



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	10MHz
Connectivity	I <sup>2</sup> C, 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)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-VQFN (5x5)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega48v-10mj">https://www.e-xfl.com/product-detail/microchip-technology/atmega48v-10mj</a>

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in [Table 6-9 on page 32](#).

**Table 6-9.** Start-up times for the internal calibrated RC Oscillator clock selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset ( $V_{CC} = 5.0V$ )	SUT1..0
BOD enabled	6 CK	14CK <sup>(1)</sup>	00
Fast rising power	6 CK	14CK + 4.1 ms	01
Slowly rising power	6 CK	14CK + 65 ms <sup>(2)</sup>	10
Reserved			11

Note: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4.1 ms to ensure programming mode can be entered.  
2. The device is shipped with this option selected.

### 6.6.1 Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

#### • Bits 7..0 – CAL7..0: Oscillator Calibration Value

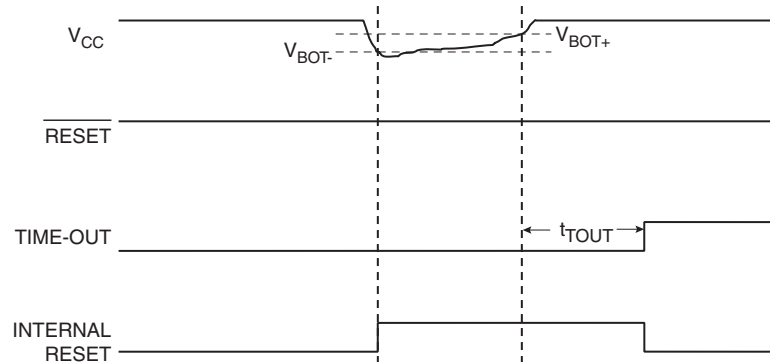
The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. The factory-calibrated value is automatically written to this register during chip reset, giving an oscillator frequency of 8.0 MHz at 25°C. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to any frequency in the range 7.3 - 8.1 MHz within  $\pm 1\%$  accuracy. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and Flash write accesses, and these write times will be affected accordingly. If the EEPROM or Flash are written, do not calibrate to more than 8.8 MHz. Otherwise, the EEPROM or Flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range. Incrementing CAL6..0 by 1 will give a frequency increment of less than 2% in the frequency range 7.3 - 8.1 MHz.

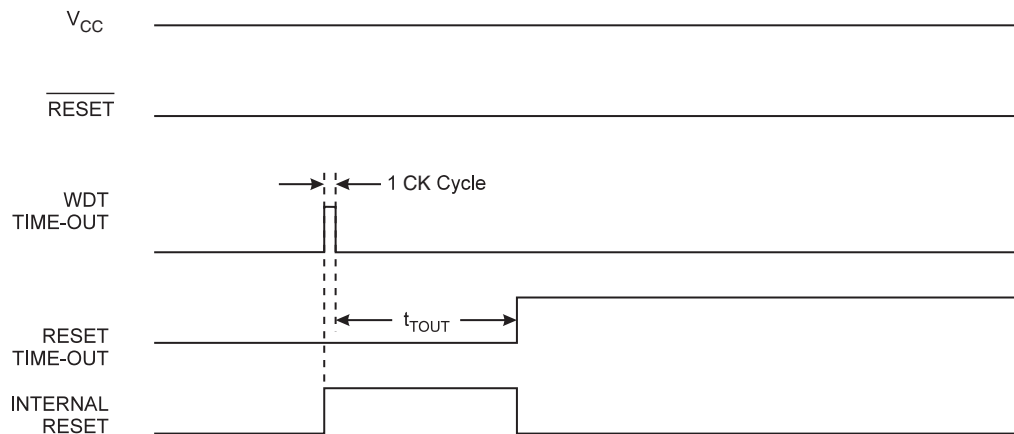
**Figure 8-5.** Brown-out Reset During Operation



## 8.0.6 Watchdog System Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period  $t_{TOUT}$ . Refer to [page 49](#) for details on operation of the Watchdog Timer.

**Figure 8-6.** Watchdog System Reset During Operation



## 8.0.7 MCU Status Register – MCUSR

The MCU Status Register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0					See Bit Description

- **Bit 7..4: Res: Reserved Bits**

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

- **Bit 3 – WDRF: Watchdog System Reset Flag**

This bit is set if a Watchdog System Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

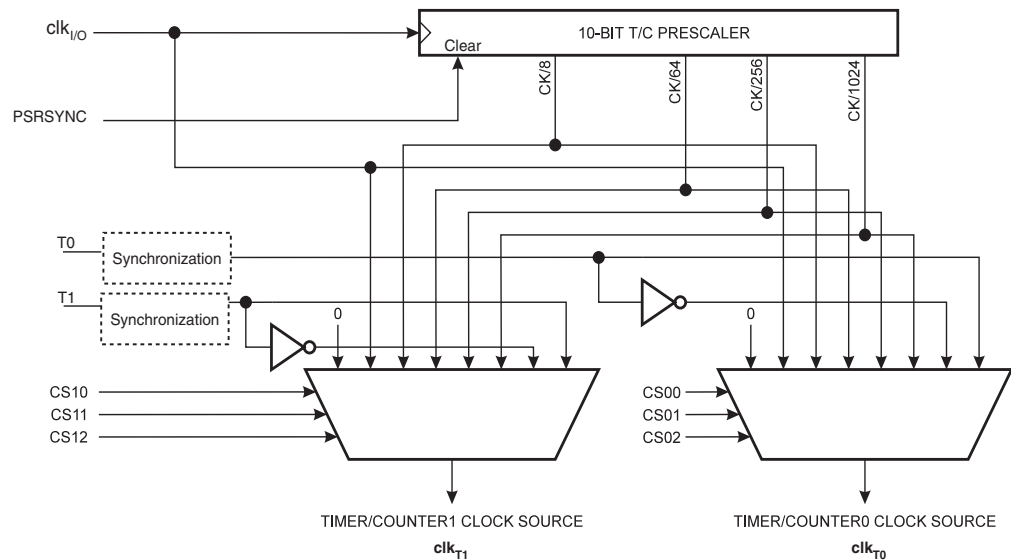
This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ( $f_{ExtClk} < f_{clk\_I/O}/2$ ) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than  $f_{clk\_I/O}/2.5$ .

An external clock source can not be prescaled.

**Figure 14-2.** Prescaler for Timer/Counter0 and Timer/Counter1<sup>(1)</sup>



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in [Figure 14-1](#).

## 15.1.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source 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 ( $\text{clk}_{T2}$ ).

The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See Section “15.4” on page 140. for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

## 15.1.2 Definitions

Many register and bit references in this document are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value and so on.

The definitions in Table 15-1 are also used extensively throughout the section.

**Table 15-1.** Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

## 15.2 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source  $\text{clk}_{T2}$  is by default equal to the MCU clock,  $\text{clk}_{IO}$ . When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see “Asynchronous Status Register – ASSR” on page 156. For details on clock sources and prescaler, see “Timer/Counter Prescaler” on page 158.

## 15.3 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 15-2 shows a block diagram of the counter and its surrounding environment.

## 15.8.2 Timer/Counter Control Register B – TCCR2B

Bit	7	6	5	4	3	2	1	0	
	<b>FOC2A</b>	<b>FOC2B</b>	–	–	<b>WGM22</b>	<b>CS22</b>	<b>CS21</b>	<b>CS20</b>	<b>TCCR2B</b>
Read/Write	W	W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC2A: Force Output Compare A**

The FOC2A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2A output is changed according to its COM2A1:0 bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A1:0 bits that determines the effect of the forced compare.

A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP.

The FOC2A bit is always read as zero.

- **Bit 6 – FOC2B: Force Output Compare B**

The FOC2B bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2B output is changed according to its COM2B1:0 bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B1:0 bits that determines the effect of the forced compare.

A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP.

The FOC2B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

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

- **Bit 3 – WGM22: Waveform Generation Mode**

See the description in the ["Timer/Counter Control Register A – TCCR2A" on page 149](#).

- **Bit 2:0 – CS22:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see [Table 15-9](#).

### Assembly Code Example<sup>(1)</sup>

```

SPI_MasterInit:
    ; Set MOSI and SCK output, all others input
    ldi r17, (1<<DD_MOSI) | (1<<DD_SCK)
    out DDR_SPI, r17
    ; Enable SPI, Master, set clock rate fck/16
    ldi r17, (1<<SPE) | (1<<MSTR) | (1<<SPR0)
    out SPCR, r17
    ret

SPI_MasterTransmit:
    ; Start transmission of data (r16)
    out SPDR, r16
Wait_Transmit:
    ; Wait for transmission complete
    sbis SPSR, SPIF
    rjmp Wait_Transmit
    ret

```

### C Code Example<sup>(1)</sup>

```

void SPI_MasterInit(void)
{
    /* Set MOSI and SCK output, all others input */
    DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
    /* Enable SPI, Master, set clock rate fck/16 */
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while (!(SPSR & (1<<SPIF)))
        ;
}

```

Note: 1. See "About Code Examples" on page 6.

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

## Assembly Code Example<sup>(1)</sup>

```

SPI_SlaveInit:
    ; Set MISO output, all others input
    ldi  r17, (1<<DD_MISO)
    out   DDR_SPI, r17
    ; Enable SPI
    ldi  r17, (1<<SPE)
    out   SPCR, r17
    ret

SPI_SlaveReceive:
    ; Wait for reception complete
    sbis  SPSR, SPIF
    rjmp SPI_SlaveReceive
    ; Read received data and return
    in   r16, SPDR
    ret

```

## C Code Example<sup>(1)</sup>

```

void SPI_SlaveInit(void)
{
    /* Set MISO output, all others input */
    DDR_SPI = (1<<DD_MISO);
    /* Enable SPI */
    SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
    /* Wait for reception complete */
    while (!(SPSR & (1<<SPIF)))
    ;
    /* Return Data Register */
    return SPDR;
}

```

Note: 1. See "About Code Examples" on page 6.



- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If  $\overline{SS}$  is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to [Figure 16-3](#) and [Figure 16-4](#) for an example. The CPOL functionality is summarized below:

**Table 16-2.** CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to [Figure 16-3](#) and [Figure 16-4](#) for an example. The CPOL functionality is summarized below:

**Table 16-3.** CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

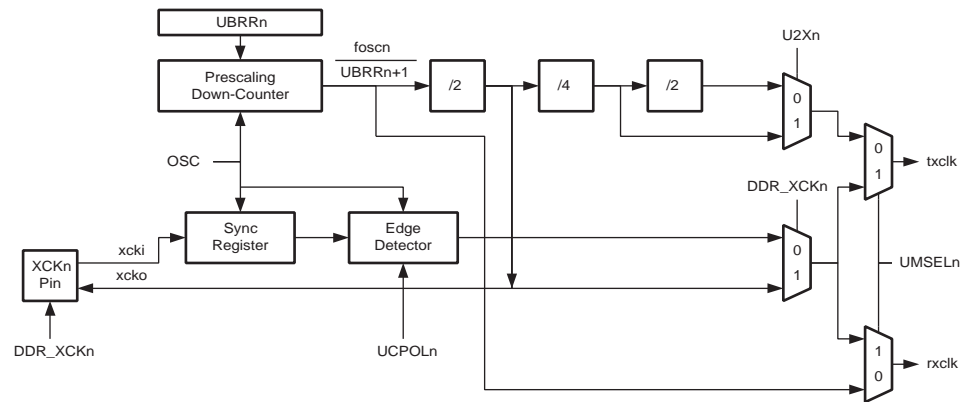
- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency  $f_{osc}$  is shown in the following table:

**Table 16-4.** Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

**Figure 17-2.** Clock Generation Logic, Block Diagram



Signal description:

<b>txclk</b>	Transmitter clock (Internal Signal).
<b>rxclk</b>	Receiver base clock (Internal Signal).
<b>xcki</b>	Input from XCK pin (internal Signal). Used for synchronous slave operation.
<b>xcko</b>	Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
<b>fosc</b>	XTAL pin frequency (System Clock).

### 17.2.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to [Figure 17-2](#).

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock ( $f_{osc}$ ), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output ( $= f_{osc}/(UBRRn+1)$ ). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSELn, U2Xn and DDR\_XCKn bits.

## 17.6.3 Receive Complete Flag and Interrupt

The USART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXCn) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXENn = 0), the receive buffer will be flushed and consequently the RXCn bit will become zero.

When the Receive Complete Interrupt Enable (RXCIEn) in UCSRnB is set, the USART Receive Complete interrupt will be executed as long as the RXCn Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDRn in order to clear the RXCn Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

## 17.6.4 Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FEn), Data OverRun (DORn) and Parity Error (UPEn). All can be accessed by reading UCSRnA. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSRnA must be read before the receive buffer (UDRn), since reading the UDRn I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRnA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FEn) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FEn Flag is zero when the stop bit was correctly read (as one), and the FEn Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FEn Flag is not affected by the setting of the USBSn bit in UCSRnC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRnA.

The Data OverRun (DORn) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. If the DORn Flag is set there was one or more serial frame lost between the frame last read from UDRn, and the next frame read from UDRn. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. The DORn Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPEn) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPEn bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see ["Parity Bit Calculation" on page 173](#) and ["Parity Checker" on page 181](#).

## 17.6.5 Parity Checker

The Parity Checker is active when the high USART Parity mode (UPMn1) bit is set. Type of Parity Check to be performed (odd or even) is selected by the UPMn0 bit. When enabled, the Parity Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (UPEn) Flag can then be read by software to check if the frame had a Parity Error.

## Assembly Code Example<sup>(1)</sup>

```

USART_MSPIM_Transfer:
    ; Wait for empty transmit buffer
    sbis UCSRnA, UDREn
    rjmp USART_MSPIM_Transfer
    ; Put data (r16) into buffer, sends the data
    out UDRn,r16
    ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
    sbis UCSRnA, RXCn
    rjmp USART_MSPIM_Wait_RXCn
    ; Get and return received data from buffer
    in r16, UDRn
    ret

```

## C Code Example<sup>(1)</sup>

```

unsigned char USART_Receive( void )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRnA & (1<<UDREn)) );
    /* Put data into buffer, sends the data */
    UDRn = data;
    /* Wait for data to be received */
    while ( !(UCSRnA & (1<<RXCn)) );
    /* Get and return received data from buffer */
    return UDRn;
}

```

Note: 1. See "About Code Examples" on page 6.

### 18.5.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and is always read as zero.

### 18.5.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

## 18.6 USART MSPIM Register Description

The following section describes the registers used for SPI operation using the USART.

### 18.6.1 USART MSPIM I/O Data Register - UDRn

The function and bit description of the USART data register (UDRn) in MSPI mode is identical to normal USART operation. See "USART I/O Data Register n– UDRn" on page 187.

### 18.6.2 USART MSPIM Control and Status Register n A - UCSRnA

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

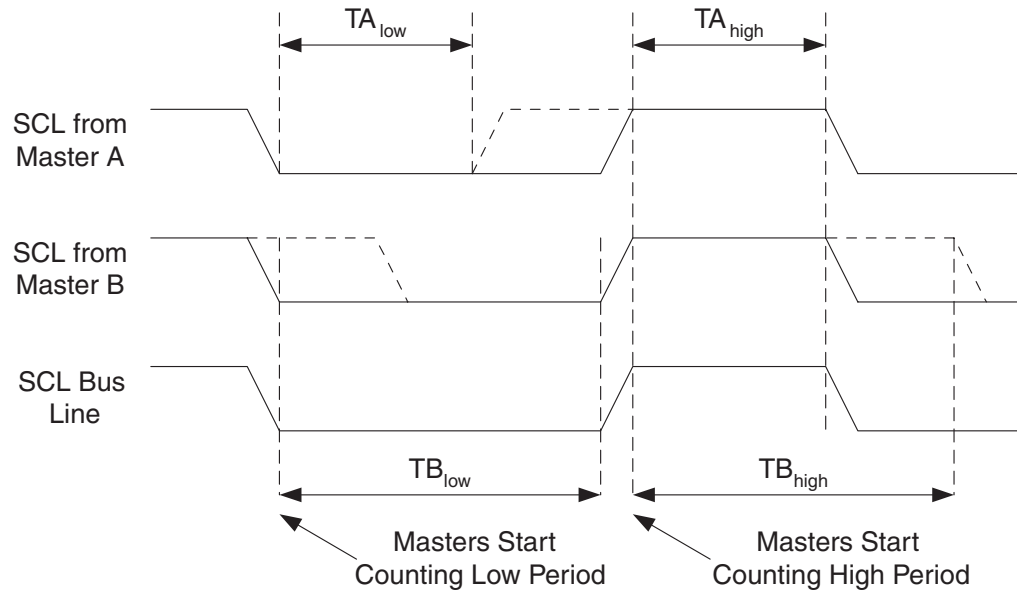
## 19.4 Multi-master Bus Systems, Arbitration and Synchronization

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

After a repeated START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

**Table 19-3. Status codes for Master Transmitter Mode**

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received SLA+R will be transmitted; Logic will switch to Master Receiver mode
		Load SLA+R	0	0	1	X	
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action or	0	1	1	X	
		No TWDR action	1	1	1	X	
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action or	0	1	1	X	
		No TWDR action	1	1	1	X	
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action or	0	1	1	X	
		No TWDR action	1	1	1	X	
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action or	0	1	1	X	
		No TWDR action	1	1	1	X	
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered A START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	X	

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

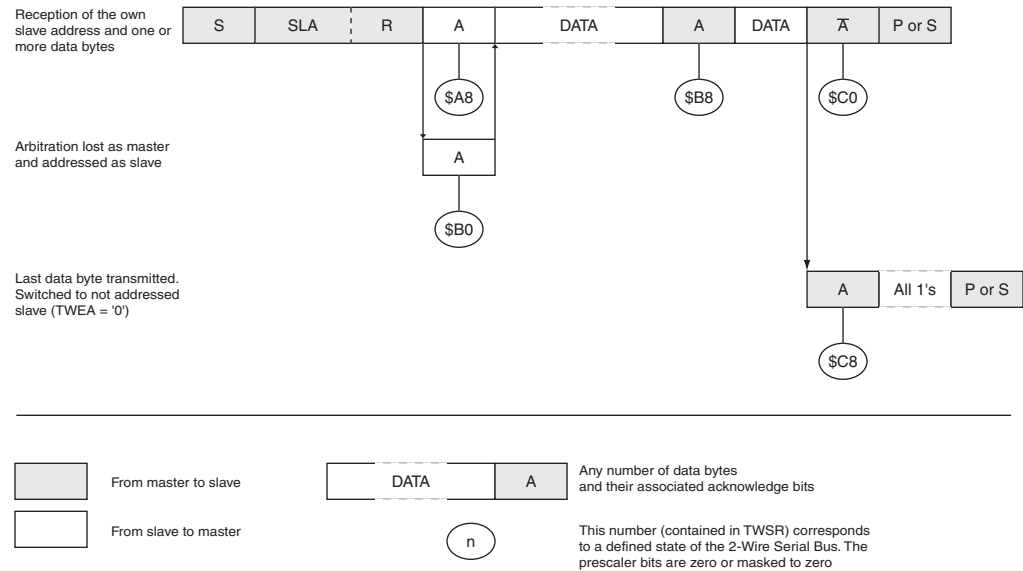
When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in [Table 19-5](#). The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these Sleep modes.

**Figure 19-19. Formats and States in the Slave Transmitter Mode**



## 19.8.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see [Table 19-7](#).

Status 0xF8 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 0x00 indicates that a bus error has occurred during a 2-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 19-7. Miscellaneous States**

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xF8	No relevant state information available; TWINT = "0"	No TWDR action	No TWCR action				Wait or proceed current transfer
0x00	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.

## 19.8.6 Combining Several TWI Modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

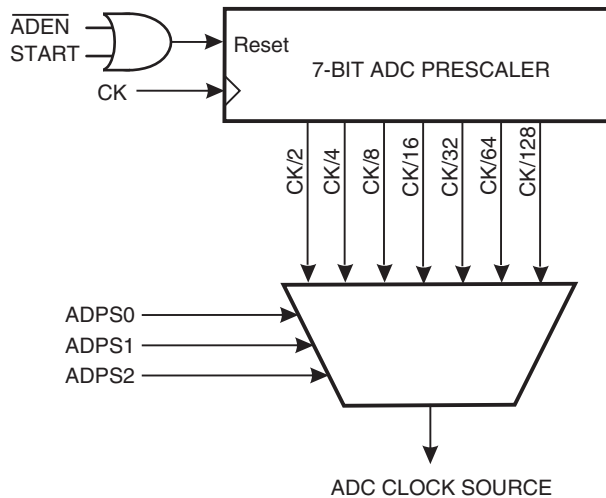
1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.
3. The reading must be performed.
4. The transfer must be finished.



If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

## 21.3 Prescaling and Conversion Timing

**Figure 21-3.** ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

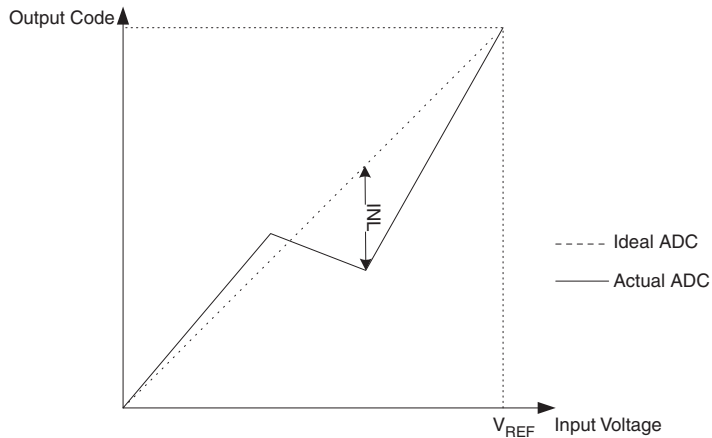
A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

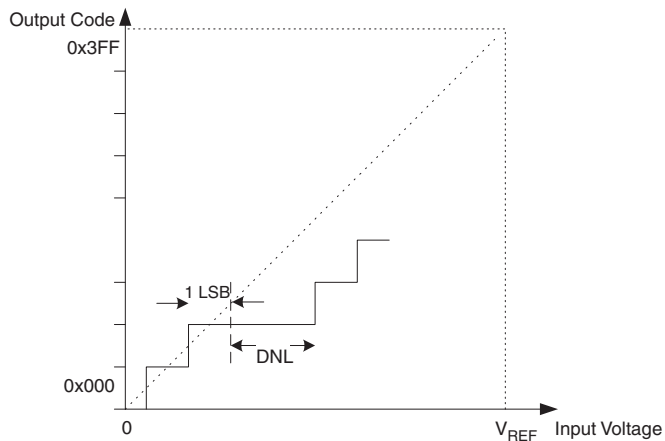
In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see [Table 21-1 on page 244](#).

**Figure 21-12. Integral Non-linearity (INL)**



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

**Figure 21-13. Differential Non-linearity (DNL)**



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

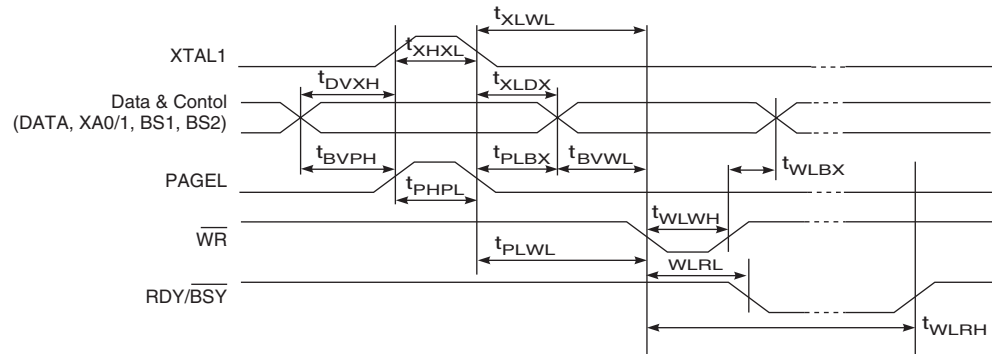
## 25.7.14 Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 287 for details on Command and Address loading):

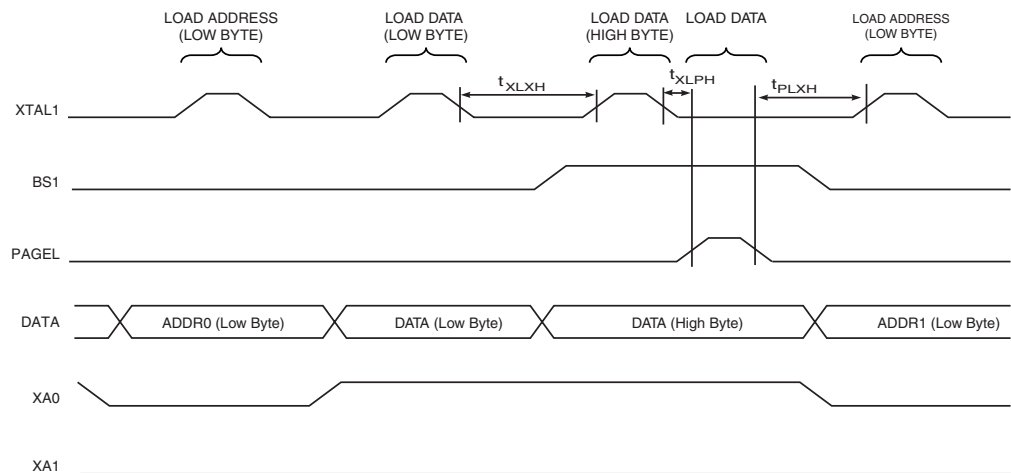
1. A: Load Command "0000 1000".
2. B: Load Address Low Byte, 0x00.
3. Set  $\overline{OE}$  to "0", and BS1 to "1". The Calibration byte can now be read at DATA.
4. Set  $\overline{OE}$  to "1".

## 25.7.15 Parallel Programming Characteristics

**Figure 25-7.** Parallel Programming Timing, Including some General Timing Requirements



**Figure 25-8.** Parallel Programming Timing, Loading Sequence with Timing Requirements<sup>(1)</sup>



Note: 1. The timing requirements shown in Figure 25-7 (i.e.,  $t_{DVXH}$ ,  $t_{XHL}$ , and  $t_{XLDX}$ ) also apply to load-  
ing operation.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	250
(0x7B)	ADCSRB	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	253
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	251
(0x79)	ADCH	ADC Data Register High byte								253
(0x78)	ADCL	ADC Data Register Low byte								253
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	154
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	133
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	104
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	87
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	87
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	87
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	84
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	OSCCAL	Oscillator Calibration Register								32
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	40
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	35
(0x60)	WDTCR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	52
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	–	–	–	–	–	(SP10) <sup>5</sup>	SP9	SP8	11
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	11
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	SPMIE	(RWWSB) <sup>5</sup>	–	(RWWSR) <sup>5</sup>	BLBSET	PGWRT	PGERS	SELFPRGEN	269
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	–	–	PUD	–	–	IVSEL	IVCE	
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	37
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	Reserved	–	–	–	–	–	–	–	–	
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	236
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPI Data Register								166
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	166
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	164
0x2B (0x4B)	GPOR2	General Purpose I/O Register 2								24
0x2A (0x4A)	GPOR1	General Purpose I/O Register 1								24
0x29 (0x49)	Reserved	–	–	–	–	–	–	–	–	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								
0x26 (0x46)	TCNT0	Timer/Counter0 (8-bit)								
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	PSRAS1	PSRSYN0	137/158
0x22 (0x42)	EEARH	(EEPROM Address Register High Byte) <sup>5</sup>								19
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								19
0x20 (0x40)	EEDR	EEPROM Data Register								19
0x1F (0x3F)	EEDR	–	–	EEP1	EEP0	EERIE	EEMPE	EEPE	EERE	19
0x1E (0x3E)	GPOR0	General Purpose I/O Register 0								24
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	85
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	85

## 29. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
<b>ARITHMETIC AND LOGIC INSTRUCTIONS</b>					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	RdI,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	RdI,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow 0x00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
<b>BRANCH INSTRUCTIONS</b>					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
JMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP <sup>(1)</sup>	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL <sup>(1)</sup>	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow STACK$	None	4
RETI		Interrupt Return	$PC \leftarrow STACK$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if $(Rd = Rr)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z, N, V, C, H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z, N, V, C, H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z, N, V, C, H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if $(Rr(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if $(Rr(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if $(P(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch if Equal	if $(Z = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRNE	k	Branch if Not Equal	if $(Z = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCS	k	Branch if Carry Set	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCC	k	Branch if Carry Cleared	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRSH	k	Branch if Same or Higher	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLO	k	Branch if Lower	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRMI	k	Branch if Minus	if $(N = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRPL	k	Branch if Plus	if $(N = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTS	k	Branch if T Flag Set	if $(T = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTC	k	Branch if T Flag Cleared	if $(T = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if $(V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if $(V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2