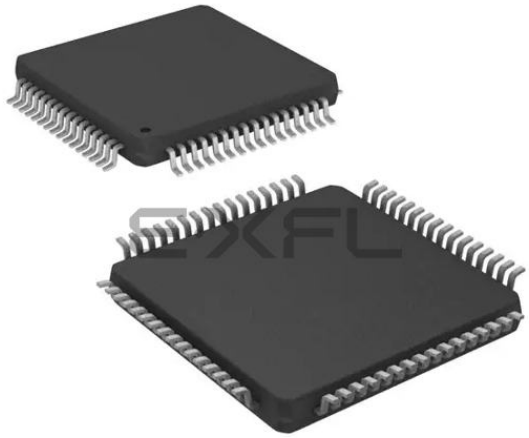


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

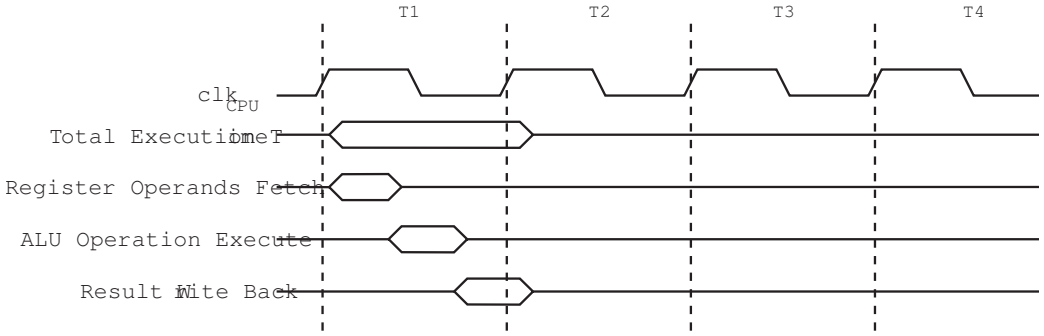
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	53
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	4K 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	64-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at90can128-15az

Figure 3-6 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 3-6. Single Cycle ALU Operation



3.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 335 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 60. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to "Interrupts" on page 60 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 320.

3.8.1 Interrupt Behavior

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

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

- **Bit 3..2 – SRW11, SRW10: Wait-state Select Bits for Upper Sector**

The SRW11 and SRW10 bits control the number of wait-states for the upper sector of the external memory address space, see Table 4-4.

- **Bit 1..0 – SRW01, SRW00: Wait-state Select Bits for Lower Sector**

The SRW01 and SRW00 bits control the number of wait-states for the lower sector of the external memory address space, see Table 4-4.

Table 4-4. Wait States[1]

SRWn1	SRWn0	Wait States
0	0	No wait-states
0	1	Wait one cycle during read/write strobe
1	0	Wait two cycles during read/write strobe
1	1	Wait two cycles during read/write and wait one cycle before driving out new address

Note: 1. n = 0 or 1 (lower/upper sector).
For further details of the timing and wait-states of the External Memory Interface, see Figures 4-6 through Figures 4-9 for how the setting of the SRW bits affects the timing.

4.5.7 External Memory Control Register B – XMCRB

Bit	7	6	5	4	3	2	1	0	
	XMBK	–	–	–	–	XMM2	XMM1	XMM0	XMCRB
Read/Write	R/W	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– XMBK: External Memory Bus-keeper Enable**

Writing XMBK to one enables the bus keeper on the AD7:0 lines. When the bus keeper is enabled, it will ensure a defined logic level (zero or one) on AD7:0 when they would otherwise be tri-stated. Writing XMBK to zero disables the bus keeper. XMBK is not qualified with SRE, so even if the XMEM interface is disabled, the bus keepers are still activated as long as XMBK is one.

- **Bit 6..4 – Reserved Bits**

These are reserved bits and will always read as zero. When writing to this address location, write these bits to zero for compatibility with future devices.

- **Bit 2..0 – XMM2, XMM1, XMM0: External Memory High Mask**

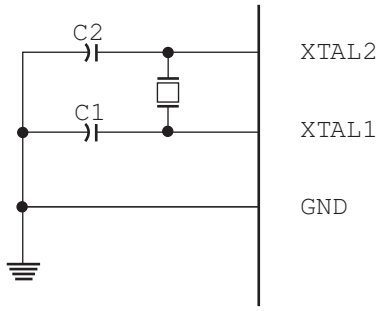
When the External Memory is enabled, all Port C pins are default used for the high address byte. If the full address space is not required to access the External Memory, some, or all, Port C pins can be released for normal Port Pin function as described in Table 4-5. As described in “Using all 64KB Locations of External Memory” on page 35, it is possible to use the XMMn bits to access all 64KB locations of the External Memory.

5.4 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 5-2. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 5-3. For ceramic resonators, the capacitor values given by the manufacturer should be used. For more information on how to choose capacitors and other details on Oscillator operation, refer to the Multi-purpose Oscillator Application Note.

Figure 5-2. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 5-3.

Table 5-3. Crystal Oscillator Operating Modes

CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100[1]	0.4 - 0.9	12 - 22
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 - 16.0	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 5-4.

8.2 Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

8.2.1 MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	JTD	–	–	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – IVSEL: Interrupt Vector Select**

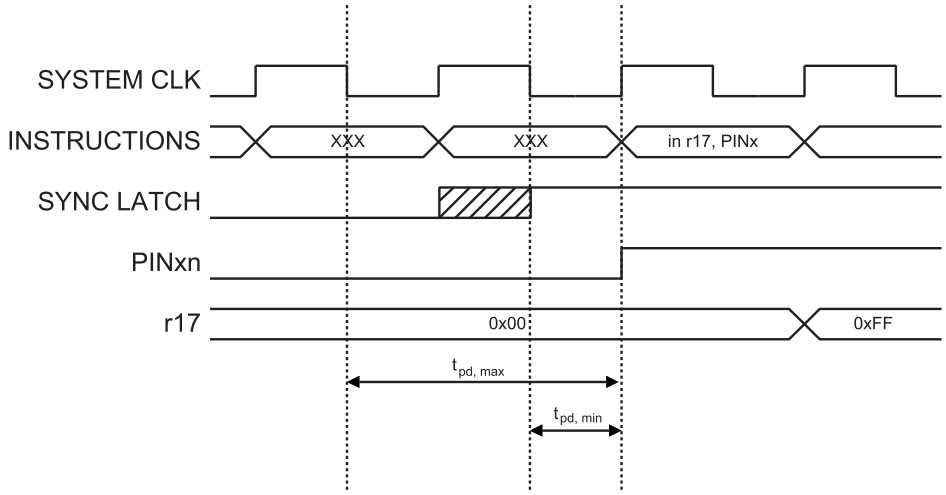
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 320 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

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 the section "Boot Loader Support – Read-While-Write Self-Programming" on page 320 for details on Boot Lock bits.

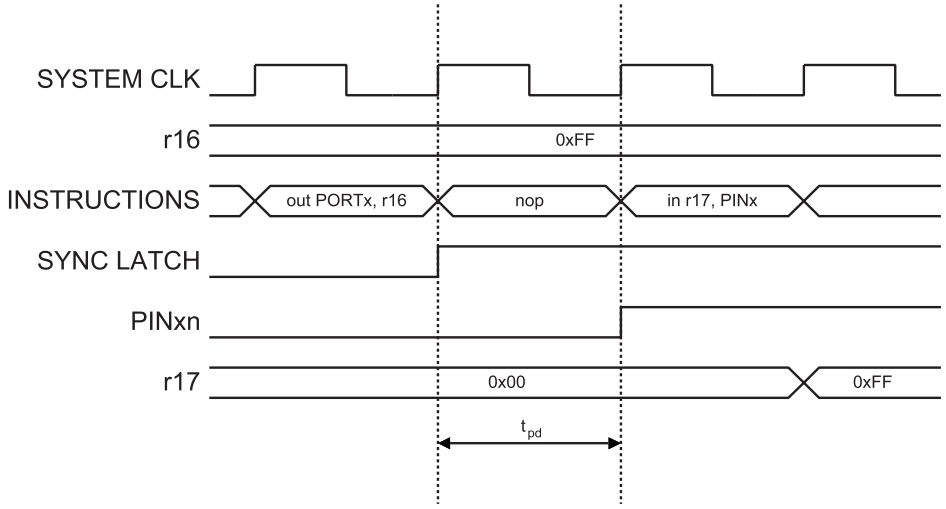
Figure 9-3. Synchronization when Reading an Externally Applied Pin value



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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 9-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is 1 system clock period.

Figure 9-4. Synchronization when Reading a Software Assigned Pin Value



above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.

9.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode). The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-5 shows how the port pin control signals from the simplified Figure 9-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

TXCAN, CAN Transmit Data (Data output pin for the CAN). When the CAN is enabled, this pin is configured as an output regardless of the value of DDD5.

XCK1, USART1 External clock. The Data Direction Register (DDD5) controls whether the clock is output (DDD5 set) or input (DDD45 cleared). The XCK1 pin is active only when the USART1 operates in Synchronous mode.

- **ICP1 – Port D, Bit 4**

ICP1, Input Capture Pin1. The PD4 pin can act as an input capture pin for Timer/Counter1.

- **INT3/TXD1 – Port D, Bit 3**

INT3, External Interrupt source 3. The PD3 pin can serve as an external interrupt source to the MCU.

TXD1, Transmit Data (Data output pin for the USART1). When the USART1 Transmitter is enabled, this pin is configured as an output regardless of the value of DDD3.

- **INT2/RXD1 – Port D, Bit 2**

INT2, External Interrupt source 2. The PD2 pin can serve as an External Interrupt source to the MCU.

RXD1, Receive Data (Data input pin for the USART1). When the USART1 receiver is enabled this pin is configured as an input regardless of the value of DDD2. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD2 bit.

- **INT1/SDA – Port D, Bit 1**

INT1, External Interrupt source 1. The PD1 pin can serve as an external interrupt source to the MCU.

SDA, Two-wire Serial Interface Data. When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PD1 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.

- **INT0/SCL – Port D, Bit 0**

INT0, External Interrupt source 0. The PD0 pin can serve as an external interrupt source to the MCU.

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PD0 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.

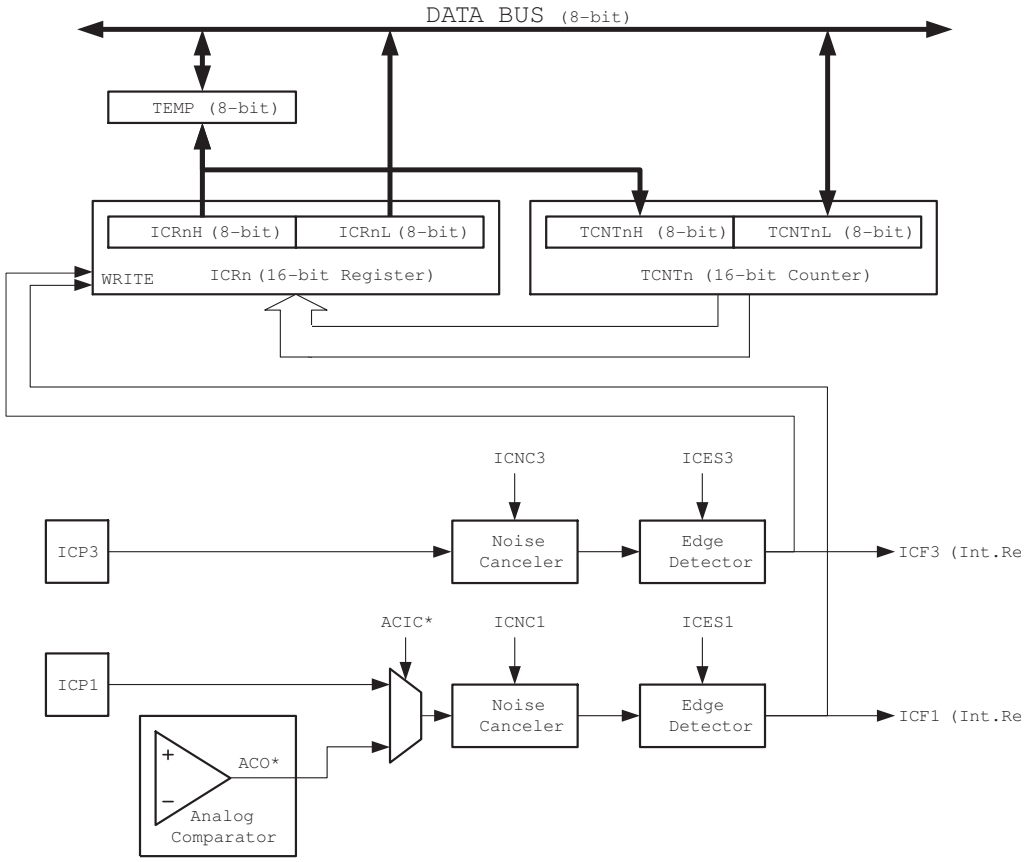
The Timer/Counter Overflow Flag (TOVn) is set according to the mode of operation selected by the WGMn3:0 bits. TOVn can be used for generating a CPU interrupt.

13.6 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICPn 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 13-3. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 13-3. Input Capture Unit Block Diagram



Note: The Analog Comparator Output (ACO) can only trigger the Timer/Counter1 IC Unit– not Timer/Counter3.

When a change of the logic level (an event) occurs on the Input Capture pin (ICPn), 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 (TCNTn) is written to the Input Capture Register (ICRn). The Input Capture Flag (ICFn) is set at the same system clock as the TCNTn value is copied into ICRn Register. If enabled (ICIEn = 1), the Input Capture Flag generates an Input Capture interrupt. The ICFn flag is automatically

Table 13-3 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 13-3. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM[1]

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGMn3=0: Normal port operation, OCnA/OCnB/OCnC disconnected. WGMn3=1: Toggle OCnA on Compare Match, OCnB/OCnC reserved.
1	0	Clear OCnA/OCnB/OCnC on Compare Match when up-counting. Set OCnA/OCnB/OCnC on Compare Match when downcounting.
1	1	Set OCnA/OCnB/OCnC on Compare Match when up-counting. Clear OCnA/OCnB/OCnC on Compare Match when downcounting.

Note: 1. A special case occurs when OCnA/OCnB/OCnC equals TOP and COMnA1/COMnB1/COMnC1 is set. See “Phase Correct PWM Mode” on page 130. for more details.

- **Bit 1:0 – WGMn1:0: Waveform Generation Mode**

Combined with the WGMn3:2 bits found in the TCCRnB Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 13-4. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See “Modes of Operation” on page 126.).

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 154.

14.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM21:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

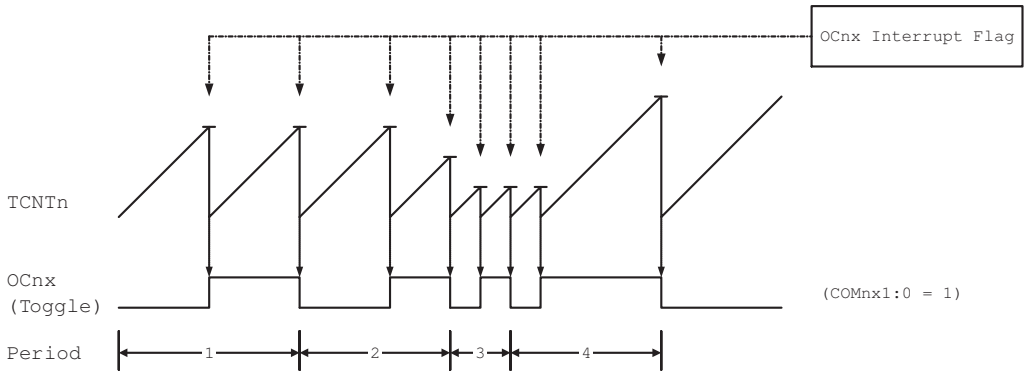
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

14.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM21:0 = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

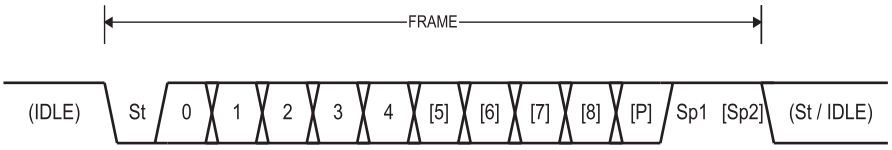
The timing diagram for the CTC mode is shown in Figure 14-6. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.

Figure 14-6. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current 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.

Figure 17-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

The frame format used by the USARTn is set by the UCSZn2:0, UPMn1:0 and USBSn bits in UCSRnB and UCSRnC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USARTn Character SiZe (UCSZn2:0) bits select the number of data bits in the frame. The USARTn Parity mode (UPMn1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USARTn Stop Bit Select (USBSn) bit. The Receiver ignores the second stop bit. An FEn (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

17.5.2 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

$\mathbf{P_{even}}$	Parity bit using even parity
$\mathbf{P^{odd}}$	Parity bit using odd parity
$\mathbf{d_n}$	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

17.6 USART Initialization

The USARTn has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USARTn operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are 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

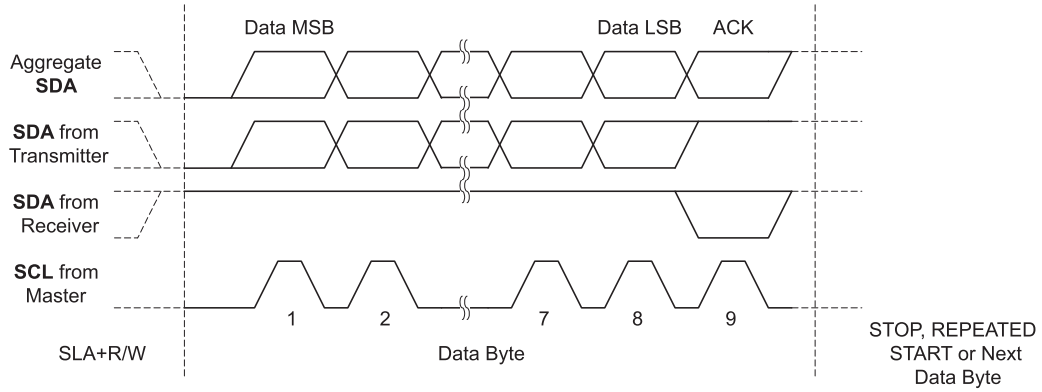
Table 17-10. Examples of UBRRn Settings for Commonly Frequencies (Continued)

Baud Rate (bps)	$f_{clk_{io}}$ = 3.6864 MHz				$f_{clk_{io}}$ = 4.0000 MHz				$f_{clk_{io}}$ = 7.3728 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
500k	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max.[1]	230.4 Kbps		460.8 Kbps		250 Kbps		0.5 Mbps		460.8 Kbps		921.6 Kbps	

Note: 1. UBRRn = 0, Error = 0.0%

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 18-5. Data Packet Format

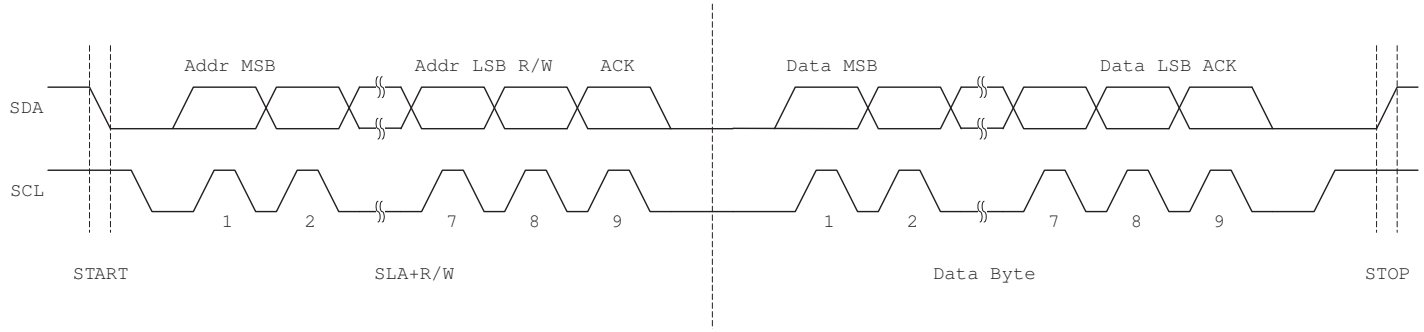


18.3.5 Combining Address and Data Packets Into a Transmission

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 18-6 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 18-6. Typical Data Transmission



18.4 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:

18.5.3 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.

18.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the 7-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.

18.5.5 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 available. As long as the TWINT flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a slave

The application writes the TWSTA bit to one when it desires to become a master on the Two-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the Two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Reserved Bit**

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

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT flag is high.

18.6.3 TWI Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

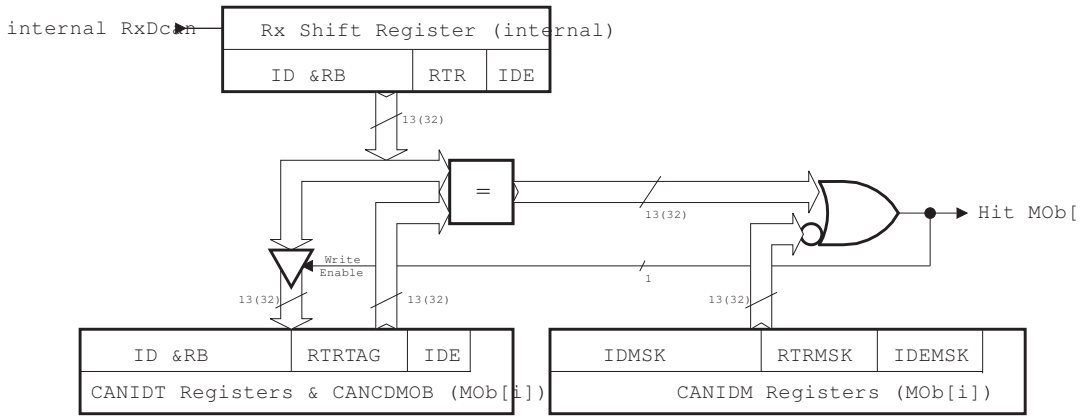
- **Bits 7.3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the Two-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

Figure 19-10. Acceptance Filter Block Diagram



Note: Examples:

Full filtering: to accept only ID = 0x317 in part A.
- ID MSK = 111 1111 1111 $_b$
- ID TAG = 011 0001 0111 $_b$

Partiel filtering: to accept ID from 0x310 up to 0x317 in part A.
- ID MSK = 111 1111 1000 $_b$
- ID TAG = 011 0001 0xxx $_b$

No filtering: to accept all ID's from 0x000 up to 0x7FF in part A.
- ID MSK = 000 0000 0000 $_b$
- ID TAG = xxx xxxx xxxx $_b$

19.5.3 MOb Page

Every MOb is mapped into a page to save place. The page number is the MOb number. This page number is set in CANPAGE register. The number 15 is reserved for factory tests.

CANHPMOB register gives the MOb having the highest priority in CANSIT registers. It is formatted to provide a direct entry for CANPAGE register. Because CANHPMOB codes CANSIT registers, it will be only updated if the corresponding enable bits (ENRX, ENTX, ENERR) are enabled (c.f. Figure 19-14).

19.5.4 CAN Data Buffers

To preserve register allocation, the CAN data buffer is seen such as a FIFO (with address pointer accessible) into a MOb selection.This also allows to reduce the risks of un-controlled accesses.

There is one FIFO per MOb. This FIFO is accessed into a MOb page thanks to the CAN message register.

The data index (INDX) is the address pointer to the required data byte. The data byte can be read or write. The data index is automatically incremented after every access if the AINC* bit is reset. A roll-over is implemented, after data index=7 it is data index=0.

The first byte of a CAN frame is stored at the data index=0, the second one at the data index=1, ...

- **Bit 3:1 – PHS12:0: Phase Segment 1**

This phase is used to compensate for phase edge errors. This segment may be lengthened by the re-synchronization jump width.

$$Tphs1 = Tscl \times (PHS1\ [2:0] + 1)$$

- **Bit 0 – SMP: Sample Point(s)**
  - 0 - once, at the sample point.
  - 1 - three times, the threefold sampling of the bus is the sample point and twice over a distance of a 1/2 period of the Tscl. The result corresponds to the majority decision of the three values.

19.10.11 CAN Timer Control Register - CANTCON

Bit	7	6	5	4	3	2	1	0	
	TPRSC7	TPRSC6	TPRSC5	TPRSC4	TPRSC3	TPRSC2	TRPSC1	TPRSC0	CANTCON
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 – TPRSC7:0: CAN Timer Prescaler**

Prescaler for the CAN timer upper counter range 0 to 255. It provides the clock to the CAN timer if the CAN controller is enabled.

$$Tclk_{CANTIM} = Tclk_{IO} \times 8 \times (CANTCON\ [7:0] + 1)$$

19.10.12 CAN Timer Registers - CANTIML and CANTIMH

Bit	7	6	5	4	3	2	1	0	
	CANTIM7	CANTIM6	CANTIM5	CANTIM4	CANTIM3	CANTIM2	CANTIM1	CANTIM0	CANTIML
	CANTIM15	CANTIM14	CANTIM13	CANTIM12	CANTIM11	CANTIM10	CANTIM9	CANTIM8	CANTIMH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:0 - CANTIM15:0: CAN Timer Count**

CAN timer counter range 0 to 65,535.

19.10.13 CAN TTC Timer Registers - CANTTCL and CANTTCH

Bit	7	6	5	4	3	2	1	0	
	TIMTTC7	TIMTTC6	TIMTTC5	TIMTTC4	TIMTTC3	TIMTTC2	TIMTTC1	TIMTTC0	CANTTCL
	TIMTTC15	TIMTTC14	TIMTTC13	TIMTTC12	TIMTTC11	TIMTTC10	TIMTTC9	TIMTTC8	CANTTCH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:0 - TIMTTC15:0: TTC Timer Count**

CAN TTC timer counter range 0 to 65,535.

Table 25-15. Serial Programming Instruction Set (Continued)

Set **a** = address high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care

Instruction	Instruction Format[1]				Operation[1]
	Byte 1	Byte 2[2]	Byte 3	Byte4	
Load EEPROM Memory Page (page access)	1100 0001	0000 0000	0000 0**bbb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM Memory Page (page access)	1100 0010	000x **aaaa**	**bbbb b000**	xxxx xxxx	Write EEPROM page at address **a**:**b**.
Read Lockbits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. "0"=programmed, "1"=unprogrammed. See Table 25-1 on page 335 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = "0" to program Lock bits. See Table 25-1 on page 335 for details.
Read Signature Byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse Low bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 25-5 on page 337 for details.
Write Fuse High bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 25-4 on page 336 for details.
Write Extended Fuse Bits	1010 1100	1010 0100	xxxx xxxx	**xxxx iiii**	Set bits = "0" to program, "1" to unprogram. See Table 25-3 on page 336 for details.
Read Fuse Low bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. "0"=programmed, "1"=unprogrammed. See Table 25-5 on page 337 for details.
Read Fuse High bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse High bits. "0"=programmed, "1"=unprogrammed. See Table 25-4 on page 336 for details.
Read Extended Fuse Bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read Extended Fuse bits. "0"=programmed, "1"=unprogrammed. See Table 25-3 on page 336 for details.
Read Calibration Byte	0011 1000	000x xxxx	0000 0000	**oooo oooo**	Read Calibration Byte
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = "1", a programming operation is still busy. Wait until this bit returns to "0" before applying another command.

Notes: 1. All bytes are represented by binary digits (0b...).

2. Address bits exceeding PCMSB and EEAMSB (see Table 25-11 on page 340 and Table 25-12 on page 340) are don't care.

25.9 JTAG Programming Overview

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI, and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCR must be cleared. Alternatively, if the JTD bit is set, the external reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in Running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using

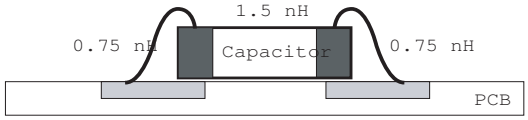
26. Decoupling Capacitors

The operating frequency (i.e. system clock) of the processor determines in 95% of cases the value needed for microcontroller decoupling capacitors.

The hypotheses used as first evaluation for decoupling capacitors are:

- The operating frequency (f_{op}) supplies itself the maximum peak levels of noise. The main peaks are located at f_{op} and 2 • f_{op}.
- An SMC capacitor connected to 2 micro-vias on a PCB has the following characteristics:
  - 1.5 nH from the connection of the capacitor to the PCB,
  - 1.5 nH from the capacitor intrinsic inductance.

Figure 26-1. Capacitor description



According to the operating frequency of the product, the decoupling capacitances are chosen considering the frequencies to filter, f_{op} and 2 • f_{op}.

The relation between frequencies to cut and decoupling characteristics are defined by:

$$f_{op} = \frac{1}{2\Pi\sqrt{LC_1}} \quad \text{and} \quad 2 \bullet f_{op} = \frac{1}{2\Pi\sqrt{LC_2}}$$

where:

- L: the inductance equivalent to the global inductance on the Vcc/Gnd lines.
- C_1 & C_2: decoupling capacitors (C_1 = 4 • C_2).

Then, in normalized value range, the decoupling capacitors become:

Table 26-1. Decoupling Capacitors vs. Frequency

f_{op}, operating frequency	C_1	C_2
16 MHz	33 nF	10 nF
12 MHz	56 nF	15 nF
10 MHz	82 nF	22 nF
8 MHz	120 nF	33 nF
6 MHz	220 nF	56 nF
4 MHz	560 nF	120 nF

These decoupling capacitors must to be implemented as close as possible to each pair of power supply pins:

- 21-22 and 52-53 for logic sub-system,
- 64-63 for analogical sub-system.

Nevertheless, a bulk capacitor of 10-47 µF is also needed on the power distribution network of the PCB, near the power source.

For further information, please refer to Application Notes AVR040 “EMC Design Considerations“ and AVR042 “Hardware Design Considerations“ on the Atmel web site.

27.8 ADC Characteristics([1]

Table 27-5. ADC Characteristics, Single Ended Channels

Symbol	Parameter	Condition	Min[2]	Typ[2]	Max[2]	Units
	Resolution	Single Ended Conversion		10		Bits
	Absolute accuracy (Included INL, DNL, Quantization Error, Gain and Offset Error)	Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz		1.5	3	LSB
		Single Ended Conversion V_{REF} = 4V, Vcc = 4V			3	LSB
		Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz Noise Reduction Mode		1.5	3	LSB
		Single Ended Conversion V_{REF} = 4V, Vcc = 4V Noise Reduction Mode			3	LSB
	Integral Non-linearity (INL)	Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz		0.6	1.5	LSB
	Differential Non-linearity (DNL)	Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz		0.3	1.5	LSB
	Gain Error	Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz	– 3	0	+ 3	LSB
	Offset Error	Single Ended Conversion V_{REF} = 4V, Vcc = 4V ADC clock = 200 kHz	– 2.5	1	+ 2.5	LSB
	Clock Frequency	Free Running Conversion	50		1000	kHz
	Conversion Time	Free Running Conversion	65		260	µs
AV_{CC}	Analog Supply Voltage		V_{CC} – 0.3 [3]		V_{CC} + 0.3 [4]	V
V_{REF}	External Reference Voltage		2.0		AV_{CC}	V
V_{IN}	Input voltage		GND		V_{REF}	V
	Input bandwidth			38.5		kHz
V_{INT}	Internal Voltage Reference		2.4	2.56	2.7	V
R_{REF}	Reference Input Resistance		24	32	40	kΩ
R_{AIN}	Analog Input Resistance			100		MΩ

Notes:
1. 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.
2. Values are guidelines only.
3. Minimum for AV_{CC} is 2.7 V.
4. Maximum for AV_{CC} is 5.5 V