bit Mode

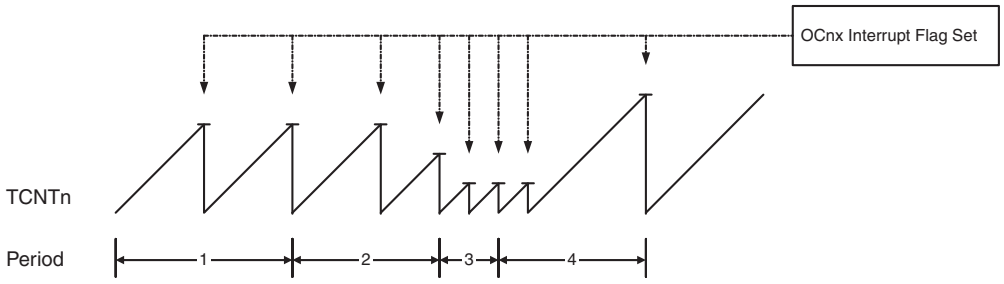
In the normal mode, the counter (TCNTnL) is incrementing until it overruns when it passes its maximum 8-bit value (MAX = 0xFF) and then restarts from the bottom (0x00), see Table 16-2 on page 76 for bit settings. The Overflow Flag (TOVn) will be set in the same timer clock cycle as the TCNTnL becomes 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.

16.5.2 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 16-2 on page 76 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 16-3 on page 77. The counter value (TCNTn) increases until a Compare Match occurs between TCNTn and OCRnA, and then counter (TCNTn) is cleared.

Figure 16-3. 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 since the CTC mode does not have the double buffering feature. 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.

16.5.3 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 16-2 on page 76 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.

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

In Clear Timer on Compare 16-bit mode, OCRAnA/B Registers are used to manipulate the counter resolution, see Table 16-2 on page 76 for bit settings. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches OCRnA/B, where OCRnB represents the eight most significant bits and OCRnA represents the eight least significant bits. OCRnA/B 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 OCRnA/B 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.

16.5.5 8-bit Input Capture Mode

The Timer/Counter can be used in a 8-bit Input Capture mode, see Table 16-2 on page 76 for bit settings. For full description, see ”Input Capture Unit” on page 78.

16.5.6 16-bit Input Capture Mode

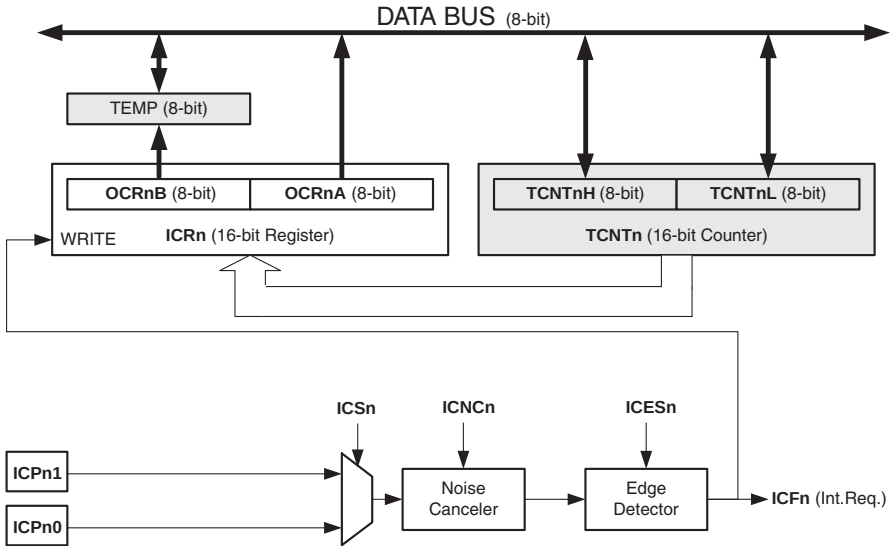
The Timer/Counter can also be used in a 16-bit Input Capture mode, see Table 16-2 on page 76 for bit settings. For full description, see ”Input Capture Unit” on page 78.

16.6 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 (ICP00), or alternatively via the osi_posedge pin on the Oscillator Sampling Interface (ICP01). 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 16-4 on page 79. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 16-4. 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 82.

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.

16.6.1 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 (ICS0/1) bit. 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.

16.6.2 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.

The noise canceller should only be used for ICP01 (Port PC0).

16.6.3 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 16-3. Timer/Counter0 Input Capture Source (ICS)

ICS0	Source
0	ICP00: osi_posedge pin from OSI module(1) (2)
1	ICP01: Port PC0

Notes: 1. See "OSI – Oscillator Sampling Interface" on page 27 for details.
2. The noise canceller cannot be used with this setting.

Table 16-4. Timer/Counter1 Input Capture Source (ICS)

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

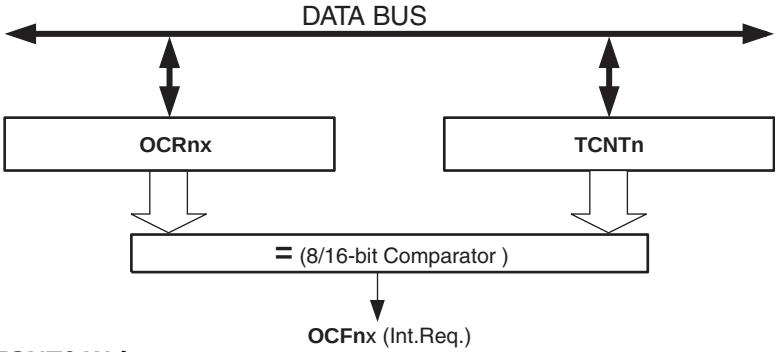
Note: 1. The noise canceller cannot be used with this setting.

16.7 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 Regisers, 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 16-5 on page 81 shows a block diagram of the Output Compare unit.

Figure 16-5. Output Compare Unit, Block Diagram



16.7.1 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 OCRnA/B to be initialized to the same value as TCNTn without triggering an interrupt when the Timer/Counter clock is enabled.

16.7.2 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 OCRnA/B value, the Compare Match will be missed.

16.8 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 16-6 on page 81 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 16-6. Timer/Counter Timing Diagram, no Prescaling

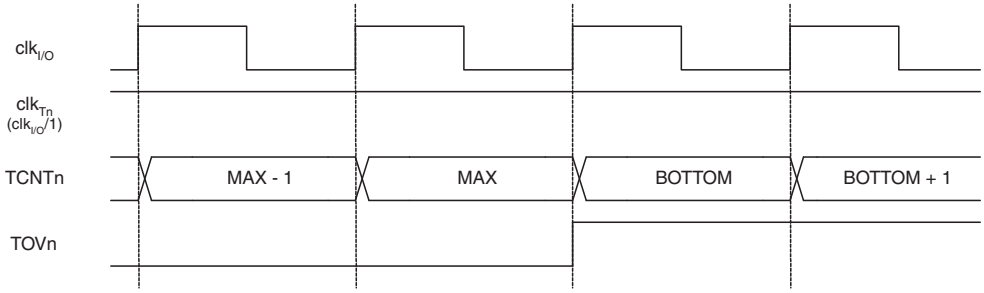


Figure 16-7 on page 82 shows the same timing data, but with the prescaler enabled.

Figure 16-7. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

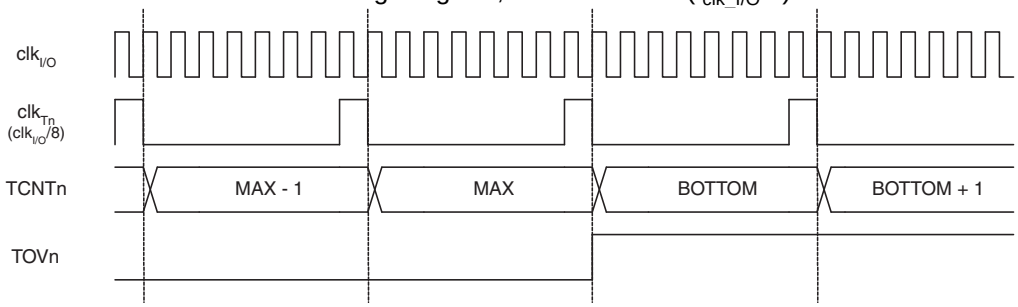


Figure 16-8 on page 82 shows the setting of OCFnA and OCFnB in Normal mode.

Figure 16-8. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk\ I/O}/8$)

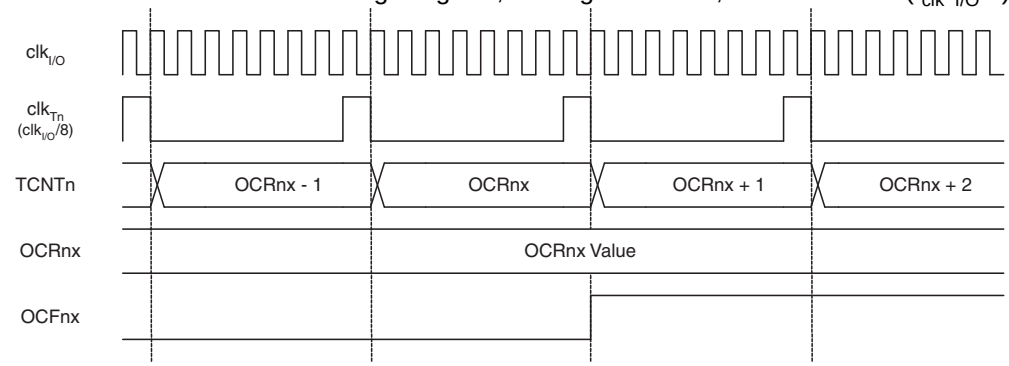
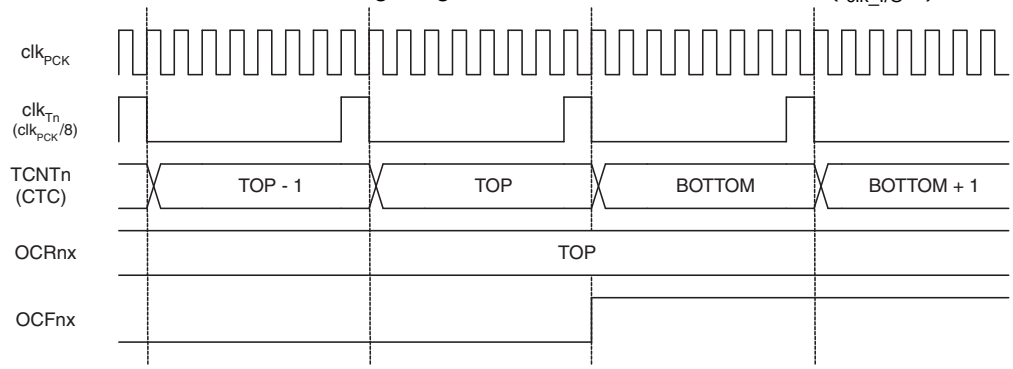


Figure 16-9 on page 82 shows the setting of OCFnA and the clearing of TCNTn in CTC mode.

Figure 16-9. Timer/Counter Timing Diagram, CTC mode, with Prescaler ($f_{clk\_I/O}/8$)



16.9 Accessing Registers in 16-bit Mode

In 16-bit mode (the TCWn bit is set to one) the TCNTnH/L and OCRnA/B or TCNTnL/H 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 OCRnA/B 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 OCRnA/B 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 5.

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 5.

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 OCRnA/B 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 5.

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

16.9.1 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.

16.10 Register Description

16.10.1 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 82.

- **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 logic one, this bit enables the input capture function in Timer/Counter to be triggered by the alternative Input Capture Source. To make the comparator trigger the Timer/Counter Input Capture interrupt, the ICIEn bit in the Timer Interrupt Mask Register (TIMSK) must be set. See Table 16-3 on page 80 and Table 16-4 on page 80.

- **Bits 2:0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega4HVD/8HVD 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 16-6 on page 81. 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 81).

16.10.2 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.

16.10.3 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 82.

16.10.4 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 82.

16.10.5 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 82.

16.10.6 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 “11.” on page 51.) 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 Timer/Counter Interrupt Flag Register – TIFRn.

- **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 Timer/Counter n Interrupt Flag Register – TIFRn.

- **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 Timer/Counter n Interrupt Flag Register – TIFRn.

16.10.7 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.

17. ADC - Analog-to-Digital Converter

17.1 Features

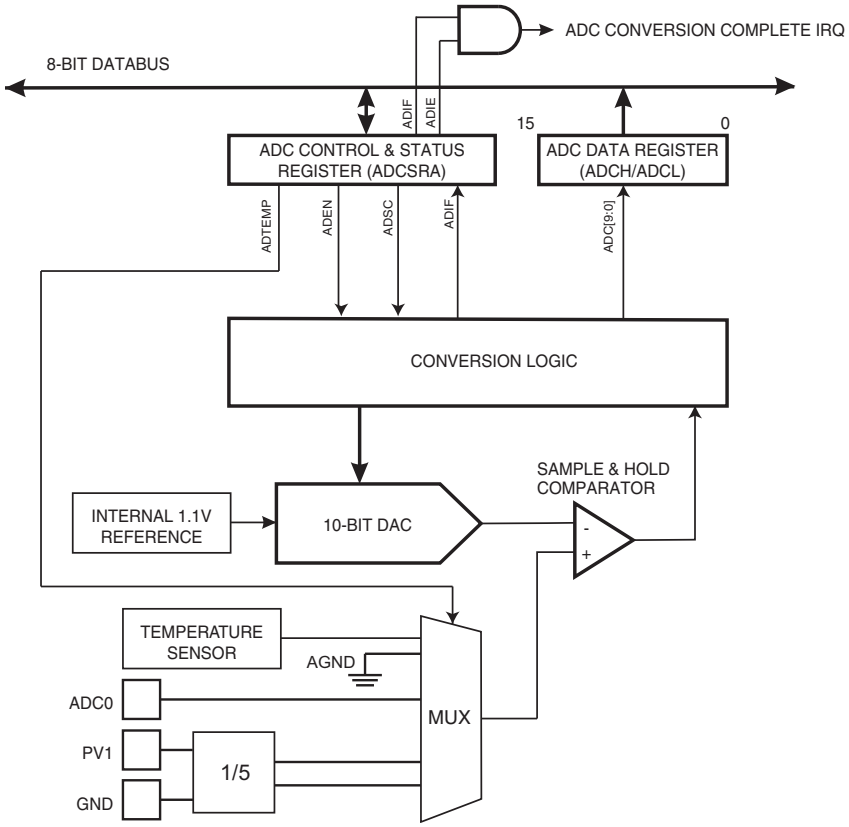
- **10-bit Resolution**
- **78 µs Conversion Time @ clk_{ADC} = 167 kHz**
- **Up to 13 kSPS at Maximum Resolution**
- **External Input Channel with 0 - 5V Input Voltage Range**
- **External Input Channel (ADC0) with 0 - 1V Input Voltage Range**
- **Internal Temperature Sensor Input Channel**
- **1.1V ADC Reference Voltage (typical value)**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The ATmega4HVD/8HVD features a 10-bit successive approximation ADC. The single-ended voltage inputs refer to 0V.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 17-1.

Internal reference voltage of nominally 1.1V is provided on-chip. External input on PV1 pin is divided by 5 internally to be within the range of the internal reference votlage.

Figure 17-1. Analog-to-Digital Converter Block Schematic



17.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. For the PV1 pin the minimum value represents GND and the maximum value represents 5 times the internal 1.1V reference voltage. For the ADC0 pin the minimum value represents AGND and the maximum value represents the internal 1.1V reference voltage.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. The result is presented right adjusted.

ADCL must be read first, then ADCH, to ensure that the content of the data registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

17.3 Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed.

17.4 Conversion Timing

When initiating a conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. If the system clock prescaler setting is changed during a ADC conversion, the conversion result may be invalid and should be discarded.

When PV1 or ADC0 is selected, a conversion takes 13 ADC clock cycles. When the internal temperature sensor is selected, the conversion takes 27 ADC cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 27 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 16.5 ADC clock cycles after the start of the first conversion. When a conversion is complete, the result is written to the ADC Data Registers, the ADC Interrupt Flag (ADIF) is set, and ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

Figure 17-2. ADC Timing Diagram, First Conversion (Single Conversion Mode)

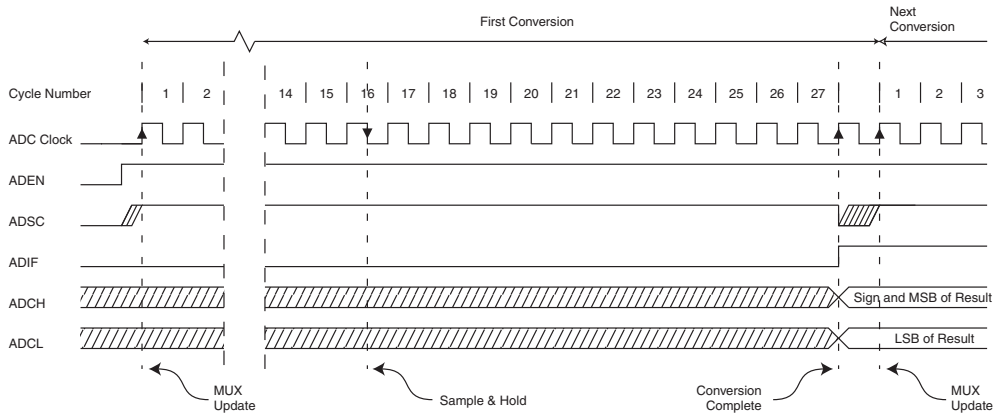


Figure 17-3. ADC Timing Diagram, Single Conversion

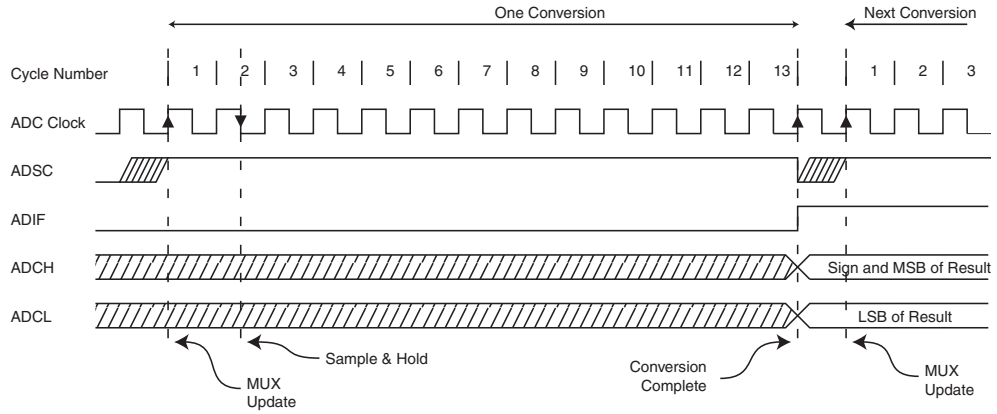


Table 17-1. ADC Conversion Time, except VTEMP input.

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	15.5	27
Normal conversions	1.5	13

17.5 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) is taken from the internal 1.1V bandgap reference.

17.6 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting and the ADC conversion complete interrupt is enabled.
2. Enter ADC Noise Reduction mode. The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If

another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering sleep modes other than Idle mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

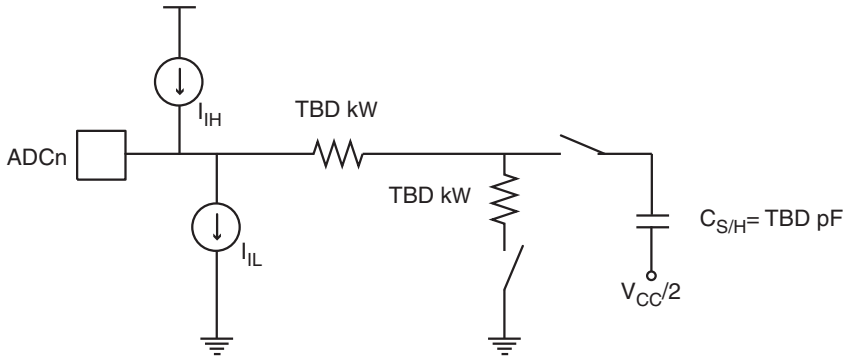
17.6.1 Analog Input Circuitry

The analog input circuitry is illustrated in Figure 17-4 An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, which can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 17-4. Analog Input Circuitry



17.6.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
2. Use the ADC noise canceler function to reduce induced noise from the CPU.
3. If any port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

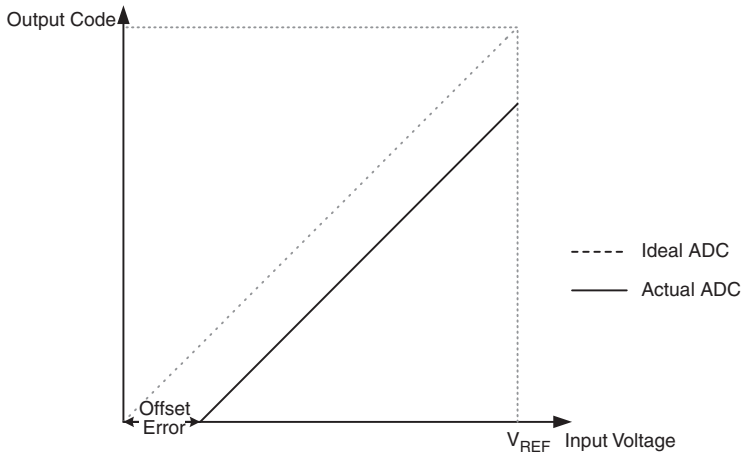
17.6.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

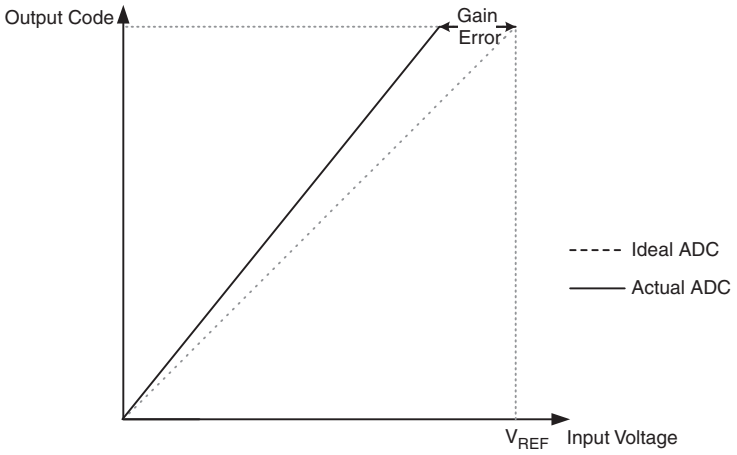
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 17-5. Offset Error



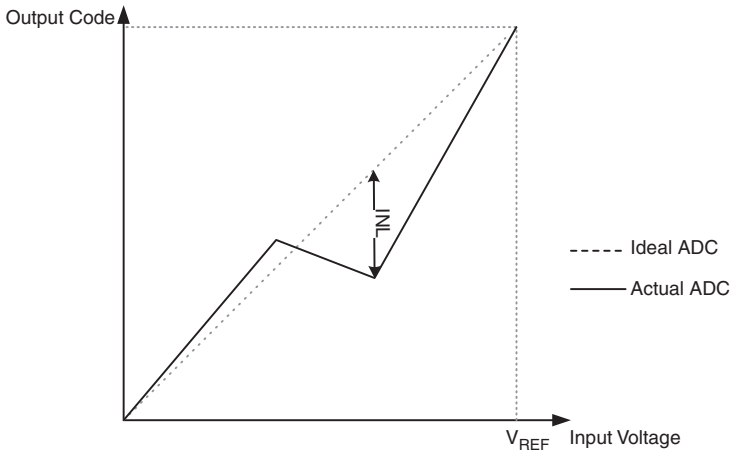
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 17-6. Gain Error



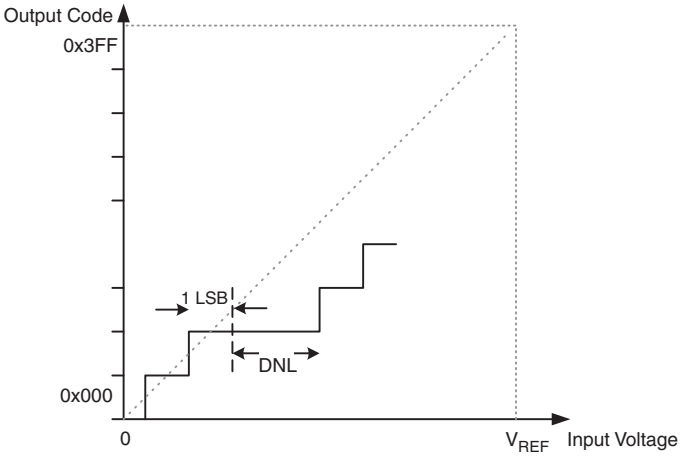
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 17-7. Integral Non-linearity (INL)



- Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 17-8. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ± 0.5 LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ± 0.5 LSB.

17.7 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{5 \cdot V_{REF}}$$

where V_{IN} is the voltage on input pin and V_{REF} the voltage reference. 0x000 represents analog ground, and 0x3FF represents 5 times the reference voltage minus one LSB.

17.7.1 Temperature Measurement using the internal temperature sensor – TBD

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC channel. Selecting the ADC Temperature channel by writing the ADTEMP bit in ADCSRA register enables the temperature sensor. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor. The measured voltage has a linear relationship to the temperature as described in Table 17-2 on page 96. The voltage sensitivity is approximately 1 mV / °C and the accuracy of the temperature measurement is +/- 10°C.

Table 17-2. Temperature vs. Sensor Output Voltage (Typical values)

Temperature (°C)	-45 °C	+27 °C	+105 °C
Voltage (mV)	225 mV	297 mV	327 mV

The values described in Table 17-2 on page 96 are typical values. However, due to the process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software. The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip, as a part of the production test. The software calibration can be done utilizing the formula:

$$Temperature = k \cdot V_{TEMP} + T_{OS}$$

where V_{TEMP} is the ADC reading of the temperature sensor signal, *k* is a fixed coefficient (TBD, 1.07 mV/°C) and T_{OS} is the temperature sensor offset value determined and stored into EEPROM as a part of production test.

17.8 Register Description

17.8.1 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	-	ADIF	ADIE	-	ADMUX1	ADMUX0	ADCSRA
Read/Write	R/W	R/W	R	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

Write this bit to one to start each conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 27 ADC clock cycles. The conversion time will then be 13 ADC cycles if ADTEMP is set to zero, or 27 ADC cycles if ADTEMP is set to one. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – RES: Reserved bits**

These bits are reserved, and will always read as zero.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

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

- **Bit 2 – RES: Reserved bit**

This bit isreserved, and will always read as zero.

- **Bit 1:0 – ADMUX1:0: ADC Channel Selection Bits**

These bits selects the analog input that should be connected to the ADC according to Table 17-3. If these bits are changed during a conversion, the change will not be effective until the conversion is complete.

Table 17-3. ADC Channel Input Selection

ADMUX1:0	Input
00	VCELL
01	VTEMP
10	Reserved
11	ADC0

17.8.2 ADCL and ADCH – The ADC Data Register

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	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 an ADC conversion is complete, the result is found in these two registers.When ADCL is read, the ADC Data Register is not updated until ADCH is read.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in ”ADC Conversion Result” on page 96.

17.8.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	PB0DID	DIDR0
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 – RES: Reserved**

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

- **Bit 0 – PB0DID: PB0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the Port B pin is disabled. The PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the PB0 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.

18. Voltage Reference

ATmega4HVD/8HVD features an internal bandgap reference. This reference is an input reference to the ADC, the Battery Protection module, the Voltage Regulator, and the Black-out Detection. The bandgap reference voltage cannot be observed directly at any output in normal operation. During factory testing, the bandgap reference will be calibrated. This final calibration cannot be changed by software. See ”System and Reset Characteristics” on page 144 for details.

19. Voltage Regulator

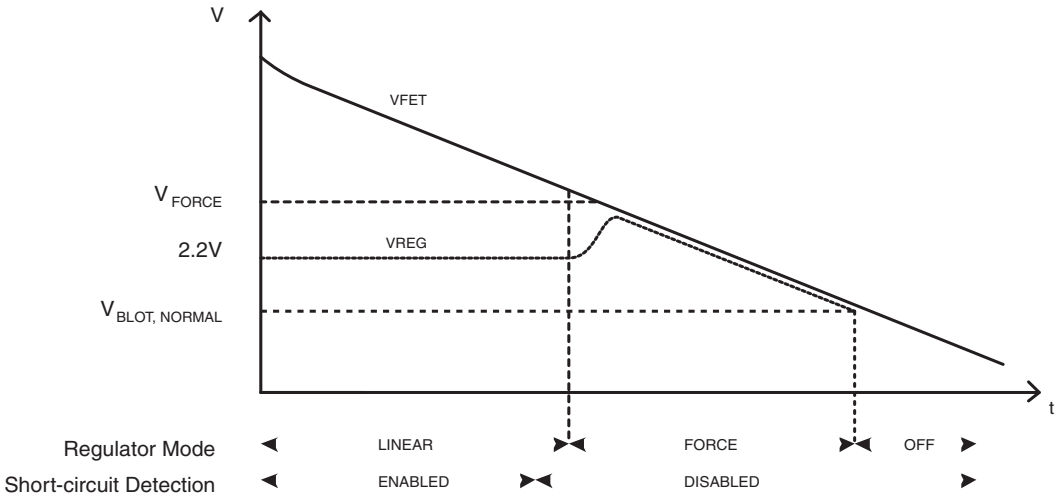
19.1 Features

- **Linear regulation giving a fixed output voltage (VREG) of 2.2V for VFET > V_{FORCE}**
- **Output voltage forced as close as possible to VFET for VFET ≤ V_{FORCE} to facilitate low voltage operation.**
- **Regulator Short-circuit Detection (RSCD), disconnecting VFET from VREG if VFET drops below the Regulator Short-Circuit Detection voltage (V_{RSCD})**
- **RSCD enabling/disabling in software**

19.2 Overview

The Voltage Regulator is a linear regulator that nominally provides a fixed output voltage (VREG). However, to facilitate efficient low-voltage operation with minimum voltage drop caused by the regulator, voltage regulation is automatically switched off when VFET is below V_{FORCE} (see ”Electrical Characteristics” on page 142). VREG as a function VFET and regulator is illustrated in Figure 19-1. A discontinuity in VREG can be observed as the point where the regulator switches from linear mode to FORCE mode, as VREG is being forced as close as possible to VFET. The voltage drop from VFET to VREG in this mode depends on the load current on VREG.

Figure 19-1. VREG/VCC as function of VFET



The Voltage regulator mode depending on operating conditions and SW settings, are summarized in Table 19-1.

Table 19-1.

VFET Voltage	Regular Short-circuit Detection	Regular mode
VFET > V_{FORCE}	Enabled/Disabled	Normal Linear Regulation
VFET< V_{FORCE}	Enabled	Battery Pack Short mode
VFET < V_{FORCE}	Disabled	Force mode

19.3 Battery Pack Short mode

The Voltage Regulator has a separate Short-Circuit Detection mode (RSCD) that can be enabled or disabled by SW. This mode should always be enabled except when operating at VFET voltages below V_{FORCE} (see TBD-electrical chara). The mode is intended for sustaining operation during short spikes on VFET that can occur for instance during a battery pack insertion. The mode is entered when VFET drops below V_{FORCE} and Regulator Short-circuit Detection is enabled. In the Battery Pack Short mode, VFET is temporarily disconnected from VREG to avoid a quick drop in the voltage regulator output. The chip will be completely powered by the external reservoir capacitor (CREG). This allows the chip to operate a certain time before entering BLOD reset (power-off) even if the VFET voltage is too low for the voltage regulator to operate.

The maximum time that the chip can operate in the Battery Pack Short mode is given by the size of the external reservoir capacitor and the actual power drawn from VREG. The VREG voltage must stay above $V_{BLOT,normal}$ to avoid that the chip enters power-off. If a battery pack short occurs when VREG = 2.2V and $V_{BLOT,\ normal}$ is 2.0V the chip can operate for a time given by:

$$T = \frac{c \cdot \Delta(v)}{I_{AVG}} = \frac{CREG \cdot 0.2V}{I_{AVG}}$$

where I_{AVG} represents the average current drawn from CREG. For CREG = 2.2 µF and I_{AVG} = 100 µA, this time equals 4.4 ms. Refer to Table 19-2 on page 102 for an example of Regulator Short-circuit Detection.

19.4 Regulator Force mode

The regulator Force mode is designed to be able to operate the chip even at very low cell voltages. This mode is automatically entered when the VFET voltage drops below V_{FORCE-} provided that the Regulator Short-circuit Detection is disabled. To ensure operation down to minimum VFET level, Regulator Short-circuit Detection should always be disabled before VFET reaches V_{FORCE-} during discharge.

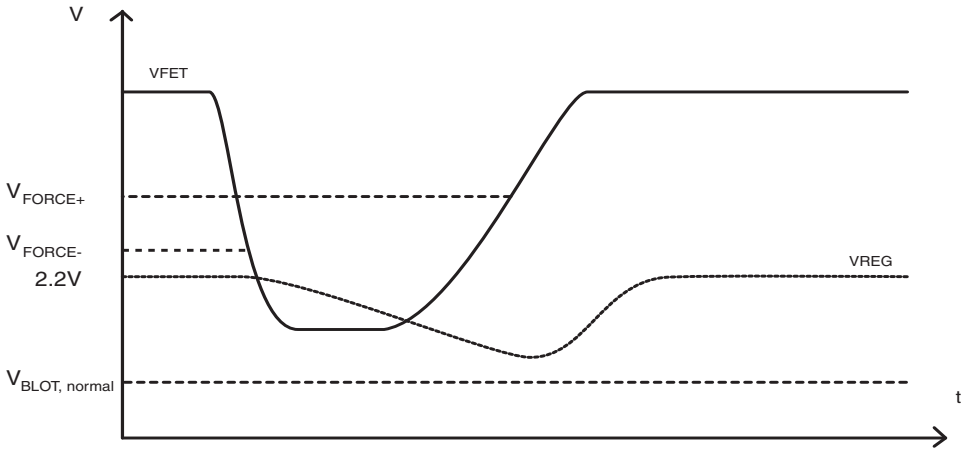
An example of VREG voltage as function of VFET when using the voltage regulator as intended, is shown in figure 66. When VFET > V_{FORCE}, VREG is regulated to 2.2V (nominal value). When VFET approaches V_{FORCE}, Regulator Short-circuit Detection is disabled, and when VFET passes V_{FORCE-}, the regulator enters FORCE mode. When this occurs, a discontinuity in VREG can be observed. This is caused by the fact that VREG is no longer regulated and is forced as close to VFET as possible. A small difference V_{DROP} between VFET and VREG can be observed. This voltage drop depends on the load current. Typical vales can be found in TBD-El.Chara. The chip will continue operation in FORCE mode until VREG reaches the BLOD level and the chip enters power-off.

When using VREG as reference for external measurements, for instance as reference for an external thermistor as shown in TBD-operating circuit, it may be required to know the accurate value of VREG. In FORCE mode, VREG will range from V_{FORCE} to $VFET_{min}$, as opposed to normal regulation where VREG has a constant value. Since ATmega4/8HVD has no method of measuring VREG directly, it is recommended to measure the cell voltage and estimate VREG based on loss through Charge FET (if applied) and voltage drop from VFET to VREG according to the following formula:

$$VREG_{FORCE} = V_{Cell} - V_{DS-CFET} - V_{DROP}$$

V_{DROP} depends on actual current drawn from VREG.

Figure 19-2. Regulator Short-circuit Detection Example



19.5 Register Description

19.5.1 ROCR – Regulator Operating Condition Register

Bit	7	6	5	4	3	2	1	0	
	ROCS	-	-	-	-	RSCDEN	ROCWIF	ROCWIE	ROCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ROCS: ROC Status**

When the VFET voltage is below V_{FORCE}, this bit is set. When this is set, it is either an indication that the Voltage Regulator has entered the Battery Pack short mode (if RSCDEN is set) or that the Voltage regulator is operating in FORCE mode (if RSCDEN is cleared).

- **Bit 6:3 – Res: Reserved**

These bits are reserved and will always read as zero.

- **Bit 2 – RSCDEN: Regulator Short-Circuit Detection Enable**

If this bit is set, the Voltage Regulator will enter the Battery Pack Short mode if VFET drops below V_{FORCE}. If the bit is cleared, the Voltage Regulator will enter FORCE mode under the same condition. The ROCS bit can be used to indicate if VFET is above or below the V_{FORCE} threshold.

The RSCDEN bit should normally never be set when VFET is below V_{FORCE} since this will eventually lead to a power-off condition when CREG has been discharged. For this reason, the default value of RSCDEN is always 0. For the same reason, RSCDEN should always be cleared when operating in DUVR mode. During charging, when VFET passes V_{FORCE+} and the ROCS bit is cleared, the RSCDEN bit must be set to be able to sustain operation during short spikes on VFET.

- **Bit 1 – ROCWIF: ROC Warning Interrupt Flag**

: The ROCWIF flag is set if RSCDEN is set and ROCS goes high. The bit can be cleared by writing a logic '1' to its bit location.

- **Bit 0 – ROCWIE: ROC Warning Interrupt Enable**

This bit enables the interrupt caused by the ROCWIF Flag.

20. Battery Protection

20.1 Features

- **Short-circuit Protection**
- **Discharge Over-current Protection**
- **Charge Over-current Protection**
- **External Protection Input**
- **Programmable and Lockable Detection Levels and Reaction Times**
- **Autonomous Operation Independent of CPU**

20.2 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 three different programmable detection levels: Short-circuit Detection Level, Discharge Over-current Detection Level, and Charge Over-current Detection Level. There are two different programmable delays for activating Current Battery Protection: Short-circuit Reaction Time and Over-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 nominally 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.

In addition, the module offers an External Protection Input. The activation of the External Protection Input operates independently of the rest of the battery protection mechanisms. The activation/deactivation of this protection is instantaneously controlled from the External Protection Input port, and will not deactivate or affect the other battery protection mechanisms.

The effect of the various battery protection types is given in Table 20-1.

Table 20-1. 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
External Protection Input	Entry and/or Exit	Disabled	Disabled	Operational

In order to reduce power consumption, Short-circuit and Discharge Over-current Protection are automatically deactivated when the D-FET is disabled. The Charge Over-current is disabled when the C-FET is disabled. Note that Charge Over-current Protection is never automatically disabled when the chip is operated in DUVR mode. Also note that none of the current protections are deactivated by External Protection Input. To save power during an External Protection event, DFE and CFE should be cleared in the FCSR register. Make also sure that the chip is not operated in DUVR mode.

The Current Battery Protection (CBP) monitors the cell current by sampling the shunt resistor voltage (R_{SENSE}) connected between the NI and GND 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 and Over-current. This ensures a reliable clock source, offset cancellation and low power consumption.

20.3 Short-circuit Protection

The Short-circuit detection is provided to enable a fast response time to very large discharge currents. If the voltage over R_{SENSE} 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.

20.4 Discharge Over-current Protection

If the voltage over R_{SENSE} 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.

20.5 Charge Over-current Protection

If the voltage at the GND/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.

The Short-circuit and Over-current 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” on page 108 for register descriptions.

20.6 External Protection Input

The External Protection Input disables both FETs (Charge FET and Discharge FET) immediately (asynchronously) when the voltage on PC1 is pulled high (logic ‘1’) by the External Protection circuitry. It is also used to disable DUVR mode if DUVR mode is enabled. Note that, unlike a Battery Protection event, the External Protection input does not affect the status of the FCSR (CFE, DFE, DUVRD) bits. When the ‘high’ condition disappears, the FET disabling is released immediately. DUVR mode is automatically re-entered if enabled.

The feature is automatically enabled when the chip starts up, and can be disabled before locking the BPCR register. When locking the BPCR register, the External Protection feature is also locked. The feature should be disabled if it is not used.

When External Protection Input is enabled, an override enable signal is set to PC1, configuring the pin as digital input. The port may be set up to give an interrupt when the pin value changes. The protection status can be read from the port register.

Note that the External Protection Input is default enabled. This means that after reset (and during reset) the port is default overridden to External Protection Input, independent of the port register setting. The user must disable the External Protection Input before the port can be used as a normal port.

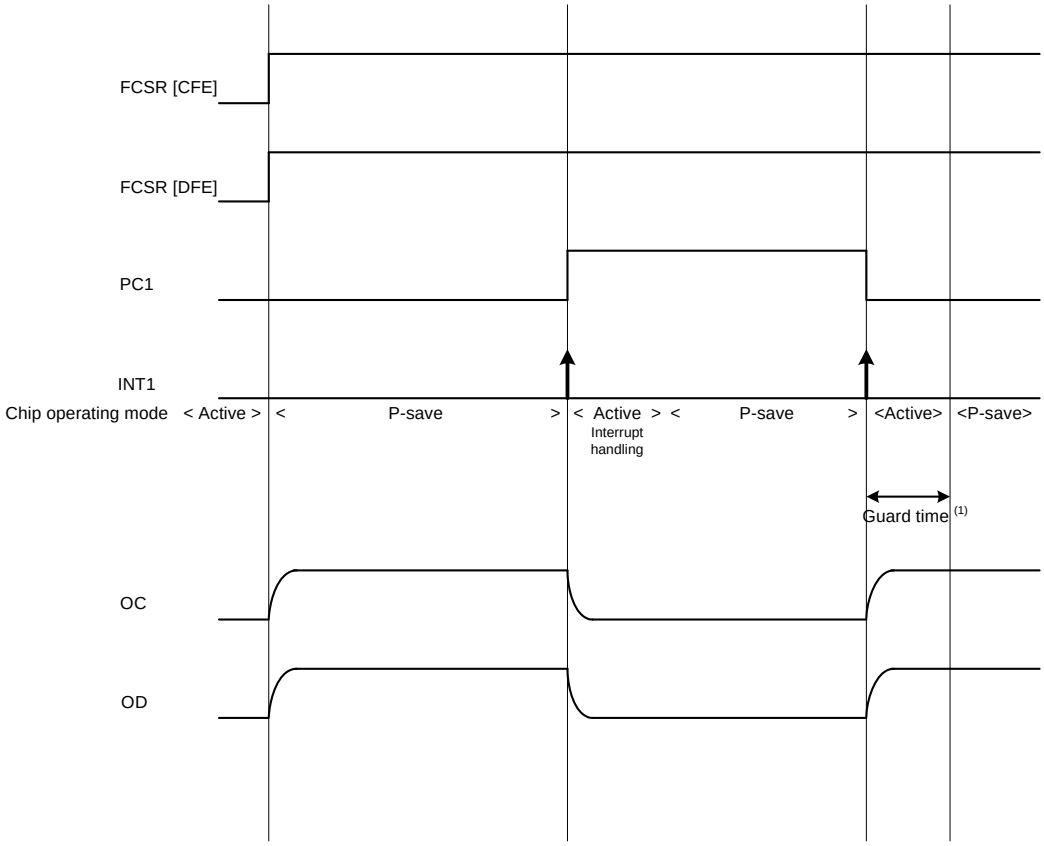
It is recommended that the external interrupt on the External Protection Input port (PC1) is configured to ‘any edge’ to generate an interrupt to the microcontroller when using this feature, indicating that the FET protection status has changed. Refer to ”External Interrupt” on page 53. By reading the pin register, the External Protection status can be determined. If the pin register is set, it means that External Protection is triggered and the FET control signals (FCSR (CFE, DFE and DUVRD)) are overridden so the FETs are disabled. In the opposite case External Protection violation is not present.

To ensure a safe exit from the External Protection Input condition, the FETs and DUVR mode should be disabled by SW when an External Protection condition is detected. This enables software to control completely when the FETs are switched ON again.

If not disabled by SW, the FETs will be re-enabled once the External Protection condition disappears (PC1 pin equals '0'). This feature may be useful to filter out glitches on the PC1 pin, for instance caused by temporary high voltages on the BATT pin when connecting a charger (refer to ”Operating Circuit” on page 141). In this case, SW does not have to take any action to re-enable the FETs.

Note however that compared to SW switching on the FETs, the switch ON time for the FETs is multiplied by a factor 10 when the FETs are disabled and re-enabled by External Protection Input and the chip is operated in sleep mode. This is caused by the internal clock system and power saving functions of the chip. For this reason, if the switch ON time of the transistors is critical, External Interrupt on PC1 should always be enabled when using the External Protection Input feature. The interrupt should be configured to trig on any edge and will ensure that the chip wakes up from sleep and thus enables a normal switch ON time for the FETs. To ensure a fast rise time on the OC and OD pins, SW must either remain in ACTIVE or IDLE mode during FET charging or disable and re-enable the OC/OD bits before entering Power-save sleep mode. For an example on correct External Protection Input usage, refer to Figure 20-1.

Figure 20-1. Example in External protection Input

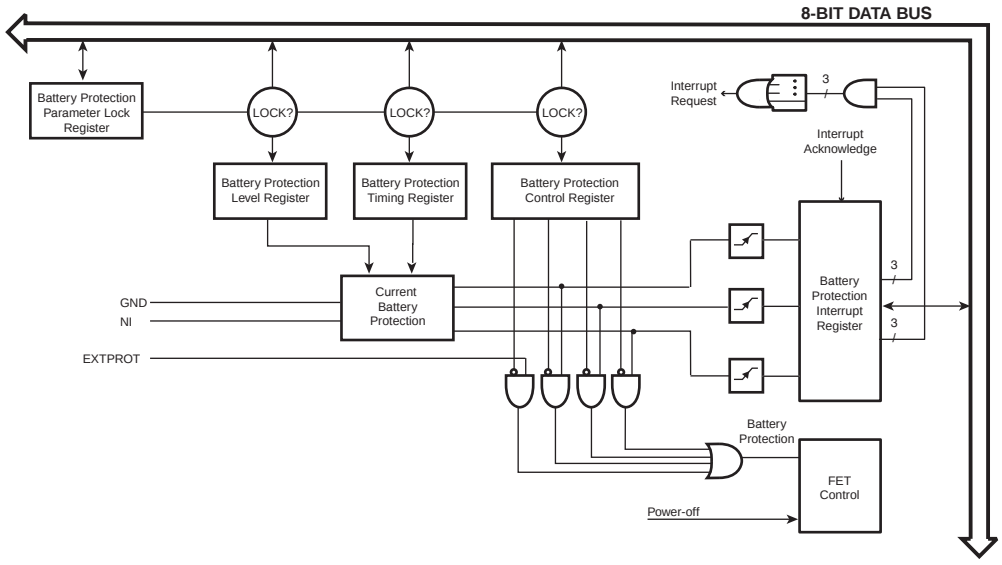


Note: 1. To ensure that the FET switch ON time is as expected, the chip should remain in Active/Idle mode during this time period. Alternatively, SW may switch off the FETs while External Protection Input is active, and re-enable FETs on next INT1 interrupt. In this case, SW may enter Power-save without considering the switch ON time of the FETs.

20.7 Battery Protection CPU Interface

The Battery Protection CPU Interface is illustrated in Figure 20-2.

Figure 20-2. Battery Protection CPU Interface



Each protection originating from the Current Battery Protection module has an Interrupt Flag. All enabled flags are combined into a single battery protection 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. An interrupt event for the External Protection Input can be generated by enabling the external interrupt for the input port.

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.

20.8 Register Description

20.8.1 BPPLR – Battery Protection Parameter Lock Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	**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, BPOCTR, BPSCTR, 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.

20.8.2 BPCR – Battery Protection Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	EPID	SCD	DOCD	COCD	-	-	BPCR
Read/Write	R	R	R/W	R/W	R/W	R/W	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **bit 5 – EPID: External Protection Input Disable**

When this bit is set, the External Protection Input is disabled and any External Protection Input will be ignored. Note that this bit overrides the GPIO functionallity in the External Protection Input port. If not using the External Protection Input feature, it is recommended that this bit is always set.

- **Bit 4 – SCD: Short Circuit Protection Disable**

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 Disable**

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 1:0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

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.

20.8.3 BPSCTR – Battery Protection Short-current Timing Register

Bit	7	6	5	4	3	2	1	0	
	–	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-current 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 20-2 on page 110.

Table 20-2. 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	(23.0 - 86.0 µs) + T_d[3]
0x01	(23.0 - 86.0 µs) + T_d[3]
0x02	(85.5 - 148.5 µs) + T_d[3]
0x03	(148.0 - 211.0 µs) + T_d[3]
...	...
0x7E	(7.84 - 7.90 ms) + T_d[3]
0x7F	(7.90 - 7.96 ms) + T_d[3]

Notes:
1. The actual value depends on the actual frequency of the "Ultra Low Power RC Oscillator" on page 24. See "Electrical Characteristics" on page 142.
2. Initial value: SCPT[0x10](1ms).
3. An additional delay up to 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 55 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.

20.8.4 BPOCTR – Battery Protection Over-current Timing Register

Bit	7	6	5	4	3	2	1	0	
	–	–	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 20-3 on page 111.

Table 20-3. 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 24. See "Electrical Characteristics" on page 142.
2. Initial value.
3. An additional delay up to 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 0.2 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 BPOCTR register is written. Any writing to the BPOCTR register during this period will be ignored.

20.8.5 BPSCD – Battery Protection Short-circuit Detection Level Register

Bit	7	6	5	4	3	2	1	0	
	SCDL[7:0]								BPSCD
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	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 20-4 on page 112. This register should always be written as one-hot.

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.

20.8.6 BPDOCD – Battery Protection Discharge-Over-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
	DOCDL[7:0]								BPDOCD
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	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 20-4 on page 112. This register should always be written as one-hot.

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.

20.8.7 BPCOCD – Battery Protection Charge-Over-current Detection Level Register

Bit	7	6	5	4	3	2	1	0	
	COCDL[7:0]								BPCOCD
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	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 20-4 on page 112. This register should always be written as one-hot.

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.

Table 20-4. DL[7:0] with corresponding R_{SENSE} Current for all Current Detection Levels (R_{SENSE} = 10 mΩ, VREF = 1.100 ± 0.005V)

Current Protection Detection Level			
DL[7:0]	**Min.**	**Typ.**	**Max.**
0x01	0.5A	2.0A	3.5A
0x02	1.0A	2.5A	4.0A
0x04	1.5A	3.0A	4.5A
0x08	2.0A	3.5A	5.0A
0x10	2.5A	4.0A	5.5A
0x20	3.0A	4.5A	6.0A
0x40	3.5A	5.0A	6.5A
0x80	4.5A	6.0A	7.5A
All other values	Reserved		

20.8.8 BPIMSK – Battery Protection Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	SCIE	DOCIE	COCIE	-	-	BPIMSK
Read/Write	R	R	R	R/W	R/W	R/W	R	R	
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 1:0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

20.8.9 BPIFR – Battery Protection Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	SCIF	DOCIF	COCIF	-	-	BPIFR
Read/Write	R	R	R	R/W	R/W	R/W	R	R	
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 1:0 – Res: Reserved Bit**

These bits are reserved and will always read as zero.

21. FET Control

21.1 Overview

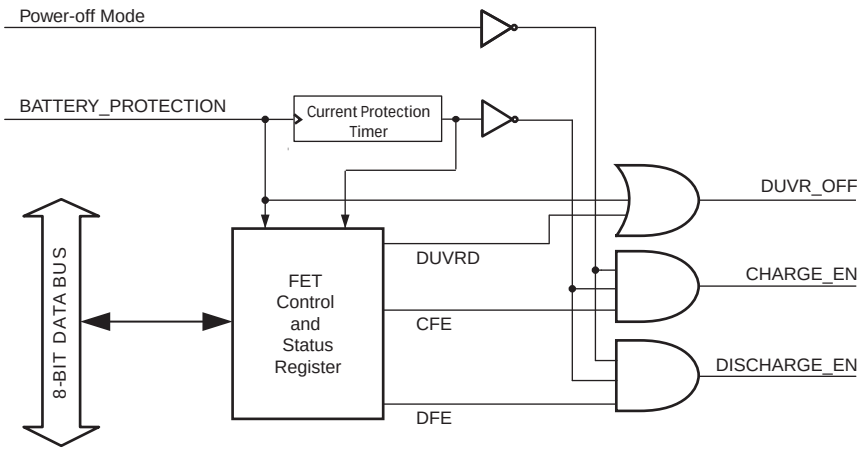
The FET control is used to enable and disable the Charge FET and Discharge FET. Normally, the FETs are enabled and disabled by SW writing to the FET Control and Status Register (FCSR). However, the autonomous Battery Protection circuitry will if necessary override SW settings to protect the battery cells 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 21-1.

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 Under-voltage 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 21-1. FET Control Block Diagram



21.1.1 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.

21.2 FET Driver

21.2.1 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**

21.2.2 Overview

The ATmega4HVD/8HVD includes a FET Driver. The FET Driver is designed for driving N-channel FETs used as high side switch in 1-Cell Li-Ion battery pack. A block diagram of the FET driver is shown in Figure 21-2.

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 2.3 Volts in 1-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 26-6 on page 146 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 21-3 on page 116 and Table 26-6 on page 146 for details..

Figure 21-2. FET Driver block diagram.

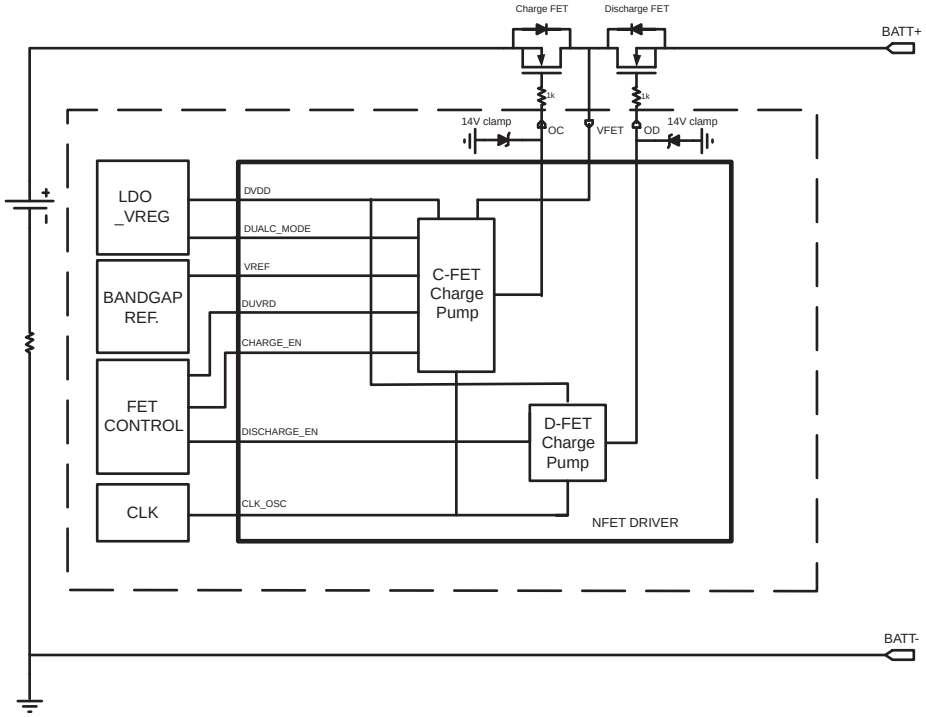
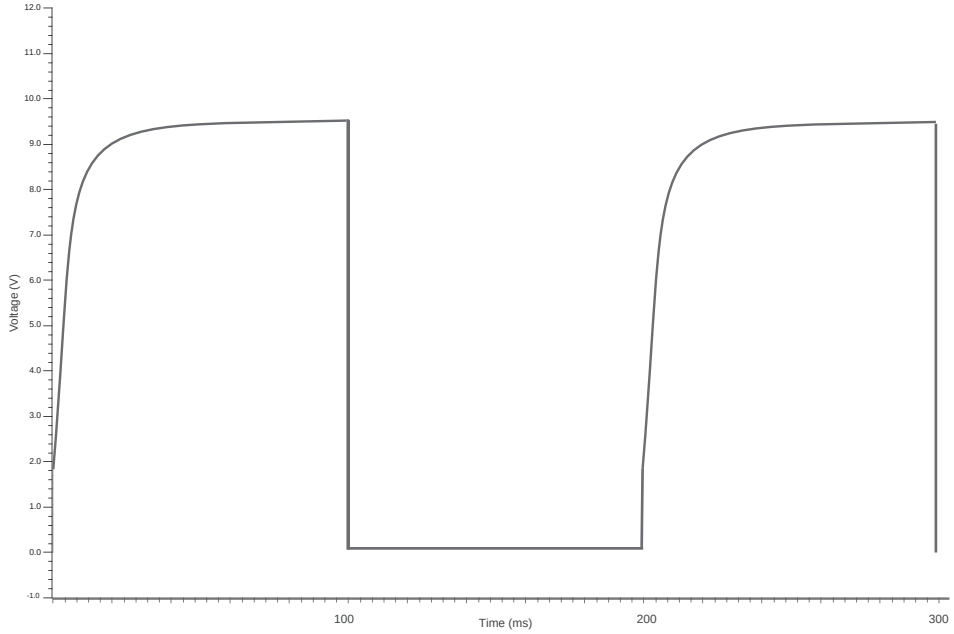


Figure 21-3. Switching NFET on and off during NORMAL operation



21.3 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).

ATmega4HVD/8HVD 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 BLOD reset and switch off both the Charge- and Discharge FET.

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.

21.4 Register Description

21.4.1 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 ATmega4HVD/8HVD, 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 DUVR mode. See ”DUVR – Deep Under-Voltage Recovery Mode operation” on page 117 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), DUVR 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 cleard, the charge FET will be enabled by DUVT 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.

22. debugWIRE On-chip Debug System

22.1 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**

22.2 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.

22.3 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 22-1. The debugWIRE Setup

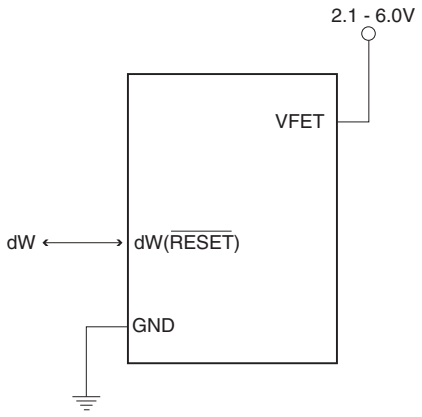


Figure 22-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/($\overline{\text{RESET}}$) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the $\overline{\text{RESET}}$ pin directly to V_{CC} will not work.

- Capacitors connected to the $\overline{\text{RESET}}$ pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

22.4 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.

22.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset ($\overline{\text{RESET}}$). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e. when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (AVR Studio).

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.

22.6 Register Description

22.6.1 DWDR – debugWire Data Register

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

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.

23. Self-Programming the Flash

23.1 Overview

In ATmega4HVD/8HVD, there is no Read-While-Write support, and no separate Boot Loader Section. The SPM instruction can be executed from the entire Flash.

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.

23.2 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.

23.3 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 RWWSRE 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.

23.4 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.

23.5 Addressing the Flash During Self-Programming

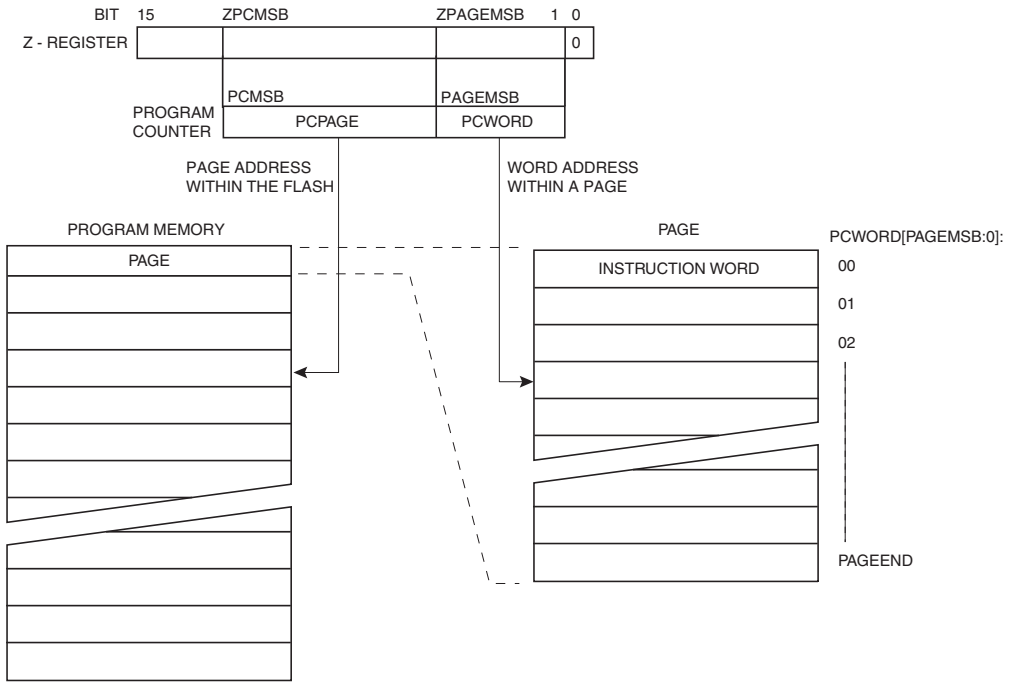
The Z-pointer is used to address the SPM commands.

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 24-6 on page 131), 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 23-1. 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 23-1. Addressing the Flash During SPM[1]



Note: 1. The different variables used in Figure 23-1 are listed in Table 24-6 on page 131.

23.5.1 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 (EEWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

23.5.2 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 24-1 on page 129 and Table 24-2 on page 129 for how the different settings of the Lock bits affect the Flash access.

23.5.3 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 upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. 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 24-4 on page 130 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.

23.5.4 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. Sec-

ondly, 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. The internal Black-out Detection circuit will issue an internal reset immediately and take the chip in power-off after 4 CPU cycles if the operating voltage drops below the detection level V_{BLOD}. Alternatively, 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.

23.5.5 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 23-1 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 23-1. 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
SLOW RC FREQ(5)	0BH
ULP RC FREQ(6)	0AH
Reserved	0CH
Reserved	0DH
Batt Prot Adjust Factor(7)	0EH
Reserved	0FH:11H
VPTAT CAL L	12H
VPTAT CAL H(8)	13H

Table 23-1. Signature Row Addressing (Continued)

Signature Byte Description	Z-Pointer Address
ADC Cell Gain Calibration Word L	14H
ADC Cell Gain Calibration Word H(9)	15H
ADC Cell Offset(10)	16H
Reserved	17H
ADC ADC0 Gain Calibration Word L	18H
ADC ADC0 Gain Calibration Word H(11)	19H
ADC ADC0 Offset(12)	1AH
Reserved	1BH
HOT TEMP(13)	1CH
Reserved	1DH

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 [@ HOT].
4. Characterized Slow RC oscillator frequency temperature drift prediction value.
5. Slow RC oscillator frequency in kHz [@ HOT].
6. ULP RC oscillator frequency in kHz [@ HOT].
7. BattProt Adjust Factor is an adjustment for additional battery protection trigger level accuracy, signed byte value. (Negative value means right shift, positive value means left shift.)
8. Calibration Word used to calculate the absolute temperature in Kelvin from a VTEMP conversion.
9. Calibration Word used to compensate for gain error in ADC Cell input 0.
10. Calibration Byte used to compensate for offset in ADC Cell input 0.
11. Calibration Word used to compensate for gain error in ADC input ADC0.
12. Calibration Byte used to compensate for offset in ADC input ADC0.
13. Calibration Temperature in °C.

All other addresses are reserved for future use.

23.5.6 Programming Time for Flash when Using SPM

The Calibrated Fast RC Oscillator is used to time Flash accesses. Table 23-2 shows the typical programming time for Flash accesses from the CPU.

Table 23-2. SPM Programming Time(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 time is per individual operation.

Table 23-3. Explanation of different variables used in Figure 23-1 and the mapping to the Z-pointer for ATmega4HVD

Variable		Corresponding Z-value	Description
PCMSB	10		Most significant bit in the Program Counter. (The Program Counter is 11 bits PC[10:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires six bits PC[4:0]).
ZPCMSB		Z11	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[10:5]	Z11:Z6	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[4:0]	Z5:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

Table 23-4. Explanation of different variables used in Figure 23-1 and the mapping to the Z-pointer for ATmega8HVD

Variable		Corresponding Z-value	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires six bits PC[4:0]).
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[11:5]	Z12:Z6	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[4:0]	Z5:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

23.6 Register Description

23.6.1 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	
	–	–	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 124 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 123 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 “10001”, “01001”, “00101”, “00011” or “00001” in the lower five bits will have no effect.

24. Memory Programming

24.1 Program And Data Memory Lock Bits

The ATmega4HVD/8HVD provides two Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 24-2. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 24-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No	Description	Default Value
	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 24-2. 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 programming mode.(1)
3	0	0	The Fuse bits are locked and further programming and verification of the Flash and EEPROM is disabled programming mode.(1)

Notes: 1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

24.2 Fuse Bits

The ATmega4HVD/8HVD has two Fuse bytes. Table 24-4 and Table 24-3 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.

24.2.1 High Byte

Table 24-3. Fuse High Byte

Bit No	Fuse High Byte	Description	Default Value
7:2	–	–	1 (unprogrammed)
1	OSCSEL1	Oscillator Select 1	1 (unprogrammed)
0	OSCSEL0	Oscillator Select 0	0 (programmed)

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.

24.2.2 Low Byte

Table 24-4. 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 8-1 on page 23 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 48 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.

24.2.3 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.

24.3 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 ATmega4HVD/8HVD is given in Table 24-5.

Table 24-5. Device ID

	Signature Bytes Address		
Part	**0x000**	**0x001**	**0x002**
ATmega4HVD	0x1E	TBD	TBD
ATmega8HVD	0x1E	0x93	0x12

24.4 Calibration Bytes

The ATmega4HVD/8HVD has a calibration byte for the Fast RC Oscillator. This byte resides in a high byte in the signature address space. During Reset, the calibration byte for the Fast RC Oscillator is automatically written into the corresponding calibration register.

24.5 Page Size

Table 24-6. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega4HVD	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATmega8HVD	4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 24-7. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega4HVD	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATmega8HVD	256 bytes	4 bytes	EEA[1:0]	64	EEA[8:2]	8

24.6 Serial Downloading

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 24-8 on page 132, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 24-1. Serial Programming and Verify

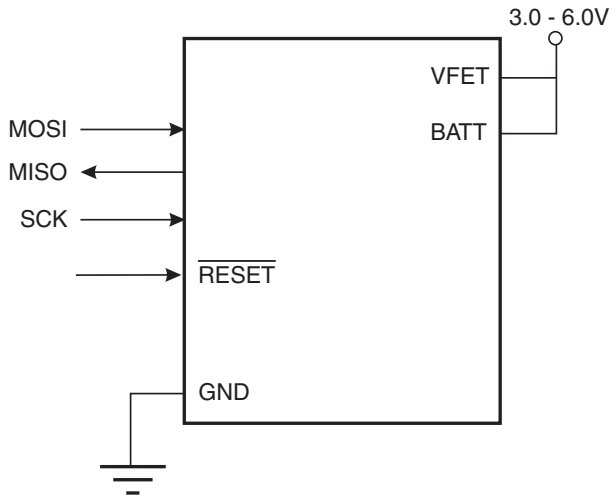


Table 24-8. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
SCK	PB1	I	Serial Clock
MISO	PB2	O	Serial Data out
MOSI	PC1	I	Serial Data in

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

24.6.1 Serial Programming Algorithm

When writing serial data to the ATmega4HVD/8HVD, data is clocked on the rising edge of SCK.

When reading data from the ATmega4HVD/8HVD, data is clocked on the falling edge of SCK. See Figure 26-1 and Figure 26-2 for timing details.

To program and verify the ATmega4HVD/8HVD in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 24-10):

1. Power-up sequence:
   Apply minimum 3V between VFET and GND, and BATT 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 24-9.) 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 24-9.) 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/$\overline{BSY}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 24-9). 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, $\overline{RESET}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{RESET}$ to “1”.
   Turn V_{CC} power off.

Table 24-9. 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

24.6.2 Serial Programming Instruction set

Table 24-10 on page 133 and Figure 24-2 on page 135 describes the Instruction set.

Table 24-10. 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	$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	$00	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

Table 24-10. Serial Programming Instruction Set (Continued)

Instruction/Operation	Instruction Format			
	Byte 1	**Byte 2**	**Byte 3**	**Byte4**
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

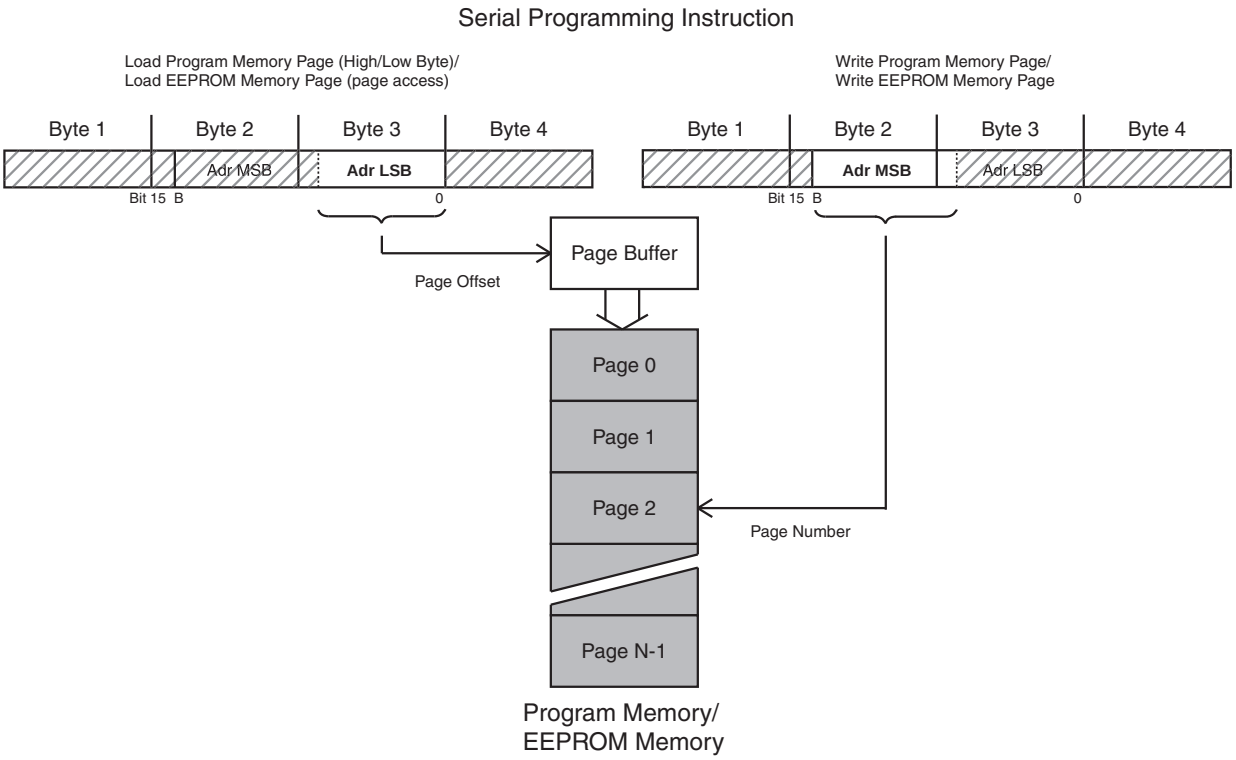
Notes:
1. Not all instructions are applicable for all parts.
2. a = address.
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature 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 24-2 on page 135.

Figure 24-2. Serial Programming Instruction example



24.7 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 ATmega4HVD/8HVD.

Table 24-11. Pin Name Mapping

Signal Name in High-voltage Serial Programming Mode	Pin Name	I/O	Function
SDO	PB1	O	Serial Data Output
SDI	PC1	I	Serial Data Input
SII	PB0	I	Serial Instruction Input
SCI	PC0	I	Serial Clock Input (min. 220ns period)

The prog_enable minimum period for the Serial Clock Input (SCI) during High-voltage Serial Programming is 220 ns.

Table 24-12. Pin Values Used to Enter Programming Mode

Symbol	Pin Name	Value
Prog_enable[0]	PB0	0
Prog_enable[1]	PB1	0
Prog_enable[2]	NC	0
Prog_enable[3]	PB2	0

24.8 High-voltage Serial Programming Algorithm

To program and verify the ATmega4HVD/8HVD in the High-voltage Serial Programming mode, the following sequence is recommended (See instruction formats in Table 24-14):

24.8.1 Enter High-voltage Serial Programming Mode

The following algorithm puts the device in (High-voltage) Serial Programming mode:

1. Set Prog_enable pins listed in Table 24-12 on page 135 to “0000”, RESET pin to 0V and V_{CC} to 0V. VFET should not be connected.
2. Apply 3.0 - 3.5V between V_{CC} and GND, and between BATT 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 24-12 on page 135 to “0000”, RESET pin to 0V, V_{CC} to 0V. VFET should not be connected.
2. Apply 3.0 - 3.5V between V_{CC} and GND, and between BATT 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 drie 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 24-13. 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

24.8.2 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.

24.8.3 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 24-14).
2. Wait after Instr. 3 until SDO goes high for the “Chip Erase” cycle to finish.
3. Load Command “No Operation”.

24.8.4 Programming the Flash

The Flash is organized in pages, see Table 24-10 on page 133. 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 24-14).
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 ATmega4HVD/8HVD, data is clocked on the rising edge of the serial clock, see Figure 24-4, Figure 26-3 and Table 26-9 for details.

Figure 24-3. Addressing the Flash which is Organized in Pages

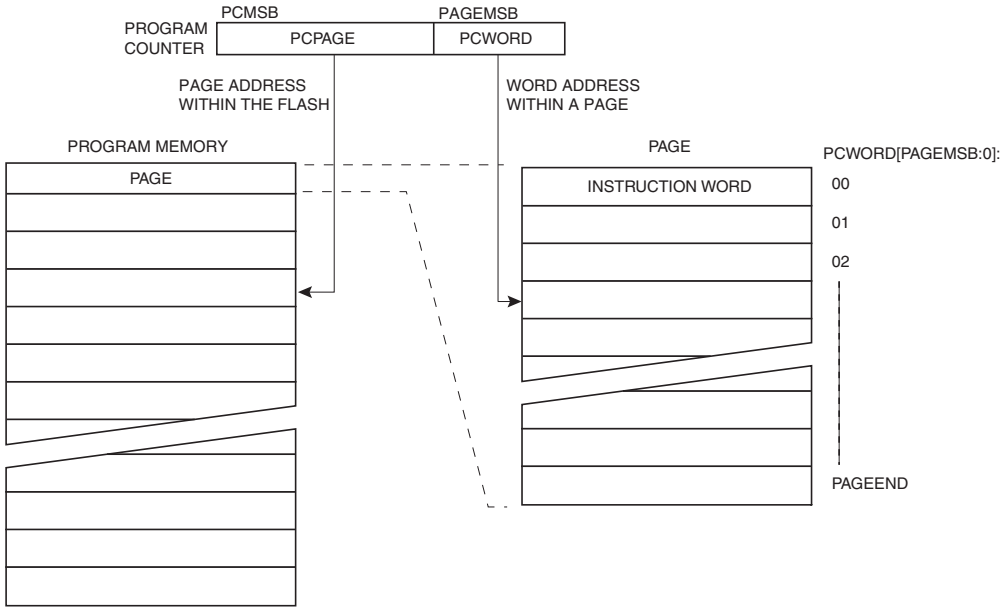
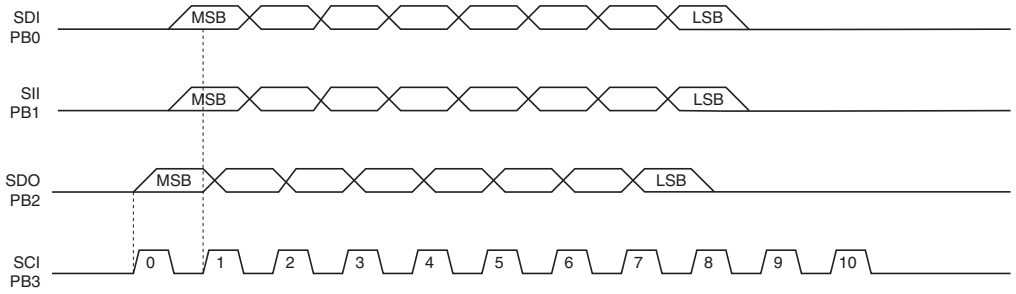


Figure 24-4. High-voltage Serial Programming Waveforms



24.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 26-8 on page 148. 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 24-14):

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.

End Page Programming by Loading Command “No Operation”.

24.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to Table 24-14):

1. Load Command "Read Flash".
2. Read Flash Low and High Bytes. The contents at the selected address are available at serial output SDO.

24.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Table 24-14):

1. Load Command “Read EEPROM”.
2. Read EEPROM Byte. The contents at the selected address are available at serial output SDO.

24.8.8 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 24-14.

24.8.9 Reading the Signature Bytes and Calibration Byte

The algorithms for reading the Signature bytes and Calibration byte are shown in Table 24-14.

24.8.10 Power-off sequence

Set SCI to “0”. Set RESET to “1”. Turn V_{CC} power off.

Table 24-14. High-voltage Serial Programming Instruction Set for ATmega4HVD/8HVD

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(1) 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_000**a_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(1) 0_0000_1100_00 x_xxxx_xxxx_xx	0_000**a_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_0**bbb_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(2)	SDI SII SDO	0_0**bbb_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 24-14. High-voltage Serial Programming Instruction Set for ATmega4HVD/8HVD (Continued)

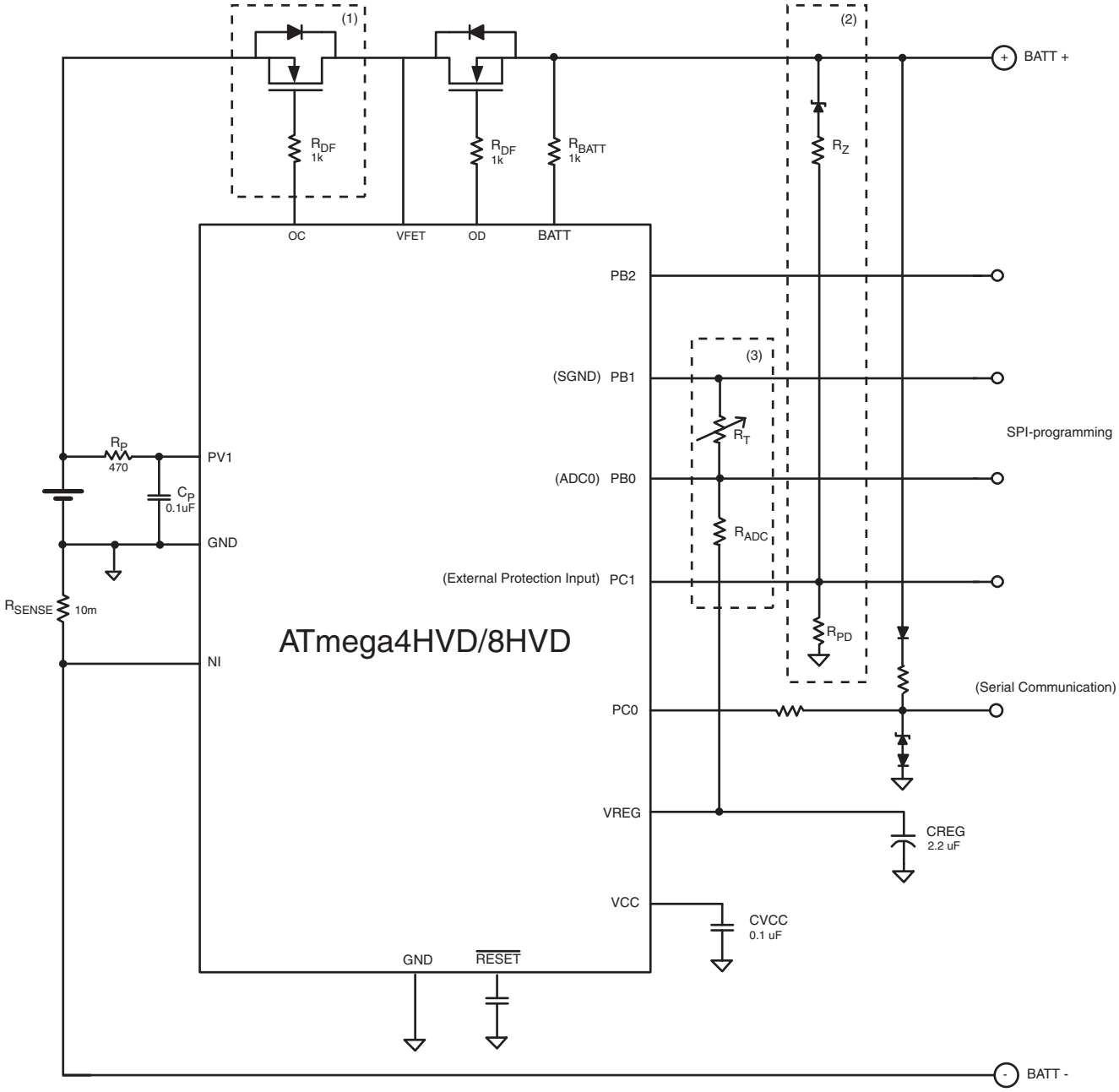
Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Read EEPROM Byte	SDI	0_**bbbb_bbbb**_00	0_**aaaa_aaaa**_00	0_0000_0000_00	0_0000_0000_00	
	SII	0_0000_1100_00	0_0001_1100_00	0_0110_1000_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**q_qqqq_qqq**0_00	
Write Fuse Low Bits	SDI	0_0100_0100_00	0_**A987_6543**_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write **A - 3** = “0” to program the Fuse bit.
	SII	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Write Lock Bits	SDI	0_0010_0000_00	0_0000_00**21**_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write **2 - 1** = “0” to program the Lock Bit.
	SII	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Read Fuse Low Bits	SDI	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00		Reading **A - 3** = “0” means the Fuse bit is programmed.
	SII	0_0100_1100_00	0_0110_1000_00	0_0110_1100_00		
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**A_9876_543**x_xx		
Read Lock Bits	SDI	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00		Reading **2, 1** = “0” means the Lock bit is programmed.
	SII	0_0100_1100_00	0_0111_1000_00	0_0111_1100_00		
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_x**21**x_xx		
Read Signature Bytes	SDI	0_0000_1000_00	0_0000_00**bb**_00	0_0000_0000_00	0_0000_0000_00	Repeats Instr 2 4 for each signature byte address.
	SII	0_0100_1100_00	0_0000_1100_00	0_0110_1000_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**q_qqqq_qqq**x_xx	
Read Calibration Byte	SDI	0_0000_1000_00	0_0000_0000_00	0_0000_0000_00	0_0000_0000_00	
	SII	0_0100_1100_00	0_0000_1100_00	0_0111_1000_00	0_0111_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**p_pppp_ppp**x_xx	
Load “No Operation” Command	SDI	0_0000_0000_00				
	SII	0_0100_1100_00				
	SDO	x_xxxx_xxxx_xx				

Note: **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, **1** = Lock Bit1, **2** = Lock Bit2, **3** = CKSEL Fuse, **4** = SUT0 Fuse, **5** = SUT1 Fuse, Fuse, **A** = WDTON Fuse, **9** = EESAVE Fuse, **8** = SPIEN Fuse, **7** = DWEN Fuse, **6** = SELFPRGEN Fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in High-voltage Serial Programming, only in SPI Programming.

25. Operating Circuit

Figure 25-1. Operating Circuit Diagram



Notes:
1. Optional. The chip can operate without Charge FET.
2. Optional. Only needed for External Protection Input.
3. Optional. Only if external thermistor is used.

26. Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature	-20°C to +85°C
Storage Temperature	-65°C to +150°C
Voltage on PB0 - PB2, VCC, and NI with respect to Ground	-0.5V to VCC +0.5V
Voltage on PV1 and BATT with respect to Ground	-0.5V to + 8.0V
Voltage on PC0 and PC1 with respect to Ground	-0.5V to + 5.0V
Voltage on VFET with respect to Ground	-0.5V to + 12V
Maximum Voltage in $\overline{\text{RESET}}$	-0.5V to + 13V
Voltage on OC, OD and BATT with respect to Ground	-0.5V to + 12V
Maximum Operating Voltage on VFET	6.0V

*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.

26.1 DC Characteristics

Table 26-1. Electrical Characteristics[1], T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

	Parameter	Condition	Min	Typ	Max	Unit
Supply Current	Active	1 MHz, All PRR bits set		420		µA
	idle			150		
	ADC Noise Reduction			290		
	Active	1 MHz, Only ADC Enabled		1.2		mA
	Idle			300		µA
	ADC Noise Reduction	4 MHz, Only ADC Enabled		360		
	Power-save	DUVR mode Disabled, Battery protection Enabled, OC/OD Enabled		35	TBD	µA
		DUVR mode Disabled Battery protection Disabled, OC/OD Disabled		23		
	Power-off	VFET < 4.0V		0.1	1	

Table 26-1. Electrical Characteristics[1], T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted) (Continued)

	Parameter	Condition	Min	Typ	Max	Unit
Voltage Regulator	Regulated Output Voltage, VREG (Linear regulation mode)	VFET = 3.0	2.0	2.2	2.4	V
	Regulator Output Voltage, VREG (Force mode)			VFET - V_{DROP}	V_{FORCE}[2]	V
	Operating Voltage, VFET	Active mode, EEPROM writing	2.3		6.0	V
		Active mode, no EEPROM writing	2.1		6.0	V
	Regulator Force mode level (V_{FORCE})	Rising edge		2.50	2.7	V
		Falling edge		2.35	2.55	V
	Regulator Force level hysteresis			150		mV
	Regulator drop in Force mode (VFET - VREG)	VFET = 2.2V, I_{load} = 1 mA		50		

Notes: 1. All Electrical Characteristics contained in this data sheet are based on initial characterization of actual silicon. These values are preliminary values.
2. TBD

26.2 Oscillator Characteristics

Table 2. Oscillator Characteristics, T_A = -20°C to 85°C, V_{CC} = 2.2V (unless otherwise noted)

	Parameter	Condition	Min	Typ	Max	Unit
Slow RC Oscillator	Frequency		91	131	171	kHz
	Frequency Prediction Error[1]				1	%
	Frequency drift as function of V_{CC} [1][2]	$2.1 \le V_{CC} \le 2.7$		4		
ULP RC Oscillator	Frequency		89	128	167	kHz
Fast RC Oscillator	After initial calibration		1.9	2.0	2.1	MHz
	FOSCCAL frequency step size		0.5	1	2	%
	Frequency drift as function of V_{CC} [1][2]	$2.1 \le V_{CC} \le 2.7$, T_A = 25°C		0.5		%

Notes: 1. Not tested in production.
2. Relevant if regulatoris operated in Force mode.

26.3 System and Reset Characteristics

Table 26-2. Power-on, Reset, BLOD, and Voltage Reference Characteristics[1], T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Threshold Voltage[3]		2.5	3.0[4]	3.4	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage	VREG = 2.2V	0.4		1.9	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin[2]			900		ns
$V_{BLOT, NORMAL}$	BLOD Threshold Voltage[5][6]		1.7	1.8	2.0	V
$V_{BLOT, START-UP}$	BLOD Threshold Voltage			2.3	2.5	V
t_{BLOD}	Min Pulse Width on Black-out Reset[2]			2		µs
$t_{BLODTOUT}$	Black-out Power-off Delay[2]			4		CK
V_{REF}	Bandgap reference voltage after calibration		1.03	1.1	1.17	V

Notes:
1. Values are guidelines only. Actual values are TBD.
2. Not tested in production.
3. The ATmega4HVD/8HVD will start up when the voltage at the BATT pin exceeds V_{POT}.
4. The BATT pin monitors the charger voltage. Because of the voltage drop across the body diode of the Discharge FET, the supply voltage at the VFET pin is approximately 0.5V below the V_{POT} level when the ATmega4HVD/8HVD starts up. That is why typical V_{POT} level is set to 3.0V.
5. V_{BLOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{REG} = V_{BLOT, NORMAL}$ during the production test. This guarantees that a Black-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The Black-out detector monitors the output voltage of the regulator, VREG. V_{BLOT} level is therefore not directly related to the V_{POT} level.
6. In normal operation, the BLOD detection level will always be lower than the minimum voltage regulator output voltage when the voltage regulator is operated within its specifications.

26.4 External Interrupts Characteristics

Table 26-3. Asynchronous External Interrupt Characteristics, T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

26.5 General I/O Lines Characteristics

Table 26-4. T_A = -20°C to 85°C, VFET = 2.1V to 4.2V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}[1]	Input Low Voltage, Except RESET pin	V_{CC} = 2.2V	-0.5		$0.2V_{CC}$	V
V_{IL1}	Input Low Voltage RESET pin	V_{CC} = 2.2V	-0.5		$0.2V_{CC}$	
V_{IH}[1]	Input High Voltage, Except RESET pin	V_{CC} = 2.2V	$0.7V_{CC}$[2]		V_{CC} + 0.5[3]	V
V_{IH1}	Input High Voltage, RESET pin	V_{CC} = 2.2V	$0.9V_{CC}$[2]		V_{CC} + 0.5[3]	V
V_{OL}	Output Low Voltage	I_{OL} = 1mA, V_{CC} = 2.2V			0.2	V
V_{OH}	Output High Voltage	I_{OH} = -1mA, V_{CC} = 2.2V	2.0			V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 2.2V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 2.2V, 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/PC1.
2. “Min” means the lowest value where the pin is guaranteed to be read as high.
3. “Max” means the highest value where the pin is guaranteed to be read as low.
4. Although each I/O port can sink more than the test conditions (1 mA at V_{CC} = 2.2V) 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 (1 mA at V_{CC} = 2.2V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all IOH should not exceed 2 mA.

Table 26-5. PC0/PC1 Characteristics, T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.3VCC	V
V_{IH}	Input High-voltage		0.6VCC	5	V
V_{OL}	Output Low-voltage	350 µA sink current	0	0.4	V
t_r[1]	Rise Time			300	ns
t_{of}[1]	Output Fall Time from V_{IHmin} to V_{ILmax}	C_b < 400 pF[2]		250	ns
t_{SP}[1]	Spikes Suppressed by Input Filter		0	TBD	ns
I_{IH}	Input Leakage Current	V_{IH} = 4.2V		5	µA
C_i[1]	Capacitance			10	pF

Notes:
1. Not tested in production.
2. C_b = capacitance of one bus line in pF.

Table 26-6. FET Driver Outputs specification[1], T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

Parameter	Condition	Min.	Typ.	Max.	Units
VFET DC level[2]	1 cell DUVR operation	2.0	2.3	2.5	V
VFET ripple[2]	1 cell DUVR operation		±0.05		V
OC, OD clamping voltage			TBD		V
OC, OD[3]	Normal ON operation	VFET+2.5	VFET+ 5	VFET+ 7	V
OC, OD	Normal OFF operation		0.0	0.1	V
Risetime[2] (OC, OD, 0 - 90 %)	Normal ON operation		0.5	10	ms
Falltime[2] (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 scales approximately proportional to the capacitive loading.
3. VFET = 4V, I_{load} = 500 nA.

ADC Characteristics

Table 26-7. ADC Characteristics[2], T_A = -20°C to 85°C, VFET = 2.4 to 4.2V (unless otherwise noted)

Symbol	Parameter	Condition	Min[1]	Typ[1]	Max[1]	Units
ADC	Resolution				10	Bits
	Absolute Accuracy (Including INL, DNL, Quantization Error, Gain and Offset Error)	VCELL > 1V				LSB
	Integral Non-Linearity (INL)	VCELL > 1V V_{CC} = 2.2V ADC clock = 167 kHz		2		LSB
	Differential Non-Linearity (DNL)	VCELL > 1V V_{CC} = 2.2V ADC clock = 167 kHz		0.5		LSB
	Gain Error	VCELL > 1V V_{CC} = 2.2V ADC clock = 167 kHz		4		LSB
	Offset error	VCELL > 1V V_{CC} = 2.2V ADC clock = 167 kHz		2		LSB
	PrescaledClock Frequency	Fast RC Osc = 8 MHz		167		kHz
	Conversion Time (VCELL, ADC0)	ADC clock = 167 kHz		78		µs
	Conversion Time (VTEMP)	ADC clock = 167 kHz		163		µs
	Input Voltage (VCELL)		GND		5	V
	Input Voltage (ADC0)	VCELL > 1V	GND		1	V
	Input Voltage (VTEMP)	VCELL > 1V	GND		1	V
	Input Bandwidth			7.5		kHz
	Analog Input Resistance			100		MΩ
Temp. Sensor	V_{PTAT}, Voltage Proportional to Absolute Temperature			1.07		mV/K
	Absolute Accuracy				TBD	K

Notes: 1. Values are guidelines only.
2. All 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.

26.6 Programming Characteristics

26.6.1 Serial Programming

Figure 26-1. Serial Programming Waveforms

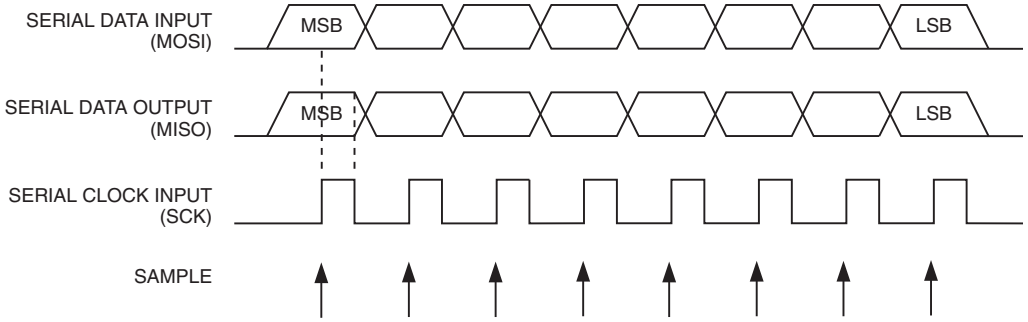


Figure 26-2. Serial Programming Timing

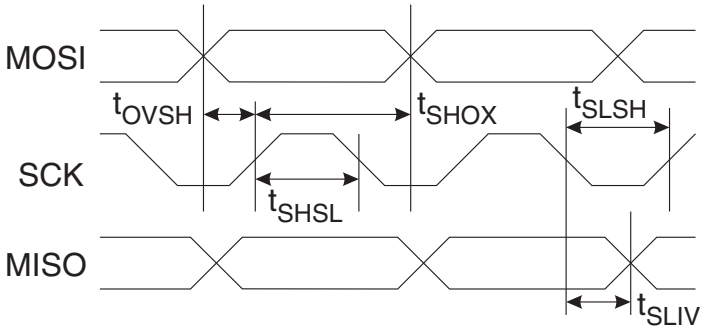


Table 26-8. Serial Programming Characteristics, T_A = -40°C to 85°C, VFET = 2.1 - 6.0V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (ATmega4HVD/8HVD, V_{CC} = 2.2V)	0		4	MHz
t_{CLCL}	Oscillator Period (ATmega4HVD/8HVD, V_{CC} = 2.2V)	50			ns
t_{SHSL}	SCK Pulse Width High	3 t_{CLCL}*			ns
t_{SLSH}	SCK Pulse Width Low	3 t_{CLCL}*			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	TBD	TBD	TBD	ns

26.6.2 High-voltage Serial Programming

Figure 26-3. High-voltage Serial Programming Timing

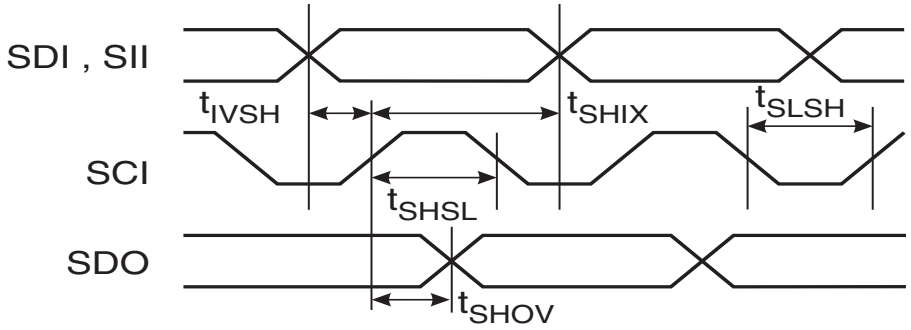


Table 26-9. High-voltage Serial Programming Characteristics T_A = 25°C ± 10%, V_{CC} = 3.5V ± 10% (Unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
t_{SHSL}	SCI Pulse Width High	$1/f_{ck}$			ns
t_{SLSH}	SCI Pulse Width Low	$1/f_{ck}$			ns
t_{IVSH}	SDI, SII Valid to SCI High	50			ns
t_{SHIX}	SDI, SII Hold after SCI High	50			ns
t_{SHOV}	SCI High to SDO Valid		16		ns
t_{WLWH_PFB}	Wait after Instr. 3 for Write Fuse Bits		2.5		ms

27. Typical Characteristics – TBD

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

28. 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	
(0xFD)	BPCR	–	–	EPID	SCD	DOCD	COCD	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	BPOCTR	–	–	OCTR[5:0]						
(0xFA)	BPSCTR	–	SCTR[6:0]							
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
(0xF8)	Reserved	–	–	–	–	–	–	–	–	
(0xF7)	BPCOCD	COCDL[7:0]								
(0xF6)	BPDOCD	DOCDL[7:0]								
(0xF5)	BPSCD	SCDL[7:0]								
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	BPIFR	–	–	–	SCIF	DOCIF	COCIF	–	–	
(0xF2)	BPIMSK	–	–	–	SCIE	DOCIE	COCIE	–	–	
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	FCSR	–	–	–	–	DUVRD	CPS	DFE	CFE	
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	Reserved	–	–	–	–	–	–	–	–	
(0xE8)	Reserved	–	–	–	–	–	–	–	–	
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	Reserved	–	–	–	–	–	–	–	–	
(0xE5)	Reserved	–	–	–	–	–	–	–	–	
(0xE4)	Reserved	–	–	–	–	–	–	–	–	
(0xE3)	Reserved	–	–	–	–	–	–	–	–	
(0xE2)	Reserved	–	–	–	–	–	–	–	–	
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	Reserved	–	–	–	–	–	–	–	–	
(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)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	ROCR	ROCS	–	–	–	–	RSCDEN	RSCWIF	RSCWIE	
(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								
(0x88)	OCR1A	Timer/Counter1 - Output Compare Register A								
(0x87)	Reserved	–	–	–	–	–	–	–	–	
(0x86)	Reserved	–	–	–	–	–	–	–	–	
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	Reserved	–	–	–	–	–	–	–	–	
(0x81)	TCCR1B	–	–	–	–	–	CS12	CS11	CS10	
(0x80)	TCCR1A	TCW1	ICEN1	ICNC1	ICES1	ICS1	–	–	WGM10	
(0x7F)	Reserved	–	–	–	–	–	–	–	–	
(0x7E)	DIDR0	–	–	–	–	–	–	–	PB0DID	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	Reserved	–	–	–	–	–	–	–	–	
(0x7B)	Reserved	–	–	–	–	–	–	–	–	
(0x7A)	ADCSRA	ADEN	ADSC	–	ADIF	ADIE	–	ADMUX1	ADMUX0	
(0x79)	ADCH	–	–	–	–	–	–	ADC9	ADC8	
(0x78)	ADCL	ADC[7:0]								
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	Reserved	–	–	–	–	–	–	–	–	
(0x6F)	TIMSK1	–	–	–	–	ICIE1	OCIE1B	OCIE1A	TOIE1	
(0x6E)	TIMSK0	–	–	–	–	ICIE0	OCIE0B	OCIE0A	TOIE0	
(0x6D)	Reserved	–	–	–	–	–	–	–	–	
(0x6C)	Reserved	–	–	–	–	–	–	–	–	
(0x6B)	Reserved	–	–	–	–	–	–	–	–	
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	
(0x68)	Reserved	–	–	–	–	–	–	–	–	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	FOSCCAL	Fast Oscillator Calibration Register								
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR0	–	–	PRVRM	–	PRSPI	PRTIM1	PRTIM0	PRADC	
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	–	–	CLKPS1	CLKPS0	
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	–	–	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	–	CKOE	PUD	–	–	–	–	
0x34 (0x54)	MCUSR	–	–	–	OCDRF	WDRF	–	EXTRF	PORF	
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	DWDR	debugWIRE Data Register								
0x30 (0x50)	Reserved	–	–	–	–	–	–	–	–	
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	Reserved	–	–	–	–	–	–	–	–	
0x2D (0x4D)	Reserved	–	–	–	–	–	–	–	–	
0x2C (0x4C)	Reserved	–	–	–	–	–	–	–	–	
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								
0x29 (0x49)	OCR0B	Timer/Counter0 - Output Compare Register A								
0x28 (0x48)	OCR0A	Timer/Counter0 - Output Compare Register B								
0x27 (0x47)	TCNT0H	Timer/Counter0 - Counter Register High Byte								
0x26 (0x46)	TCNT0L	Timer/Counter0 - Counter Register Low Byte								
0x25 (0x45)	TCCR0B	–	–	–	–	–	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	TCW0	ICEN0	ICNC0	ICES0	ICS0	–	–	WGM00	
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	–	PSR	
0x22 (0x42)	Reserved	–	–	–	–	–	–	–	–	
0x21 (0x41)	EEARL	EEPROM Address Register								
0x20 (0x40)	EEDR	EEPROM Data Register								
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	

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	
0x16 (0x36)	TIFR1	–	–	–	–	ICF1	OCF1B	OCF1A	TOV1	
0x15 (0x35)	TIFR0	–	–	–	–	ICF0	OCF0B	OCF0A	TOV0	
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	–	–	–	–	–	–	PORTC1	PORTC0	
0x07 (0x27)	Reserved	–	–	–	–	–	–	–	–	
0x06 (0x26)	PINC	–	–	–	–	–	–	PINC1	PINC0	
0x05 (0x25)	PORTB	–	–	–	–	–	PORTB2	PORTB1	PORTB0	
0x04 (0x24)	DDRB	–	–	–	–	–	DDB2	DDB1	DDB0	
0x03 (0x23)	PINB	–	–	–	–	–	PINB2	PINB1	PINB0	
0x02 (0x22)	Reserved	–	–	–	–	–	–	–	–	
0x01 (0x21)	Reserved	–	–	–	–	–	–	–	–	
0x00 (0x20)	Reserved	–	–	–	–	–	–	–	–	

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 ATmega4HVD/8HVD 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.

29. 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	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	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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
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
OUT	P, Rr	Out Port	P ← Rr	None	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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

30. Ordering Information

30.1 ATmega4HVD

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
1 - 4 MHz	2.0 - 2.4V	ATmega4HVD-4MX	18M1	-20 - 85°C

Note: This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

Package Type	
18M1	18-pad (Staggered Dual-row) 6.5 x 3.5 x 0.80 mm Body. 3.20 x 2.00 mm Exposed Pad, (MLF)

30.2 ATmega8HVD

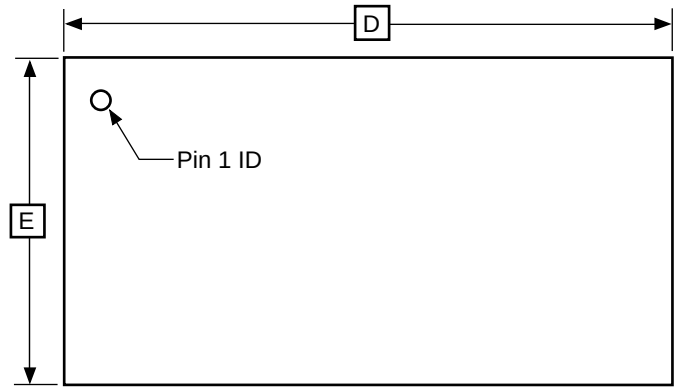
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
1 - 4 MHz	2.0 - 2.4V	ATmega8HVD-4MX	18M1	-20 - 85°C

Note: This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

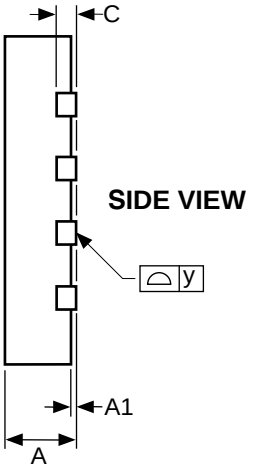
Package Type	
18M1	18-pad (Staggered Dual-row) 6.5 x 3.5 x 0.80 mm Body. 3.20 x 2.00 mm Exposed Pad, (MLF)

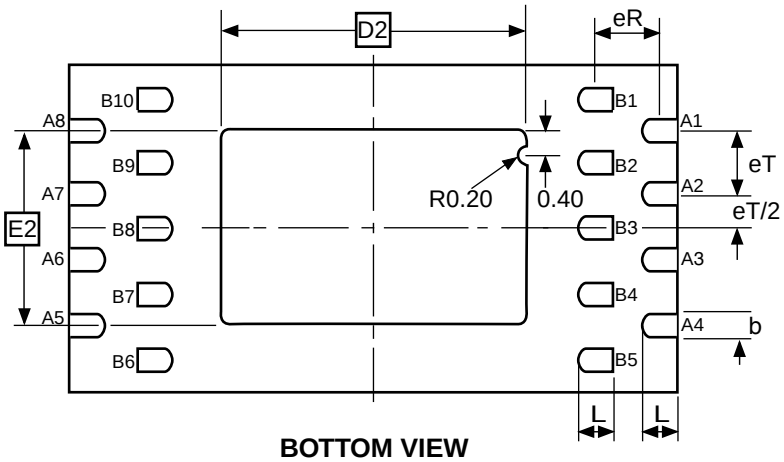
31. Packaging Information

31.1 18M1



TOP VIEW





BOTTOM VIEW

Note: 1. The terminal #1 ID is a Laser-marked Feature.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.70	0.75	0.80	
A1	0.00	0.02	0.05	
b	0.23	0.28	0.33	
C		0.20 REF		
D	6.40	6.50	6.60	
D2	3.15	3.20	3.25	
E	3.40	3.50	3.60	
E2	1.95	2.00	2.05	
eT	–	0.70	–	
eR	–	0.70	–	
L	0.35	0.40	0.45	
y	0.00	–	0.075	

3/21/07

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**18M1,** 18-pad (Staggered Dual-row), 6.5 x 3.5 x 0.80 mm Body, 3.20 x 2.00 mm Exposed Pad, MicroLeadFrame® Package (MLF)	18M1	B

32. Errata

32.1 ATmega4HVD

32.1.1 All revisions

No known errata.

32.2 ATmega8HVD

32.2.1 All revisions

No known errata.

33. Datasheet Revision History

33.1 Rev. B - 09/08

1. Updated Table 20-2 on page 110 and Table 20-3 on page 111 in the Register summary of section of ”Battery Protection” on page 104.

33.2 Rev. A - 09/08

1. Initial revision.

Table of Contents

Features *1*

1 Pin Configurations *2*

1.1 Pin Descriptions 3

2 Overview *4*

3 Resources *5*

4 Data Retention *5*

5 About Code Examples *5*

6 AVR CPU Core *6*

6.1 Overview 6

6.2 ALU – Arithmetic Logic Unit 7

6.3 Status Register 7

6.4 General Purpose Register File 9

6.5 Stack Pointer 10

6.6 Instruction Execution Timing 11

6.7 Reset and Interrupt Handling 11

7 AVR Memories *14*

7.1 Overview 14

7.2 In-System Reprogrammable Flash Program Memory 14

7.3 SRAM Data Memory 15

7.4 EEPROM Data Memory 16

7.5 I/O Memory 16

7.6 Register Description 17

8 System Clock and Clock Options *22*

8.1 Clock Systems and their Distribution 22

8.2 Clock Sources 23

8.3 Calibrated Fast RC Oscillator 23

8.4 Slow RC Oscillator 24

8.5 Ultra Low Power RC Oscillator 24

8.6 CPU, I/O, Flash, and ADC Clock 24

8.7 Watchdog Timer and Battery Protection 24

8.8 Clock Startup Sequence 24

8.9 Clock Output 25

8.10System Clock Prescaler 25
8.11ADC Clock Prescaler 26
8.12OSI – Oscillator Sampling Interface 27
8.13Register Description 29

9 Power Management and Sleep Modes 32

9.1Sleep Modes 32
9.2Idle Mode 34
9.3ADC Noise Reduction 34
9.4Power-save Mode 34
9.5Power-off Mode 35
9.6Power Reduction Register 35
9.7Minimizing Power Consumption 35
9.8Register Description 36

10 System Control and Reset 38

10.1Resetting the AVR 38
10.2Reset Sources 38
10.3External Reset 40
10.4Watchdog Reset 40
10.5Black-out Detection 40
10.6ATmega4HVD/8HVD Start-up Sequence 42
10.7Watchdog Timer 45
10.8Register Description 48

11 Interrupts 51

11.1Interrupt Vectors in ATmega4HVD/8HVD 51

12 External Interrupt 53

12.1Register Description 53

13 High Voltage I/O Ports 56

13.1High Voltage Ports as General Digital Outputs 57
13.2Alternate Port Functions 57
13.3Register Description 60

14 Low Voltage I/O-Ports 61

14.1Overview 61
14.2Low Voltage Ports as General Digital I/O 62
14.3Alternate Port Functions 66

14.4Register Description 70

15 Timer/Counter0 and Timer/Counter1 Prescalers 71

15.1Overview 71
15.2Internal Clock Source 71
15.3Prescaler Reset 71
15.4External Clock Source 72
15.5Register Description 72

16 Timer/Counter(T/C0,T/C1) 74

16.1Features 74
16.2Overview 74
16.3Timer/Counter Clock Sources 75
16.4Counter Unit 75
16.5Modes of Operation 76
16.6Input Capture Unit 78
16.7Output Compare Unit 80
16.8Timer/Counter Timing Diagrams 81
16.9Accessing Registers in 16-bit Mode 82
16.10Register Description 86

17 ADC - Analog-to-Digital Converter 90

17.1Features 90
17.2Operation 91
17.3Starting a Conversion 91
17.4Conversion Timing 91
17.5ADC Voltage Reference 92
17.6ADC Noise Canceler 92
17.7ADC Conversion Result 96
17.8Register Description 97

18 Voltage Reference 99

19 Voltage Regulator 100

19.1Features 100
19.2Overview 100
19.3Battery Pack Short mode 101
19.4Regulator Force mode 101
19.5Register Description 102

20 Battery Protection 104

20.1Features 104

20.2Overview 104

20.3Short-circuit Protection 105

20.4Discharge Over-current Protection 105

20.5Charge Over-current Protection 105

20.6External Protection Input 106

20.7Battery Protection CPU Interface 108

20.8Register Description 108

21 FET Control 114

21.1Overview 114

21.2FET Driver 115

21.3DUVR – Deep Under-Voltage Recovery Mode operation 117

21.4Register Description 117

22 debugWIRE On-chip Debug System 119

22.1Features 119

22.2Overview 119

22.3Physical Interface 119

22.4Software Break Points 120

22.5Limitations of debugWIRE 120

22.6Register Description 120

23 Self-Programming the Flash 121

23.1Overview 121

23.2Performing Page Erase by SPM 121

23.3Filling the Temporary Buffer (Page Loading) 121

23.4Performing a Page Write 122

23.5Addressing the Flash During Self-Programming 122

23.6Register Description 127

24 Memory Programming 129

24.1Program And Data Memory Lock Bits 129

24.2Fuse Bits 129

24.3Signature Bytes 130

24.4Calibration Bytes 131

24.5Page Size 131

24.6Serial Downloading 131

24.7High-voltage Serial Programming 135
24.8High-voltage Serial Programming Algorithm 136

25 Operating Circuit 141

26 Electrical Characteristics 142

26.1DC Characteristics 142
26.2Oscillator Characteristics 143
26.3System and Reset Characteristics 144
26.4External Interrupts Characteristics 144
26.5General I/O Lines Characteristics 145
26.6Programming Characteristics 148

27 Typical Characteristics – TBD 150

28 Register Summary 151

29 Instruction Set Summary 155

30 Ordering Information 158

30.1ATmega4HVD 158
30.2ATmega8HVD 159

31 Packaging Information 160

31.118M1 160

32 Errata 161

32.1ATmega4HVD 161
32.2ATmega8HVD 161

33 Datasheet Revision History 162

33.1Rev. B - 09/08 162
33.2Rev. A - 09/08 162

Table of Contents i