Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atmega168-15az

5. AVR ATmega48/88/168 Memories

This section describes the different memories in the Atmel® ATmega48/88/168. The AVR® architecture has two main memory spaces, the data memory and the program memory space. In addition, the Atmel ATmega48/88/168 features an EEPROM memory for data storage. All three memory spaces are linear and regular.

5.1 In-System Reprogrammable Flash Program Memory

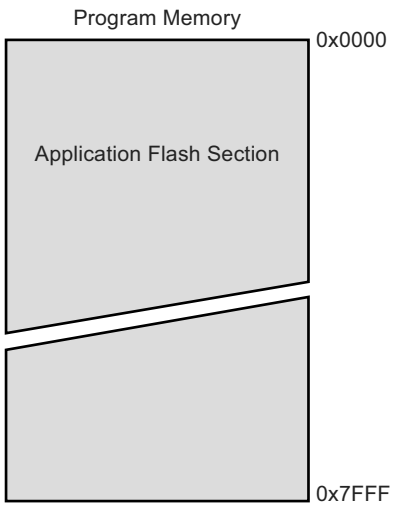
The Atmel ATmega48/88/168 contains 4/8/16K bytes on-chip in-system reprogrammable flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the flash is organized as 2/4/8K x 16. For software security, the flash program memory space is divided into two sections, boot loader section and application program section in Atmel ATmega88 and ATmega168. ATmega48 does not have separate boot loader and application program sections, and the SPM instruction can be executed from the entire flash. See SELFPRGEN description in Section 23.4.1 “Store Program Memory Control and Status Register – SPMCSR” on page 225 and Section 24.5.1 “Store Program Memory Control and Status Register – SPMCSR” on page 233 for more details.

The flash memory has an endurance of at least 75,000 write/erase cycles. The Atmel ATmega48/88/168 program counter (PC) is 11/12/13 bits wide, thus addressing the 2/4/8K program memory locations. The operation of boot program section and associated boot lock bits for software protection are described in detail in Section 23. “Self-Programming the Flash, ATmega48” on page 223 and Section 24. “Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168” on page 229. Section 25. “Memory Programming” on page 242 contains a detailed description on flash programming in SPI- or parallel programming mode.

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 Section 4.7 “Instruction Execution Timing” on page 13.

Figure 5-1. Program Memory Map, ATmega48

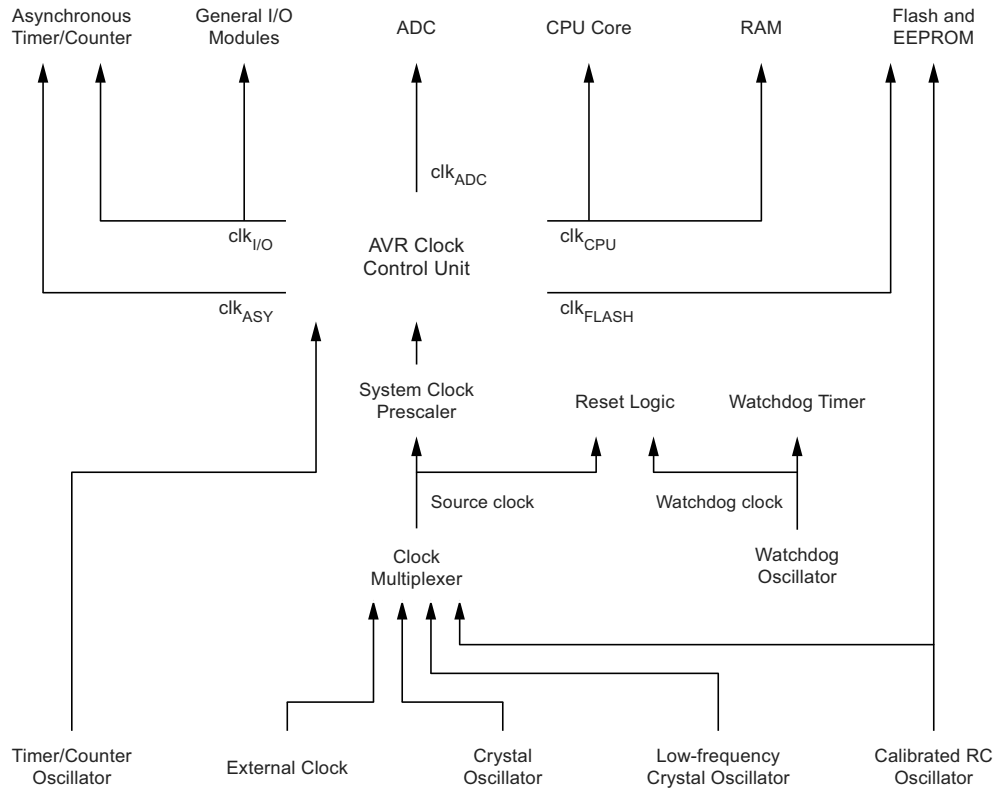


6. System Clock and Clock Options

6.1 Clock Systems and their Distribution

Figure 6-1 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 Section 7. “Power Management and Sleep Modes” on page 33. The clock systems are detailed below.

Figure 6-1. Clock Distribution



6.1.1 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.

6.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. 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. Also note that start condition detection in the USI module is carried out asynchronously when $clk_{I/O}$ is halted, TWI address recognition in all sleep modes.

6.1.3 Flash Clock – clk_{FLASH}

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

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

Table 8-5. Watchdog Timer Configuration

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

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

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

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

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

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

- **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 running. The different prescaling values and their corresponding time-out periods are shown in Table 8-6.

Table 8-6. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16ms
0	0	0	1	4K (4096) cycles	32ms
0	0	1	0	8K (8192) cycles	64ms
0	0	1	1	16K (16384) cycles	0.125s
0	1	0	0	32K (32768) cycles	0.25s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0s
0	1	1	1	256K (262144) cycles	2.0s
1	0	0	0	512K (524288) cycles	4.0s
1	0	0	1	1024K (1048576) cycles	8.0s
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		

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the status register is unaffected by the automatic disabling.

Note: If interrupt vectors are placed in the boot loader section and boot lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and boot lock bit BLB12 is programed, interrupts are disabled while executing from the boot loader section. Refer to Section 24. “Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168” on page 229 for details on boot lock bits.

This bit is not available in Atmel® ATmega48.

- **Bit 0 – IVCE: Interrupt Vector Change Enable**

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See code example below.

Assembly Code Example

```
Move_interrupts:
        ; Enable change of Interrupt Vectors
        ldi     r16, (1<<IVCE)
        out     MCUCR, r16
        ; Move interrupts to Boot Flash section
        ldi     r16, (1<<IVSEL)
        out     MCUCR, r16
        ret
```

C Code Example

```
void Move_interrupts(void)
{
        /* Enable change of Interrupt Vectors */
        MCUCR = (1<<IVCE);
        /* Move interrupts to Boot Flash section */
        MCUCR = (1<<IVSEL);
}
```

This bit is not available in Atmel ATmega48.

- **Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in EIMSK are set (one), the MCU will jump to the corresponding 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 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding 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 INT0 is configured as a level interrupt.

11.4 Pin Change Interrupt Control Register - PCICR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
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..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega48/88/168, and will always read as zero.

- **Bit 2 - PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI2 interrupt vector. PCINT23..16 pins are enabled individually by the PCMSK2 register.

- **Bit 1 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT14..8 pins are enabled individually by the PCMSK1 register.

- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

11.5 Pin Change Interrupt Flag Register - PCIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
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..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega48/88/168, and will always read as zero.

- **Bit 2 - PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT23..16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in PCICR are set (one), the MCU will jump to the corresponding 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.

- **Bit 1 - PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT14..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding 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.

- **Bit 0 - PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding 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.

11.6 Pin Change Mask Register 2 – PCMSK2

Bit	7	6	5	4	3	2	1	0	
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
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..0 – PCINT23..16: Pin Change Enable Mask 23..16**

Each PCINT23..16-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.7 Pin Change Mask Register 1 – PCMSK1

Bit	7	6	5	4	3	2	1	0	
	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	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 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega48/88/168, and will always read as zero.

- **Bit 6..0 – PCINT14..8: Pin Change Enable Mask 14..8**

Each PCINT14..8-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.8 Pin Change Mask Register 0 – PCMSK0

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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..0 – PCINT7..0: Pin Change Enable Mask 7..0**

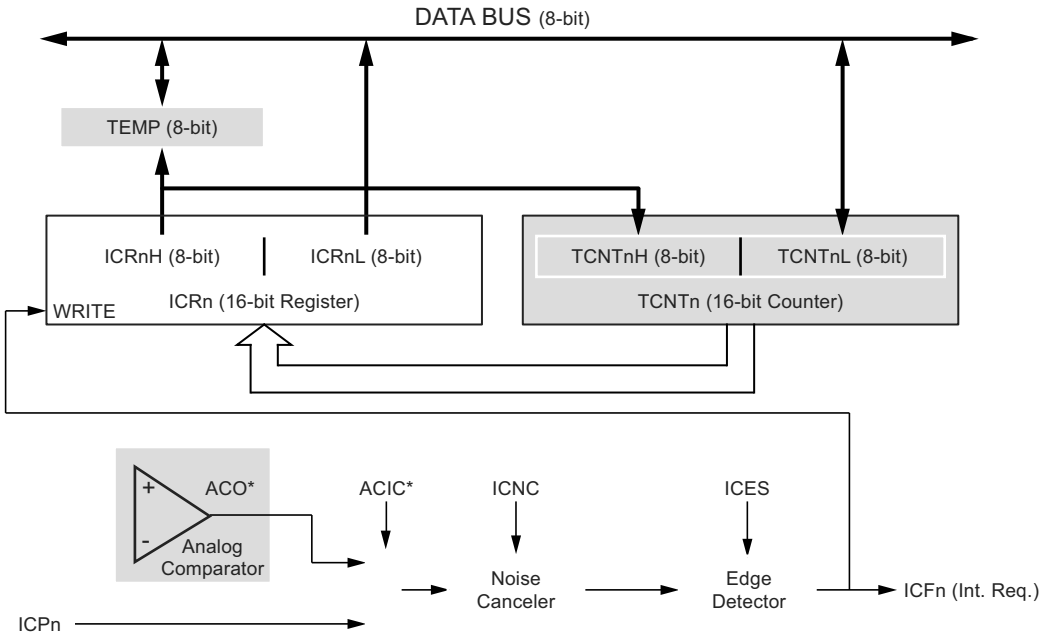
Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

14.5 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 indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the analog-comparator unit. 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 14-3. The elements of the block diagram that are not directly a part of the input capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

Figure 14-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the input capture pin (ICP1), alternatively on the analog comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the input capture register (ICR1). The input capture flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (ICIE1 = 1), the Input Capture flag generates an input capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the input capture register (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP register.

The ICR1 register can only be written when using a waveform generation mode that utilizes the ICR1 register for defining the counter’s TOP value. In these cases the waveform generation mode (WGM13:0) bits must be set before the TOP value can be written to the ICR1 register. When writing the ICR1 register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to Section 14.2 “Accessing 16-bit Registers” on page 96.

Figure 14-11 shows the same timing data, but with the prescaler enabled.

Figure 14-11.Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)

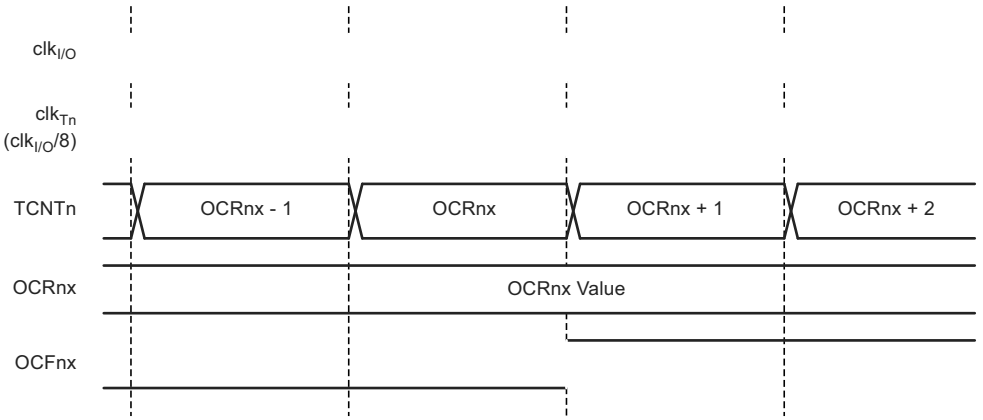


Figure 14-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 14-12.Timer/Counter Timing Diagram, no Prescaling

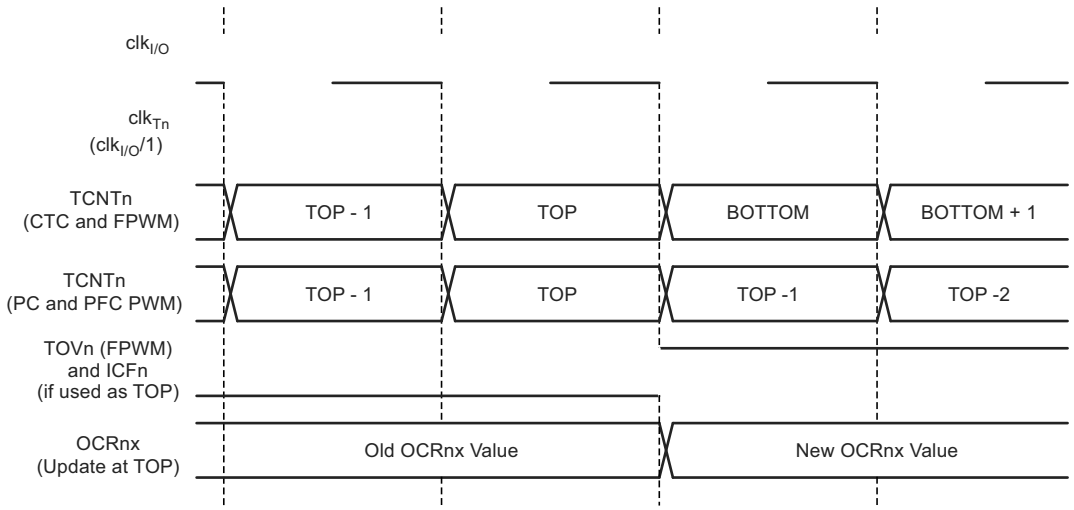


Table 14-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

14.10.3 Timer/Counter1 Control Register C – TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in clear timer on compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

14.10.4 Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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 two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for 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 Section 14.2 “Accessing 16-bit Registers” on page 96.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x registers.

Writing to the TCNT1 register blocks (removes) the compare match on the following timer clock for all compare units.

Table 17-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 17-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRR}n + 1)}$	$\boldsymbol{UBRR}n = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(\boldsymbol{UBRR}n + 1)}$	$\boldsymbol{UBRR}n = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRR}n + 1)}$	$\boldsymbol{UBRR}n = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL registers, (0-4095)

Some examples of UBRRn values for some system clock frequencies are found in Table 17-9 on page 165.

17.2.2 Double Speed Operation (U2Xn)

The transfer rate can be doubled by setting the U2Xn bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the transmitter, there are no downsides.

17.2.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 17-2 on page 148 for details.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCKn clock frequency is limited by the following equation:

$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

17.2.4 Synchronous Clock Operation

When synchronous mode is used (UMSELn = 1), the XCKn pin will be used as either clock input (slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxDn) is sampled at the opposite XCKn clock edge of the edge the data output (TxDn) is changed.

18.4.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the transmitter and the receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the global interrupt flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is no ongoing transmissions during the period the registers are changed. The TXCn flag can be used to check that the transmitter has completed all transfers, and the RXCn flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example[1]

```
USART_Init:
        clr r18
        out UBRRnH,r18
        out UBRRnL,r18
        ; Setting the XCKn port pin as output, enables master mode.
        sbi XCKn_DDR, XCKn
        ; Set MSPI mode of operation and SPI data mode 0.
        ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
        out UCSRnC,r18
        ; Enable receiver and transmitter.
        ldi r18, (1<<RXENn)|(1<<TXENn)
        out UCSRnB,r18
        ; Set baud rate.
        ; IMPORTANT: The Baud Rate must be set after the transmitter is
                     enabled!
        out UBRRnH, r17
        out UBRRnL, r18
        ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
        UBRRn = 0;
        /* Setting the XCKn port pin as output, enables master mode. */
        XCKn_DDR |= (1<<XCKn);
        /* Set MSPI mode of operation and SPI data mode 0. */
        UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
        /* Enable receiver and transmitter. */
        UCSRnB = (1<<RXENn)|(1<<TXENn);
        /* Set baud rate. */
        /* IMPORTANT: The Baud Rate must be set after the transmitter
                      is enabled   */
        UBRRn = baud;
}
```

Note: 1. The example code assumes that the part specific header file is included. For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

Table 19-2. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see Section 19.5.2 “Bit Rate Generator Unit” on page 181. The value of TWPS1..0 is used in the equation.

19.6.4 TWI Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
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	

In transmit mode, TWDR contains the next byte to be transmitted. In receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI interrupt flag (TWINT) is set by hardware. Note that the data register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined.

In the case of a lost bus arbitration, no data is lost in the transition from master to slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

- **Bits 7..0 – TWD: TWI Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire serial bus.

19.6.5 TWI (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
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	0	

The TWAR should be loaded with the 7-bit slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a slave transmitter or receiver, and not needed in the master modes. In multi master systems, TWAR must be set in masters which can be addressed as slaves by other masters.

The LSB of TWAR is used to enable recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bits 7..1 – TWA: TWI (Slave) Address Register**

These seven bits constitute the slave address of the TWI unit.

- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a general call given over the 2-wire serial bus.

20.3 Analog Comparator Multiplexed Input

It is possible to select any of the ADC7..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 20-2. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the analog comparator.

Table 20-2. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

20.3.1 Digital Input Disable Register 1 – DIDR1

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	AIN1D	AIN0D	DIDR1
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 unused bits in the Atmel® ATmega48/88/168, and will always read as zero.

- **Bit 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

21.5.2 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:

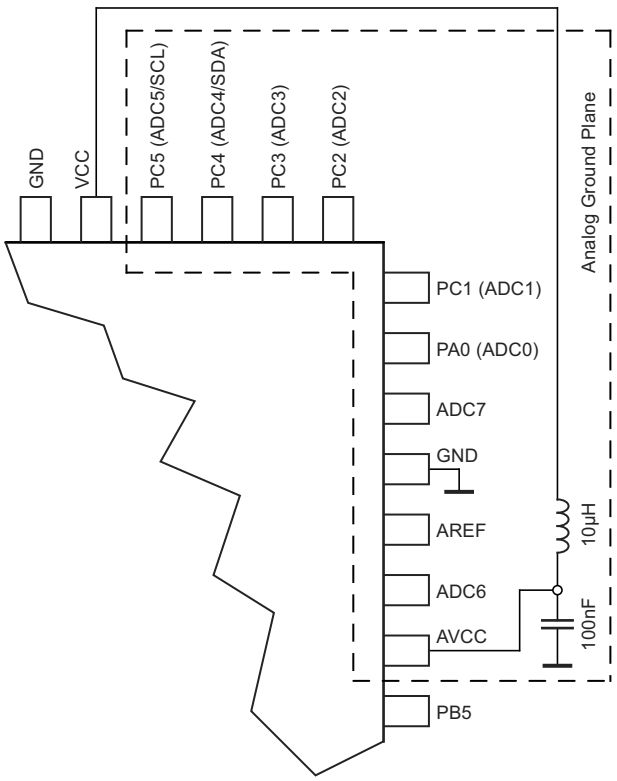
a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. The AV_{CC} pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 21-9.

c. Use the ADC noise canceler function to reduce induced noise from the CPU.

d. If any ADC [3..0] port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress. However, using the 2-wire interface (ADC4 and ADC5) will only affect the conversion on ADC4 and ADC5 and not the other ADC channels.

Figure 21-9. ADC Power Connections



25.7.9 Programming the Fuse High Bits

The algorithm for programming the use high bits is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and data loading):

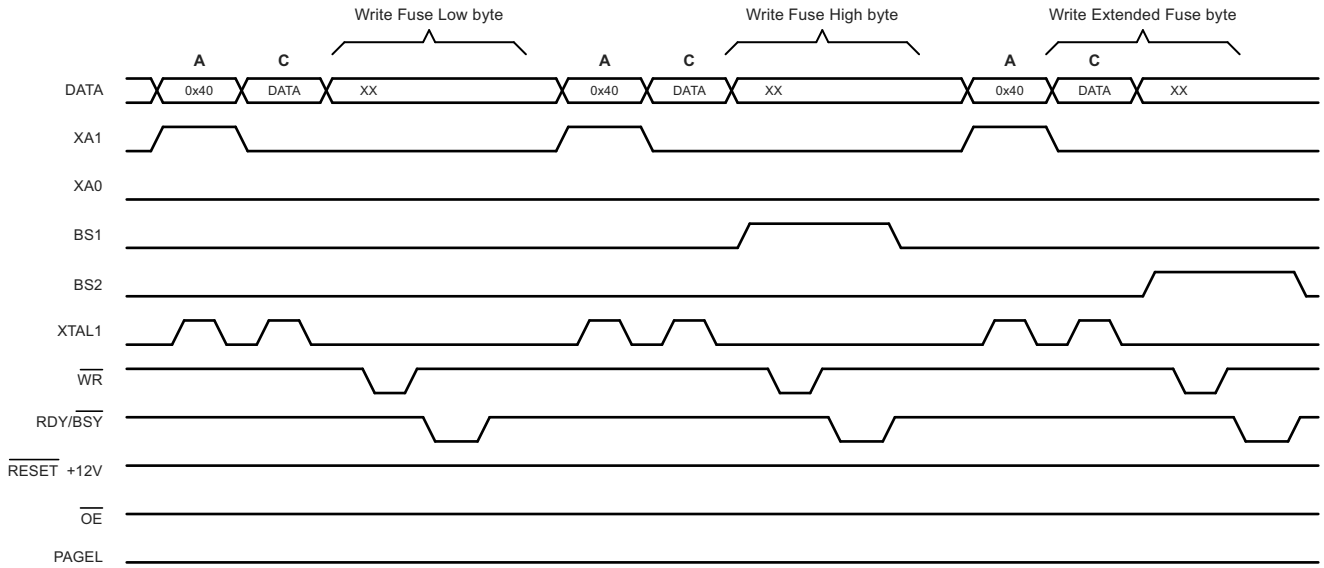
1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

25.7.10 Programming the Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and data loading):

1. 1. A: Load command “0100 0000”.
2. 2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. 3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. 4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. 5. Set BS2 to “0”. This selects low data byte.

Figure 25-5. Programming the FUSES Waveforms



25.7.11 Programming the Lock Bits

The algorithm for programming the lock bits is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and data loading):

1. A: Load command “0010 0000”.
2. C: Load data low byte. Bit n = “0” programs the lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the boot lock bits by any external programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

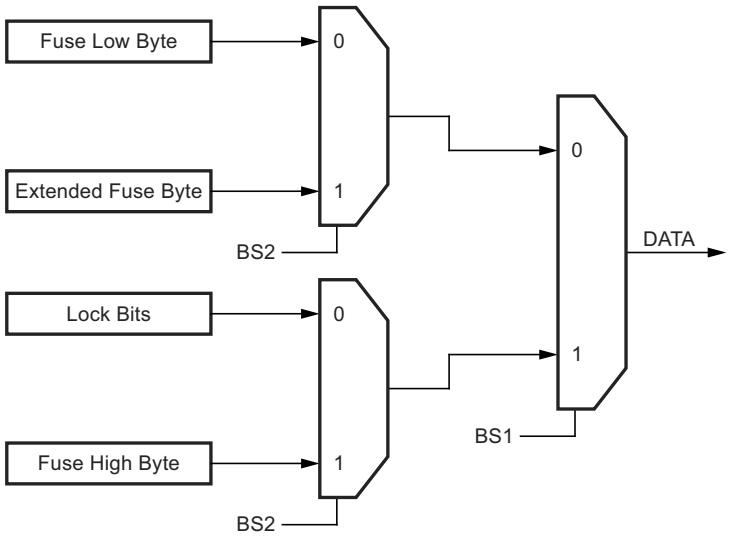
The lock bits can only be cleared by executing chip erase.

25.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the fuse and lock bits is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command loading):

1. A: Load command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the fuse low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the fuse high bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the extended fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 25-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



25.7.13 Reading the Signature Bytes

The algorithm for reading the signature bytes is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and address loading):

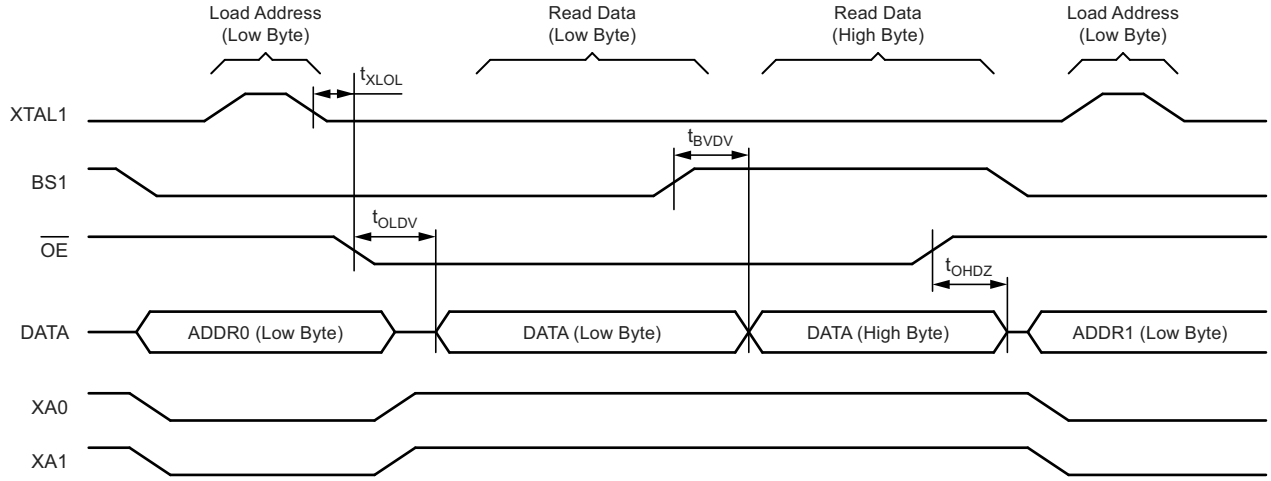
1. A: Load command “0000 1000”.
2. B: Load address low byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

25.7.14 Reading the Calibration Byte

The algorithm for reading the calibration byte is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and address loading):

1. A: Load command “0000 1000”.
2. B: Load address low byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

Figure 25-9. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 25-7 on page 254 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

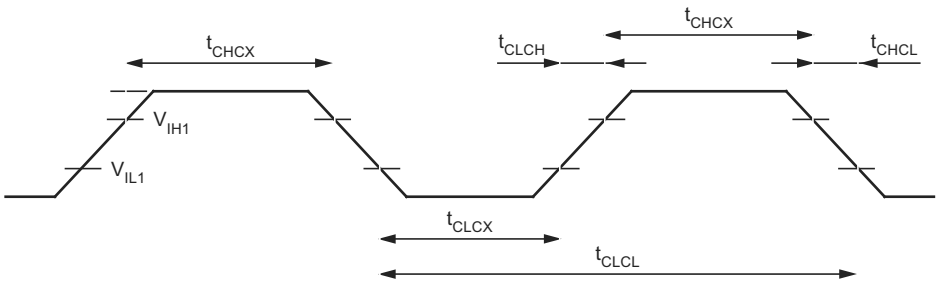
Table 25-15. Parallel Programming Characteristics, V_{CC} = 5V ±10%

Parameter	Symbol	Min	Typ	Max	Unit
Programming enable voltage	V_{PP}	11.5		12.5	V
Programming enable current	I_{PP}			250	µA
Data and control valid before XTAL1 high	t_{DVXH}	67			ns
XTAL1 low to XTAL1 high	t_{XLXH}	200			ns
XTAL1 pulse width high	t_{XHXL}	150			ns
Data and control hold after XTAL1 low	t_{XLDX}	67			ns
XTAL1 low to $\overline{WR}$ low	t_{XLWL}	0			ns
XTAL1 low to PAGEL high	t_{XLPH}	0			ns
PAGEL low to XTAL1 high	t_{PLXH}	150			ns
BS1 valid before PAGEL high	t_{BVPH}	67			ns
PAGEL pulse width high	t_{PHPL}	150			ns
BS1 hold after PAGEL low	t_{PLBX}	67			ns
BS2/1 hold after $\overline{WR}$ low	t_{WLBX}	67			ns
PAGEL low to $\overline{WR}$ low	t_{PLWL}	67			ns
BS1 valid to $\overline{WR}$ low	t_{BVWL}	67			ns
$\overline{WR}$ pulse width low	t_{WLWH}	150			ns
$\overline{WR}$ low to RDY/$\overline{BSY}$ low	t_{WLRL}	0		1	µs
$\overline{WR}$ low to RDY/$\overline{BSY}$ high(1)	t_{WLRH}	3.7		4.5	ms
$\overline{WR}$ Low to RDY/$\overline{BSY}$ high for chip erase(2)	t_{WLRH_CE}	7.5		9	ms
XTAL1 low to $\overline{OE}$ low	t_{XLOL}	0			ns
BS1 valid to DATA valid	t_{BVDV}	0		250	ns
$\overline{OE}$ low to DATA valid	t_{OLDV}			250	ns
$\overline{OE}$ high to DATA tri-stated	t_{OHDZ}			250	ns

Notes: 1. t_{WLRH} is valid for the write flash, write EEPROM, write fuse bits and write lock bits commands.
2. t_{WLRH_CE} is valid for the chip erase command.

26.3 External Clock Drive Waveforms

Figure 26-1. External Clock Drive Waveforms



26.4 External Clock Drive

Table 26-1. External Clock Drive

Parameter	Symbol	V_{CC}=2.7 to 5.5V		V_{CC}=4.5 to 5.5V		Unit
		Min.	Max.	Min.	Max.	
Oscillator frequency	$1/t_{CLCL}$	0	8	0	16	MHz
Clock period	t_{CLCL}	125		62.5		ns
High time	t_{CHCX}	50		25		ns
Low time	t_{CLCX}	50		25		ns
Rise time	t_{CLCH}		1.6		0.5	µs
Fall time	t_{CHCL}		1.6		0.5	µs
Change in period from one clock cycle to the next	Dt_{CLCL}		2		2	%

26.5 Maximum Speed versus V_{CC}

Maximum frequency is dependent on $V_{CC.}$ As shown in Figure 26-2, the maximum frequency versus V_{CC} curve is linear between 2.7V < V_{CC} < 4.5V.

Figure 26-2. Maximum Frequency versus V_{CC}, ATmega48/88/168

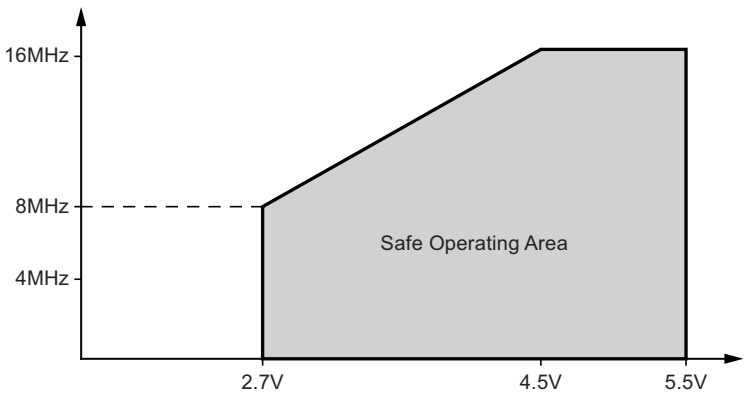


Figure 28-9. Output High Voltage versus Output High Current (V_{CC} = 3V)

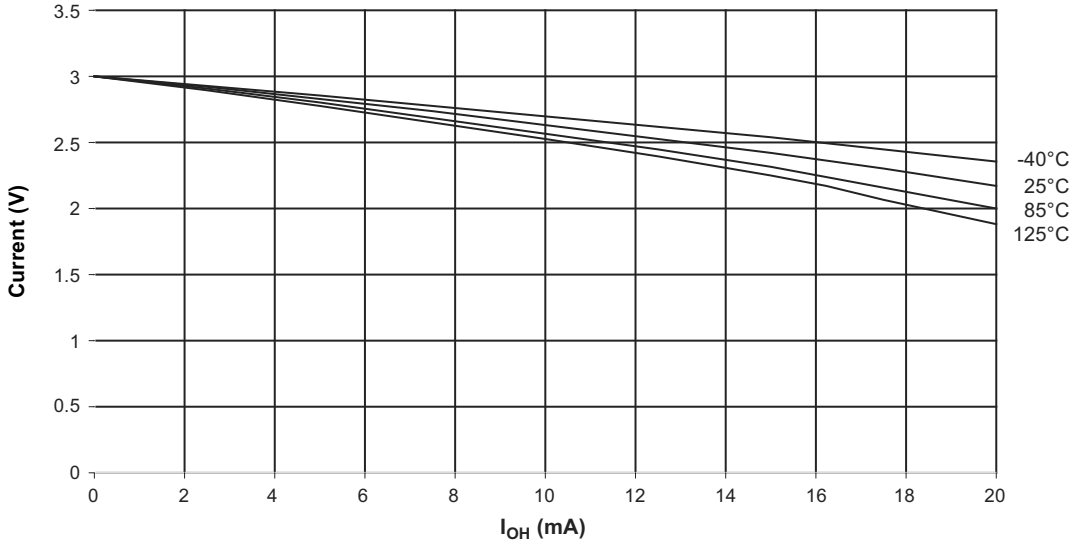
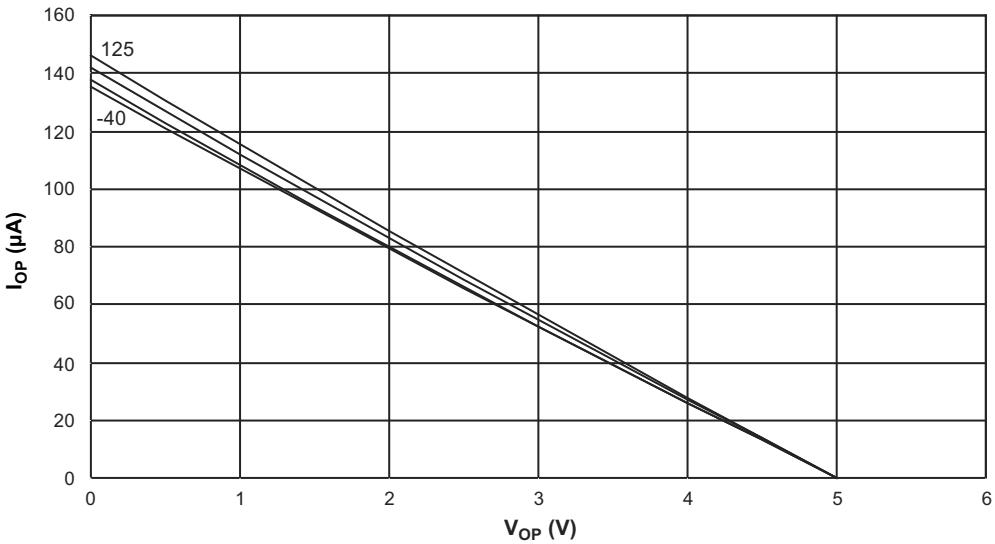


Figure 28-10. Reset Pull-up Resistor Current versus Reset Pin Voltage (V_{CC} = 5V)



28.1.7 Peripheral Units

Figure 28-27. Analog to Digital Converter GAIN versus V_{CC}

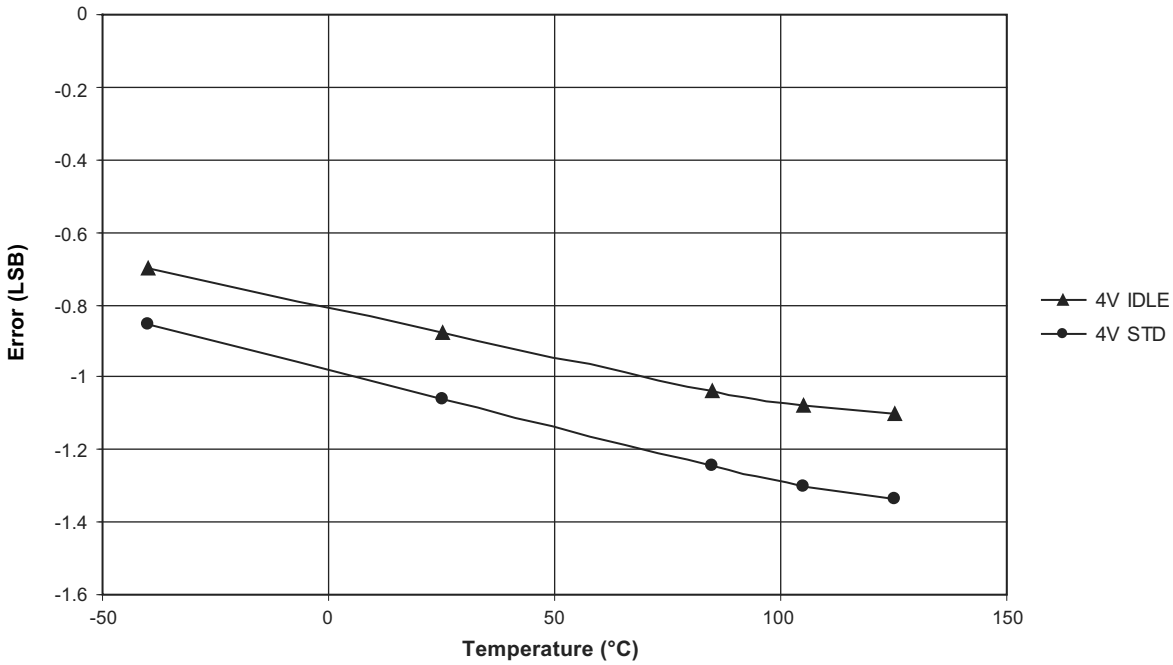


Figure 28-28. Analog to Digital Converter OFFSET versus V_{CC}