



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	8MHz
Connectivity	I ² 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)	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	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega8l-8ai

Unconnected pins

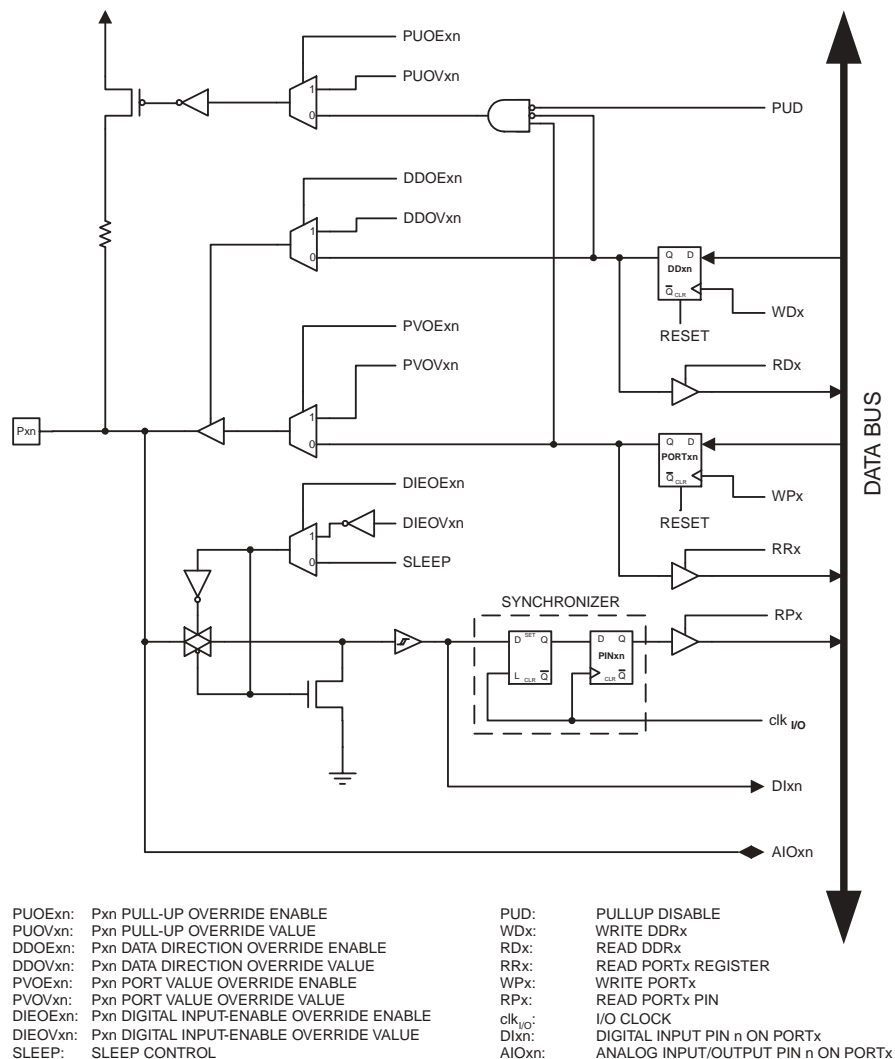
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 25 shows how the port pin control signals from the simplified Figure 22 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 25. Alternate Port Functions⁽¹⁾



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

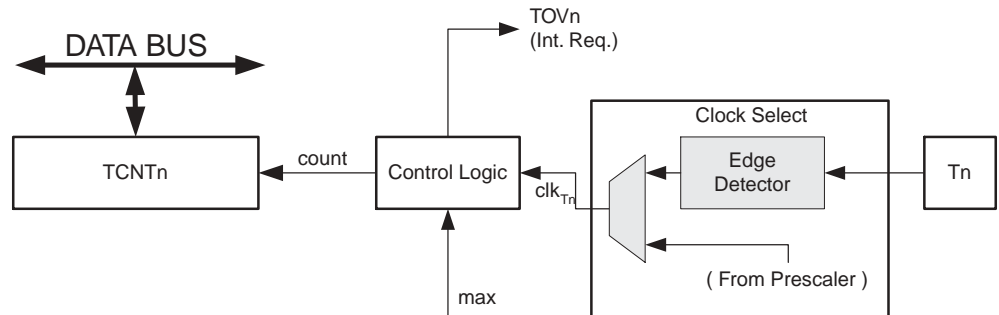
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 72.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable counter unit. Figure 27 shows a block diagram of the counter and its surroundings.

Figure 27. Counter Unit Block Diagram



Signal description (internal signals):

- count** Increment TCNT0 by 1.
- clk_{Tn}** Timer/Counter clock, referred to as clk_{T0} in the following.
- max** Signalize that TCNT0 has reached maximum value.

The counter is incremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the clock select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

Operation

The counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (MAX = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. A new counter value can be written anytime.

Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 28 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 28. Timer/Counter Timing Diagram, No Prescaling

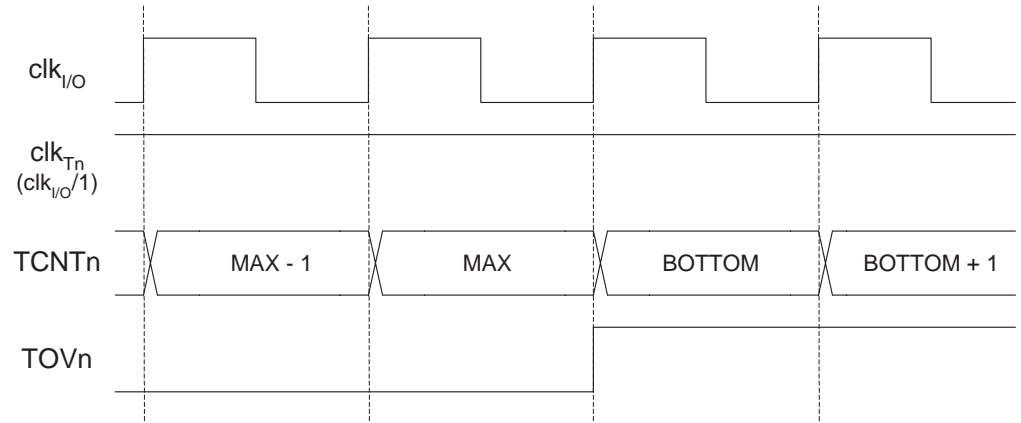
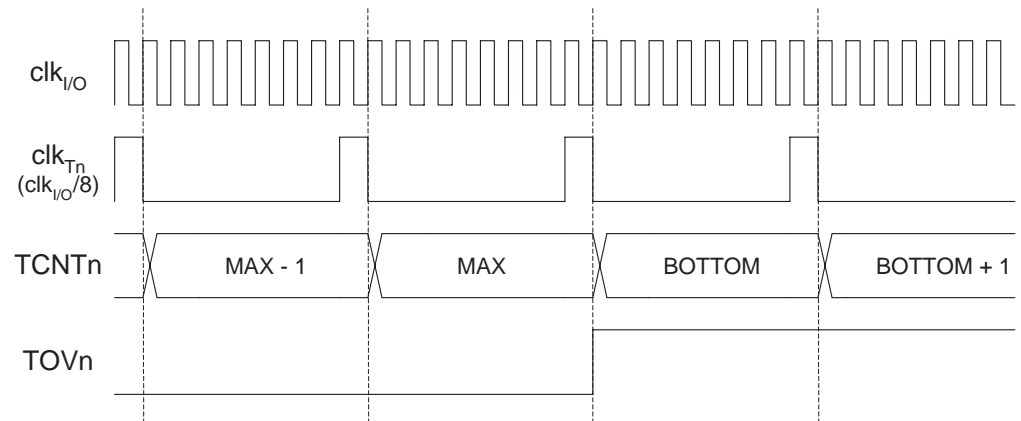


Figure 29 shows the same timing data, but with the prescaler enabled.

Figure 29. Timer/Counter Timing Diagram, with Prescaler ($f_{\text{clk}_{I/O}}/8$)



8-bit Timer/Counter2 with PWM and Asynchronous Operation

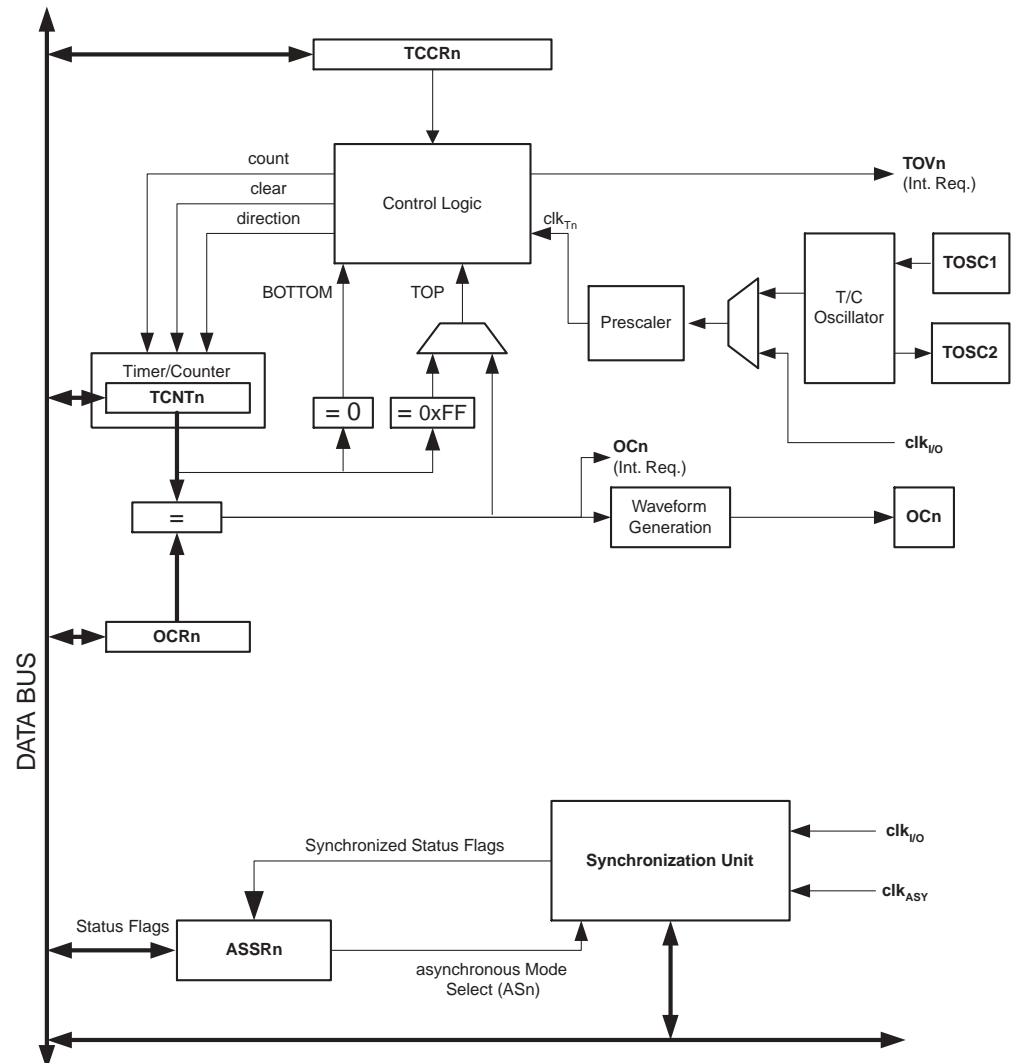
Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single Channel Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV2 and OCF2)**
- **Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock**

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 45. For the actual placement of I/O pins, refer to “Pin Configurations” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “8-bit Timer/Counter Register Description” on page 115.

Figure 45. 8-bit Timer/Counter Block Diagram



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 (FOC2) bit. Forcing Compare Match will not set the OCF2 Flag or reload/clear the timer, but the OC2 pin will be updated as if a real Compare Match had occurred (the COM21:0 bits settings define whether the OC2 pin is set, cleared or toggled).

Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any Compare Match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2 to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2 value, the Compare Match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

The setup of the OC2 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2 value is to use the Force Output Compare (FOC2) strobe bit in Normal mode. The OC2 Register keeps its value even when changing between waveform generation modes.

Be aware that the COM21:0 bits are not double buffered together with the compare value. Changing the COM21:0 bits will take effect immediately.

Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the Receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received (i.e., a complete serial frame is present in the Receive Shift Register), the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example⁽¹⁾

```
USART_Receive:
    ; Wait for data to be received
    sbis UCSRA, RXC
    rjmp USART_Receive
    ; Get and return received data from buffer
    in    r16, UDR
    ret
```

C Code Example⁽¹⁾

```
unsigned char USART_Receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSRA & (1<<RXC)) )
        ;
    /* Get and return received data from buffer */
    return UDR;
}
```

Note: 1. The example code assumes that the part specific header file is included.

The function simply waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.

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.

described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.

6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR 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 TWSR, 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 TWCR, instructing the TWI 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 will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the 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:

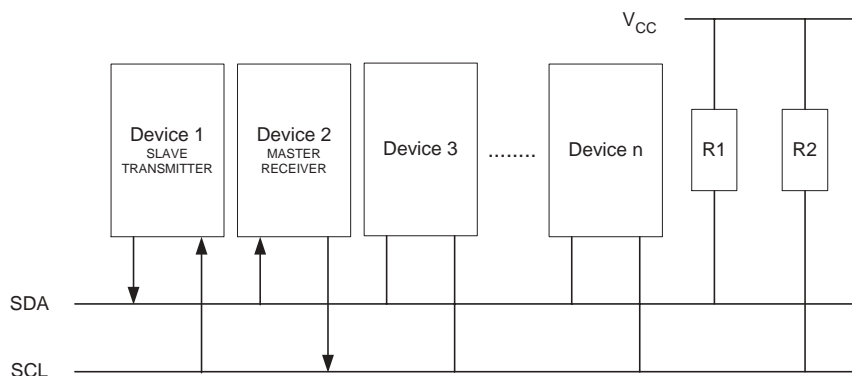
- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded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

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.

Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 84). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 84. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

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

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

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

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 69. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the Two-wire Serial Bus.

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

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

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)

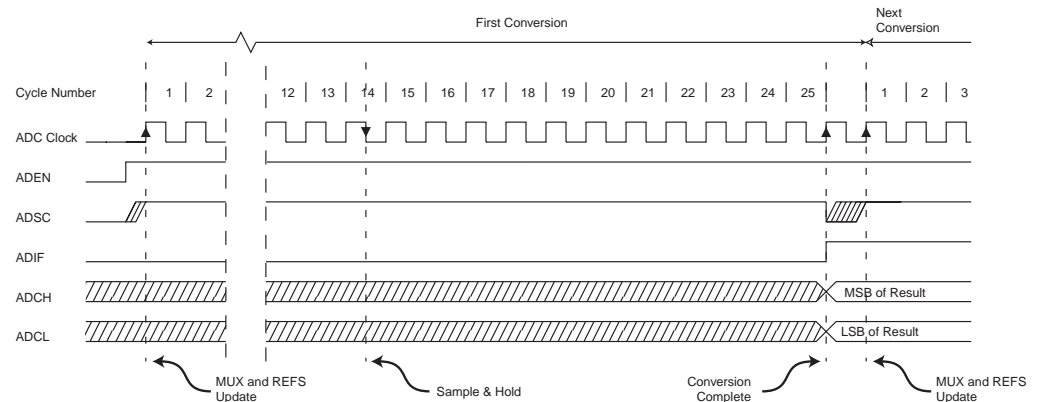
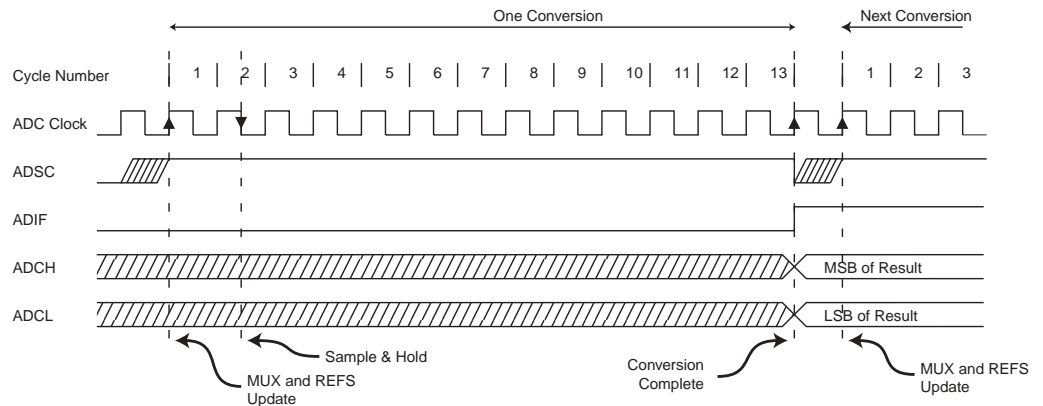


Figure 93. ADC Timing Diagram, Single Conversion



Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock Bits, write the desired data to R0, write “X0001001” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The only accessible Lock Bits are the Boot Lock Bits that may prevent the Application and Boot Loader section from any software update by the MCU.

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

See Table 78 and Table 79 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 SP MEN are set in SPMCR. 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, 6, 1, and 0 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 SPMCR

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 SPMCR 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 SP MEN bits in SPMCR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SP MEN bits are set in SPMCR, the value of the Lock Bits will be loaded in the destination register. The BLBSET and SP MEN 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 SP MEN 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 bits is similar to the one described above for reading the Lock Bits. To read the Fuse Low bits, load the Z-pointer with 0x0000 and set the BLBSET and SP MEN bits in SPMCR. When an LPM instruction is executed within three cycles after the BLBSET and SP MEN bits are set in the SPMCR, the value of the Fuse Low bits (FLB) will be loaded in the destination register as shown below. Refer to Table 88 on page 221 for a detailed description and mapping of the fuse low bits.

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

Similarly, when reading the Fuse High bits, load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SP MEN bits are set in the SPMCR, the value of the Fuse High bits (FHB) will be loaded in the destination register as shown below. Refer to Table 87 on page 220 for detailed description and mapping of the fuse high bits.

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

Fuse and Lock Bits that are programmed, will be read as zero. Fuse and Lock Bits that are unprogrammed, will be read as one.

Programming the Fuse High Bits

The algorithm for programming the Fuse high bits is as follows (refer to “Programming the Flash” on page 225 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low 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 data byte.
4. Give WR a negative pulse and wait for RDY/BSY to go high.
5. Set BS1 to “0”. This selects low data byte.

Programming the Lock Bits

The algorithm for programming the Lock Bits is as follows (refer to “Programming the Flash” on page 225 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.
3. Give WR a negative pulse and wait for RDY/BSY to go high.

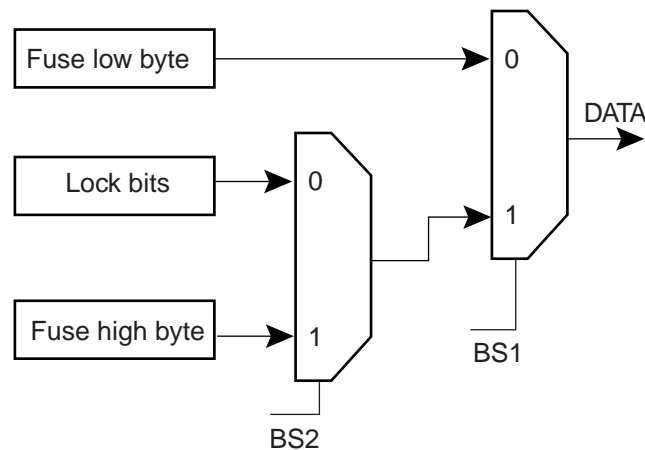
The Lock Bits can only be cleared by executing Chip Erase.

Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock Bits is as follows (refer to “Programming the Flash” on page 225 for details on Command loading):

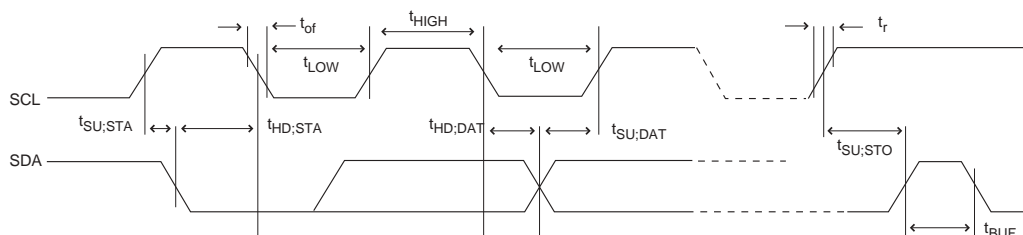
1. A: Load Command “0000 0100”.
2. Set OE to “0”, BS2 to “0”, and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set OE to “0”, BS2 to “1”, and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “0”, and BS1 to “1”. The status of the Lock Bits can now be read at DATA (“0” means programmed).
5. Set OE to “1”.

Figure 108. Mapping Between BS1, BS2 and the Fuse- and Lock Bits During Read



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	SS low to out	Slave		15		
10	SCK period	Slave	$4 \cdot t_{ck}$			
11	SCK high/low ⁽¹⁾	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 SS high	Slave	20			
17	SS high to tri-state	Slave		10		
18	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 129. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

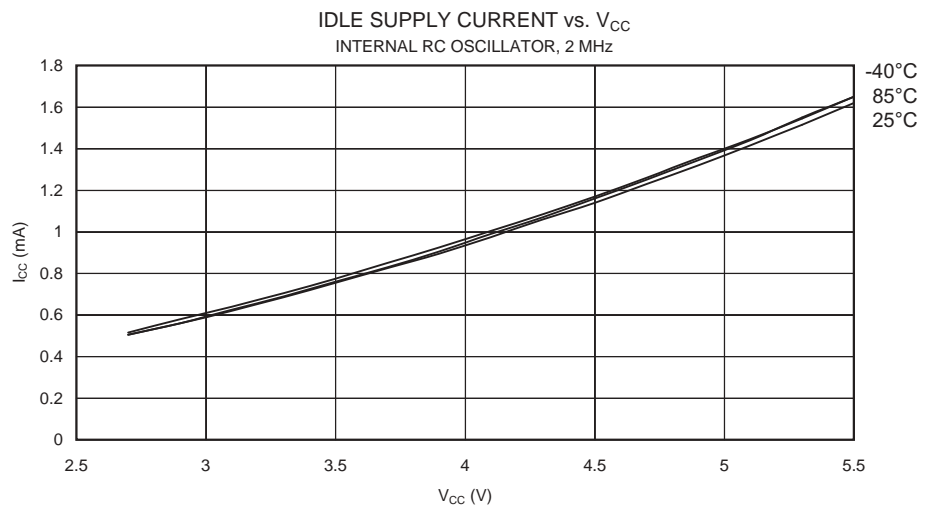


Figure 130. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

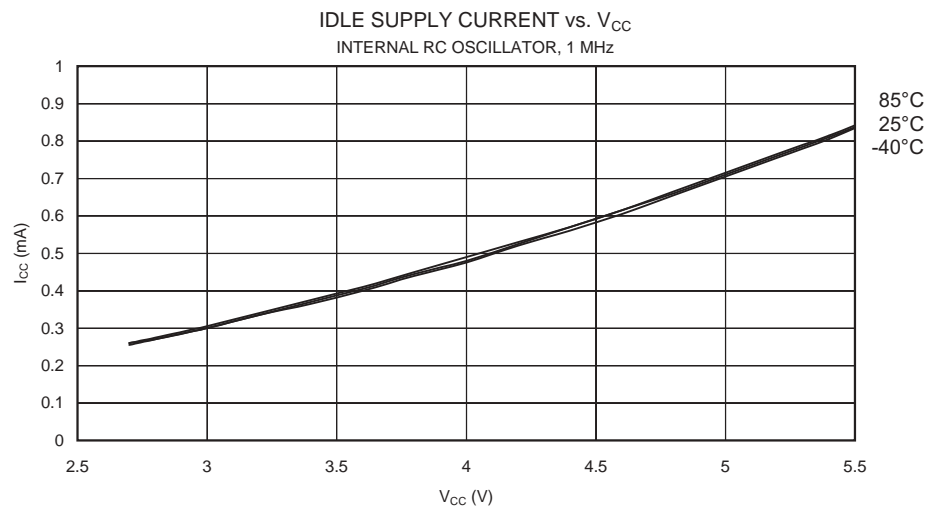


Figure 145. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC} = 5V$)

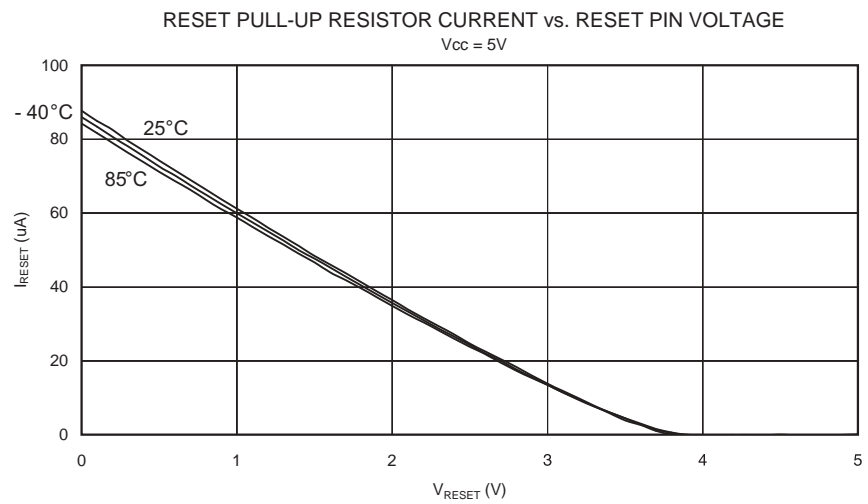


Figure 146. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC} = 2.7V$)

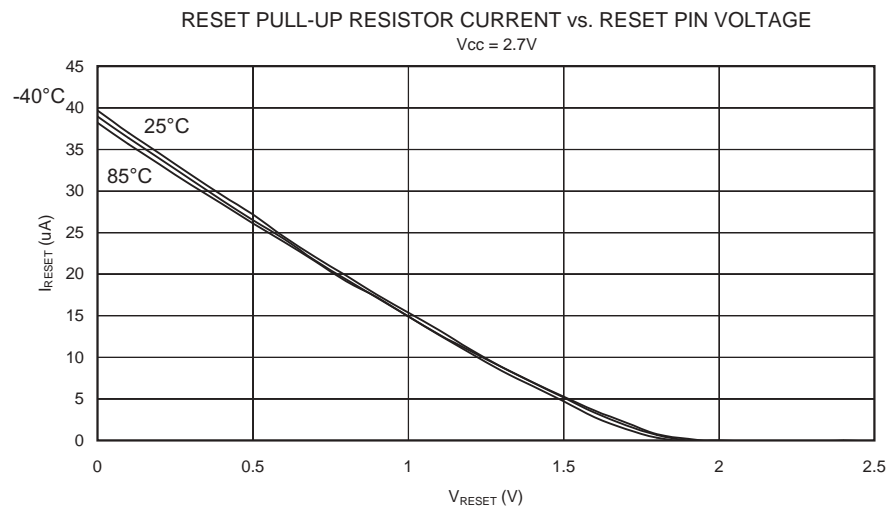


Figure 175. Calibrated 4 MHz RC Oscillator Frequency vs. Oscscal Value

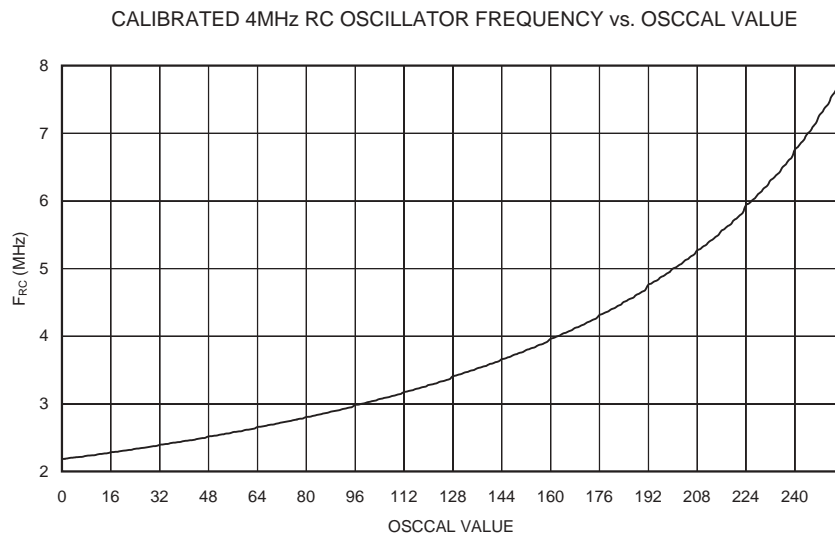


Figure 176. Calibrated 2 MHz RC Oscillator Frequency vs. Temperature

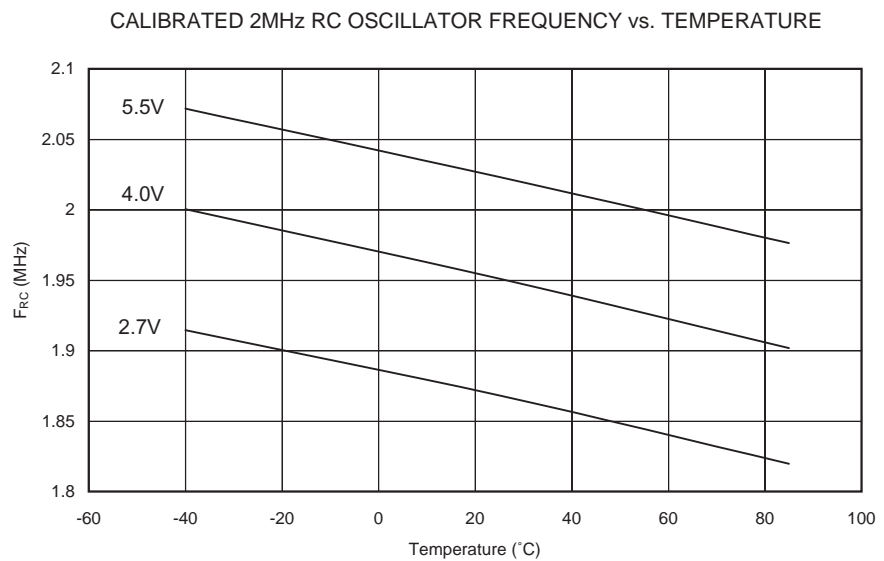


Figure 177. Calibrated 2 MHz RC Oscillator Frequency vs. V_{CC}

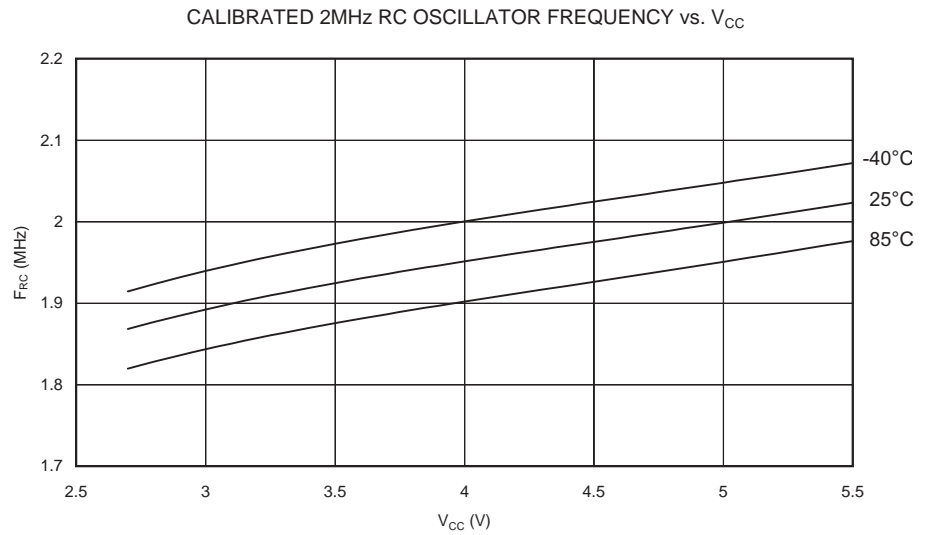
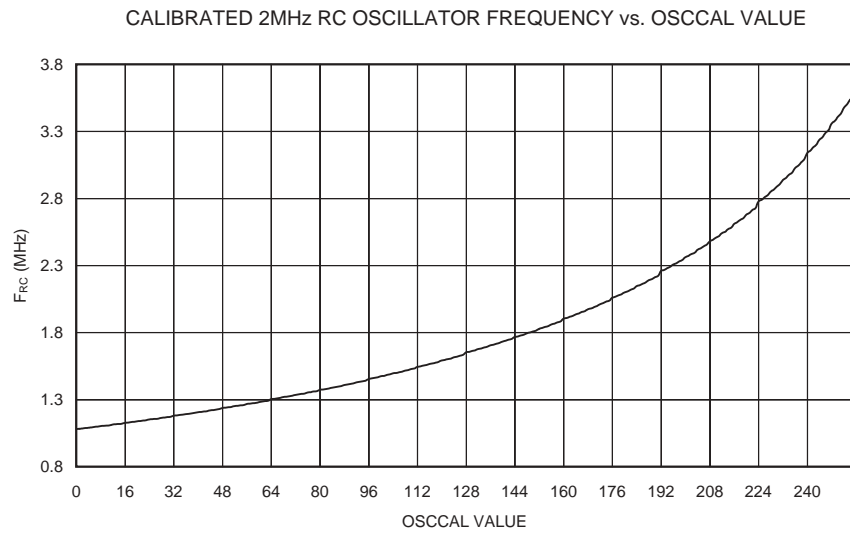


Figure 178. Calibrated 2 MHz RC Oscillator Frequency vs. Oscal Value



Changes from Rev. 2486B-12/01 to Rev. 2486C-03/02

All page numbers refers to this document.

1 Updated TWI Chapter.

More details regarding use of the TWI Power-down operation and using the TWI as Master with low TWBRR values are added into the datasheet.

Added the note at the end of the “Bit Rate Generator Unit” on page 167.

Added the description at the end of “Address Match Unit” on page 167.

2 Updated Description of OSCCAL Calibration Byte.

In the datasheet, it was not explained how to take advantage of the calibration bytes for 2, 4, and 8 MHz Oscillator selections. This is now added in the following sections:

Improved description of “Oscillator Calibration Register – OSCCAL” on page 29 and “Calibration Byte” on page 221.

3 Added Some Preliminary Test Limits and Characterization Data.

Removed some of the TBD’s in the following tables and pages:

Table 3 on page 24, Table 15 on page 36, Table 16 on page 40, Table 17 on page 42, “TA = -40°C to 85°C, VCC = 2.7V to 5.5V (unless otherwise noted)” on page 237, Table 99 on page 239, and Table 102 on page 241.

4 Updated Programming Figures.

Figure 104 on page 222 and Figure 112 on page 232 are updated to also reflect that AVCC must be connected during Programming mode.

5 Added a Description on how to Enter Parallel Programming Mode if RESET Pin is Disabled or if External Oscillators are Selected.

Added a note in section “Enter Programming Mode” on page 224.

8-bit Timer/Counter Register Description	115
Asynchronous Operation of the Timer/Counter	117
Timer/Counter Prescaler.....	121
Serial Peripheral Interface – SPI.....	122
\overline{SS} Pin Functionality.....	126
Data Modes	129
USART	130
Overview.....	130
Clock Generation	131
Frame Formats	134
USART Initialization.....	135
Data Transmission – The USART Transmitter	137
Data Reception – The USART Receiver	140
Asynchronous Data Reception	144
Multi-processor Communication Mode	148
Accessing UBRRH/UCSRC Registers.....	149
USART Register Description	150
Examples of Baud Rate Setting.....	156
Two-wire Serial Interface	160
Features.....	160
Two-wire Serial Interface Bus Definition.....	160
Data Transfer and Frame Format.....	161
Multi-master Bus Systems, Arbitration and Synchronization	164
Overview of the TWI Module	166
TWI Register Description.....	168
Using the TWI	171
Transmission Modes.....	175
Multi-master Systems and Arbitration.....	188
Analog Comparator	190
Analog Comparator Multiplexed Input	192
Analog-to-Digital Converter.....	193
Features.....	193
Starting a Conversion	195
Prescaling and Conversion Timing.....	195
Changing Channel or Reference Selection	197
ADC Noise Canceler.....	198
ADC Conversion Result.....	202