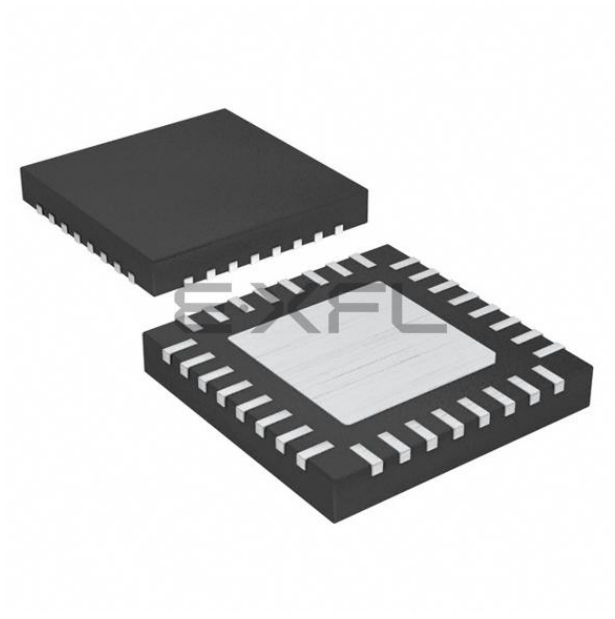


**Welcome to E-XFL.COM**

## 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 | 20MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 27 |
| 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-VFQFN (5x5) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atmega48pb-mur |

4. Tape & Reel.

| Package Type | |
|---|---|
| 32A | 32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP) |
| 32MS1 | 32-pad, 5.0x5.0x0.9mm body, Lead Pitch 0.50mm, Very-thin Fine pitch, Quad Flat No Lead Package (VFQFN) |

## 3.3. ATmega168PB

| Speed [MHz] | Power Supply [V] | Ordering Code(2) | Package(1) | Operational Range |
|---|---|---|---|---|
| 20 | 1.8 - 5.5 | ATmega168PB-AU<br>ATmega168PB-AUR(3)<br>ATmega168PB-MU<br>ATmega168PB-MUR(3) | 32A<br>32A<br>32MS1<br>32MS1 | Industrial<br>(-40°C to +85°C) |
| | | ATmega168PB-AN<br>ATmega168PB-ANR(3)<br>ATmega168PB-MN<br>ATmega168PB-MNR(3) | 32A<br>32A<br>32MS1<br>32MS1 | Industrial<br>(-40°C to +105°C) |

Note: 1. This device can also be supplied in wafer form. Contact your local Atmel sales office for detailed ordering information and minimum quantities.

2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive).Also Halide free and fully Green.

3. Tape & Reel.

| Package Type | |
|---|---|
| 32A | 32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP) |
| 32MS1 | 32-pad, 5.0x5.0x0.9mm body, Lead Pitch 0.50mm, Very-thin Fine pitch, Quad Flat No Lead Package (VFQFN) |

# 11. Capacitive Touch Sensing

## 11.1. QTouch Library

The Atmel® QTouch® Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR® microcontrollers. The QTouch Library includes support for the Atmel QTouch and Atmel QMatrix® acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

The QTouch Library is FREE and downloadable from the Atmel website at the following location: http://www.atmel.com/technologies/touch/. For implementation details and other information, refer to the Atmel QTouch Library User Guide - also available for download from the Atmel website.

**Table 14-10. Start-Up Times for the Internal Calibrated RC Oscillator Clock Selection - SUT**

| Power Conditions | Start-Up Time from Power-down and Power-Save | Additional Delay from Reset ($V_{CC}$ = 5.0V) | SUT[1:0] |
|---|---|---|---|
| BOD enabled | 6 CK | 19CK(1) | 00 |
| Fast rising power | 6 CK | 19CK + 4.1ms | 01 |
| Slowly rising power | 6 CK | 19CK + 65ms(2) | 10 |
| Reserved | | | 11 |

**Note:**

1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 19CK + 4.1ms to ensure programming mode can be entered.
2. The device is shipped with this option selected.

By changing the OSCCAL register from SW, it is possible to get a higher calibration accuracy than by using the factory calibration.

When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-Out. For more information on the pre-programmed calibration value.

**Related Links**

Clock Characteristics on page 397

## 14.6. 128kHz Internal Oscillator

The 128kHz internal Oscillator is a low power Oscillator providing a clock of 128kHz. The frequency is nominal at 3V and 25°C. This clock may be select as the system clock by programming the CKSEL Fuses to '0011':

**Table 14-11. 128kHz Internal Oscillator Operating Modes**

| Nominal Frequency(1) | CKSEL[3:0] |
|---|---|
| 128kHz | 0011 |

**Note:**

1. The 128kHz oscillator is a very low power clock source, and is not designed for high accuracy.

When this clock source is selected, start-up times are determined by the SUT Fuses:

**Table 14-12. Start-Up Times for the 128kHz Internal Oscillator**

| Power Conditions | Start-Up Time from Power-down and Power-save | Additional Delay from Reset | SUT[1:0] |
|---|---|---|---|
| BOD enabled | 6 CK | 19CK(1) | 00 |
| Fast rising power | 6 CK | 19CK + 4ms | 01 |
| Slowly rising power | 6 CK | 19CK + 64ms | 10 |
| Reserved | | | 11 |

**Note:**

| Address | Labels | | Code Comments |
|---|---|---|---|
| 0x000 | rjmp | RESET | ; Reset Handler |
| 0x001 | rjmp | EXT_INT0 | ; IRQ0 Handler |
| 0x002 | rjmp | EXT_INT1 | ; IRQ1 Handler |
| 0x003 | rjmp | PCINT0 | ; PCINT0 Handler |
| 0x004 | rjmp | PCINT1 | ; PCINT1 Handler |
| 0x005 | rjmp | PCINT2 | ; PCINT2 Handler |
| 0x006 | rjmp | WDT | ; Watchdog Timer Handler |
| 0x007 | rjmp | TIM2_COMPA | ; Timer2 Compare A Handler |
| 0X008 | rjmp | TIM2_COMPB | ; Timer2 Compare B Handler |
| 0x009 | rjmp | TIM2_OVF | ; Timer2 Overflow Handler |
| 0x00A | rjmp | TIM1_CAPT | ; Timer1 Capture Handler |
| 0x00B | rjmp | TIM1_COMPA | ; Timer1 Compare A Handler |
| 0x00C | rjmp | TIM1_COMPB | ; Timer1 Compare B Handler |
| 0x00D | rjmp | TIM1_OVF | ; Timer1 Overflow Handler |
| 0x00E | rjmp | TIM0_COMPA | ; Timer0 Compare A Handler |
| 0x00F | rjmp | TIM0_COMPB | ; Timer0 Compare B Handler |
| 0x010 | rjmp | TIM0_OVF | ; Timer0 Overflow Handler |
| 0x011 | rjmp | SPI_STC | ; SPI Transfer Complete Handler |
| 0x012 | rjmp | USART_RXC | ; USART, RX Complete Handler |
| 0x013 | rjmp | USART_UDRE | ; USART, UDR Empty Handler |
| 0x014 | rjmp | USART_TXC | ; USART, TX Complete Handler |
| 0x015 | rjmp | ADC | ; ADC Conversion Complete Handler |

### 18.2.3. External Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

**Name:** EIFR
**Offset:** 0x3C
**Reset:** 0x00
**Property:** When addressing as I/O Register: address offset is 0x1C

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | INTF1 | INTF0 |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 will be set. If the I-bit in SREG and the INT1 bit in EIMSK are set, 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 '1' to it. This flag is always cleared when INT1 is configured as a level interrupt.

**Bit 0 – INTF0: External Interrupt Flag 0**

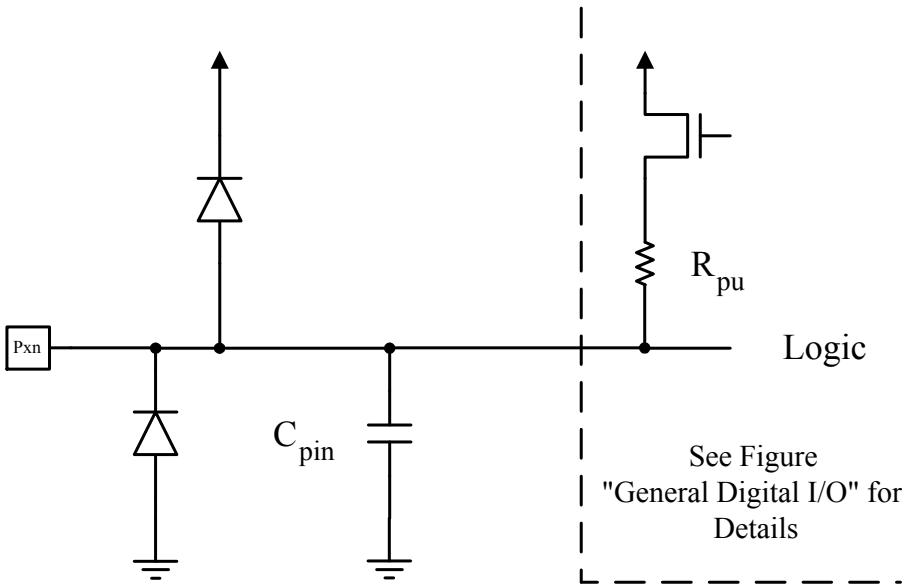
When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 will be set. If the I-bit in SREG and the INT0 bit in EIMSK are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it. This flag is always cleared when INT0 is configured as a level interrupt.

# 19. I/O-Ports

## 19.1. Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both $V_{CC}$ and Ground as indicated in the following figure.

**Figure 19-1. I/O Pin Equivalent Schematic**




All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing '1' to a bit in the PINx Register will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer to the individual module sections for a full description of the alternate functions.

Enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is down counting.

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

Be aware that the COMnx[1:0] bits are not double buffered together with the compare value. Changing the COMnx[1:0] bits will take effect immediately.

## 20.6. Compare Match Output Unit

The Compare Output mode bits in the Timer/Counter Control Register A (TCCR1A.COM1x) have two functions:

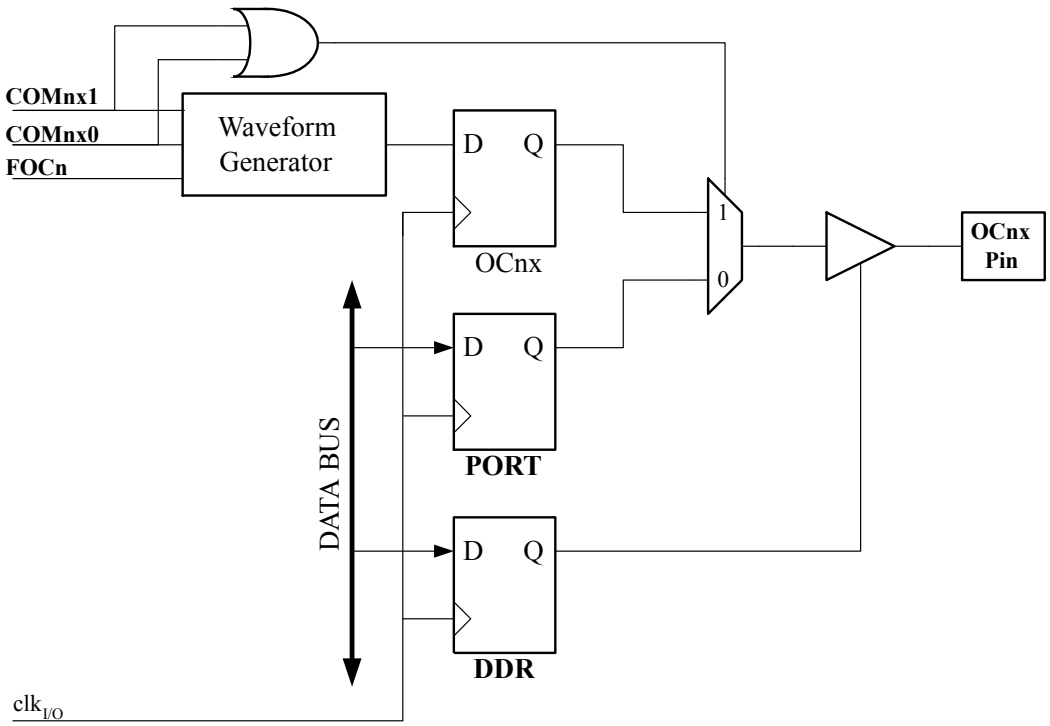
- The Waveform Generator uses the COM1x bits for defining the Output Compare (OC1x) register state at the next compare match.
- The COM1x bits control the OC1x pin output source

The figure below shows a simplified schematic of the logic affected by COM1x. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers that are affected by the COM1x bits are shown, namely PORT and DDR.

On system reset the OC1x Register is reset to 0x00.

**Note:** 'OC1x state' is always referring to internal OC1x *registers*, not the OC1x *pin*.

**Figure 20-4. Compare Match Output Unit, Schematic**




**Note:** The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled

1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM0A[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

**Table 20-5. Compare Output Mode, Phase Correct PWM Mode(1)**

| COM0A1 | COM0A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A disconnected. |
| 0 | 1 | WGM02 = 0: Normal Port Operation, OC0A Disconnected.<br>WGM02 = 1: Toggle OC0A on Compare Match. |
| 1 | 0 | Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting. |
| 1 | 1 | Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting. |

**Note:**

1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

**Bits 5:4 – COM0Bn: Compare Output Mode for Channel B [n = 1:0]**

These bits control the Output Compare pin (OC0B) behavior. If one or both of the COM0B[1:0] bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B[1:0] bits depends on the WGM0[2:0] bit setting. The table shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to a normal or CTC mode (non- PWM).

**Table 20-6. Compare Output Mode, non-PWM**

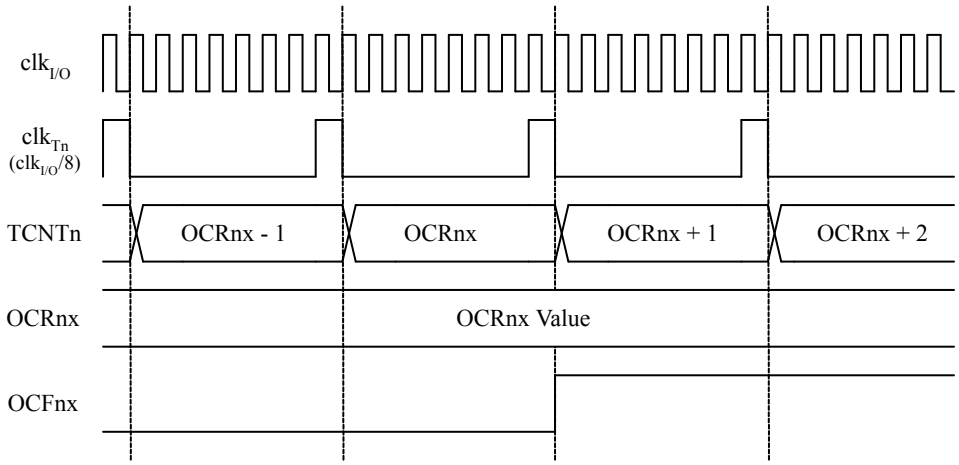
| COM0B1 | COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0B disconnected. |
| 0 | 1 | Toggle OC0B on Compare Match. |
| 1 | 0 | Clear OC0B on Compare Match. |
| 1 | 1 | Set OC0B on Compare Match. |

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to fast PWM mode.

**Table 20-7. Compare Output Mode, Fast PWM(1)**

| COM0B1 | COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0B disconnected. |
| 0 | 1 | Reserved |

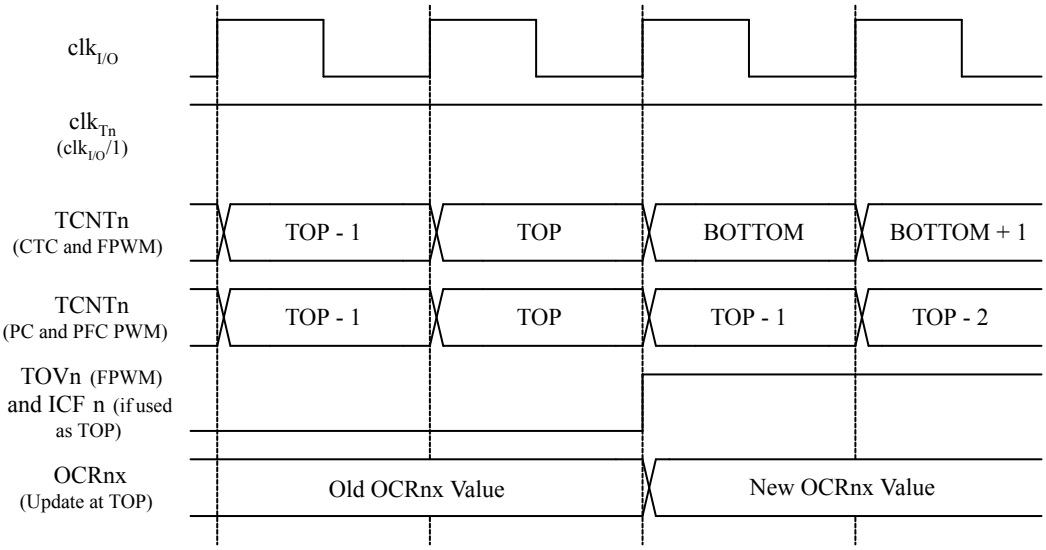
**Figure 21-11. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)**




**Note:** The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B).

The next figure shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

**Figure 21-12. Timer/Counter Timing Diagram, no Prescaling.**




**Note:** The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

### 21.12.5. TC1 Counter High byte

**Name:** TCNT1H
**Offset:** 0x85
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCNT1H[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TCNT1H[7:0]: Timer/Counter 1 High byte**
Refer to TCNT1L.

### 21.12.13. TC1 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

**Name:** TIFR1
**Offset:** 0x36
**Reset:** 0x00
**Property:** When addressing as I/O Register: address offset is 0x16

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | ICF1 | | | OCF1B | OCF1A | TOV1 |
| Access | | | R/W | | | R/W | R/W | R/W |
| Reset | | | 0 | | | 0 | 0 | 0 |

**Bit 5 – ICF1: Input Capture Flag**
This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM1[3:0] to be used as the TOP value, the ICF Flag is set when the counter reaches the TOP value.

ICF is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF can be cleared by writing a logic one to its bit location.

**Bit 2 – OCF1B: Output Compare B Match Flag**
This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a Forced Output Compare (FOCB) strobe will not set the OCFB Flag.

OCFB is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCFB can be cleared by writing a logic one to its bit location.

**Bit 1 – OCF1A: Output Compare A Match Flag**
This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOCA) strobe will not set the OCFA Flag.

OCFA is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCFA can be cleared by writing a logic one to its bit location.

**Bit 0 – TOV1: Overflow Flag**
The setting of this flag is dependent of the WGM1[3:0] bits setting. In Normal and CTC modes, the TOV Flag is set when the timer overflows. Refer to the Waveform Generation Mode bit description for the TOV Flag behavior when using another WGM1[3:0] bit setting.

TOV is automatically cleared when the Timer/Counter 1 Overflow Interrupt Vector is executed. Alternatively, TOV can be cleared by writing a logic one to its bit location.

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.

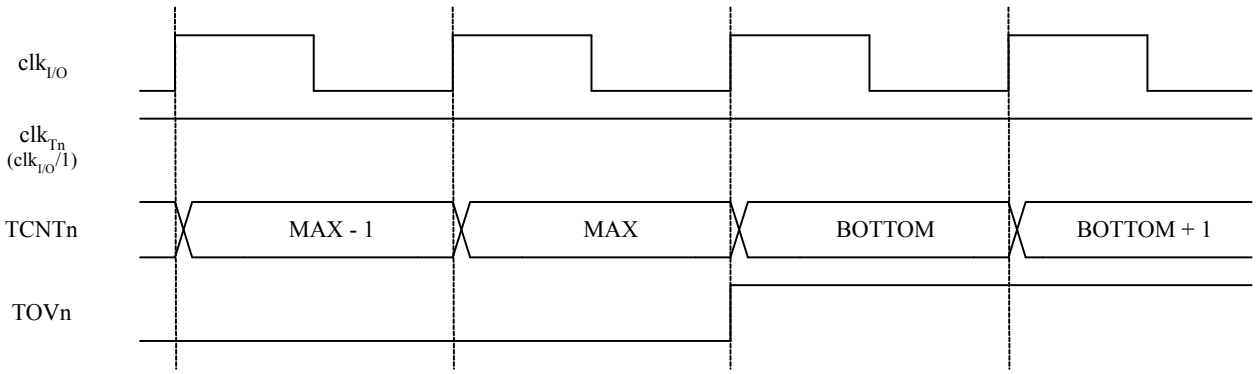
At the very start of period 2 in the above figure OC2x has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR2A changes its value from MAX, as shown in the preceeding figure. When the OCR2A value is MAX the OC2 pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OC2 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 OC2 change that would have happened on the way up.

## 23.8. Timer/Counter Timing Diagrams

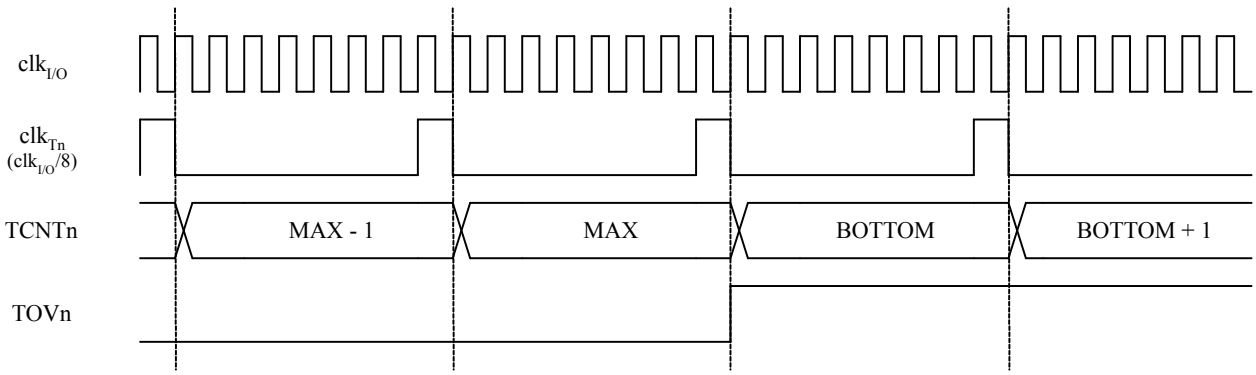
The following figures show the Timer/Counter in synchronous mode, and the timer clock ($clk_{T2}$) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. The following figure 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 23-8. Timer/Counter Timing Diagram, no Prescaling**



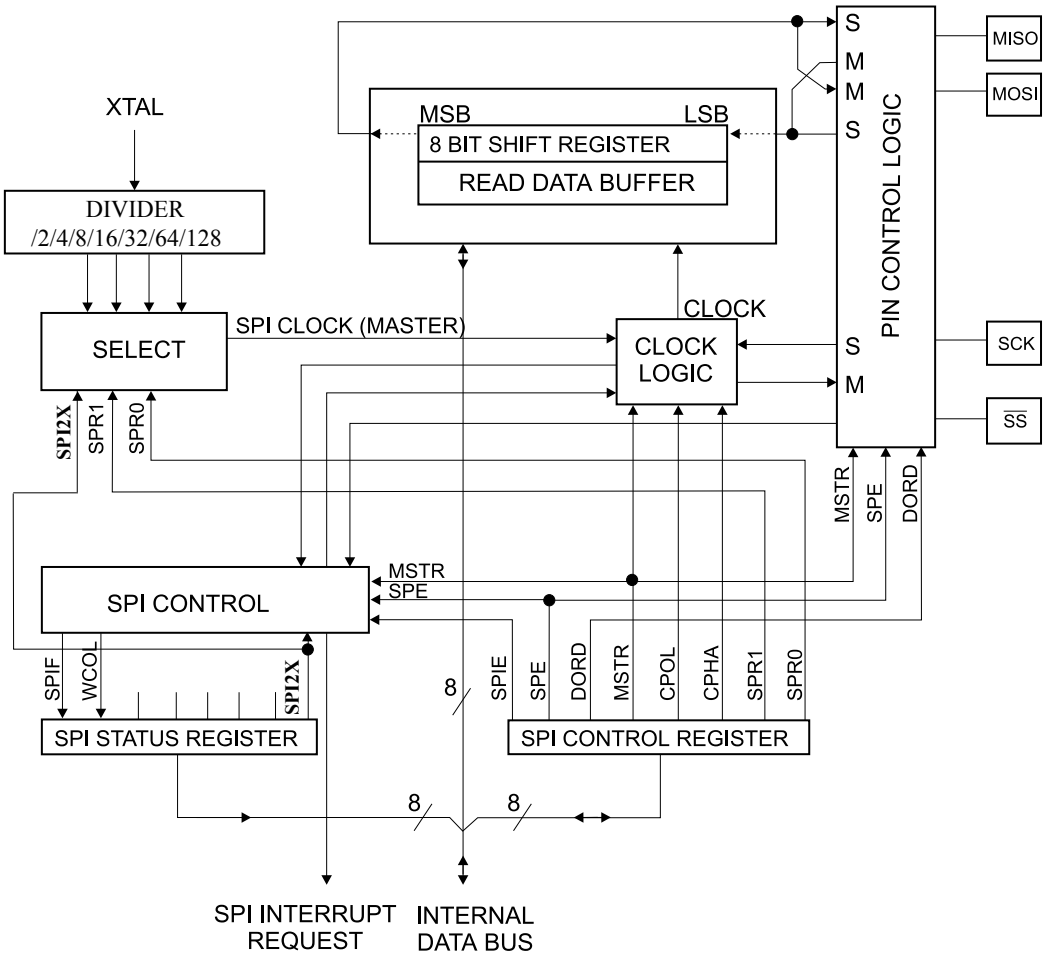

The following figure shows the same timing data, but with the prescaler enabled.

**Figure 23-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}$/8)**




The following figure shows the setting of OCF2A in all modes except CTC mode.

**Figure 24-1. SPI Block Diagram**



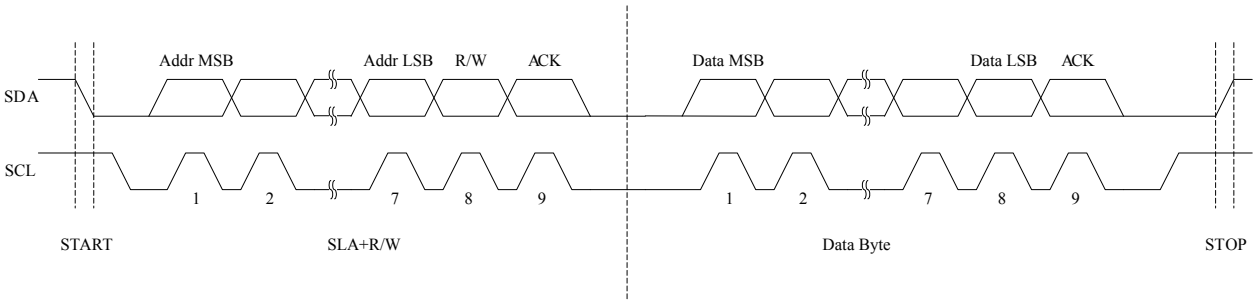

**Note:** Refer to the pinout description and the IO Port description for SPI pin placement.

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

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new

**Figure 27-6. Typical Data Transmission**




## 27.4. 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.

3. The application software should now examine the value of TWSRn, to make sure that the START condition was successfully transmitted. If TWSRn 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 TWDRn is used both for address and data. After TWDRn has been loaded with the desired SLA+W, a specific value must be written to TWCRn, instructing the TWI n hardware to transmit the SLA+W present in TWDRn. 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 TWCRn 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 TWCRn is set, and TWSRn 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 TWSRn, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSRn 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 TWDRn. Subsequently, a specific value must be written to TWCRn, instructing the TWI n hardware to transmit the data packet present in TWDRn. 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 n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
6. When the data packet has been transmitted, the TWINT Flag in TWCRn is set, and TWSRn is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSRn, to make sure that the data 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 write a specific value to TWCRn, instructing the TWI n hardware to transmit a STOP 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 n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is *not* set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI n Registers with the value relevant for the next TWI n bus cycle. As an example, TWDRn must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI n Register updates and other pending application software tasks have been completed, TWCRn is written. When writing TWCRn, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI n will then commence executing whatever operation was specified by the TWCRn setting.

The following table lists assembly and C implementation examples for TWI0. Note that the code below assumes that several definitions have been made, e.g. by using include-files.

start executing the application code. The fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

**Table 32-4. Boot Reset Fuse**

| BOOTRST | Reset Address |
|---|---|
| 1 | Reset Vector = Application Reset (address 0x0000) |
| 0 | Reset Vector = Boot Loader Reset, as described by the Boot Loader Parameters |

**Note:** '1' means unprogrammed, '0' means programmed.

## 32.7. Addressing the Flash During Self-Programming

The Z-pointer is used to address the SPM commands.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ZH (R31) | Z15 | Z14 | Z13 | Z12 | Z11 | Z10 | Z9 | Z8 |
| ZL (R30) | Z7 | Z6 | Z5 | Z4 | Z3 | Z2 | Z1 | Z0 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Since the Flash is organized in pages, the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in the following figure. The Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

**Figure 35-9. ATmega48PB/88PB: Idle Supply Current vs. $V_{CC}$ (Internal RC Oscillator, 1MHz)**

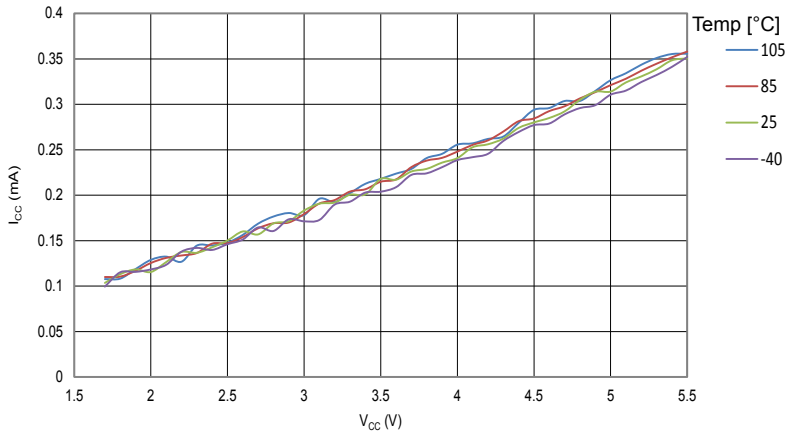
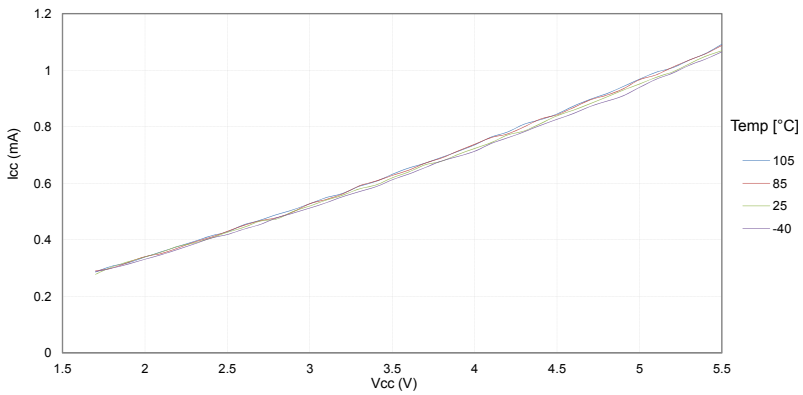



**Figure 35-10. ATmega48PB/88PB: Idle Supply Current vs. $V_{CC}$ (Internal RC Oscillator, 8MHz)**




### 35.1.3. ATmega48PB/88PB Supply Current of IO Modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the Power Reduction Register. See ”Power Reduction Register” on page 43 for details.

**Table 35-1. ATmega48PB/88PB: Additional Current Consumption for the different I/O modules (absolute values)**

| PRR bit | Typical numbers @ 25°C | | |
|---|---|---|---|
| | $V_{CC}$ = 2V, F = 1MHz | $V_{CC}$ = 3V, F = 4MHz | $V_{CC}$ = 5V, F = 8MHz |
| PRUSART0 | 4.66µA | 28.73µA | 103.38µA |
| PRTWI | 6.63µA | 41.89µA | 148.00µA |
| PRTIM2 | 6.64µA | 37.74µA | 137.36µA |
| PRTIM1 | 4.36µA | 29.65µA | 112.13µA |
| PRTIM0 | 1.61µA | 9.59µA | 32.13µA |
| PRSPI | 5.55µA | 37.15µA | 136.38µA |
| PRADC | 7.01µA | 43.31µA | 158.38µA |

**Figure 35-50. ATmega168PB: Active Supply Current vs. Frequency (1-20MHz)**

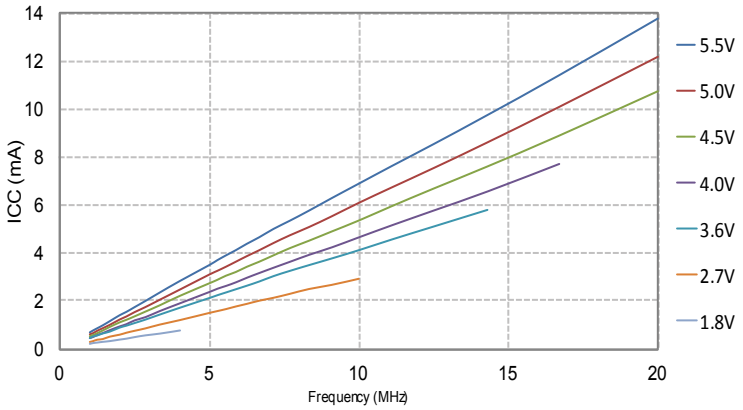



**Figure 35-51. ATmega168PB: Active Supply Current vs. $V_{CC}$ (Internal RC Oscillator, 128kHz)**

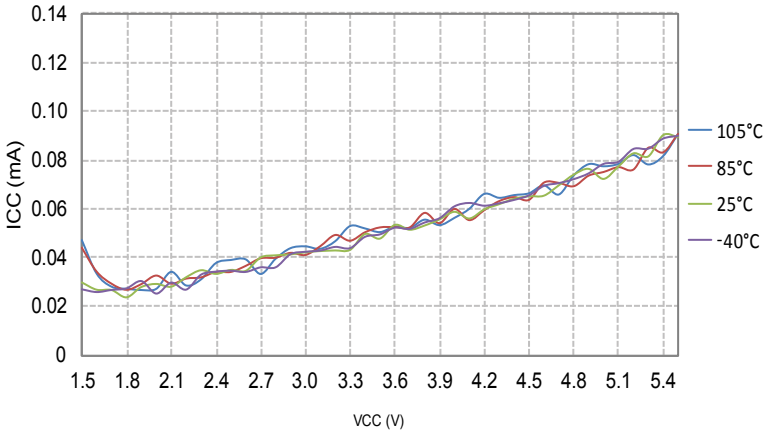



**Figure 35-52. ATmega168PB: Active Supply Current vs. $V_{CC}$ (Internal RC Oscillator, 1MHz)**

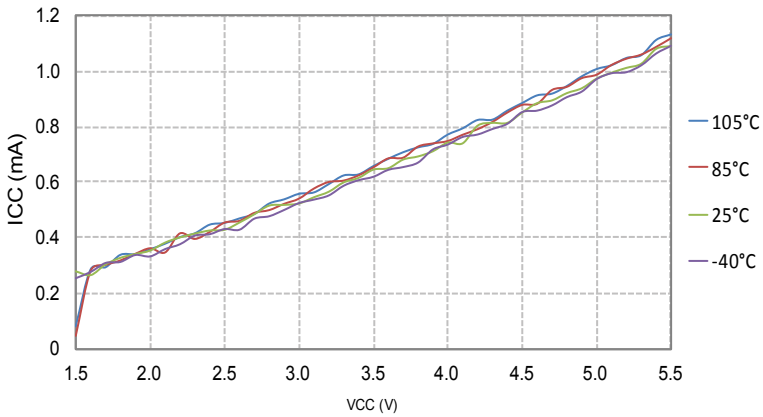
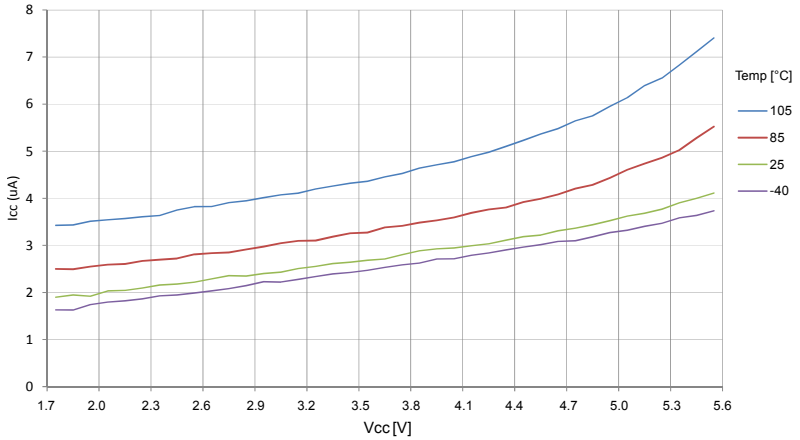
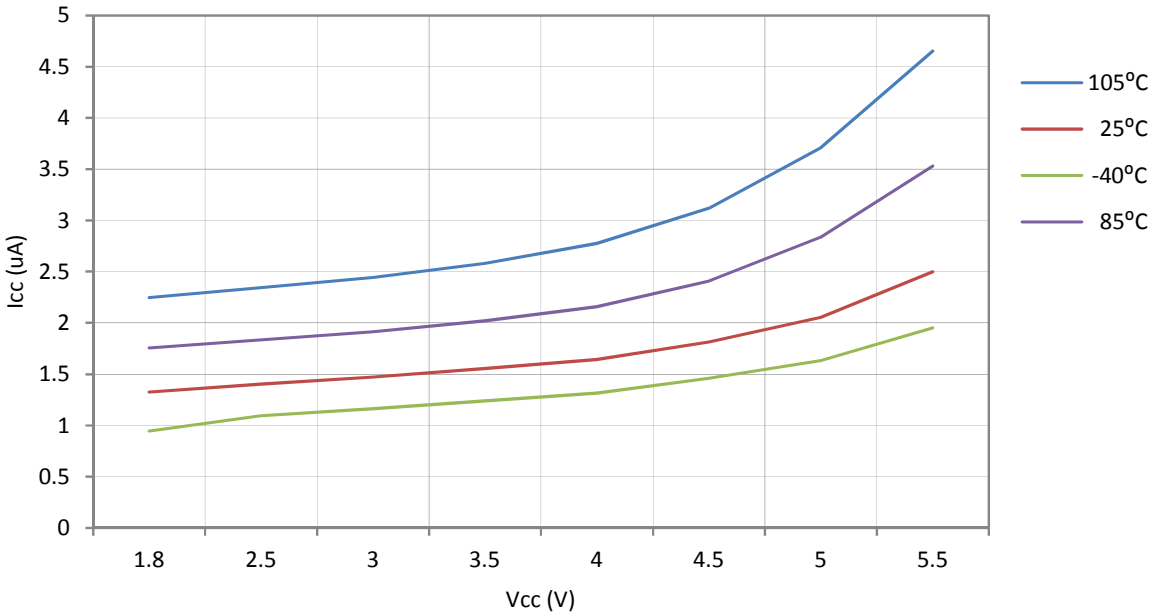


**Figure 35-60. ATmega168PB: Power-Down Supply Current vs. $V_{CC}$ (Watchdog Timer Enabled)**




## 35.2.5. Power-save Supply Current

**Figure 35-61. ATmega168PB: Power-Save Supply Current vs. $V_{CC}$**



| | |
|---|---|
| 13. | Updated ”ATmega48PB/88PB Typical Characteristics” on page 316.<br>• Added ”Power-save Supply Current” on page 323.<br>• Added ”Power-standby Supply Current” on page 323. |
| 14. | Updated the typical values in ”ATmega168PB DC Characteristics” on page 305. |
| 15. | Updated ”ATmega168PB Typical Characteristics” on page 341.<br>• Added ”Power-save Supply Current” on page 348.<br>• Added ”Power-standby Supply Current” on page 348. |

## 40.4. Rev. 42176D – 04/2015

| | |
|---|---|
| 1. | Added ”ATmega48PB/88PB DC Characteristics” on page 303. |
| 2. | Added ”ATmega48PB/88PB Typical Characteristics” on page 315. |
| 3. | Updated numbers in ”ATmega168PB DC Characteristics” on page 303 |
| 4. | Updated numbers in ”ATmega168PB Supply Current of IO Modules” on page 344. |

## 40.5. Rev. 42176C – 03/2015

| | |
|---|---|
| 1. | “Clock Characteristics” :<br>Updated factory calibration accuracy from ±10% to ±3% in ”Calibrated Internal RC Oscillator Accuracy” on page 305. |
| 2. | “Errata” :<br>Updated ”Errata ATmega48PB” on page 375, ”Errata ATmega88PB” on page 376 and ”Errata ATmega168PB” on page 377. |

## 40.6. Rev. 42176B – 11/2014

| | |
|---|---|
| 1. | Additional Delay from Reset ($V_{CC}$=5V) updated from 14CK to 19CK in the following sections / tables:<br>• “Low Power Crystal Oscillator” / Table 10-4 on page 29.<br>• “Low Frequency Crystal Oscillator” / Table 10-6 on page 30.<br>• “Low Frequency Crystal Oscillator” / Table 10-9 on page 31.<br>• “Calibrated Internal RC Oscillator” / Table 10-12 on page 32.<br>• “128kHz Internal Oscillator” / Table 10-14 on page 33.<br>• “External Clock” / Table 10-16 on page 34. |

## 40.7. Rev. 42176A - 11/2014

| | |
|---|---|
| 1. | Initial release. |