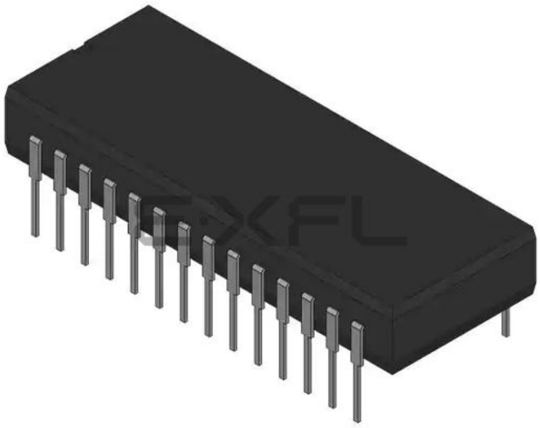


Welcome to E-XFL.COM

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	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	4KB (2K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atmega48-20pi

Table 9-3. Reset and Interrupt Vectors Placement in ATmega88[(1)]

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

Note: 1. The Boot Reset Address is shown in Table 24-6 on page 276. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega88 is:

```
Address  Labels Code                 Comments
0x000          rjmp   RESET          ; Reset Handler
0x001          rjmp   EXT_INT0       ; IRQ0 Handler
0x002          rjmp   EXT_INT1       ; IRQ1 Handler
0x003          rjmp   PCINT0         ; PCINT0 Handler
0x004          rjmp   PCINT1         ; PCINT1 Handler
0x005          rjmp   PCINT2         ; PCINT2 Handler
0x006          rjmp   WDT            ; Watchdog Timer Handler
0x007          rjmp   TIM2_COMPA     ; Timer2 Compare A Handler
0X008          rjmp   TIM2_COMPB     ; Timer2 Compare B Handler
0x009          rjmp   TIM2_OVF       ; Timer2 Overflow Handler
0x00A          rjmp   TIM1_CAPT      ; Timer1 Capture Handler
0x00B          rjmp   TIM1_COMPA     ; Timer1 Compare A Handler
0x00C          rjmp   TIM1_COMPB     ; Timer1 Compare B Handler
0x00D          rjmp   TIM1_OVF       ; Timer1 Overflow Handler
0x00E          rjmp   TIM0_COMPA     ; Timer0 Compare A Handler
0x00F          rjmp   TIM0_COMPB     ; Timer0 Compare B Handler
0x010          rjmp   TIM0_OVF       ; Timer0 Overflow Handler
0x011          rjmp   SPI_STC        ; SPI Transfer Complete Handler
0x012          rjmp   USART_RXC      ; USART, RX Complete Handler
0x013          rjmp   USART_UDRE     ; USART, UDR Empty Handler
0x014          rjmp   USART_TXC      ; USART, TX Complete Handler
0x015          rjmp   ADC            ; ADC Conversion Complete Handler
0x016          rjmp   EE_RDY         ; EEPROM Ready Handler
0x017          rjmp   ANA_COMP       ; Analog Comparator Handler
0x018          rjmp   TWI            ; 2-wire Serial Interface Handler
0x019          rjmp   SPM_RDY        ; Store Program Memory Ready Handler
;
0x01ARESET:    ldi    r16, high(RAMEND); Main program start
0x01B          out    SPH,r16        ; Set Stack Pointer to top of RAM
0x01C          ldi    r16, low(RAMEND)
0x01D          out    SPL,r16
0x01E          sei                   ; Enable interrupts
0x01F          <instr>  xxx
```

When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

OC1B, Output Compare Match output: The PB2 pin can serve as an external output for the Timer/Counter1 Compare Match B. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT2: Pin Change Interrupt source 2. The PB2 pin can serve as an external interrupt source.

- **OC1A/PCINT1 – Port B, Bit 1**

OC1A, Output Compare Match output: The PB1 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT1: Pin Change Interrupt source 1. The PB1 pin can serve as an external interrupt source.

- **ICP1/CLKO/PCINT0 – Port B, Bit 0**

ICP1, Input Capture Pin: The PB0 pin can act as an Input Capture Pin for Timer/Counter1.

CLKO, Divided System Clock: The divided system clock can be output on the PB0 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB0 and DDB0 settings. It will also be output during reset.

PCINT0: Pin Change Interrupt source 0. The PB0 pin can serve as an external interrupt source.

Table 10-4 and Table 10-5 relate the alternate functions of Port B to the overriding signals shown in Figure 10-5 on page 69. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Figure 12-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

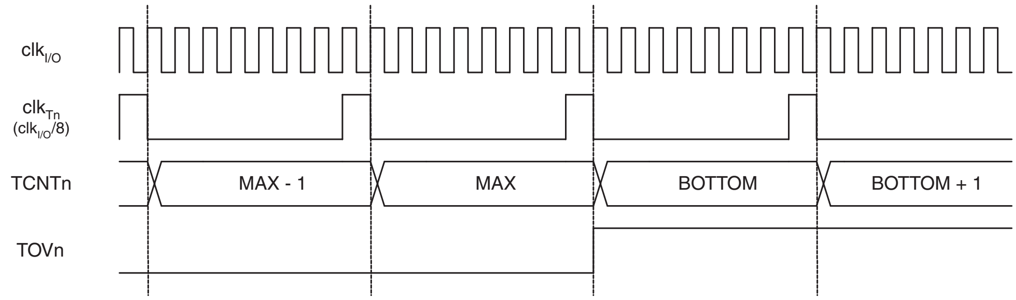


Figure 12-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 12-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

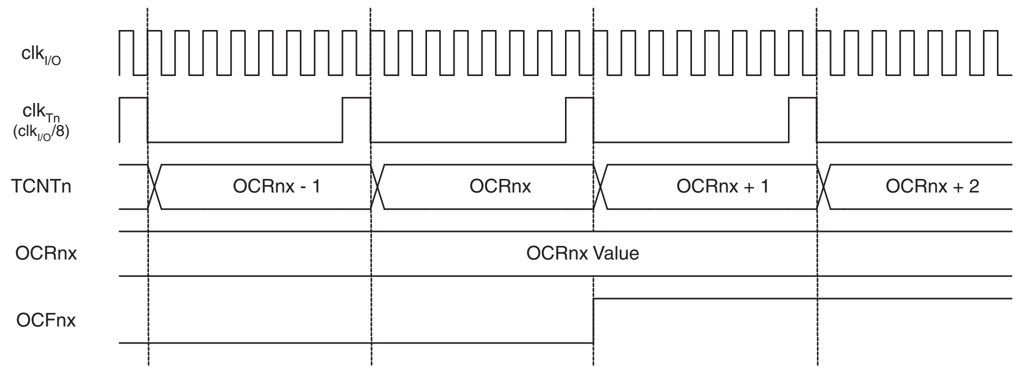


Figure 12-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 12-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)

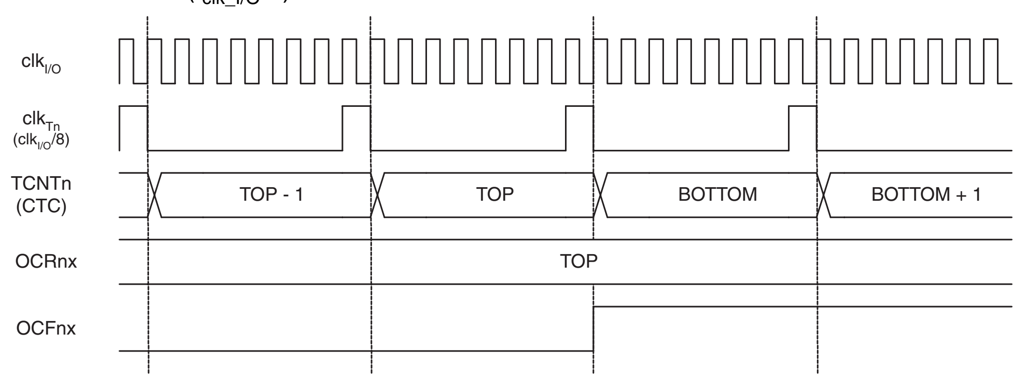
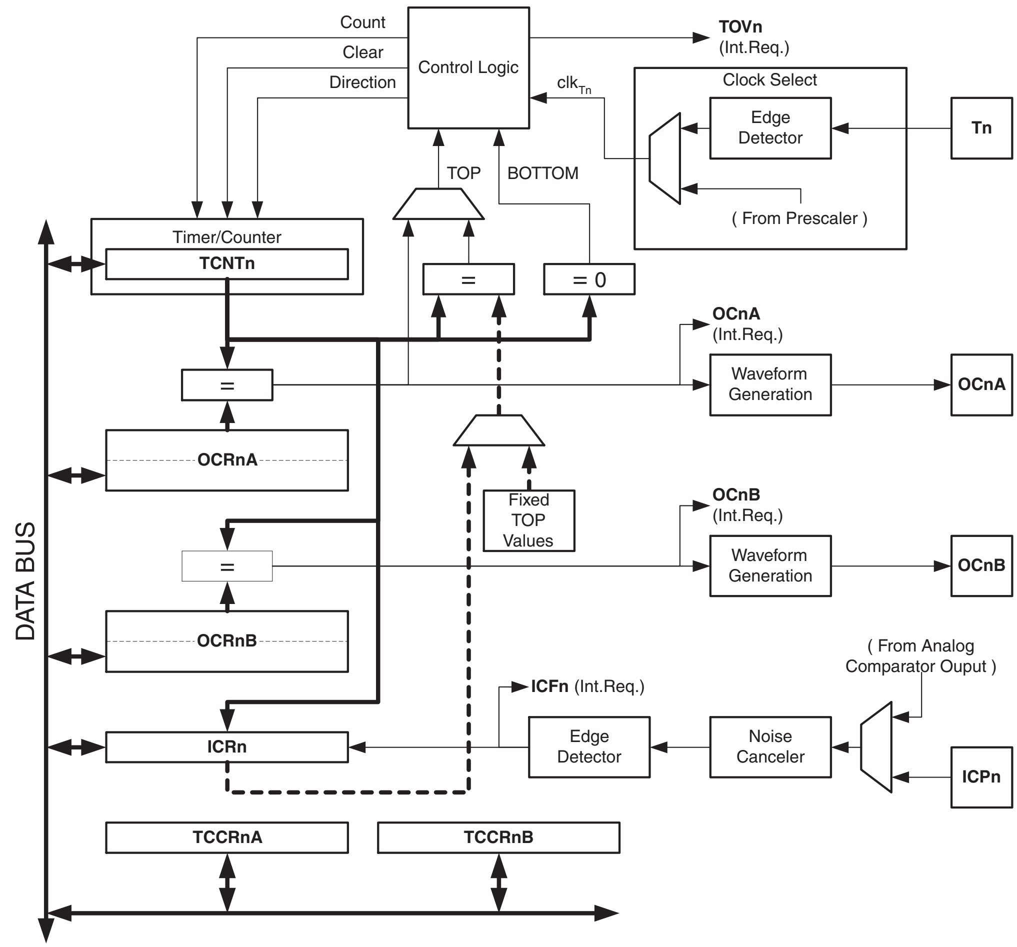


Figure 13-1. 16-bit Timer/Counter Block Diagram[1]



Note: 1. Refer to Figure 1-1 on page 2, Table 10-3 on page 71 and Table 10-9 on page 78 for Timer/Counter1 pin placement and description.

13.1.1 Registers

The *Timer/Counter* (TCNT1), *Output Compare Registers* (OCR1A/B), and *Input Capture Register* (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 108. The *Timer/Counter Control Registers* (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the *Timer Interrupt Flag Register* (TIFR1). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) 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_{T1}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC1A/B). See prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR1xH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (OCR1xL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 108.

13.6.1 Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOC1x) bit. Forcing compare match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM11:0 bits settings define whether the OC1x pin is set, cleared or toggled).

13.6.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

13.6.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

Table 15-4 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 15-4. Compare Output Mode, Phase Correct PWM Mode[1]

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match when up-counting. Set OC2A on Compare Match when down-counting.
1	1	Set OC2A on Compare Match when up-counting. Clear OC2A on Compare Match when down-counting.

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 146 for more details.

- **Bits 5:4 – COM2B1:0: Compare Match Output B Mode**

These bits control the Output Compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver.

When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. Table 15-5 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 15-5. Compare Output Mode, non-PWM Mode

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on Compare Match
1	0	Clear OC2B on Compare Match
1	1	Set OC2B on Compare Match

Table 15-6 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to fast PWM mode.

Table 15-6. Compare Output Mode, Fast PWM Mode[1]

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on Compare Match, set OC2B at TOP
1	1	Set OC2B on Compare Match, clear OC2B at TOP

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 146 for more details.

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)	$BAUD = \frac{f_{OSC}}{16(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{OSC}}{2BAUD} - 1$

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 (see page 192).

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.

18. USART in SPI Mode

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation. The Master SPI Mode (MSPIM) has the following features:

- **Full Duplex, Three-wire Synchronous Data Transfer**
- **Master Operation**
- **Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)**
- **LSB First or MSB First Data Transfer (Configurable Data Order)**
- **Queued Operation (Double Buffered)**
- **High Resolution Baud Rate Generator**
- **High Speed Operation (fXCKmax = fCK/2)**
- **Flexible Interrupt Generation**

18.1 Overview

Setting both UMSELn1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

18.2 Clock Generation

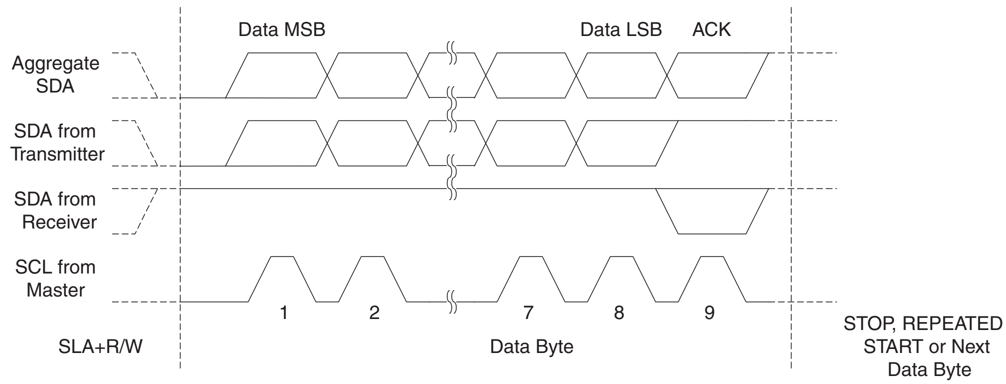
The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRRn setting can therefore be calculated using the same equations, see Table 18-1:

19.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the Receiver pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signalled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 19-5. Data Packet Format

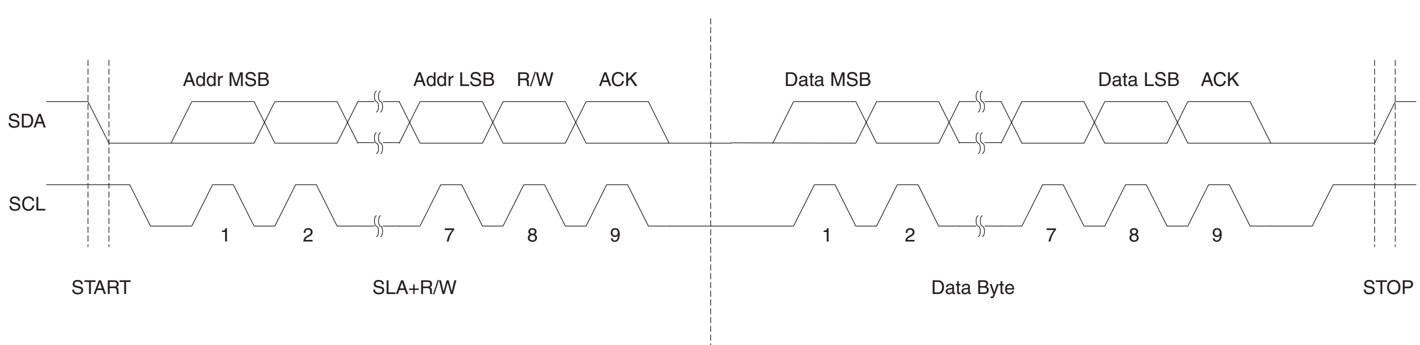


19.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the Wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 19-6 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 19-6. Typical Data Transmission



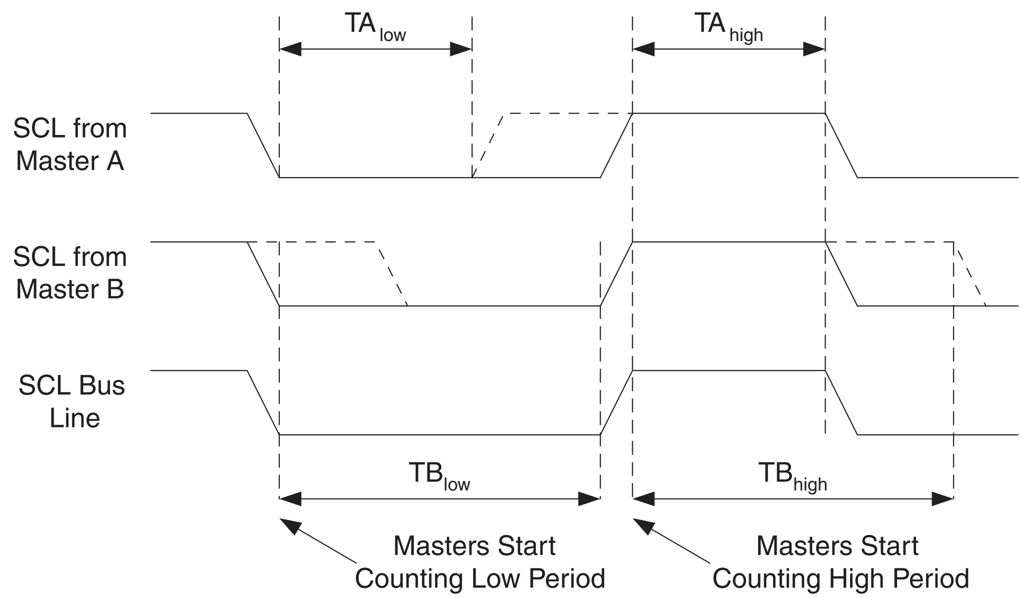
19.4 Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 19-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot (PrescalerValue)}$$

- TWBR = Value of the TWI Bit Rate Register.
- *PrescalerValue* = Value of the prescaler, see Table 19-2 on page 215.

Note: TWBR should be 10 or higher if the TWI operates in Master mode. If TWBR is lower than 10, the Master may produce an incorrect output on SDA and SCL for the reminder of the byte. The problem occurs when operating the TWI in Master mode, sending Start + SLA + R/W to a Slave (a Slave does not need to be connected to the bus for the condition to happen).

19.5.3 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

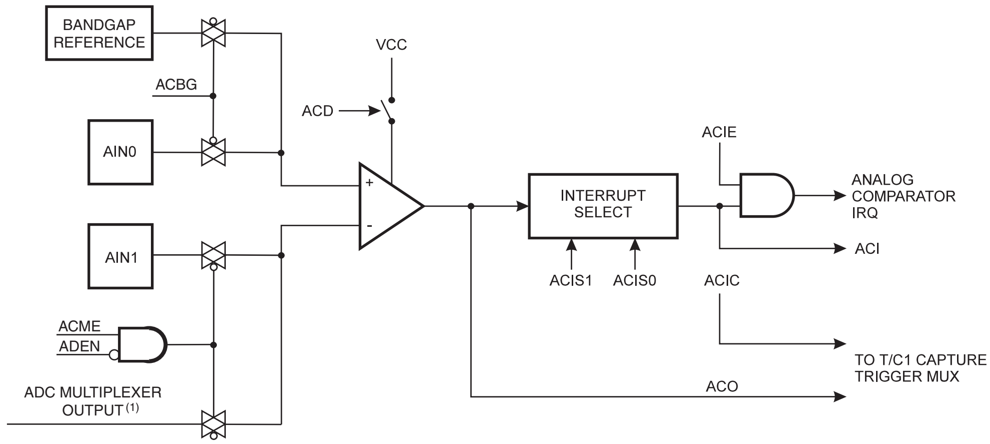
19.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to compare addresses even when the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master. If another interrupt (e.g., INT0) occurs during TWI Power-down address match and wakes up the CPU, the TWI aborts operation and return to it's idle state. If this cause any problems, ensure that TWI Address Match is the only enabled interrupt when entering Power-down.

19.5.5 Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is avail-

Figure 20-1. Analog Comparator Block Diagram[(2)]



Notes: 1. See Table 20-2 on page 238.
2. Refer to Figure 1-1 on page 2 and Table 10-9 on page 78 for Analog Comparator pin placement.

20.0.1 ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see ”Analog Comparator Multiplexed Input” on page 237.

20.0.2 Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See Section “8.1” on page 48.

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

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

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 (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 debugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

22.6.1 debugWire Data Register – DWDR

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	0	0	0	0	0	0	0	0	

The DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

23. Self-Programming the Flash, ATmega48

In ATmega48, there is no Read-While-Write support, and no separate Boot Loader Section. The SPM instruction can be executed from the entire Flash.

Table 24-2. Boot Lock Bit0 Protection Modes (Application Section)[1]

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Note: 1. "1" means unprogrammed, "0" means programmed

Table 24-3. Boot Lock Bit1 Protection Modes (Boot Loader Section)[1]

BLB1 Mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. "1" means unprogrammed, "0" means programmed

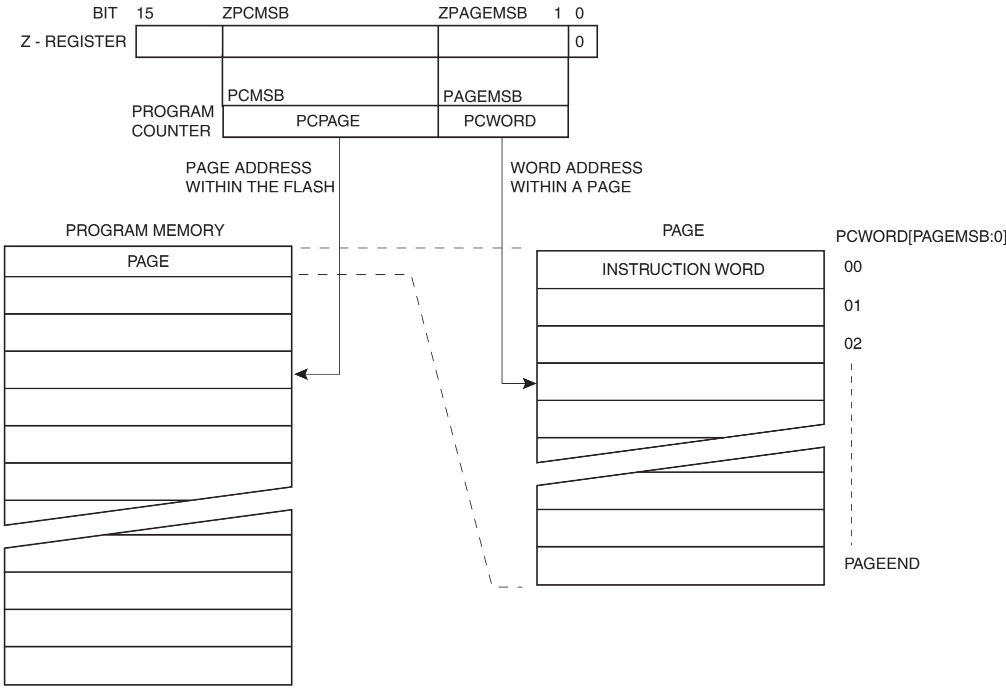
24.5 Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

Table 24-4. Boot Reset Fuse[1]

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 24-6 on page 276)

Figure 24-3. Addressing the Flash During SPM[(1)]



Note: 1. The different variables used in Figure 24-3 are listed in Table 24-8 on page 277.

24.7 Self-Programming the Flash

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 rewritten. 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. See ”Simple Assembly Code Example for a Boot Loader” on page 275 for an assembly code example.

25.4 Calibration Byte

The ATmega48/88/168 has a byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

25.5 Page Size

Table 25-8. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega48	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATmega88	4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11
ATmega168	8K words (16K bytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 25-9. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega48	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATmega88	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8
ATmega168	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

25.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega48/88/168. Pulses are assumed to be at least 250 ns unless otherwise noted.

25.6.1 Signal Names

In this section, some pins of the ATmega48/88/168 are referenced by signal names describing their functionality during parallel programming, see Figure 25-1 and Table 25-10. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 25-12.

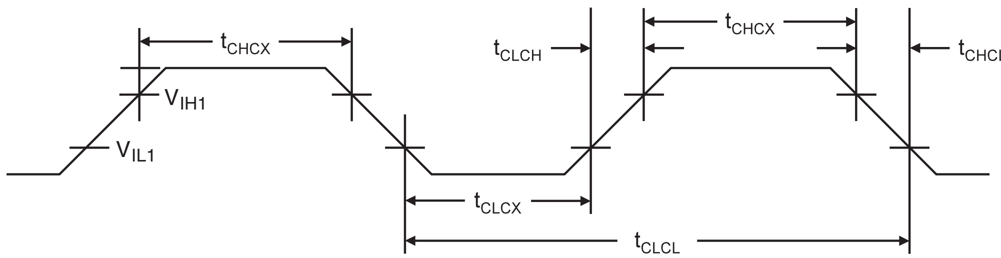
When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 25-13.

If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

5. All DC Characteristics contained in this datasheet 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
6. Values with “Power Reduction Register - PRR” disabled (0x00).

26.3 External Clock Drive Waveforms

Figure 26-1. External Clock Drive Waveforms



26.4 External Clock Drive

Table 26-1. External Clock Drive

Symbol	Parameter	V_{CC}=1.8-5.5V Min.	V_{CC}=1.8-5.5V Max.	V_{CC}=2.7-5.5V Min.	V_{CC}=2.7-5.5V Max.	V_{CC}=4.5-5.5V Min.	V_{CC}=4.5-5.5V Max.	Units
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	10	0	20	MHz
t_{CLCL}	Clock Period	250		100		50		ns
t_{CHCX}	High Time	100		40		20		ns
t_{CLCX}	Low Time	100		40		20		ns
t_{CLCH}	Rise Time		2.0		1.6		0.5	μs
t_{CHCL}	Fall Time		2.0		1.6		0.5	μs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

Note: All DC Characteristics contained in this datasheet 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.

26.5 Maximum Speed vs. V_{CC}

Maximum frequency is dependent on V_{CC}. As shown in Figure 26-2 and Figure 26-3, the Maximum Frequency vs. V_{CC} curve is linear between 1.8V < V_{CC} < 2.7V and between 2.7V < V_{CC} < 4.5V.

Table 26-2. 2-wire Serial Bus Requirements (Continued)

Symbol	Parameter	Condition	Min	Max	Units
C_i[1]	Capacitance for each I/O Pin		–	10	pF
f_{SCL}	SCL Clock Frequency	f_{CK}[4] > max($16f_{SCL}$, 250kHz)[5]	0	400	kHz
Rp	Value of Pull-up resistor	$f_{SCL} \le 100$ kHz	$\frac{V_{CC} - 0{,}4V}{3mA}$	$\frac{1000ns}{C_b}$	Ω
		$f_{SCL} > 100$ kHz	$\frac{V_{CC} - 0{,}4V}{3mA}$	$\frac{300ns}{C_b}$	Ω
$t_{HD;STA}$	Hold Time (repeated) START Condition	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
t_{LOW}	Low Period of the SCL Clock	$f_{SCL} \le 100$ kHz[6]	4.7	–	µs
		$f_{SCL} > 100$ kHz[7]	1.3	–	µs
t_{HIGH}	High period of the SCL clock	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	$f_{SCL} \le 100$ kHz	4.7	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
$t_{HD;DAT}$	Data hold time	$f_{SCL} \le 100$ kHz	0	3.45	µs
		$f_{SCL} > 100$ kHz	0	0.9	µs
$t_{SU;DAT}$	Data setup time	$f_{SCL} \le 100$ kHz	250	–	ns
		$f_{SCL} > 100$ kHz	100	–	ns
$t_{SU;STO}$	Setup time for STOP condition	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
t_{BUF}	Bus free time between a STOP and START condition	$f_{SCL} \le 100$ kHz	4.7	–	µs
		$f_{SCL} > 100$ kHz	1.3	–	µs

Notes:
1. In ATmega48/88/168, this parameter is characterized and not 100% tested.
2. Required only for $f_{SCL} > 100$ kHz.
3. C_b = capacitance of one bus line in pF.
4. f_{CK} = CPU clock frequency
5. This requirement applies to all ATmega48/88/168 2-wire Serial Interface operation. Other devices connected to the 2-wire Serial Bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the ATmega48/88/168 2-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus f_{CK} must be greater than 6 MHz for the low time requirement to be strictly met at $f_{SCL} = 100$ kHz.
7. The actual low period generated by the ATmega48/88/168 2-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus the low time requirement will not be strictly met for $f_{SCL} > 308$ kHz when $f_{CK} = 8$ MHz. Still, ATmega48/88/168 devices connected to the bus may communicate at full speed (400 kHz) with other ATmega48/88/168 devices, as well as any other device with a proper t_{LOW} acceptance margin.

Figure 27-19. I/O Pin Pull-Up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

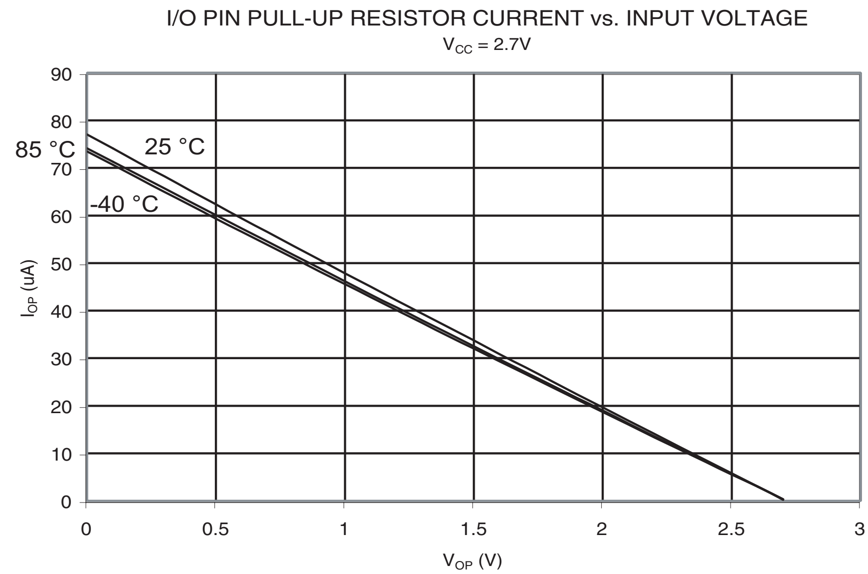
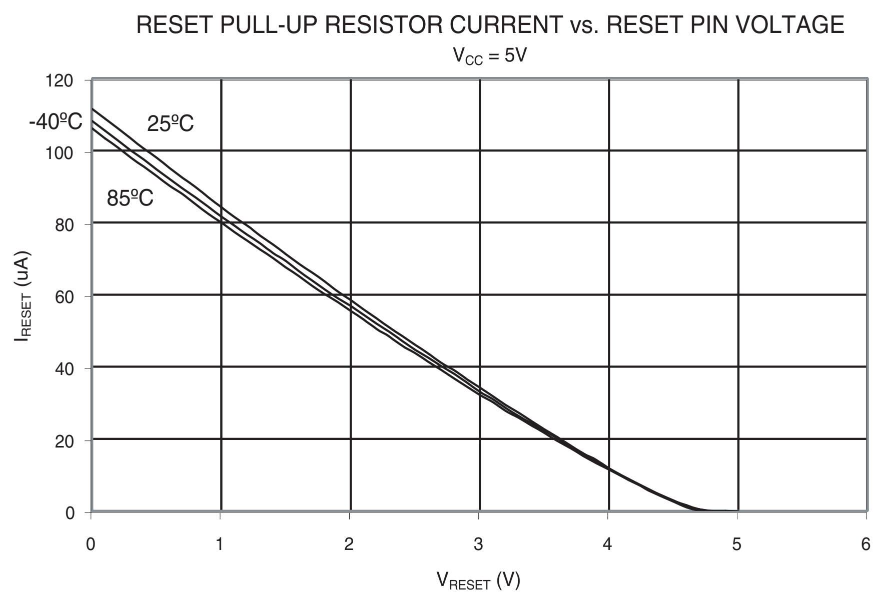


Figure 27-20. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



30.2 ATmega88

Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operational Range
10[3]	1.8 - 5.5	ATmega88V-10AI ATmega88V-10PI ATmega88V-10MI ATmega88V-10AU[2] ATmega88V-10PU[2] ATmega88V-10MU[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)
20[3]	2.7 - 5.5	ATmega88-20AI ATmega88-20PI ATmega88-20MI ATmega88-20AU[2] ATmega88-20PU[2] ATmega88-20MU[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)

Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive).Also Halide free and fully Green.
3. See Figure 26-2 on page 302 and Figure 26-3 on page 302.

Package Type	
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)