



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	16MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	54
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega165-16ai">https://www.e-xfl.com/product-detail/microchip-technology/atmega165-16ai</a>

## • Bit 1 – EEW: EEPROM Write Enable

The EEPROM Write Enable Signal EEW is the write strobe to the EEPROM. When address and data are correctly set up, the EEW bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEW, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEW becomes zero.
2. Wait until SPMEN in SPMCSR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEW in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEW.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See “Boot Loader Support – Read-While-Write Self-Programming” on page 232 for details about Boot programming.

**Caution:** An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEW bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEW has been set, the CPU is halted for two cycles before the next instruction is executed.

## • Bit 0 – EERE: EEPROM Read Enable

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEW bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

**Table 1.** EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ Programming Time
EEPROM write (from CPU)	67 584	8.5 ms

Oscillator is stopped during sleep. If the Timer/Counter2 is using the synchronous clock, the clock source is stopped during sleep. Note that even if the synchronous clock is running in Power-save, this clock is only available for the Timer/Counter2.

## Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

**Table 15.** Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators		Wake-up Sources					
	clk <sub>CPU</sub>	clk <sub>FLASH</sub>	clk <sub>IO</sub>	clk <sub>ADC</sub>	clk <sub>ASY</sub>	Main Clock Source Enabled	Timer Osc Enabled	INT0 and Pin Change	USI Start Condition	Timer2	SPM/EEPROM Ready	ADC	Other I/O
Idle			X	X	X	X	X <sup>(2)</sup>	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X <sup>(2)</sup>	X <sup>(3)</sup>	X	X <sup>(2)</sup>	X	X	
Power-down								X <sup>(3)</sup>	X				
Power-save					X		X	X <sup>(3)</sup>	X	X			
Standby <sup>(1)</sup>						X		X <sup>(3)</sup>	X				

Notes: 1. Only recommended with external crystal or resonator selected as clock source.  
2. If Timer/Counter2 is not running in asynchronous mode.  
3. For INT0, only level interrupt.

## Power Reduction Register

The Power Reduction Register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See “Supply Current of I/O modules” on page 290 for examples. In all other sleep modes, the clock is already stopped.

### Power Reduction Register - PRR

Bit	7	6	5	4	3	2	1	0	
	—	—	—	—	PRTIM1	PRSPI	PRUSART0	PRADC	PRR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

#### • Bit 7..4 - Res: Reserved bits

These bits are reserved in ATmega165 and will always read as zero.

#### • Bit 3 - PRTIM1: Power Reduction Timer/Counter1

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

## • Bit 3 – WDE: Watchdog Enable

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

In safety level 2, it is not possible to disable the Watchdog Timer, even with the algorithm described above. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 45.

## • Bits 2..0 – WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 21.

**Table 21.** Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at $V_{CC} = 3.0V$	Typical Time-out at $V_{CC} = 5.0V$
0	0	0	16K cycles	15.4 ms	14.7 ms
0	0	1	32K cycles	30.8 ms	29.3 ms
0	1	0	64K cycles	61.6 ms	58.7 ms
0	1	1	128K cycles	0.12 s	0.12 s
1	0	0	256K cycles	0.25 s	0.23 s
1	0	1	512K cycles	0.49 s	0.47 s
1	1	0	1,024K cycles	1.0 s	0.9 s
1	1	1	2,048K cycles	2.0 s	1.9 s

Note: Also see Figure 183 on page 312.

## External Interrupt Mask Register – EIMSK

Bit	7	6	5	4	3	2	1	0	
	PCIE1	PCIE0	–	–	–	–	–	INT0	EIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 7 – 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 PCINT15..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCIE1 Interrupt Vector. PCINT15..8 pins are enabled individually by the PCMSK1 Register.

### • Bit 6 – 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 PCIE0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

### • Bit 0 – INT0: External Interrupt Request 0 Enable

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

## External Interrupt Flag Register – EIFR

Bit	7	6	5	4	3	2	1	0	
	PCIF1	PCIF0	–	–	–	–	–	INTF0	EIFR
Read/Write	R/W	R/W	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 7 – PCIF1: Pin Change Interrupt Flag 1

When a logic change on any PCINT15..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in EIMSK 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 6 – PCIF0: Pin Change Interrupt Flag 0

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in EIMSK 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.

Table 28 and Table 29 relate the alternate functions of Port B to the overriding signals shown in Figure 26 on page 60. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

**Table 28.** Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/OC2A/ PCINT15	PB6/OC1B/ PCINT14	PB5/OC1A/ PCINT13	PB4/OC0A/ PCINT12
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	OC0A ENABLE
PVOV	OC2A	OC1B	OC1A	OC0A
PTOE	–	–	–	–
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1
DI	PCINT15 INPUT	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	–	–	–	–

**Table 29.** Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/MISO/ PCINT11	PB2/MOSI/ PCINT10	PB1/SCK/ PCINT9	PB0/ $\overline{SS}$ / PCINT8
PUOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
PUOV	PORTB3 • $\overline{PUD}$	PORTB2 • $\overline{PUD}$	PORTB1 • $\overline{PUD}$	PORTB0 • $\overline{PUD}$
DDOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	SPE • MSTR
DDOV	0	0	0	0
PVOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • MSTR	0
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	0
PTOE	–	–	–	–
DIEOE	PCINT11 • PCIE1	PCINT10 • PCIE1	PCINT9 • PCIE1	PCINT8 • PCIE1
DIEOV	1	1	1	1
DI	PCINT11 INPUT SPI MSTR INPUT	PCINT10 INPUT SPI SLAVE INPUT	PCINT9 INPUT SCK INPUT	PCINT8 INPUT SPI $\overline{SS}$
AIO	–	–	–	–

## Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 48 on page 113. For fast PWM mode refer to Table 49 on page 113, and for phase correct and phase and frequency correct PWM refer to Table 50 on page 114.

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

## Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (See “Compare Match Output Unit” on page 102.)

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 111.

## Normal Mode

The simplest mode of operation is the *Normal mode* (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

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

ment) 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 ( $clk_{T2}$ ).

The double buffered Output Compare Register (OCR2A) is 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 pin (OC2A). See “Output Compare Unit” on page 122. for details. The compare match event will also set the Compare Flag (OCF2A) which can be used to generate an Output Compare interrupt request.

## 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 53 are also used extensively throughout the section.

**Table 53.** 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.

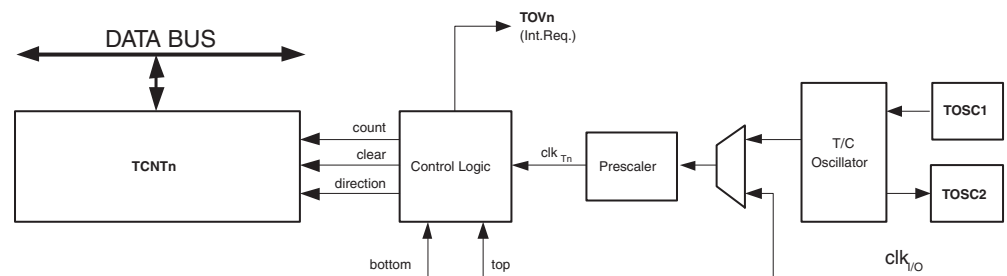
## Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source  $clk_{T2}$  is by default equal to the MCU clock,  $clk_{I/O}$ . 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 134. For details on clock sources and prescaler, see “Timer/Counter Prescaler” on page 138.

## Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 54 shows a block diagram of the counter and its surrounding environment.

**Figure 54.** Counter Unit Block Diagram



Signal description (internal signals):

- count** Increment or decrement TCNT2 by 1.
- direction** Selects between increment and decrement.
- clear** Clear TCNT2 (set all bits to zero).
- $clk_{T2}$**  Timer/Counter clock.



## SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	<b>SPIF</b>	<b>WCOL</b>	–	–	–	–	–	<b>SPI2X</b>	<b>SPSR</b>
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 7 – SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If  $\overline{SS}$  is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

### • Bit 6 – WCOL: Write COLLision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

### • Bit 5..1 – Res: Reserved Bits

These bits are reserved bits in the ATmega165 and will always read as zero.

### • Bit 0 – SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 62). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at  $f_{osc}/4$  or lower.

The SPI interface on the ATmega165 is also used for program memory and EEPROM downloading or uploading. See page 261 for serial programming and verification.

## SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	<b>MSB</b>							<b>LSB</b>	<b>SPDR</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

## Receive Complete Flag and Interrupt

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

The Receive Complete (RXC) 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 (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete interrupt will be executed as long as the RXC 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 UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

## Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (UPE). All can be accessed by reading UCSRA. 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 UCSRA must be read before the receive buffer (UDR), since reading the UDR 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 UCSRA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read (as one), and the FE 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 FE Flag is not affected by the setting of the USBF bit in UCSRC 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 UCSRA.

The Data OverRun (DOR) 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 DOR Flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPE) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see "Parity Bit Calculation" on page 153 and "Parity Checker" on page 161.

CLK or USITC strobe bits. The clock source depends of the setting of the USICS1..0 bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) are can still be used by the counter.

## USI Control Register – USICR

Bit	7	6	5	4	3	2	1	0	
	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	USICR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The Control Register includes interrupt enable control, wire mode setting, Clock Select setting, and clock strobe.

- **Bit 7 – USISIE: Start Condition Interrupt Enable**

Setting this bit to one enables the Start Condition detector interrupt. If there is a pending interrupt when the USISIE and the Global Interrupt Enable Flag is set to one, this will immediately be executed. Refer to the USISIF bit description on page 182 for further details.

- **Bit 6 – USIOIE: Counter Overflow Interrupt Enable**

Setting this bit to one enables the Counter Overflow interrupt. If there is a pending interrupt when the USIOIE and the Global Interrupt Enable Flag is set to one, this will immediately be executed. Refer to the USIOIF bit description on page 182 for further details.

- **Bit 5..4 – USIWM1..0: Wire Mode**

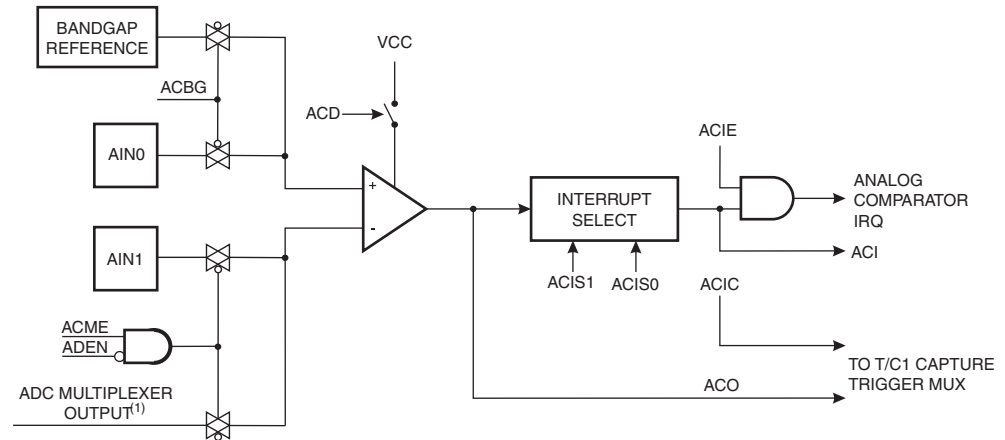
These bits set the type of wire mode to be used. Basically only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected and will always have the same function. The counter and Shift Register can therefore be clocked externally, and data input sampled, even when outputs are disabled. The relations between USIWM1..0 and the USI operation is summarized in Table 76.

## Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 82.

The Power Reduction ADC bit, PRADC, in "Power Reduction Register - PRR" on page 34 must be disabled by writing a logical zero to be able to use the ADC input MUX.

**Figure 82.** Analog Comparator Block Diagram<sup>(2)</sup>



Notes:1. See Table 79 on page 188.

2. Refer to Figure 1 on page 2 and Table 32 on page 66 for Analog Comparator pin placement.

### ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

#### • Bit 6 – ACME: Analog Comparator Multiplexer Enable

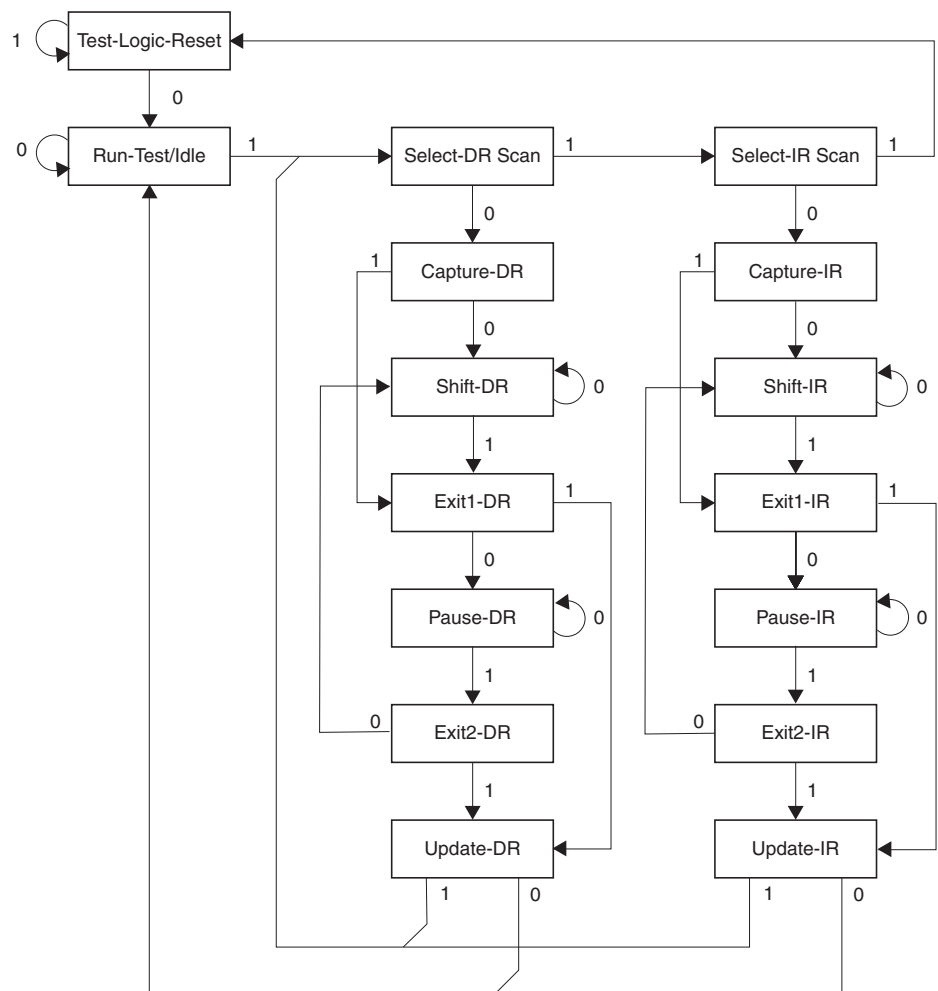
When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see "Analog Comparator Multiplexed Input" on page 188.

### Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

#### • Bit 7 – ACD: Analog Comparator Disable

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power

**Figure 98. TAP Controller State Diagram**


## TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or On-chip Debug system. The state transitions depicted in Figure 98 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-on Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR

## Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections.

See “Boundary-scan Chain” on page 216 for a complete description.

## Boundary-scan Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR\_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

### EXTEST; 0x0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-Register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

### IDCODE; 0x1

Optional JTAG instruction selecting the 32 bit ID-Register as Data Register. The ID-Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE Register is sampled into the Boundary-scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

### SAMPLE\_PRELOAD; 0x2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan chain is applied to the output latches. However, the output latches are not connected to the pins.

### AVR\_RESET; 0xC

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

## 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	
	–	–	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0						See Bit Description

### • Bit 4 – JTRF: JTAG Reset Flag

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR\_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

## Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connection.

### Scanning the Digital Port Pins

Figure 101 shows the Boundary-scan Cell for a bi-directional port pin with pull-up function. The cell consists of a standard Boundary-scan cell for the Pull-up Enable – PUExn – function, and a bi-directional pin cell that combines the three signals Output Control – OCxn, Output Data – ODxn, and Input Data – IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description

The Boundary-scan logic is not included in the figures in the datasheet. Figure 102 shows a simple digital port pin as described in the section “External Interrupts” on page 51. The Boundary-scan details from Figure 101 replaces the dashed box in Figure 102.

When no alternate port function is present, the Input Data – ID – corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction – DD Register, and the Pull-up Enable – PUExn – corresponds to logic expression  $\overline{PUD} \cdot \overline{DDxn} \cdot PORTxn$ .

Digital alternate port functions are connected outside the dotted box in Figure 102 to make the scan chain read the actual pin value. For Analog function, there is a direct connection from the external pin to the analog circuit, and a scan chain is inserted on the interface between the digital logic and the analog circuitry.

## Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits and general Lock bits, write the desired data to R0, write “X0001001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1

See Table 92 and Table 93 for how the different settings of the Boot Loader bits affect the Flash access.

If bits 5..2 in R0 are cleared (zero), the corresponding Boot Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMMEN are set in SPMCSR. The Z-pointer is don't care during this operation, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the Lock bits). For future compatibility it is also recommended to set bits 7 and 6 in R0 to “1” when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

## EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

## Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the BLBSET and SPMMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SPMMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMMEN bits will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SPMMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low byte is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the BLBSET and SPMMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the BLBSET and SPMMEN bits are set in the SPMCSR, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown below. Refer to Table 103 on page 248 for a detailed description and mapping of the Fuse Low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the Fuse High byte, load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SPMMEN bits are set in the SPMCSR, the value of the Fuse High byte (FHB) will be loaded in the destination register as shown below. Refer to Table 102 on page 248 for detailed description and mapping of the Fuse High byte.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

When reading the Extended Fuse byte, load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SPMMEN bits are set in



## Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 252 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give  $\overline{WR}$  a negative pulse and wait for RDY/ $\overline{BSY}$  to go high.

## Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 252 for details on Command and Data loading):

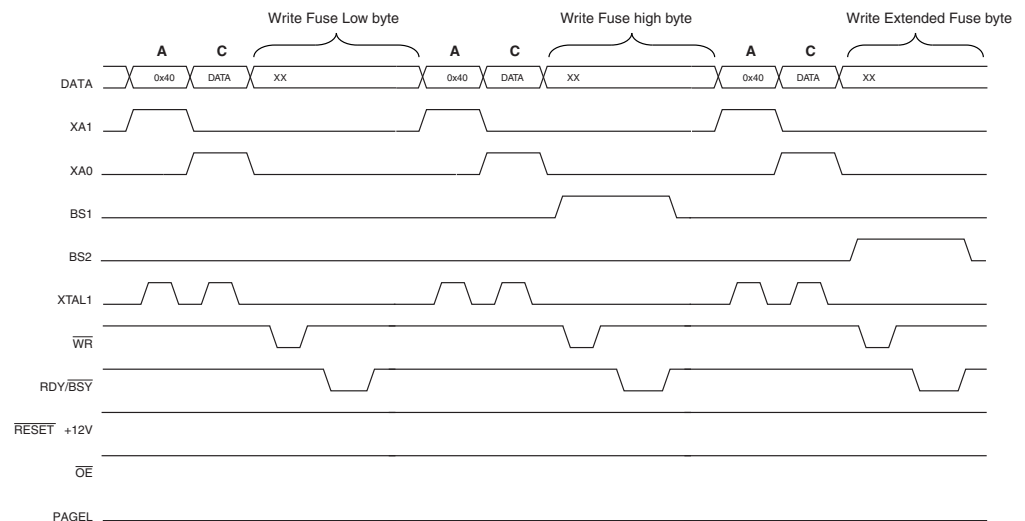
1. A: Load Command “0100 0000”.
2. C: Load Data Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high fuse byte.
4. Give  $\overline{WR}$  a negative pulse and wait for RDY/ $\overline{BSY}$  to go high.
5. Set BS1 to “0”. This selects low data byte.

## Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 252 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “0” and BS2 to “1”. This selects extended fuse byte.
4. Give  $\overline{WR}$  a negative pulse and wait for RDY/ $\overline{BSY}$  to go high.
5. Set BS2 to “0”. This selects low data byte.

**Figure 115. Programming the FUSES Waveforms**



## Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 252 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give  $\overline{WR}$  a negative pulse and wait for RDY/ $\overline{BSY}$  to go high.

The Lock bits can only be cleared by executing Chip Erase.

## ATmega165 Typical Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

All Active- and Idle current consumption measurements are done with all bits in the PRR register set and thus, the corresponding I/O modules are turned off. Also the Analog Comparator is disabled during these measurements. Table 120 and Table 121 on page 290 show the additional current consumption compared to  $I_{CC}$  Active and  $I_{CC}$  Idle for every I/O module controlled by the Power Reduction Register. See “Power Reduction Register” on page 34 for details.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

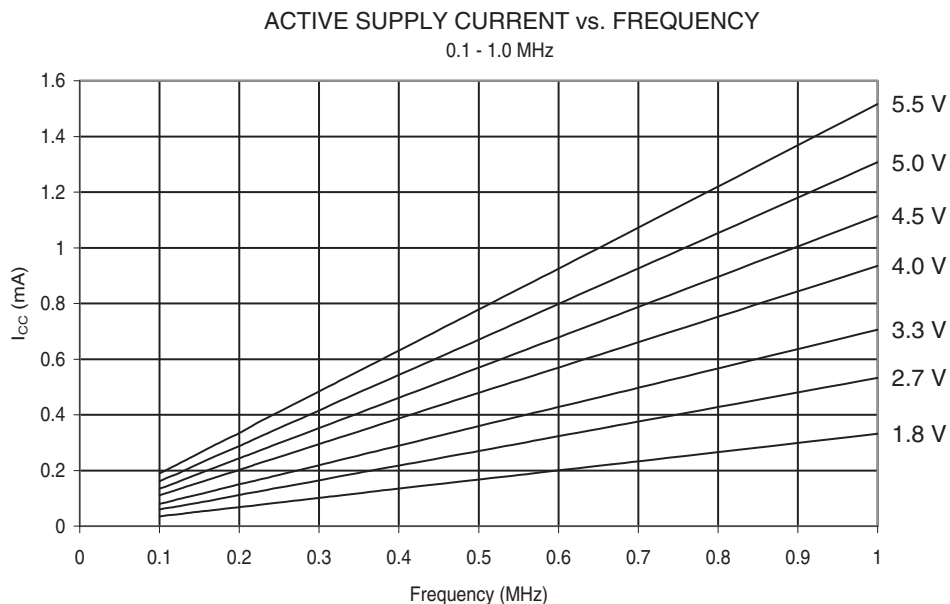
The current drawn from capacitive loaded pins may be estimated (for one pin) as  $C_L \cdot V_{CC} \cdot f$  where  $C_L$  = load capacitance,  $V_{CC}$  = operating voltage and  $f$  = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

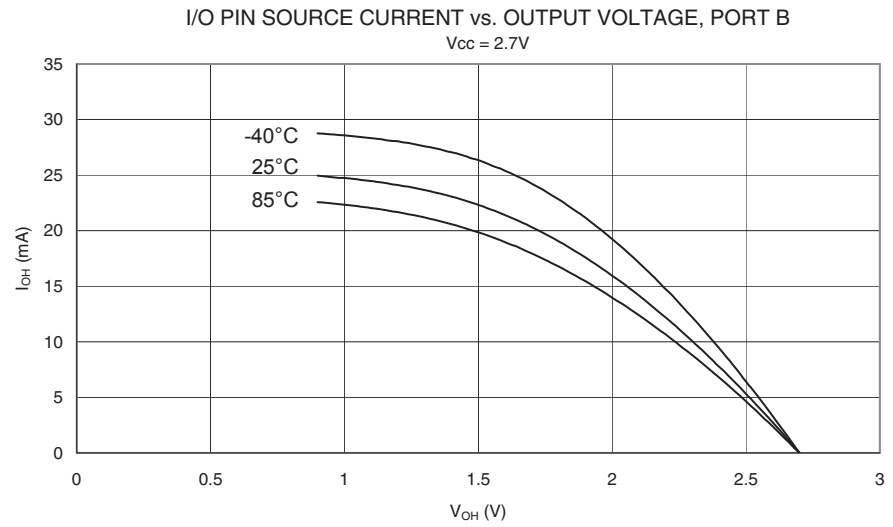
The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

## Active Supply Current

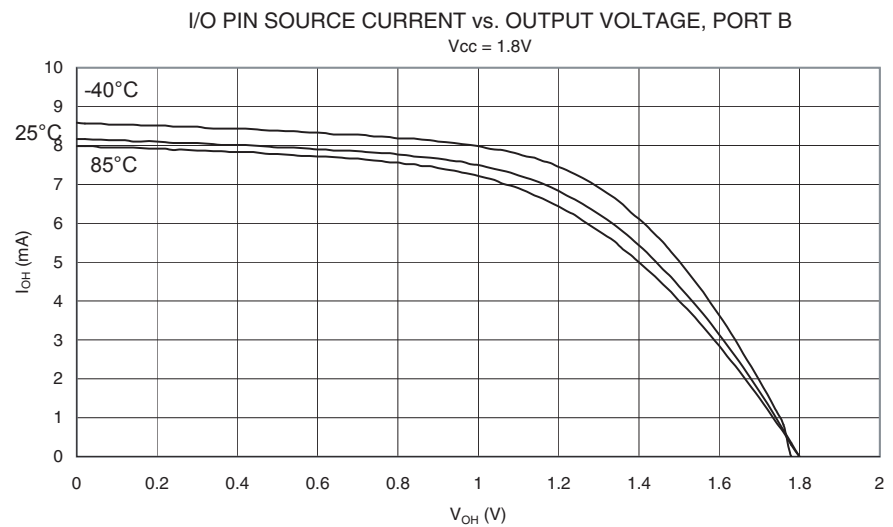
**Figure 132.** Active Supply Current vs. Frequency (0.1 - 1.0 MHz)



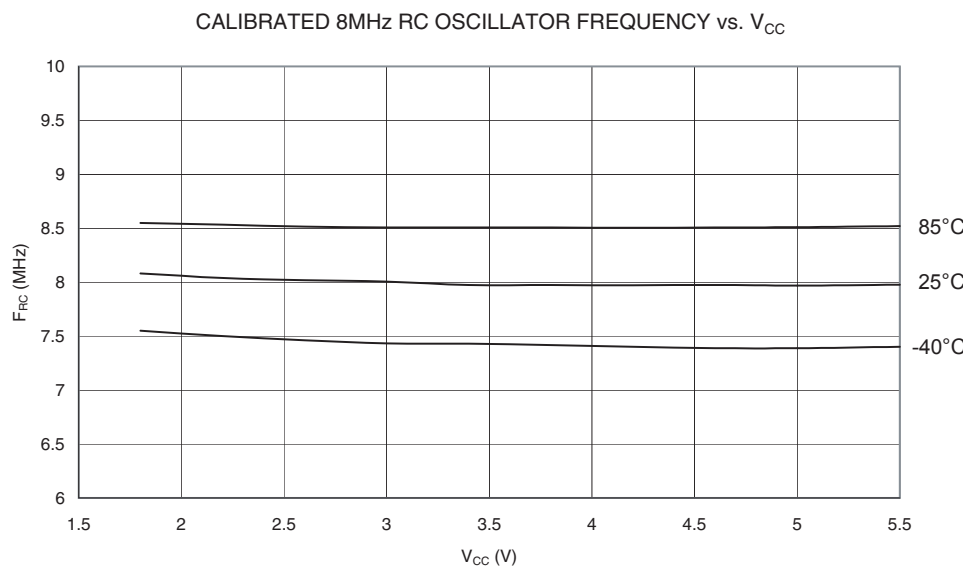
**Figure 163.** I/O Pin Source Current vs. Output Voltage, Port B ( $V_{CC} = 2.7V$ )



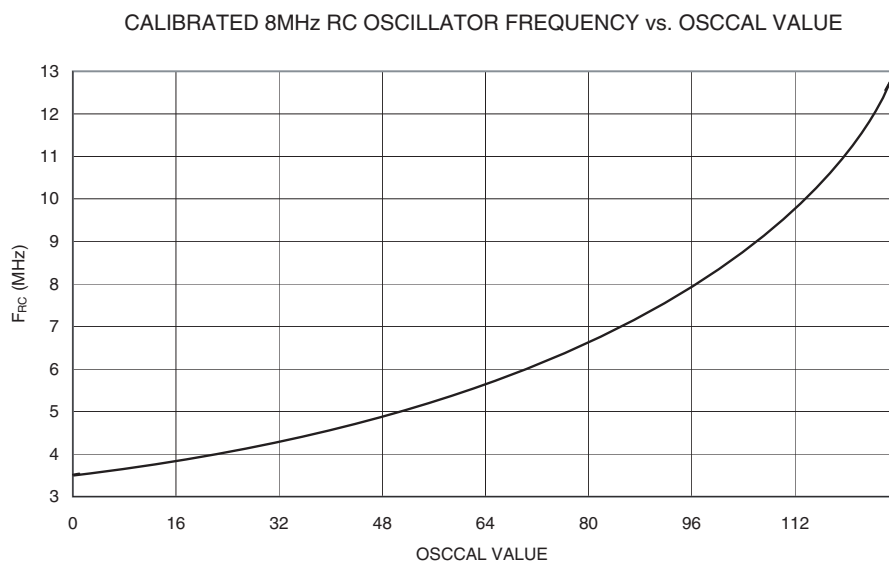
**Figure 164.** I/O Pin Source Current vs. Output Voltage, Port B ( $V_{CC} = 1.8V$ )



**Figure 185.** Calibrated 8 MHz RC Oscillator Frequency vs.  $V_{CC}$



**Figure 186.** Calibrated 8 MHz RC Oscillator Frequency vs. Oscal Value



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	—	—	—	—	—	—	—	—	
(0xBE)	Reserved	—	—	—	—	—	—	—	—	
(0xBD)	Reserved	—	—	—	—	—	—	—	—	
(0xBC)	Reserved	—	—	—	—	—	—	—	—	
(0xBB)	Reserved	—	—	—	—	—	—	—	—	
(0xBA)	USIDR	USI Data Register								181
(0xB9)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	182
(0xB8)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	183
(0xB7)	Reserved	—	—	—	—	—	—	—	—	
(0xB6)	ASSR	—	—	—	EXCLK	AS2	TCN2UB	OCR2UB	TCR2UB	134
(0xB5)	Reserved	—	—	—	—	—	—	—	—	
(0xB4)	Reserved	—	—	—	—	—	—	—	—	
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								133
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								133
(0xB1)	Reserved	—	—	—	—	—	—	—	—	
(0xB0)	TCCR2A	FOC2A	WGM20	COM2A1	COM2A0	WGM21	CS22	CS21	CS20	131
(0xAF)	Reserved	—	—	—	—	—	—	—	—	
(0xAE)	Reserved	—	—	—	—	—	—	—	—	
(0xAD)	Reserved	—	—	—	—	—	—	—	—	
(0xAC)	Reserved	—	—	—	—	—	—	—	—	
(0xAB)	Reserved	—	—	—	—	—	—	—	—	
(0xAA)	Reserved	—	—	—	—	—	—	—	—	
(0xA9)	Reserved	—	—	—	—	—	—	—	—	
(0xA8)	Reserved	—	—	—	—	—	—	—	—	
(0xA7)	Reserved	—	—	—	—	—	—	—	—	
(0xA6)	Reserved	—	—	—	—	—	—	—	—	
(0xA5)	Reserved	—	—	—	—	—	—	—	—	
(0xA4)	Reserved	—	—	—	—	—	—	—	—	
(0xA3)	Reserved	—	—	—	—	—	—	—	—	
(0xA2)	Reserved	—	—	—	—	—	—	—	—	
(0xA1)	Reserved	—	—	—	—	—	—	—	—	
(0xA0)	Reserved	—	—	—	—	—	—	—	—	
(0x9F)	Reserved	—	—	—	—	—	—	—	—	
(0x9E)	Reserved	—	—	—	—	—	—	—	—	
(0x9D)	Reserved	—	—	—	—	—	—	—	—	
(0x9C)	Reserved	—	—	—	—	—	—	—	—	
(0x9B)	Reserved	—	—	—	—	—	—	—	—	
(0x9A)	Reserved	—	—	—	—	—	—	—	—	
(0x99)	Reserved	—	—	—	—	—	—	—	—	
(0x98)	Reserved	—	—	—	—	—	—	—	—	
(0x97)	Reserved	—	—	—	—	—	—	—	—	
(0x96)	Reserved	—	—	—	—	—	—	—	—	
(0x95)	Reserved	—	—	—	—	—	—	—	—	
(0x94)	Reserved	—	—	—	—	—	—	—	—	
(0x93)	Reserved	—	—	—	—	—	—	—	—	
(0x92)	Reserved	—	—	—	—	—	—	—	—	
(0x91)	Reserved	—	—	—	—	—	—	—	—	
(0x90)	Reserved	—	—	—	—	—	—	—	—	
(0x8F)	Reserved	—	—	—	—	—	—	—	—	
(0x8E)	Reserved	—	—	—	—	—	—	—	—	
(0x8D)	Reserved	—	—	—	—	—	—	—	—	
(0x8C)	Reserved	—	—	—	—	—	—	—	—	
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								117
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								117
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								117
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								117
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								118
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								118
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								117
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								117
(0x83)	Reserved	—	—	—	—	—	—	—	—	
(0x82)	TCCR1C	FOC1A	FOC1B	—	—	—	—	—	—	116
(0x81)	TCCR1B	ICNC1	ICES1	—	WGM13	WGM12	CS12	CS11	CS10	115
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	—	—	WGM11	WGM10	113
(0x7F)	DIDR1	—	—	—	—	—	—	AIN1D	AIN0D	188
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	205