

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	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-PDIP
Purchase URL	<a href="https://www.e-xfl.com/product-detail/atmel/atmega8-16pi">https://www.e-xfl.com/product-detail/atmel/atmega8-16pi</a>

### Asynchronous Timer Clock – $clk_{ASY}$

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode. The Asynchronous Timer/Counter uses the same XTAL pins as the CPU main clock but requires a CPU main clock frequency of more than four times the Oscillator frequency. Thus, asynchronous operation is only available while the chip is clocked on the Internal Oscillator.

### ADC Clock – $clk_{ADC}$

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

## Clock Sources

The device has the following clock source options, selectable by Flash Fuse Bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

**Table 2.** Device Clocking Options Select<sup>(1)</sup>

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 3. The frequency of the Watchdog Oscillator is voltage dependent as shown in “ATmega8 Typical Characteristics”. The device is shipped with CKSEL = “0001” and SUT = “10” (1 MHz Internal RC Oscillator, slowly rising power).

**Table 3.** Number of Watchdog Oscillator Cycles

Typical Time-out ( $V_{CC} = 5.0V$ )	Typical Time-out ( $V_{CC} = 3.0V$ )	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

## Internal Voltage Reference

ATmega8 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. The 2.56V reference to the ADC is generated from the internal bandgap reference.

### 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 16. 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 BODEN Fuse).
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 16.** Internal Voltage Reference Characteristics

Symbol	Parameter	Min	Typ	Max	Units
$V_{BG}$	Bandgap reference voltage	1.15	1.23	1.35	V
$t_{BG}$	Bandgap reference start-up time		40	70	$\mu$ s
$I_{BG}$	Bandgap reference current consumption		10		$\mu$ A

When the BOOTRST Fuse is unprogrammed, the boot section size set to 2K bytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```

AddressLabels Code           Comments
$000      rjmp  RESET        ; Reset handler
;
$001  RESET:ldi   r16,high(RAMEND); Main program start
$002      out    SPH,r16      ; Set Stack Pointer to top of RAM
$003      ldi    r16,low(RAMEND)
$004      out    SPL,r16
$005      sei                      ; Enable interrupts
$006      <instr> xxx
;
.org $c01
$c01      rjmp  EXT_INT0      ; IRQ0 Handler
$c02      rjmp  EXT_INT1      ; IRQ1 Handler
...      ...      ... ;
$c12      rjmp  SPM_RDY      ; Store Program Memory Ready
Handler

```

When the BOOTRST Fuse is programmed and the boot section size set to 2K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```

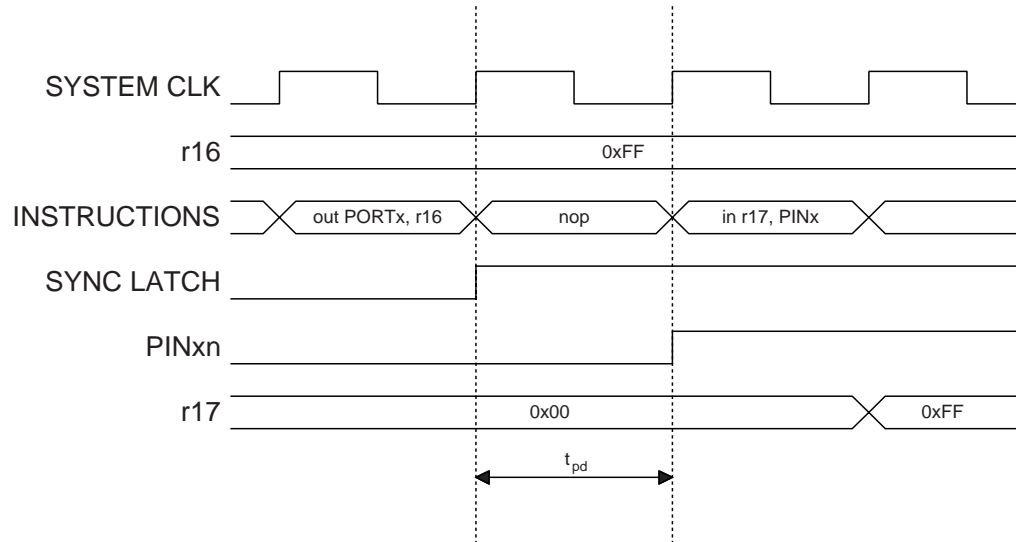
AddressLabels Code           Comments
.org $001
$001      rjmp  EXT_INT0      ; IRQ0 Handler
$002      rjmp  EXT_INT1      ; IRQ1 Handler
...      ...      ... ;
$012      rjmp  SPM_RDY      ; Store Program Memory Ready
Handler
;
.org $c00
$c00      rjmp  RESET        ; Reset handler
;
$c01  RESET:ldi   r16,high(RAMEND); Main program start
$c02      out    SPH,r16      ; Set Stack Pointer to top of RAM
$c03      ldi    r16,low(RAMEND)
$c04      out    SPL,r16
$c05      sei                      ; Enable interrupts
$c06      <instr> xxx

```

Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows  $t_{pd,max}$  and  $t_{pd,min}$ , a single signal transition on the pin will be delayed between  $\frac{1}{2}$  and  $1\frac{1}{2}$  system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a *nop* instruction must be inserted as indicated in Figure 24. The *out* instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay  $t_{pd}$  through the synchronizer is 1 system clock period.

**Figure 24.** Synchronization when Reading a Software Assigned Pin Value



update of the OCR1x 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 OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the High byte temporary register (TEMP). However, it is a good practice to read the Low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16-bit is done continuously. The High byte (OCR1xH) has to be written first. When the High byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the Low byte (OCR1xL) is written to the lower eight bits, the High byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 77.

#### **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* (FOC1x) bit. Forcing Compare Match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real Compare Match had occurred (the COM1x1:0 bits settings define whether the OC1x pin is set, cleared or toggled).

#### **Compare Match Blocking by TCNT1 Write**

All CPU writes to the TCNT1 Register will block any Compare Match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

#### **Using the Output Compare Unit**

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the Compare Match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The Compare Match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

ing at 0x0000 before the Compare Match can occur. The OCR1A Register, however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 Flag is set.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3. See Table 37 on page 96. The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR\_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the Compare Match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each Compare Match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of  $f_{OC1A} = f_{clk\_I/O}/2$  when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

## Phase Correct PWM Mode

The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the Compare Match between TCNT1 and OCR1x while upcounting, and set on the Compare Match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or

### 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 (PE). 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 USBS 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 (PE) Flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the PE 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 135 and “Parity Checker” on page 144.



This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

**Table 59.** UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

## USART Baud Rate Registers – UBRRL and UBRRHs

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the “Accessing UBRRH/UCSRC Registers” on page 149 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

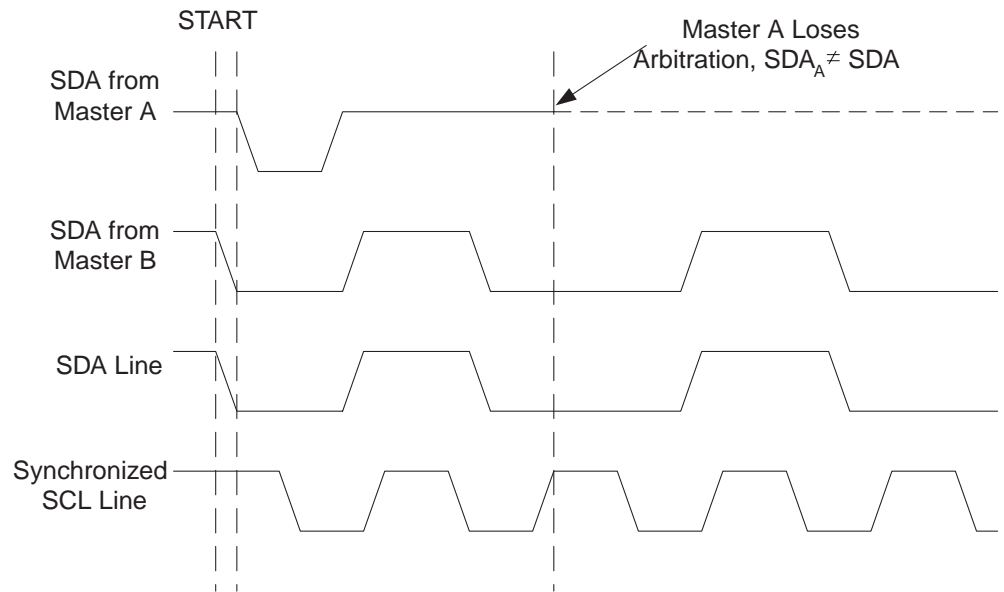
- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

**Figure 75. Arbitration Between Two Masters**

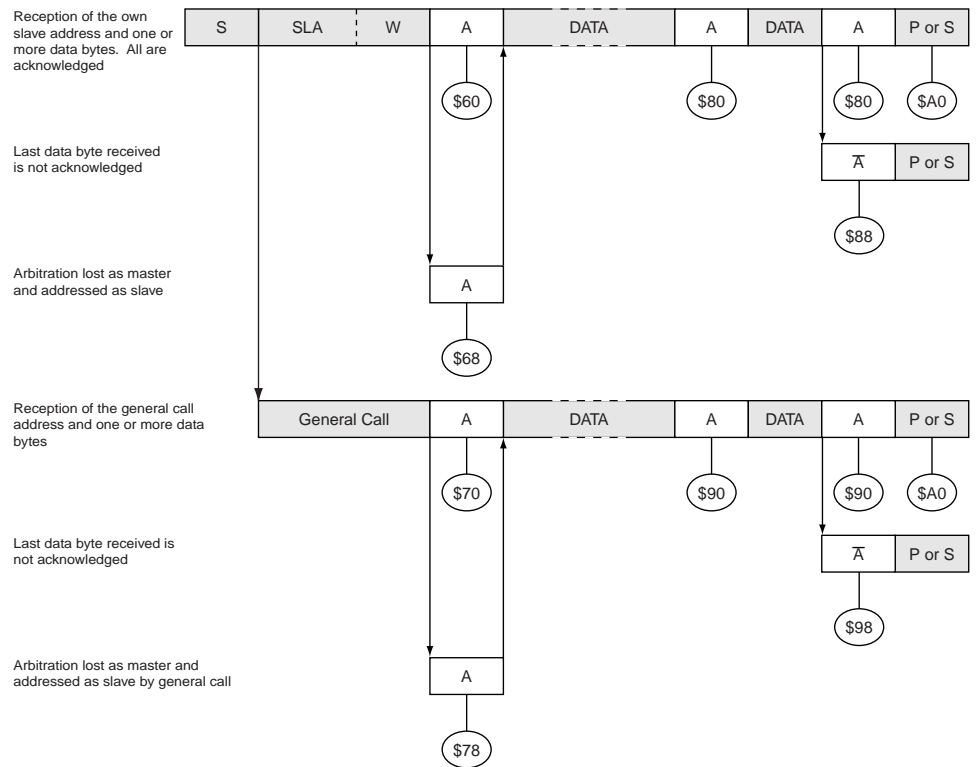




Note that arbitration is not allowed between:

- A REPEATED START condition and a data bit.
- A STOP condition and a data bit.
- A REPEATED START and a STOP condition.


It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

**Figure 83. Formats and States in the Slave Receiver Mode**



 From master to slave  
 From slave to master

 Any number of data bytes and their associated acknowledge bits

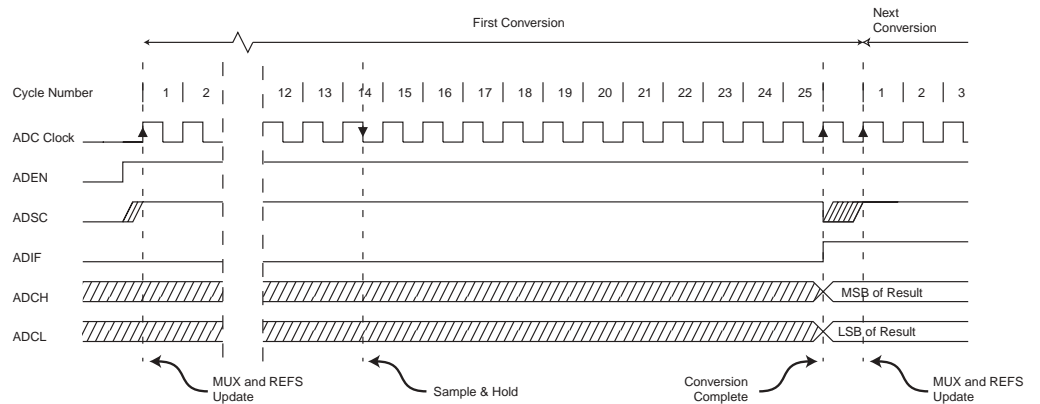
 This number (contained in TWSR) corresponds to a defined state of the Two-Wire Serial Bus. The prescaler bits are zero or masked to zero

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

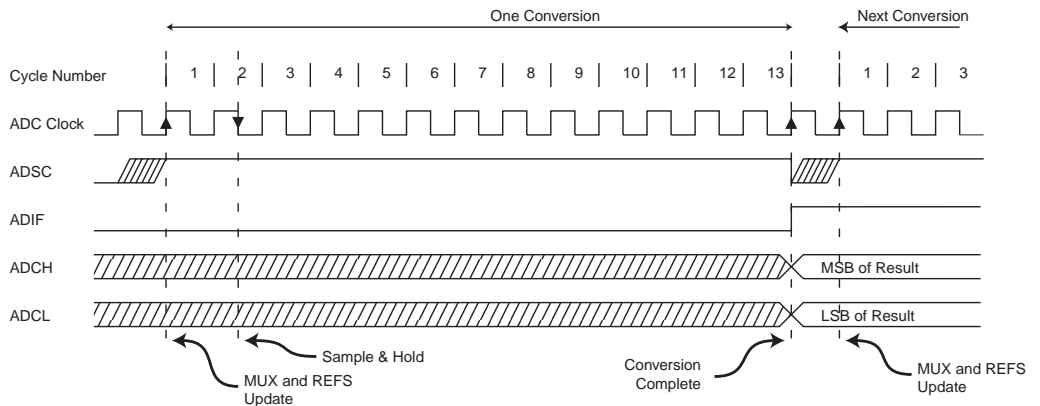
The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In single conversion mode, ADSC is cleared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

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 73.

**Figure 92.** ADC Timing Diagram, First Conversion (Single Conversion Mode)



**Figure 93.** ADC Timing Diagram, Single Conversion



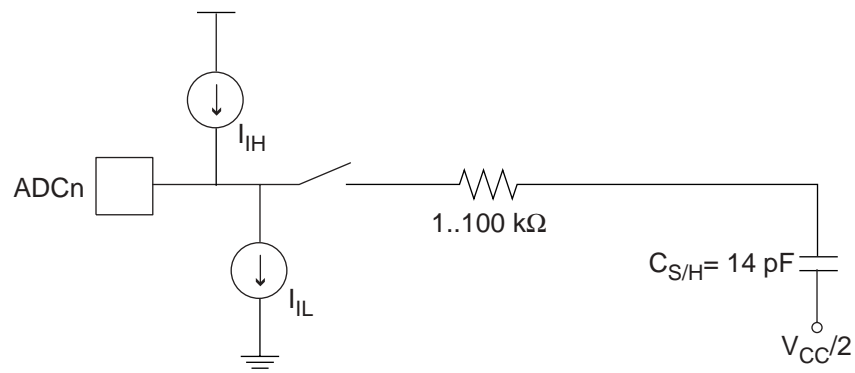
## Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 95. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 k $\Omega$  or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ( $f_{ADC}/2$ ) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

**Figure 95.** Analog Input Circuitry



## Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
2. The AVCC pin on the device should be connected to the digital  $V_{CC}$  supply voltage via an LC network as shown in Figure 96.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC [3..0] port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress. However, using the Two-wire Interface (ADC4 and ADC5) will only affect the conversion on ADC4 and ADC5 and not the other ADC channels.

Assembly Code Example for a Boot Loader” on page 216 for an assembly code example.

## Performing Page Erase by SPM

To execute page erase, set up the address in the Z-pointer, write “X0000011” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the page erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

## Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the RWWSRE bit in SPMCR. It is also erased after a System Reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

Note: If the EEPROM is written in the middle of an SPM page Load operation, all data loaded will be lost.

## Performing a Page Write

To execute page write, set up the address in the Z-pointer, write “X0000101” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the page write.
- Page Write to the NRWW section: The CPU is halted during the operation.

## Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCR is cleared. This means that the interrupt can be used instead of polling the SPMCR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in “Interrupts” on page 44.

## Consideration While Updating BLS

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

## Prevent Reading the RWW Section During Self-Programming

During Self-Programming (either page erase or page write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the self programming operation. The RWWSB in the SPMCR will be set as long as the RWW section is busy. During Self-Programming the Interrupt Vector table should be moved to the BLS as described in “Interrupts” on page 44, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See “Simple Assembly Code Example for a Boot Loader” on page 216 for an example.

## Memory Programming

### Program And Data Memory Lock Bits

The ATmega8 provides six Lock Bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 86. The Lock Bits can only be erased to “1” with the Chip Erase command.

**Table 85.** Lock Bit Byte

Lock Bit Byte	Bit No.	Description	Default Value <sup>(1)</sup>
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot lock bit	1 (unprogrammed)
BLB11	4	Boot lock bit	1 (unprogrammed)
BLB02	3	Boot lock bit	1 (unprogrammed)
BLB01	2	Boot lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

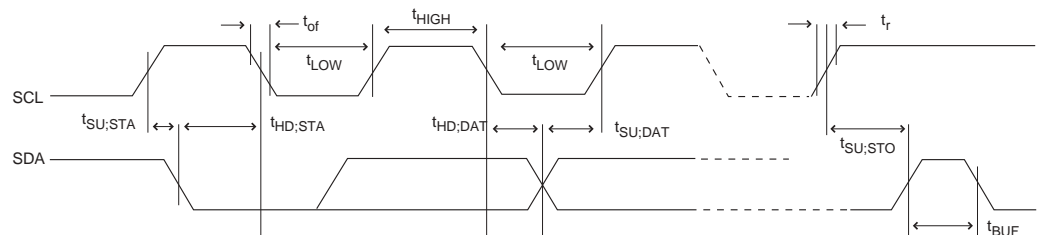
Note: 1. “1” means unprogrammed, “0” means programmed

**Table 86.** Lock Bit Protection Modes<sup>(2)</sup>

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse Bits are locked in both Serial and Parallel Programming mode. <sup>(1)</sup>
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in parallel and Serial Programming mode. The Fuse Bits are locked in both Serial and Parallel Programming modes. <sup>(1)</sup>
BLB0 Mode	BLB02	BLB01	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
BLB1 Mode	BLB12	BLB11	

5. This requirement applies to all ATmega8 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general  $f_{SCL}$  requirement.
6. The actual low period generated by the ATmega8 Two-wire Serial Interface is  $(1/f_{SCL} - 2/f_{CK})$ , thus  $f_{CK}$  must be greater than 6 MHz for the low time requirement to be strictly met at  $f_{SCL} = 100$  kHz.
7. The actual low period generated by the ATmega8 Two-wire Serial Interface is  $(1/f_{SCL} - 2/f_{CK})$ , thus the low time requirement will not be strictly met for  $f_{SCL} > 308$  kHz when  $f_{CK} = 8$  MHz. Still, ATmega8 devices connected to the bus may communicate at full speed (400 kHz) with other ATmega8 devices, as well as any other device with a proper  $t_{LOW}$  acceptance margin.

**Figure 115. Two-wire Serial Bus Timing**



## SPI Timing Characteristics

See Figure 116 and Figure 117 for details.

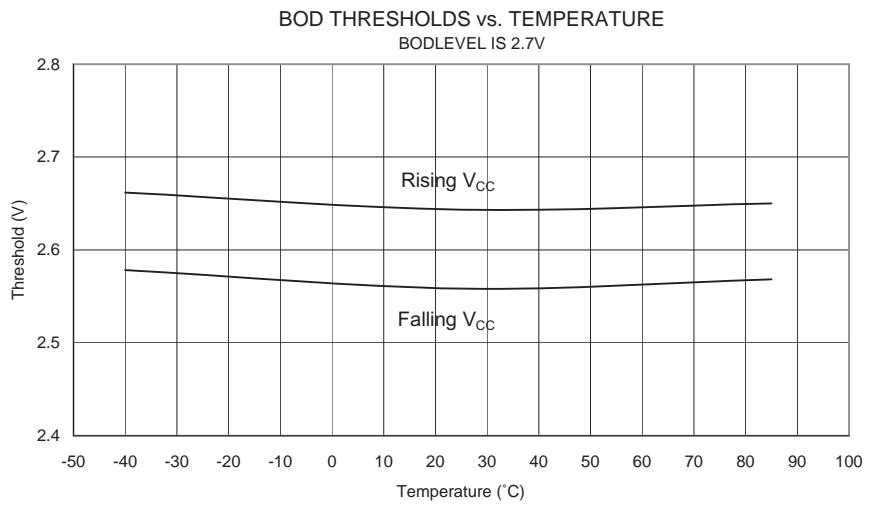
**Table 102. SPI Timing Parameters**

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 50		ns
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		$0.5 \cdot t_{SCK}$		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	$\overline{SS}$ low to out	Slave		15		
10	SCK period	Slave	$4 \cdot t_{ck}$			
11	SCK high/low <sup>(1)</sup>	Slave	$2 \cdot t_{ck}$			
12	Rise/Fall time	Slave			1.6	
13	Setup	Slave	10			
14	Hold	Slave	10			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	$\overline{SS}$ low to SCK	Slave	$2 \cdot t_{ck}$			

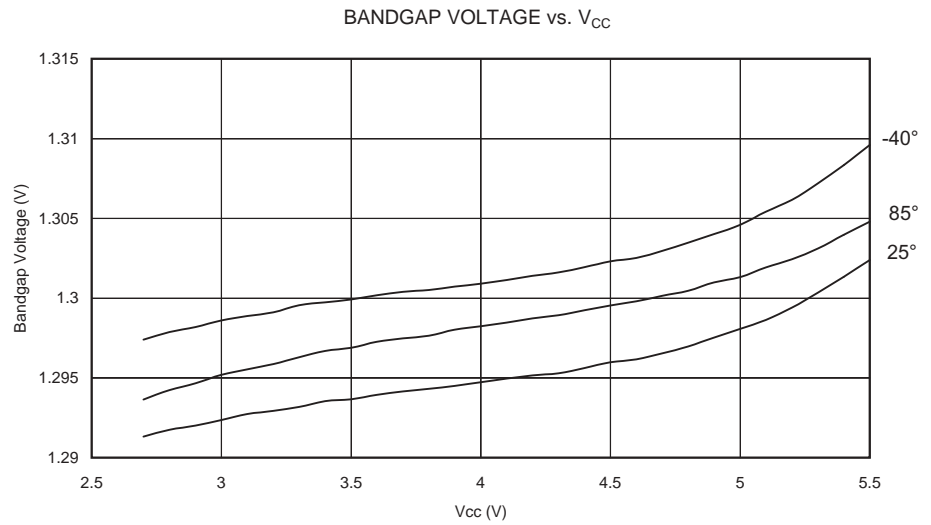
Note: 1. In SPI Programming mode the minimum SCK high/low period is:  
 -  $2t_{CLCL}$  for  $f_{CK} < 12$  MHz  
 -  $3t_{CLCL}$  for  $f_{CK} > 12$  MHz



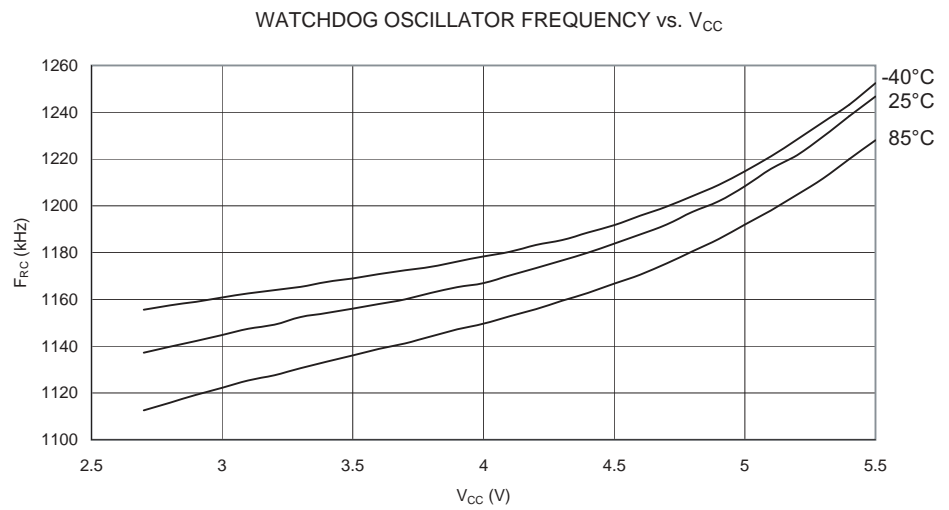
**Figure 165.** BOD Thresholds vs. Temperature (BOD Level is 2.7v)



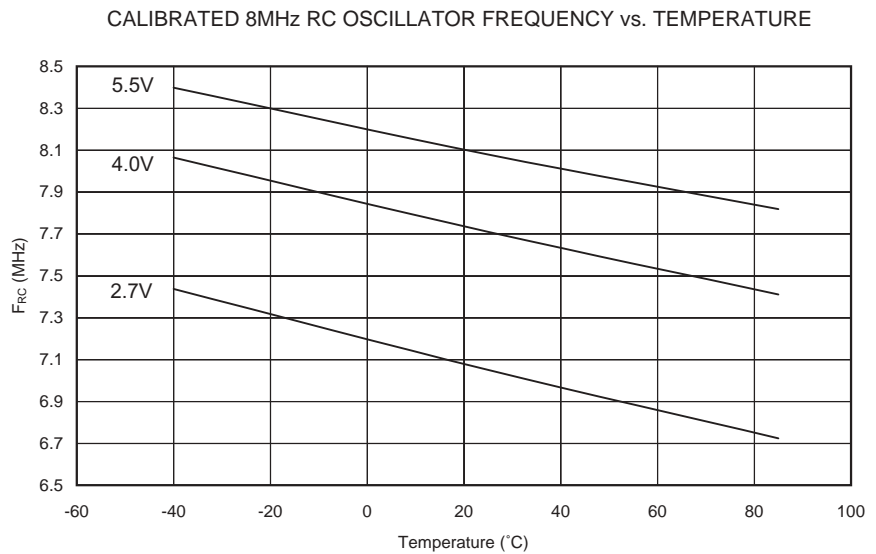
**Figure 166.** Bandgap Voltage vs.  $V_{CC}$



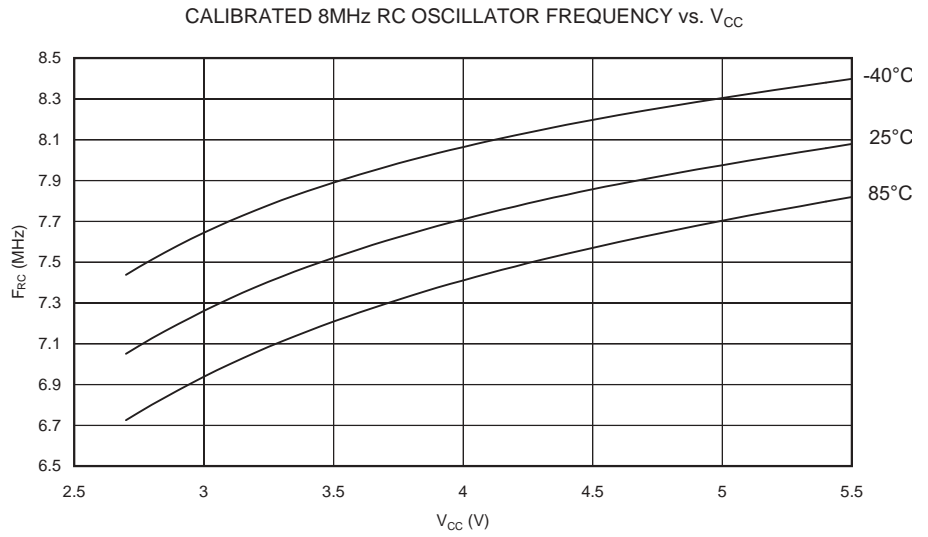
**Figure 169.** Watchdog Oscillator Frequency vs.  $V_{CC}$



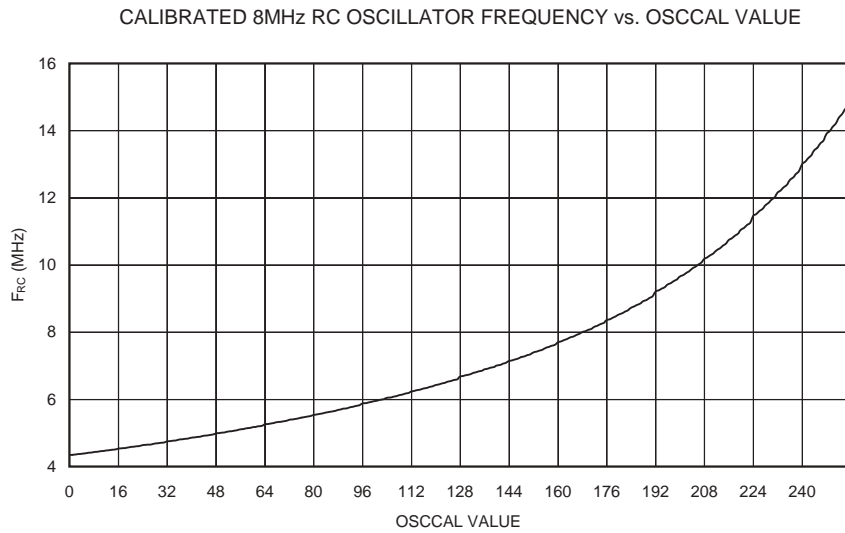
**Figure 170.** Calibrated 8 MHz RC Oscillator Frequency vs. Temperature



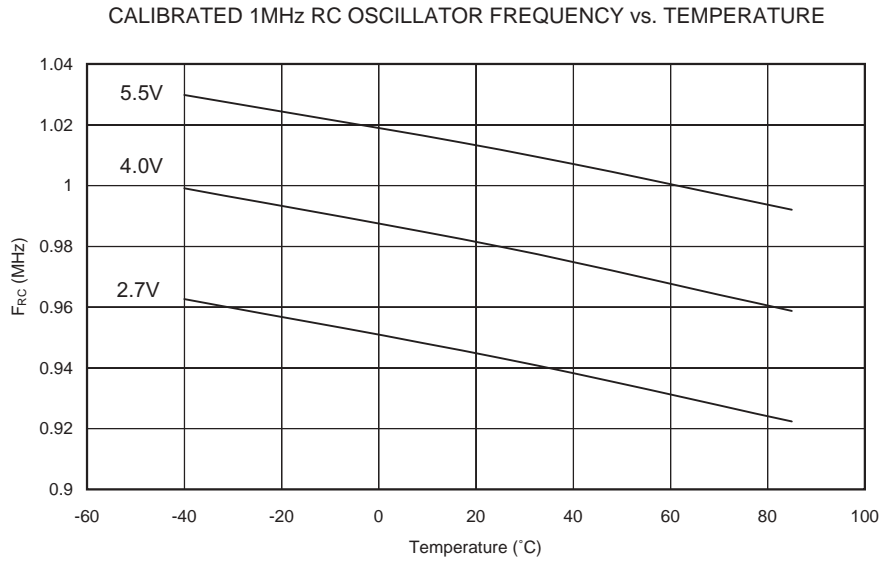
**Figure 171.** Calibrated 8 MHz RC Oscillator Frequency vs.  $V_{CC}$



**Figure 172.** Calibrated 8 MHz RC Oscillator Frequency vs. Oscscal Value



**Figure 179.** Calibrated 1 MHz RC Oscillator Frequency vs. Temperature



**Figure 180.** Calibrated 1 MHz RC Oscillator Frequency vs.  $V_{CC}$

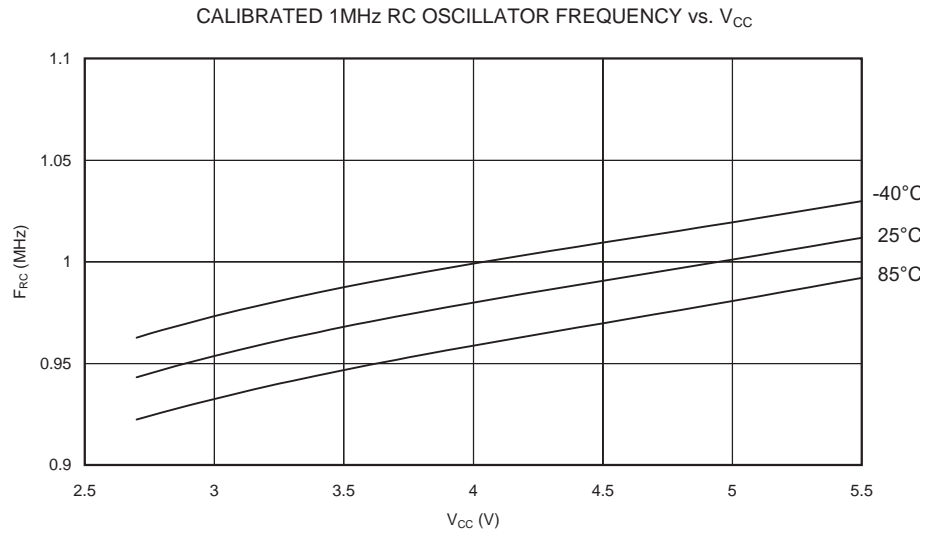


Figure 191. Reset Pulse Width vs.  $V_{CC}$

