



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	Active
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-10mur">https://www.e-xfl.com/product-detail/microchip-technology/atmega48v-10mur</a>

- **Bit 1 – EXTRF: External Reset Flag**

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

- **Bit 0 – PORF: Power-on Reset Flag**

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

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

## 8.1 Internal Voltage Reference

ATmega48/88/168 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC.

### 8.1.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in [Table 8-4](#). To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] Fuses).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

**Table 8-4.** Internal Voltage Reference Characteristics

Symbol	Parameter	Condition	Min <sup>(1)</sup>	Typ <sup>(1)</sup>	Max <sup>(1)</sup>	Units
$V_{BG}$	Bandgap reference voltage	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$	1.0	1.1	1.2	V
$t_{BG}$	Bandgap reference start-up time	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$		40	70	$\mu\text{s}$
$I_{BG}$	Bandgap reference current consumption	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$		10		$\mu\text{A}$

Note: 1. Values are guidelines only. Actual values are TBD.

becomes the inverting output of the Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin cannot be used as an I/O pin.

PCINT7: Pin Change Interrupt source 7. The PB7 pin can serve as an external interrupt source.

If PB7 is used as a clock pin, DDB7, PORTB7 and PINB7 will all read 0.

- **XTAL1/TOSC1/PCINT6 – Port B, Bit 6**

XTAL1: Chip clock Oscillator pin 1. Used for all chip clock sources except internal calibrated RC Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

TOSC1: Timer Oscillator pin 1. Used only if internal calibrated RC Oscillator is selected as chip clock source, and the asynchronous timer is enabled by the correct setting in ASSR. When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PB6 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

PCINT6: Pin Change Interrupt source 6. The PB6 pin can serve as an external interrupt source.

If PB6 is used as a clock pin, DDB6, PORTB6 and PINB6 will all read 0.

- **SCK/PCINT5 – Port B, Bit 5**

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

PCINT5: Pin Change Interrupt source 5. The PB5 pin can serve as an external interrupt source.

- **MISO/PCINT4 – Port B, Bit 4**

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB4. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

PCINT4: Pin Change Interrupt source 4. The PB4 pin can serve as an external interrupt source.

- **MOSI/OC2/PCINT3 – Port B, Bit 3**

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB3. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB3. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB3 bit.

OC2, Output Compare Match Output: The PB3 pin can serve as an external output for the Timer/Counter2 Compare Match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

PCINT3: Pin Change Interrupt source 3. The PB3 pin can serve as an external interrupt source.

- **$\overline{SS}$ /OC1B/PCINT2 – Port B, Bit 2**

$\overline{SS}$ : Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB2. As a Slave, the SPI is activated when this pin is driven low.

**Table 10-4.** Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/XTAL2/ TOSC2/PCINT7 <sup>(1)</sup>	PB6/XTAL1/ TOSC1/PCINT6 <sup>(1)</sup>	PB5/SCK/ PCINT5	PB4/MISO/ PCINT4
PUOE	$\overline{\text{INTRC}} \cdot \overline{\text{EXTCK}} + \text{AS2}$	$\overline{\text{INTRC}} + \text{AS2}$	$\text{SPE} \cdot \overline{\text{MSTR}}$	$\text{SPE} \cdot \text{MSTR}$
PUOV	0	0	$\text{PORTB5} \cdot \overline{\text{PUD}}$	$\text{PORTB4} \cdot \overline{\text{PUD}}$
DDOE	$\overline{\text{INTRC}} \cdot \overline{\text{EXTCK}} + \text{AS2}$	$\overline{\text{INTRC}} + \text{AS2}$	$\text{SPE} \cdot \overline{\text{MSTR}}$	$\text{SPE} \cdot \text{MSTR}$
DDOV	0	0	0	0
PVOE	0	0	$\text{SPE} \cdot \text{MSTR}$	$\text{SPE} \cdot \overline{\text{MSTR}}$
PVOV	0	0	SCK OUTPUT	SPI SLAVE OUTPUT
DIEOE	$\overline{\text{INTRC}} \cdot \overline{\text{EXTCK}} + \text{AS2} + \text{PCINT7} \cdot \text{PCIE0}$	$\overline{\text{INTRC}} + \text{AS2} + \text{PCINT6} \cdot \text{PCIE0}$	$\text{PCINT5} \cdot \text{PCIE0}$	$\text{PCINT4} \cdot \text{PCIE0}$
DIEOV	$(\text{INTRC} + \text{EXTCK}) \cdot \overline{\text{AS2}}$	$\text{INTRC} \cdot \overline{\text{AS2}}$	1	1
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT SCK INPUT	PCINT4 INPUT SPI MSTR INPUT
AIO	Oscillator Output	Oscillator/Clock Input	—	—

Notes: 1. INTRC means that one of the internal RC Oscillators are selected (by the CKSEL fuses), EXTCK means that external clock is selected (by the CKSEL fuses).

**Table 10-5.** Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/MOSI/ OC2/PCINT3	PB2/ $\overline{\text{SS}}$ / OC1B/PCINT2	PB1/OC1A/ PCINT1	PB0/ICP1/ PCINT0
PUOE	$\text{SPE} \cdot \text{MSTR}$	$\text{SPE} \cdot \overline{\text{MSTR}}$	0	0
PUOV	$\text{PORTB3} \cdot \overline{\text{PUD}}$	$\text{PORTB2} \cdot \overline{\text{PUD}}$	0	0
DDOE	$\text{SPE} \cdot \overline{\text{MSTR}}$	$\text{SPE} \cdot \overline{\text{MSTR}}$	0	0
DDOV	0	0	0	0
PVOE	$\text{SPE} \cdot \text{MSTR} + \text{OC2A ENABLE}$	OC1B ENABLE	OC1A ENABLE	0
PVOV	SPI MSTR OUTPUT + OC2A	OC1B	OC1A	0
DIEOE	$\text{PCINT3} \cdot \text{PCIE0}$	$\text{PCINT2} \cdot \text{PCIE0}$	$\text{PCINT1} \cdot \text{PCIE0}$	$\text{PCINT0} \cdot \text{PCIE0}$
DIEOV	1	1	1	1
DI	PCINT3 INPUT SPI SLAVE INPUT	PCINT2 INPUT SPI $\overline{\text{SS}}$	PCINT1 INPUT	PCINT0 INPUT ICP1 INPUT
AIO	—	—	—	—

## 10.4 Register Description for I/O Ports

### 10.4.1 The Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	<b>PORTB7</b>	<b>PORTB6</b>	<b>PORTB5</b>	<b>PORTB4</b>	<b>PORTB3</b>	<b>PORTB2</b>	<b>PORTB1</b>	<b>PORTB0</b>	<b>PORTB</b>
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	

### 10.4.2 The Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	<b>DDB7</b>	<b>DDB6</b>	<b>DDB5</b>	<b>DDB4</b>	<b>DDB3</b>	<b>DDB2</b>	<b>DDB1</b>	<b>DDB0</b>	<b>DDRB</b>
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	

### 10.4.3 The Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	<b>PINB7</b>	<b>PINB6</b>	<b>PINB5</b>	<b>PINB4</b>	<b>PINB3</b>	<b>PINB2</b>	<b>PINB1</b>	<b>PINB0</b>	<b>PINB</b>
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

### 10.4.4 The Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	–	<b>PORTC6</b>	<b>PORTC5</b>	<b>PORTC4</b>	<b>PORTC3</b>	<b>PORTC2</b>	<b>PORTC1</b>	<b>PORTC0</b>	<b>PORTC</b>
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### 10.4.5 The Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	–	<b>DDC6</b>	<b>DDC5</b>	<b>DDC4</b>	<b>DDC3</b>	<b>DDC2</b>	<b>DDC1</b>	<b>DDC0</b>	<b>DDRC</b>
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### 10.4.6 The Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	–	<b>PINC6</b>	<b>PINC5</b>	<b>PINC4</b>	<b>PINC3</b>	<b>PINC2</b>	<b>PINC1</b>	<b>PINC0</b>	<b>PINC</b>
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

### 10.4.7 The Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	<b>PORTD7</b>	<b>PORTD6</b>	<b>PORTD5</b>	<b>PORTD4</b>	<b>PORTD3</b>	<b>PORTD2</b>	<b>PORTD1</b>	<b>PORTD0</b>	<b>PORTD</b>
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	

### 10.4.8 The Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	<b>DDD7</b>	<b>DDD6</b>	<b>DDD5</b>	<b>DDD4</b>	<b>DDD3</b>	<b>DDD2</b>	<b>DDD1</b>	<b>DDD0</b>	<b>DDRD</b>
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	

#### 11.1.4 Pin Change Interrupt Control Register - PCICR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

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

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

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC12 Interrupt Vector. PCINT23..16 pins are enabled individually by the PCMSK2 Register.

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

When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC11 Interrupt Vector. PCINT14..8 pins are enabled individually by the PCMSK1 Register.

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

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC10 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

#### 11.1.5 Pin Change Interrupt Flag Register - PCIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

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

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

When a logic change on any PCINT23..16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

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

When a logic change on any PCINT14..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

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

```
...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in r16,TCNT1L
in r17,TCNT1H
...
```

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

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

Note: 1. See ["About Code Examples" on page 6](#).

For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBRC", "SBR", and "CBR".

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

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

```
TIM16_WriteTCNT1:
    ; Save global interrupt flag
    in r18,SREG
    ; Disable interrupts
    cli
    ; Set TCNT1 to r17:r16
    out TCNT1H,r17
    out TCNT1L,r16
    ; Restore global interrupt flag
    out SREG,r18
    ret
```

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

```
void TIM16_WriteTCNT1( unsigned int i )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Set TCNT1 to i */
    TCNT1 = i;
    /* Restore global interrupt flag */
    SREG = sreg;
}
```

Note: 1. See "About Code Examples" on page 6.  
For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBRC", "SBR", and "CBR".

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

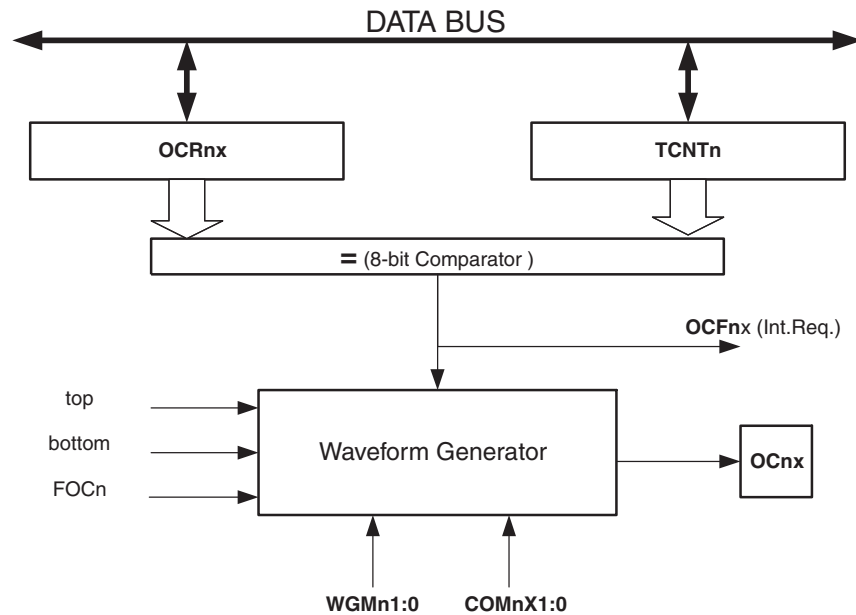
### 13.2.1 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

## 13.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter control Register B* (TCCR1B). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 135.



**Figure 15-3. Output Compare Unit, Block Diagram**


The OCR2x Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2x Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2x Buffer Register, and if double buffering is disabled the CPU will access the OCR2x directly.

#### 15.4.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2x) bit. Forcing compare match will not set the OCF2x Flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x1:0 bits settings define whether the OC2x pin is set, cleared or toggled).

#### 15.4.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

#### 15.4.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM2:0 = 3, and OCR2A when MGM2:0 = 7 (See [Table 15-4 on page 150](#)). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2x Register at the compare match between OCR2x and TCNT2 when the counter increments, and setting (or clearing) the OC2x Register at compare match between OCR2x and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

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

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

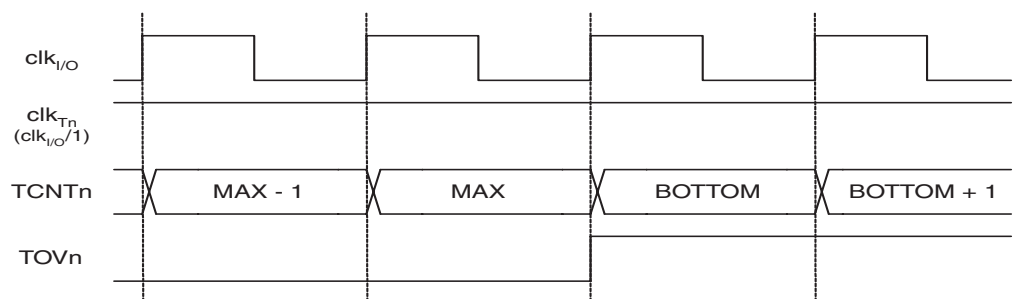
At the very start of period 2 in [Figure 15-7](#) OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR2A changes its value from MAX, like in [Figure 15-7](#). When the OCR2A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

## 15.7 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock ( $clk_{T2}$ ) is therefore shown as a clock enable signal. In asynchronous mode,  $clk_{I/O}$  should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. [Figure 15-8](#) contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

**Figure 15-8.** Timer/Counter Timing Diagram, no Prescaling



[Figure 15-9](#) shows the same timing data, but with the prescaler enabled.

## 17.4 USART Initialization

The USART 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 USART 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 RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter.

## 17.9 USART Register Description

### 17.9.1 USART I/O Data Register n– UDRn

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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 USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDREN Flag in the UCSRnA Register is set. Data written to UDRn when the UDREN Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

### 17.9.2 USART Control and Status Register n A – UCSRnA

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn Flag can be used to generate a Receive Complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). The TXCn Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn Flag can generate a Transmit Complete interrupt (see description of the TXCIEn bit).

- **Bit 5 – UDREN: USART Data Register Empty**

The UDREN Flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREN is one, the buffer is empty, and therefore ready to be written. The UDREN Flag can generate a Data Register Empty interrupt (see description of the UDRIEn bit).

**Table 18-1.** Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate <sup>(1)</sup>	Equation for Calculating UBRRn Value
Synchronous Master mode	$BAUD = \frac{f_{osc}}{2(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{2BAUD} - 1$

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

**BAUD** Baud rate (in bits per second, bps)

**f<sub>osc</sub>** System Oscillator clock frequency

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

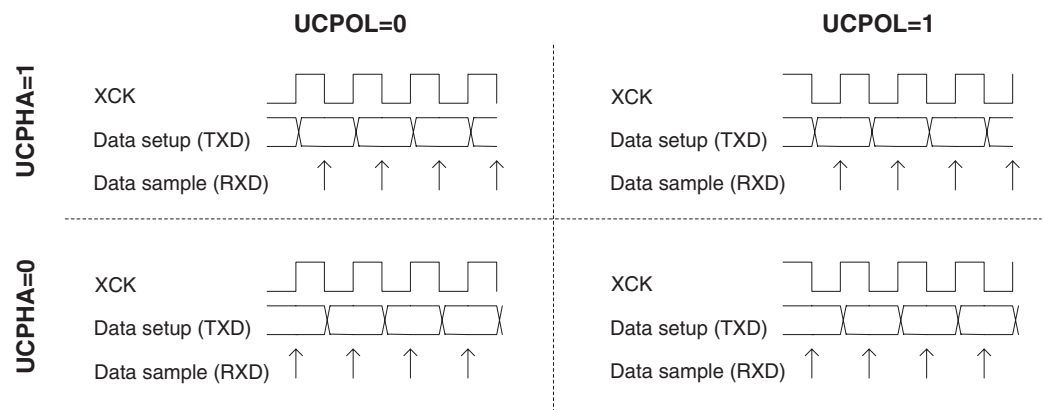
## 18.3 SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAN and UCPOLn. The data transfer timing diagrams are shown in [Figure 18-1](#). Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAN functionality is summarized in [Table 18-2](#). Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

**Table 18-2.** UCPOLn and UCPHAN Functionality-

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)
1	1	3	Setup (Falling)	Sample (Rising)

**Figure 18-1.** UCPHAN and UCPOLn data transfer timing diagrams.



that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

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

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

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

### 19.5.3 Bus Interface Unit

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

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

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

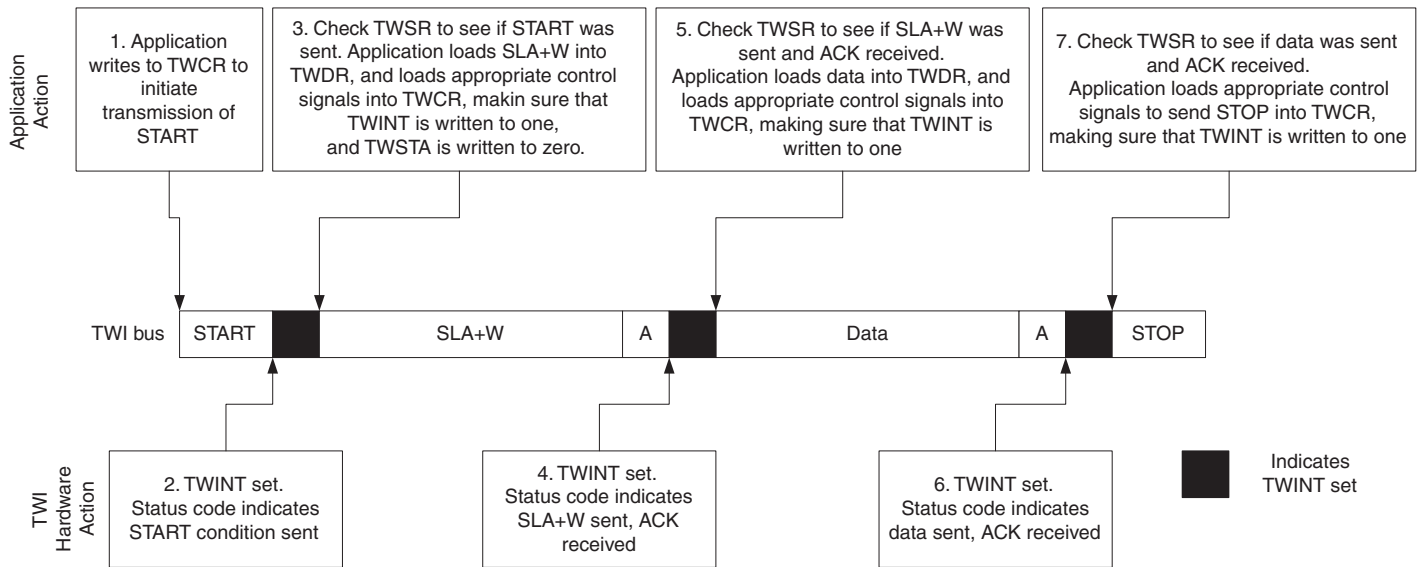
### 19.5.4 Address Match Unit

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

### 19.5.5 Control Unit

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

**Figure 19-11. Interfacing the Application to the TWI in a Typical Transmission**



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.
4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will

## 19.8 Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S: START condition

Rs: REPEATED START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\bar{A}$ : Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

In [Figure 19-13](#) to [Figure 19-19](#), circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

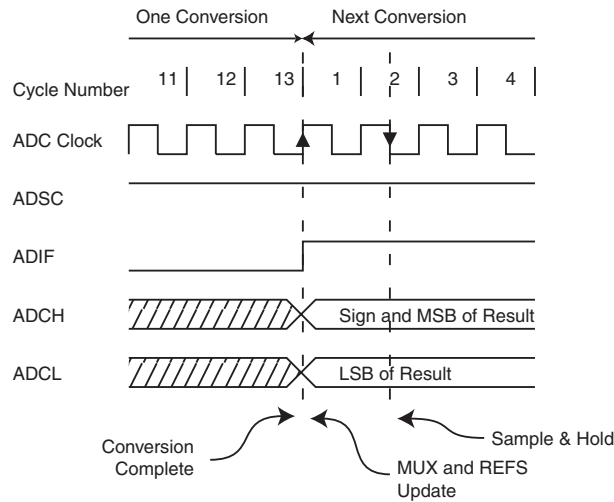
When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in [Table 19-3](#) to [Table 19-6](#). Note that the prescaler bits are masked to zero in these tables.

### 19.8.1 Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see [Figure 19-12](#)). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.



**Figure 21-7. ADC Timing Diagram, Free Running Conversion**



**Table 21-1. ADC Conversion Time**

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5

## 21.4 Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If Auto Triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADSC and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- When ADSC or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

**Table 21-5.** ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

### 21.6.5 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when DIDR0 is written.

- **Bit 5:0 – ADC5D..ADC0D: ADC5..0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC5..0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Note that ADC pins ADC7 and ADC6 do not have digital input buffers, and therefore do not require Digital Input Disable bits.

#### 24.7.14 ATmega168 Boot Loader Parameters

In [Table 24-9](#) through [Table 24-11](#), the parameters used in the description of the self programming are given.

**Table 24-9.** Boot Size Configuration, ATmega168

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - 0x1FFF	0x1F7F	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - 0x1FFF	0x1EFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - 0x1FFF	0x1DFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00 - 0x1FFF	0x1BFF	0x1C00

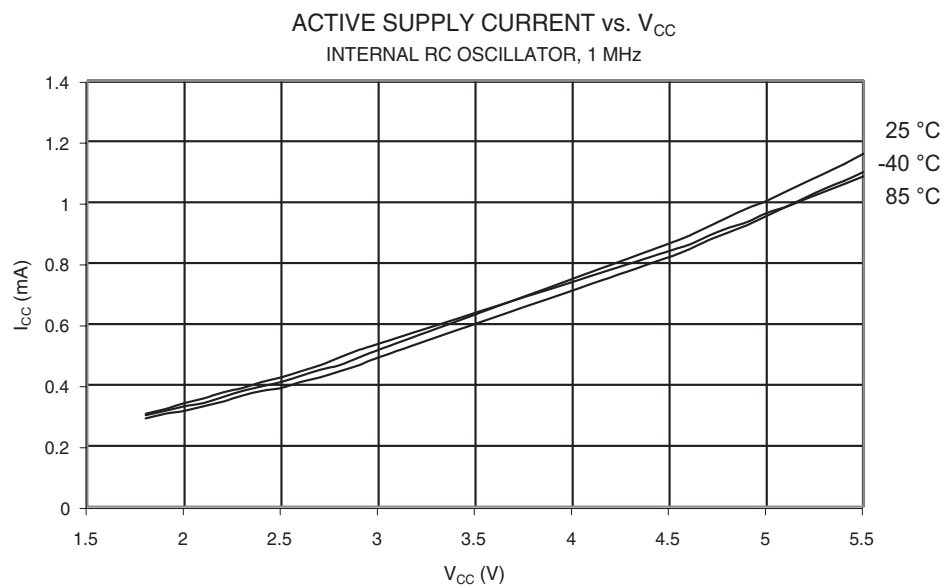
Note: The different BOOTSZ Fuse configurations are shown in [Figure 24-2](#).

**Table 24-10.** Read-While-Write Limit, ATmega168

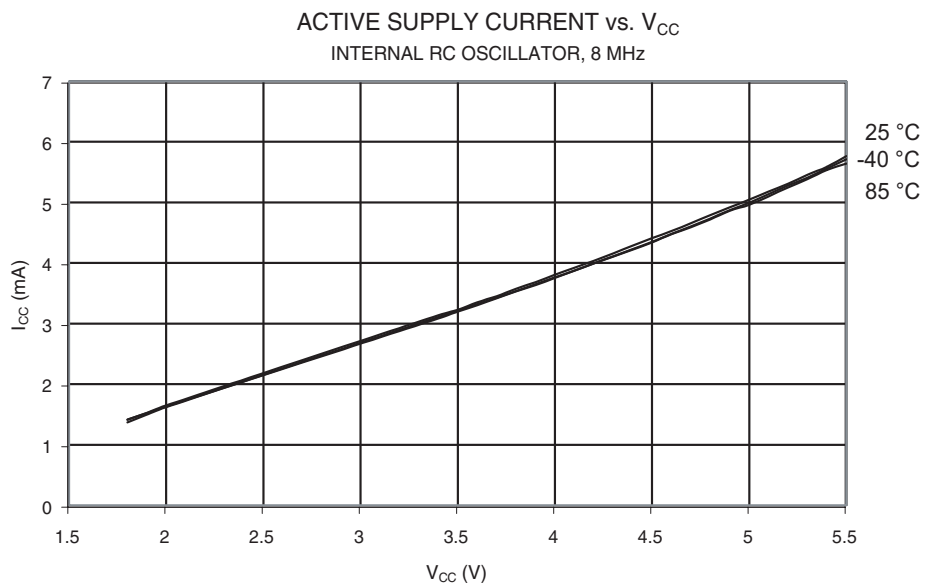
Section	Pages	Address
Read-While-Write section (RWW)	112	0x0000 - 0x1BFF
No Read-While-Write section (NRWW)	16	0x1C00 - 0x1FFF

For details about these two section, see "[NRWW – No Read-While-Write Section](#)" on page 265 and "[RWW – Read-While-Write Section](#)" on page 265

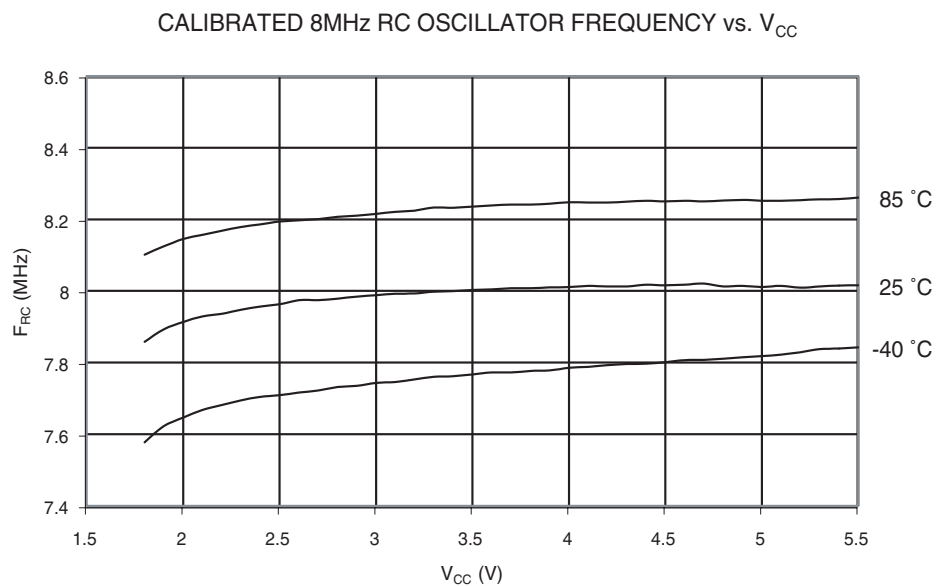
**Figure 27-4.** Active Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 1 MHz)



**Figure 27-5.** Active Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 8 MHz)



**Figure 27-41.** Calibrated 8 MHz RC Oscillator Frequency vs.  $V_{CC}$



**Figure 27-42.** Calibrated 8 MHz RC Oscillator Frequency vs. Oscal Value

