



Welcome to [E-XFL.COM](https://www.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](#)"

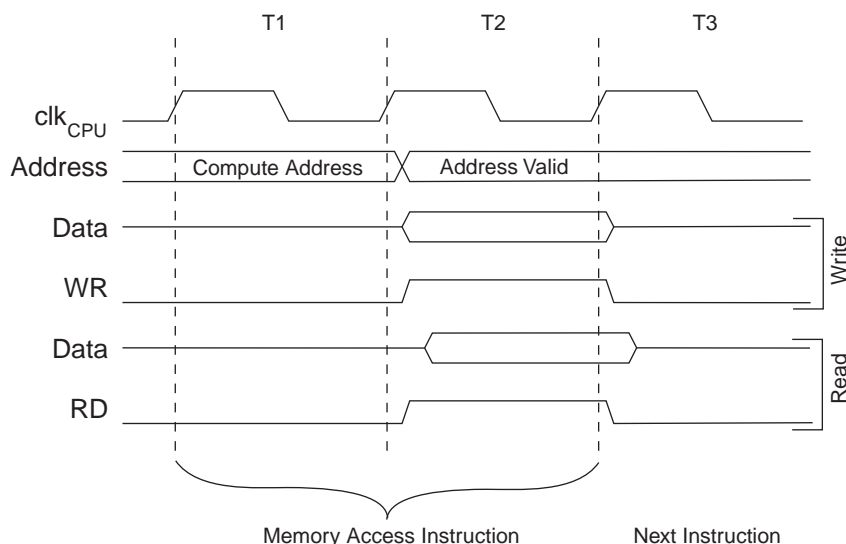
#### Details

Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	8MHz
Connectivity	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
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 ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega16l-8pj">https://www.e-xfl.com/product-detail/microchip-technology/atmega16l-8pj</a>

## 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  $\text{clk}_{\text{CPU}}$  cycles as described in [Figure 10](#).

**Figure 10.** On-chip Data SRAM Access Cycles



## EEPROM Data Memory

The ATmega16 contains 512 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 SPI, JTAG, and Parallel data downloading to the EEPROM, see [page 273](#), [page 278](#), and [page 262](#), respectively.

## 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 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 heavily filtered power supplies,  $V_{\text{CC}}$  is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See [“Preventing EEPROM Corruption” on page 22](#) for details on how to avoid problems in these situations.

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.

- **SDA – Port C, Bit 1**

SDA, Two-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC1 bit.

- **SCL – Port C, Bit 0**

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC0 is disconnected from the port and becomes the Serial Clock I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC0 bit.

[Table 29](#) and [Table 30](#) relate the alternate functions of Port C to the overriding signals shown in [Figure 26](#) on page 55.

**Table 29.** Overriding Signals for Alternate Functions in PC7..PC4

Signal Name	PC7/TOSC2	PC6/TOSC1	PC5/TDI	PC4/TDO
PUOE	AS2	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	0
DDOE	AS2	AS2	JTAGEN	JTAGEN
DDOV	0	0	0	SHIFT_IR + SHIFT_DR
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO
DIEOE	AS2	AS2	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	T/C2 OSC OUTPUT	T/C2 OSC INPUT	TDI	–

- **OC1A – Port D, Bit 5**

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **OC1B – Port D, Bit 4**

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source.

- **INT0 – Port D, Bit 2**

INT0, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source.

- **TXD – Port D, Bit 1**

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

- **RXD – Port D, Bit 0**

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

[Table 32](#) and [Table 33](#) relate the alternate functions of Port D to the overriding signals shown in [Figure 26](#) on page 55.

**Table 32.** Overriding Signals for Alternate Functions PD7..PD4

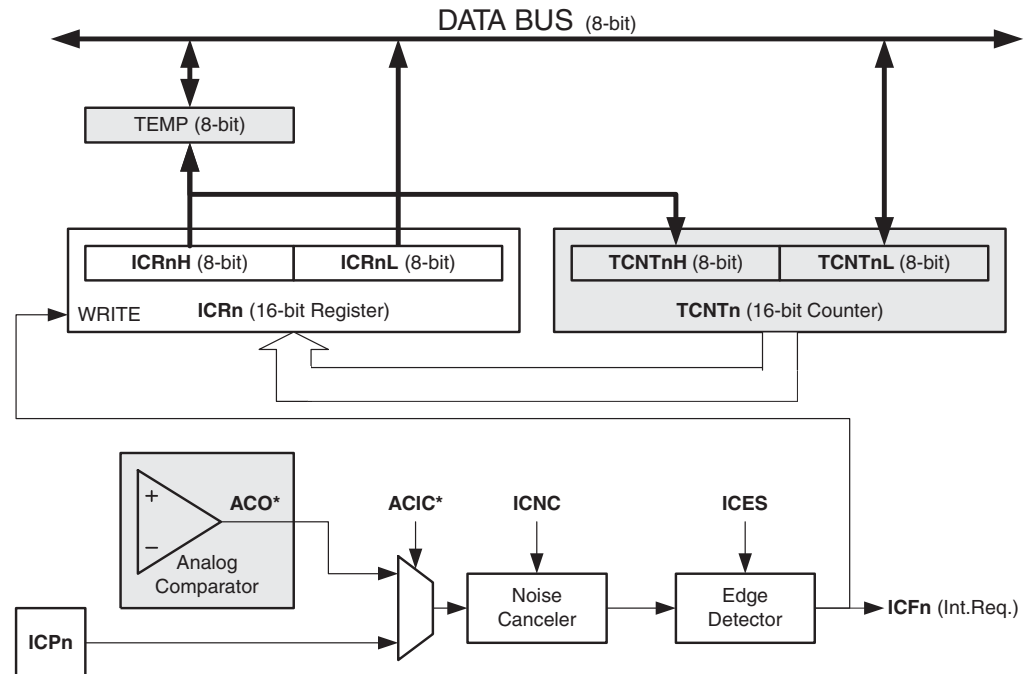
Signal Name	PD7/OC2	PD6/ICP1	PD5/OC1A	PD4/OC1B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2 ENABLE	0	OC1A ENABLE	OC1B ENABLE
PVOV	OC2	0	OC1A	OC1B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	ICP1 INPUT	–	–
AIO	–	–	–	–

## 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 42. 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 42.** 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 (TICIE1 = 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.

the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

## Phase and Frequency Correct PWM Mode

The *phase and frequency correct Pulse Width Modulation*, or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see [Figure 47](#) and [Figure 48](#)).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on [Figure 48](#). The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.

value of TCNT2, 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.

For generating a waveform output in CTC mode, the OC2 output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM21:0 = 1). The OC2 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of  $f_{OC2} = f_{clk\_I/O}/2$  when OCR2 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

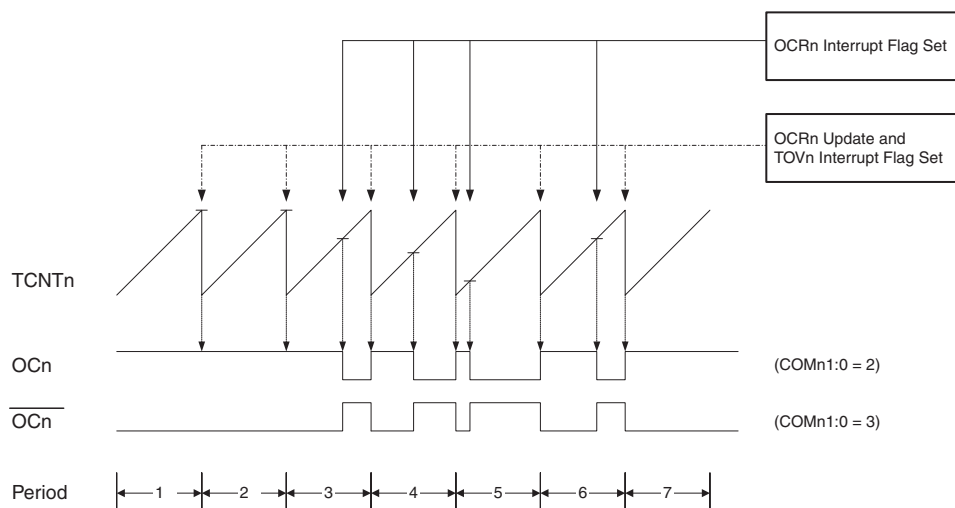
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

## Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM21:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 58. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2 and TCNT2.

**Figure 58.** Fast PWM Mode, Timing Diagram



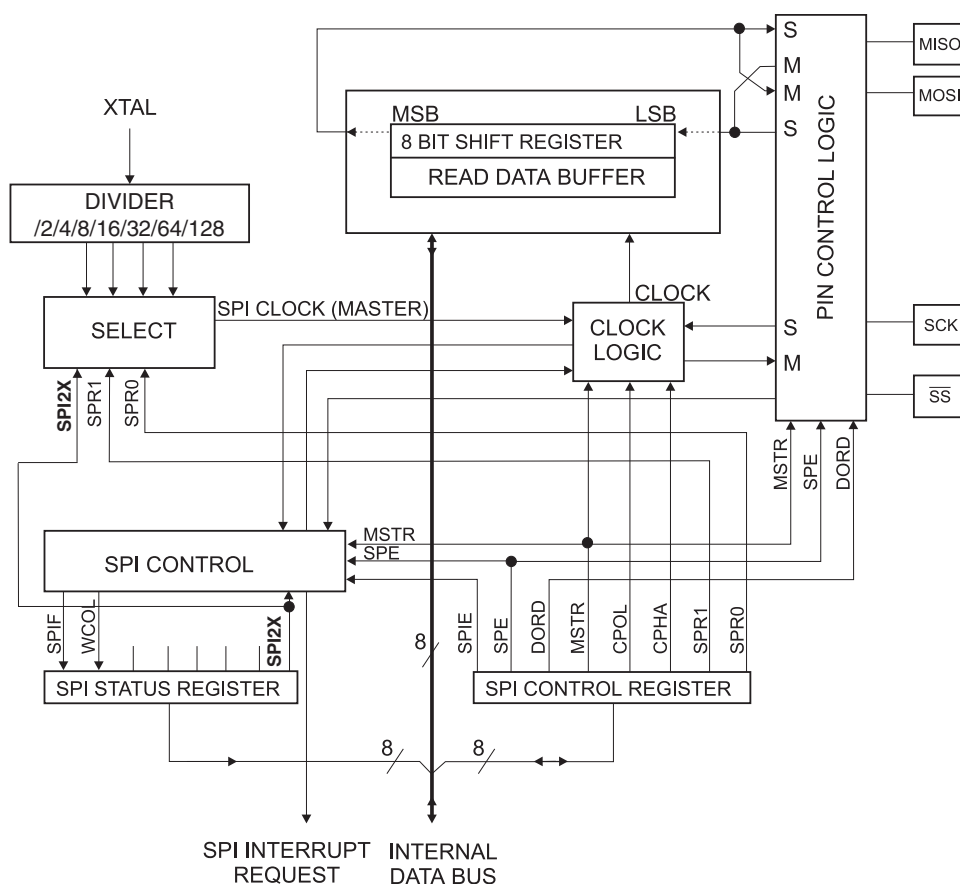
The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

## Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega16 and peripheral devices or between several AVR devices. The ATmega16 SPI includes the following features:

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

Figure 65. SPI Block Diagram<sup>(1)</sup>



Note: 1. Refer to [Figure 1 on page 2](#), and [Table 25 on page 58](#) for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in [Figure 66](#). The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select  $\overline{SS}$  pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select,  $\overline{SS}$ , line.

When configured as a Master, the SPI interface has no automatic control of the  $\overline{SS}$  line. This must be handled by user software before communication can start. When this is done, writing a

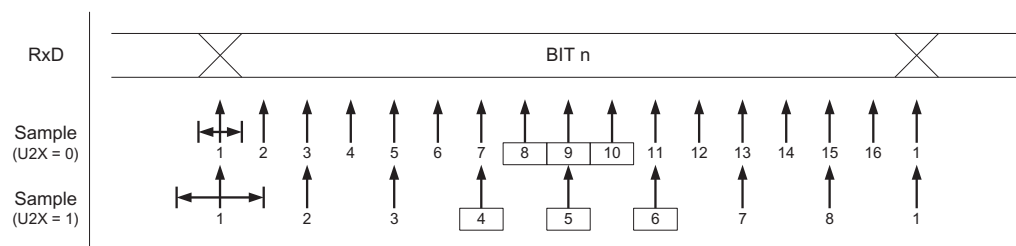


figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

## Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and 8 states for each bit in Double Speed mode. Figure 74 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

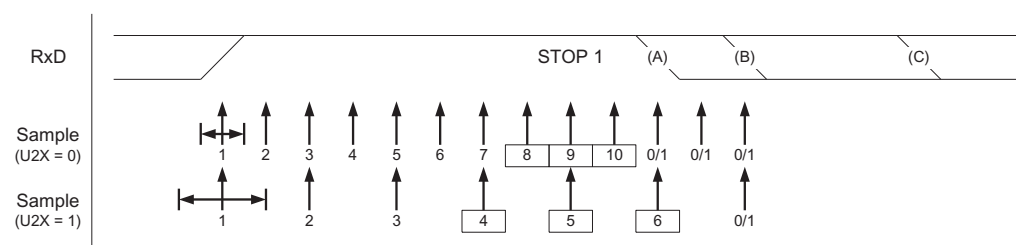
**Figure 74.** Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 75 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

**Figure 75.** Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 75. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

## Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see [Table 61](#)) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D \cdot S + S_F}$$

$$R_{fast} = \frac{(D+2)S}{(D+1)S + S_M}$$

- D Sum of character size and parity size (D = 5 to 10 bit)
- S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
- S<sub>F</sub> First sample number used for majority voting. S<sub>F</sub> = 8 for Normal Speed and S<sub>F</sub> = 4 for Double Speed mode.
- S<sub>M</sub> Middle sample number used for majority voting. S<sub>M</sub> = 9 for Normal Speed and S<sub>M</sub> = 5 for Double Speed mode.
- R<sub>slow</sub> is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R<sub>fast</sub> is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

[Table 61](#) and [Table 62](#) list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

**Table 61.** Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R <sub>slow</sub> (%)	R <sub>fast</sub> (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

**Table 62.** Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R <sub>slow</sub> (%)	R <sub>fast</sub> (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104.35	+4.35/-4.48	±1.5

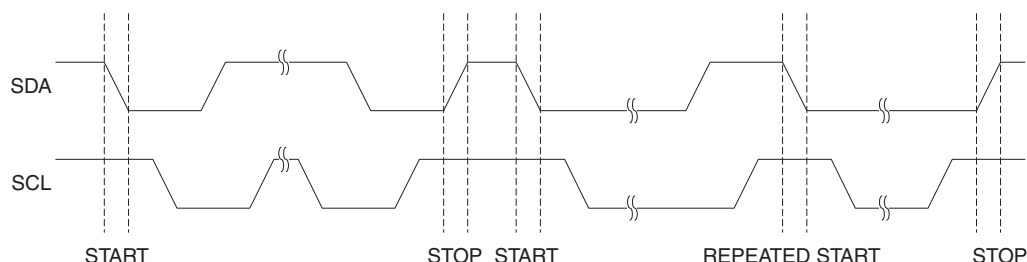
**Table 62.** Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R <sub>slow</sub> (%)	R <sub>fast</sub> (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

**Figure 78.** START, REPEATED START, and STOP Conditions



## Address Packet Format

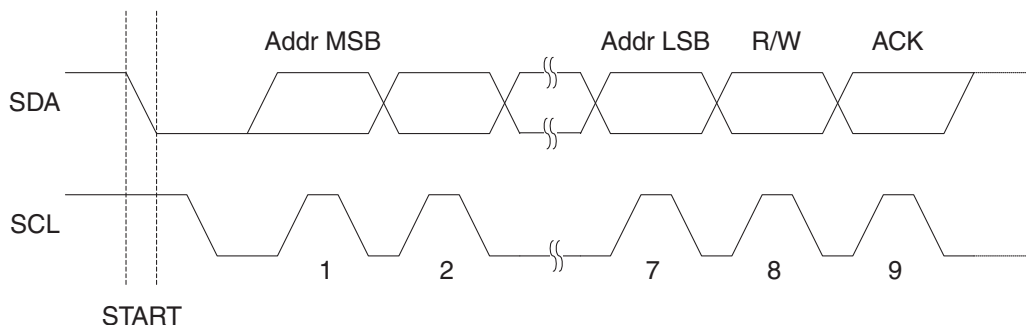
All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a Slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

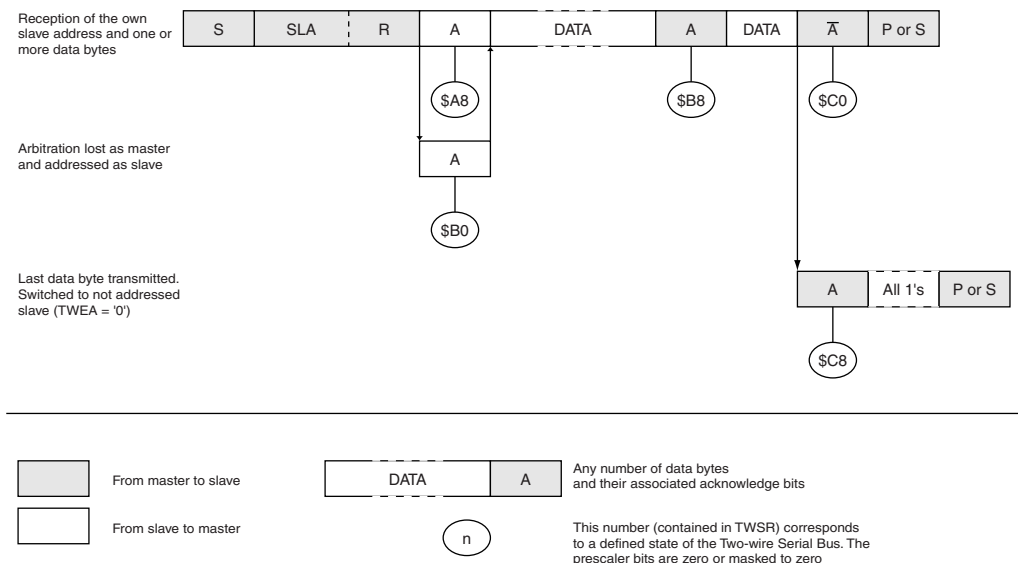
When a general call is issued, all Slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several Slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all Slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the Slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several Slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

**Figure 79.** Address Packet Format



**Figure 93. Formats and States in the Slave Transmitter Mode**



## Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see [Table 78](#).

Status \$F8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status \$00 indicates that a bus error has occurred during a Two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

**Table 78. Miscellaneous States**

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
\$F8	No relevant state information available; TWINT = "0"	No TWDR action	No TWCR action				Wait or proceed current transfer
\$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

## Analog Input Circuitry

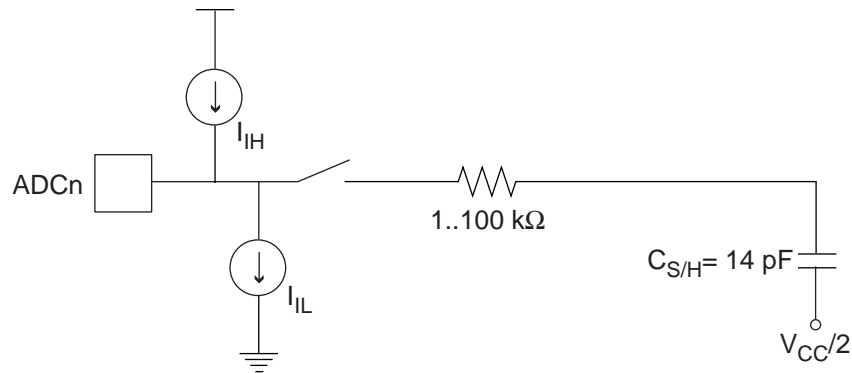
The Analog Input Circuitry for single ended channels is illustrated in Figure 105. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive 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 k $\Omega$  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, with 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.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred k $\Omega$  or less is recommended.

Signal components higher than the Nyquist frequency ( $f_{ADC}/2$ ) should not be present for either kind of channels, 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 105.** Analog Input Circuitry



## 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. Keep them well away from high-speed switching digital tracks.
2. The AVCC pin on the device should be connected to the digital  $V_{CC}$  supply voltage via an LC network as shown in [Figure 106](#).
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

## On-chip Debug Related Register in I/O Memory

### On-chip Debug Register – OCDR

Bit	7	6	5	4	3	2	1	0	
	<b>MSB/IDRD</b>							<b>LSB</b>	OCDR
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 OCDR Register provides a communication channel from the running program in the micro-controller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an Internal Flag; I/O Debug Register Dirty – IDRD – is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information.

In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

Refer to the debugger documentation for further information on how to use this register.

## Using the JTAG Programming Capabilities

Programming of AVR parts via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI and TDO. These are the only pins that need to be controlled/observed to perform JTAG programming (in addition to power pins). It is not required to apply 12V externally. The JTAGEN Fuse must be programmed and the JTD bit in the MCUSR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- Flash programming and verifying
- EEPROM programming and verifying
- Fuse programming and verifying
- Lock bit programming and verifying

The Lock bit security is exactly as in Parallel Programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section [“Programming via the JTAG Interface” on page 278](#).

## Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992

## RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an on-going programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (that is, by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control Register (SPMCR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. [See “Store Program Memory Control Register – SPMCR” on page 250.](#) for details on how to clear RWWSB.

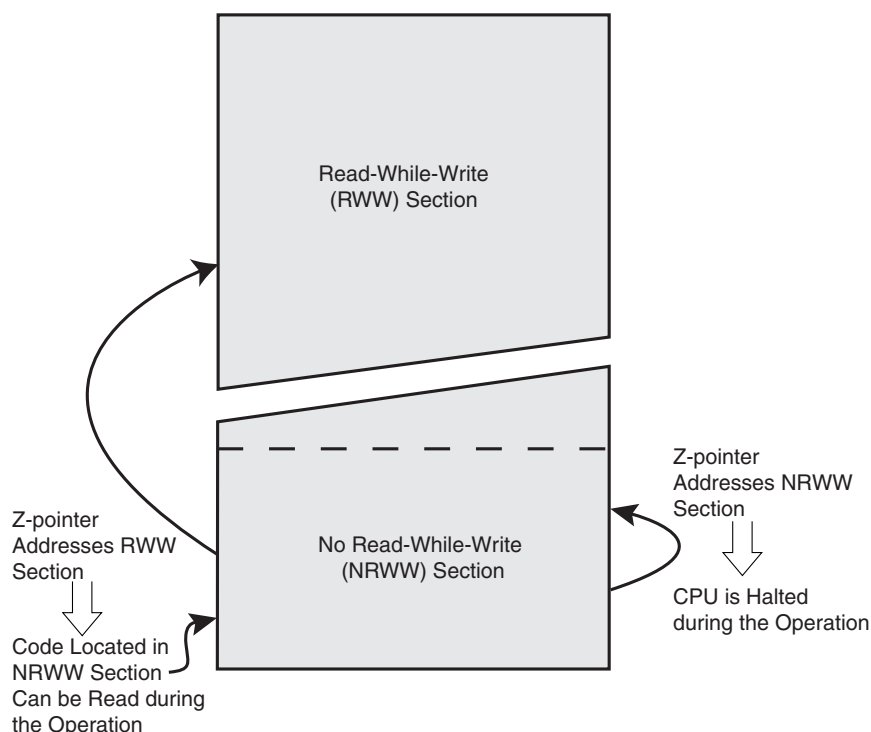
## NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire page erase or page write operation.

**Table 95.** Read-While-Write Features

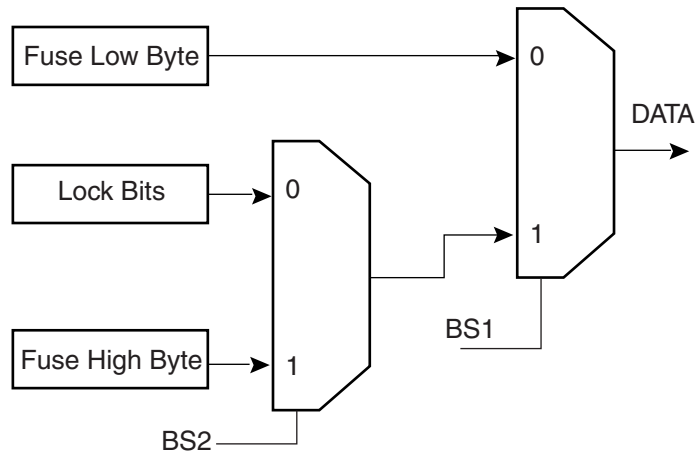
Which Section does the Z-pointer Address during the Programming?	Which Section can be Read during Programming?	Is the CPU Halted?	Read-While-Write Supported?
RWW section	NRWW section	No	Yes
NRWW section	None	Yes	No

**Figure 124.** Read-While-Write vs. No Read-While-Write





**Figure 132.** Mapping between BS1, BS2 and the Fuse- and Lock Bits during Read



## Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to [“Programming the Flash” on page 266](#) for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte (\$00 - \$02).
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”.

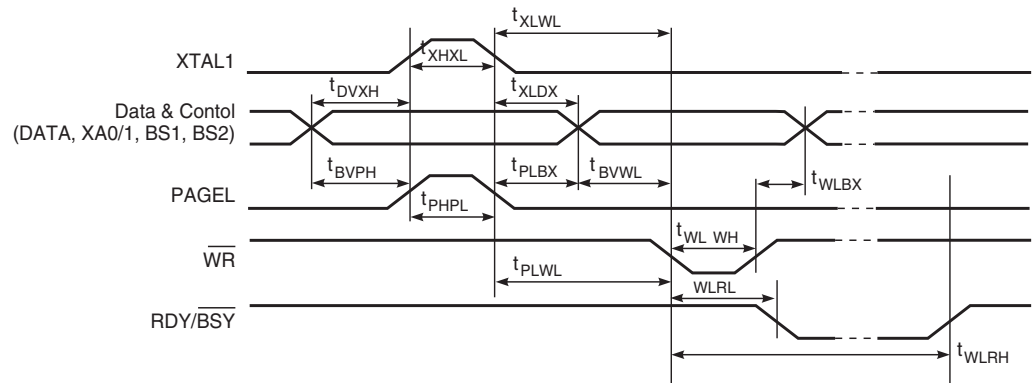
## Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to [“Programming the Flash” on page 266](#) for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, \$00.
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”.

## Parallel Programming Characteristics

**Figure 133.** Parallel Programming Timing, Including some General Timing Requirements



## Two-wire Serial Interface Characteristics

Table 120 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega16 Two-wire Serial Interface meets or exceeds these requirements under the noted conditions.

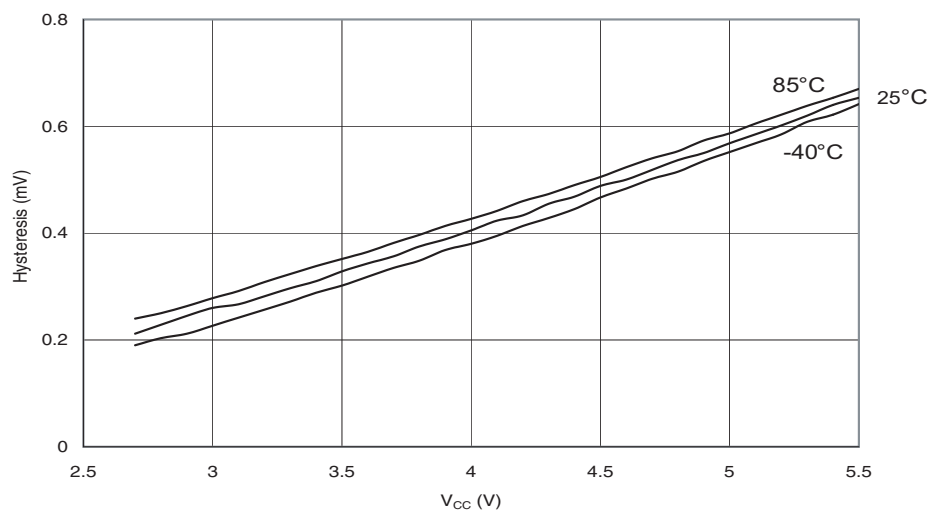
Timing symbols refer to Figure 146.

**Table 120.** Two-wire Serial Bus Requirements

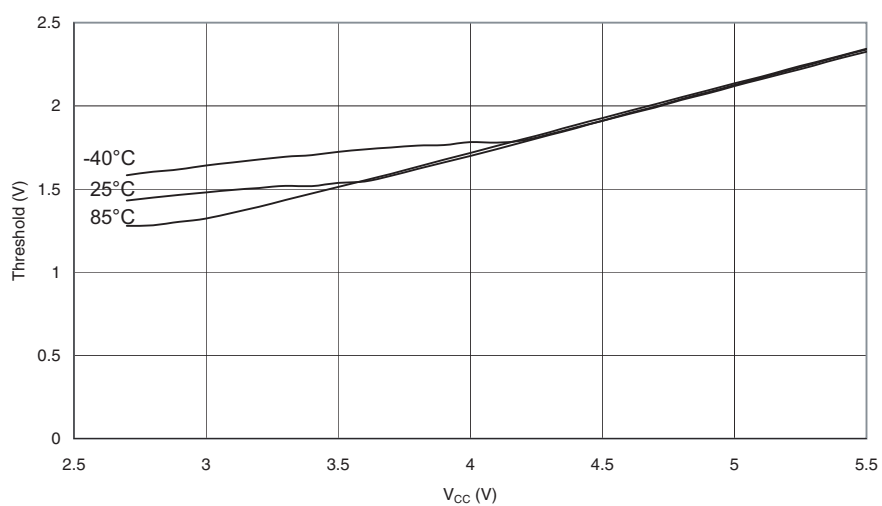
Symbol	Parameter	Condition	Min	Max	Units
$V_{IL}$	Input Low-voltage		-0.5	$0.3 V_{CC}$	V
$V_{IH}$	Input High-voltage		$0.7 V_{CC}$	$V_{CC} + 0.5$	
$V_{hys}^{(1)}$	Hysteresis of Schmitt Trigger Inputs		$0.05 V_{CC}^{(2)}$	—	
$V_{OL}^{(1)}$	Output Low-voltage	3 mA sink current	0	0.4	
$t_r^{(1)}$	Rise Time for both SDA and SCL		$20 + 0.1C_b^{(3)(2)}$	300	ns
$t_{of}^{(1)}$	Output Fall Time from $V_{IHmin}$ to $V_{ILmax}$	$10 \text{ pF} < C_b < 400 \text{ pF}^{(3)}$	$20 + 0.1C_b^{(3)(2)}$	250	
$t_{SP}^{(1)}$	Spikes Suppressed by Input Filter		0	$50^{(2)}$	
$I_i$	Input Current each I/O Pin	$0.1V_{CC} < V_i < 0.9V_{CC}$	-10	10	$\mu\text{A}$
$C_i^{(1)}$	Capacitance for each I/O Pin		—	10	pF
$f_{SCL}$	SCL Clock Frequency	$f_{CK}^{(4)} > \max(16f_{SCL}, 250\text{kHz})^{(5)}$	0	400	kHz
Rp	Value of Pull-up resistor	$f_{SCL} \leq 100 \text{ kHz}$	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{1000\text{ns}}{C_b}$	$\Omega$
		$f_{SCL} > 100 \text{ kHz}$	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{300\text{ns}}{C_b}$	
$t_{HD;STA}$	Hold Time (repeated) START Condition	$f_{SCL} \leq 100 \text{ kHz}$	4.0	—	$\mu\text{s}$
		$f_{SCL} > 100 \text{ kHz}$	0.6	—	
$t_{LOW}$	Low Period of the SCL Clock	$f_{SCL} \leq 100 \text{ kHz}$	4.7	—	
		$f_{SCL} > 100 \text{ kHz}$	1.3	—	
$t_{HIGH}$	High period of the SCL clock	$f_{SCL} \leq 100 \text{ kHz}$	4.0	—	
		$f_{SCL} > 100 \text{ kHz}$	0.6	—	
$t_{SU;STA}$	Set-up time for a repeated START condition	$f_{SCL} \leq 100 \text{ kHz}$	4.7	—	
		$f_{SCL} > 100 \text{ kHz}$	0.6	—	
$t_{HD;DAT}$	Data hold time	$f_{SCL} \leq 100 \text{ kHz}$	0	3.45	
		$f_{SCL} > 100 \text{ kHz}$	0	0.9	
$t_{SU;DAT}$	Data setup time	$f_{SCL} \leq 100 \text{ kHz}$	250	—	ns
		$f_{SCL} > 100 \text{ kHz}$	100	—	
$t_{SU;STO}$	Setup time for STOP condition	$f_{SCL} \leq 100 \text{ kHz}$	4.0	—	$\mu\text{s}$
		$f_{SCL} > 100 \text{ kHz}$	0.6	—	
$t_{BUF}$	Bus free time between a STOP and START condition	$f_{SCL} \leq 100 \text{ kHz}$	4.7	—	
		$f_{SCL} > 100 \text{ kHz}$	1.3	—	

- Notes:
1. In ATmega16, this parameter is characterized and not 100% tested.
  2. Required only for  $f_{SCL} > 100 \text{ kHz}$ .
  3.  $C_b$  = capacitance of one bus line in pF.
  4.  $f_{CK}$  = CPU clock frequency

**Figure 184.** I/O Pin Input Hysteresis vs.  $V_{CC}$



**Figure 185.** Reset Input Threshold Voltage vs.  $V_{CC}$  ( $V_{IH}$ , Reset Pin Read As '1')



## Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

### 2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

#### Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

### 3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

#### Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the first device in the chain.

### 4. Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

#### Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

## ATmega16(L) Rev. K

- First Analog Comparator conversion may be delayed
- Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

### 1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising  $V_{CC}$ , the first Analog Comparator conversion will take longer than expected on some devices.

#### Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

### 2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Prescaling and Conversion Timing 207  
Changing Channel or Reference Selection 210  
ADC Noise Canceler 211  
ADC Conversion Result 216

## ***JTAG Interface and On-chip Debug System 222***

Features 222  
Overview 222  
Test Access Port – TAP 222  
TAP Controller 224  
Using the Boundary-scan Chain 225  
Using the On-chip Debug System 225  
On-chip Debug Specific JTAG Instructions 226  
On-chip Debug Related Register in I/O Memory 227  
Using the JTAG Programming Capabilities 227  
Bibliography 227

## ***IEEE 1149.1 (JTAG) Boundary-scan 228***

Features 228  
System Overview 228  
Data Registers 228  
Boundary-scan Specific JTAG Instructions 230  
Boundary-scan Chain 232  
ATmega16 Boundary-scan Order 241  
Boundary-scan Description Language Files 245

## ***Boot Loader Support – Read-While-Write Self-Programming 246***

Features 246  
Application and Boot Loader Flash Sections 246  
Read-While-Write and no Read-While-Write Flash Sections 246  
Boot Loader Lock Bits 248  
Entering the Boot Loader Program 249  
Addressing the Flash during Self-Programming 251  
Self-Programming the Flash 252

## ***Memory Programming 259***

Program And Data Memory Lock Bits 259  
Fuse Bits 260  
Signature Bytes 261  
Calibration Byte 261  
Page Size 262  
Parallel Programming Parameters, Pin Mapping, and Commands 262  
Parallel Programming 265  
Serial Downloading 273  
Programming via the JTAG Interface 278