Microchip Technology - ATMEGA323L-4AI Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	4MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega323l-4ai

The General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector. See also “External Interrupts” on page 37.

- **Bit 6 – 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 activated. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising 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. See also “External Interrupts” on page 37.

- **Bit 5 – INT2: External Interrupt Request 2 Enable**

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control2 bit (ISC02) in the MCU Control and Status Register (MCUCSR) defines whether the external interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector. See also “External Interrupts” on page 37.

- **Bits 4..2 – Res: Reserved Bits**

These bits are reserved bits in the ATmega323 and always read as zero.

- **Bit 1 – IVSEL: Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address to the start of the Boot Flash section is determined by the BOOTSZ Fuses. Refer to the section “Boot Loader Support” on page 177 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Set the Interrupt Vector Change Enable (IVCE) bit.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will be automatically disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled in four cycles. The I-flag in the Status Register is unaffected by the automatic disabling.

- **Bits 6..4 – SM2..0: Sleep Mode Select Bits 2, 1 and 0**

These bits select between the six available sleep modes as shown in Table 8.

Table 8. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby mode and Extended Standby mode are only available with external crystals or resonators.

- **Bits 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 9. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

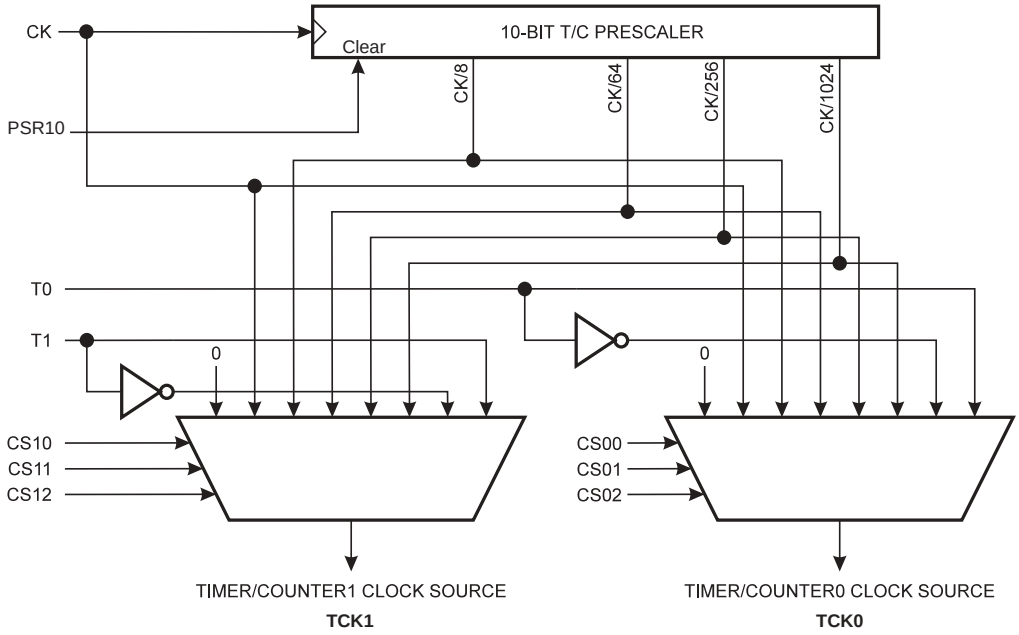
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 10. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Timer/Counters

The ATmega323 provides three general purpose Timer/Counters – two 8-bit T/Cs and one 16-bit T/C. Timer/Counter2 can optionally be asynchronously clocked from an external Oscillator. This Oscillator is optimized for use with a 32.768 kHz watch crystal, enabling use of Timer/Counter2 as a Real Time Counter (RTC). Timer/Counters 0 and 1 have individual prescaling selection from the same 10-bit prescaler. Timer/Counter2 has its own prescaler. Both these prescalers can be reset by setting the corresponding control bits in the Special Functions IO Register (SFIOR). These Timer/Counters can either be used as a timer with an internal clock time-base or as a counter with an external pin connection which triggers the counting.

Timer/Counter Prescalers

Figure 30. Prescaler for Timer/Counter0 and Timer/Counter1



For Timer/Counters 0 and 1, the four different prescaled selections are: CK/8, CK/64, CK/256, and CK/1024, where CK is the Oscillator clock. For the two Timer/Counters0 and 1, CK, external source, and stop can also be selected as clock sources. Setting the PSR10 bit in SFIOR Resets the prescaler. This allows the user to operate with a predictable prescaler. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a Prescaler Reset will affect both Timer/Counters.

Timer/Counter0 and 2 can also be used as 8-bit Pulse Width Modulators. In this mode, the Timer/Counter and the Output Compare Register serve as a glitch-free, stand-alone PWM with centered pulses. Refer to page 49 for a detailed description on this function.

Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	FOC0	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	TCCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter2 Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	TCCR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0/FOC2: Force Output Compare**

Writing a logical one to this bit, forces a change in the Compare Match output pin PB3 (Timer/Counter0) and PD7 (Timer/Counter2) according to the values already set in COMn1 and COMn0. If the COMn1 and COMn0 bits are written in the same cycle as FOC0/FOC2, the new settings will not take effect until next Compare Match or Forced Output Compare Match occurs. The Force Output Compare bit can be used to change the output pin without waiting for a Compare Match in the timer. The automatic action programmed in COMn1 and COMn0 happens as if a Compare Match had occurred, but no interrupt is generated and the Timer/Counters will not be cleared even if CTC0/CTC2 is set. The corresponding I/O pin must be set as an output pin for the FOC0/FOC2 bit to have effect on the pin. The FOC0/FOC2 bits will always be read as zero. Setting the FOC0/FOC2 bits has no effect in PWM mode.

- **Bit 6 – PWM0/PWM2: Pulse Width Modulator Enable**

When set (one) this bit enables PWM mode for Timer/Counter0 or Timer/Counter2. This mode is described on page 49.

- **Bits 5, 4 – COM01, COM00/COM21, COM20: Compare Output Mode, Bits 1 and 0**

The COMn1 and COMn0 control bits determine any output pin action following a compare match in Timer/Counter0 or Timer/Counter2. Output pin actions affect pins PB3(OC0) or PD7(OC2). This is an alternative function to an I/O port, and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 12.

Table 12. Compare Mode Select[1]

COMn1[2]	COMn0	Description
0	0	Timer/Counter Disconnected from Output Pin OCn
0	1	Toggle the OCn Output Line.
1	0	Clear the OCn Output Line (to Zero).
1	1	Set the OCn Output Line (to One).

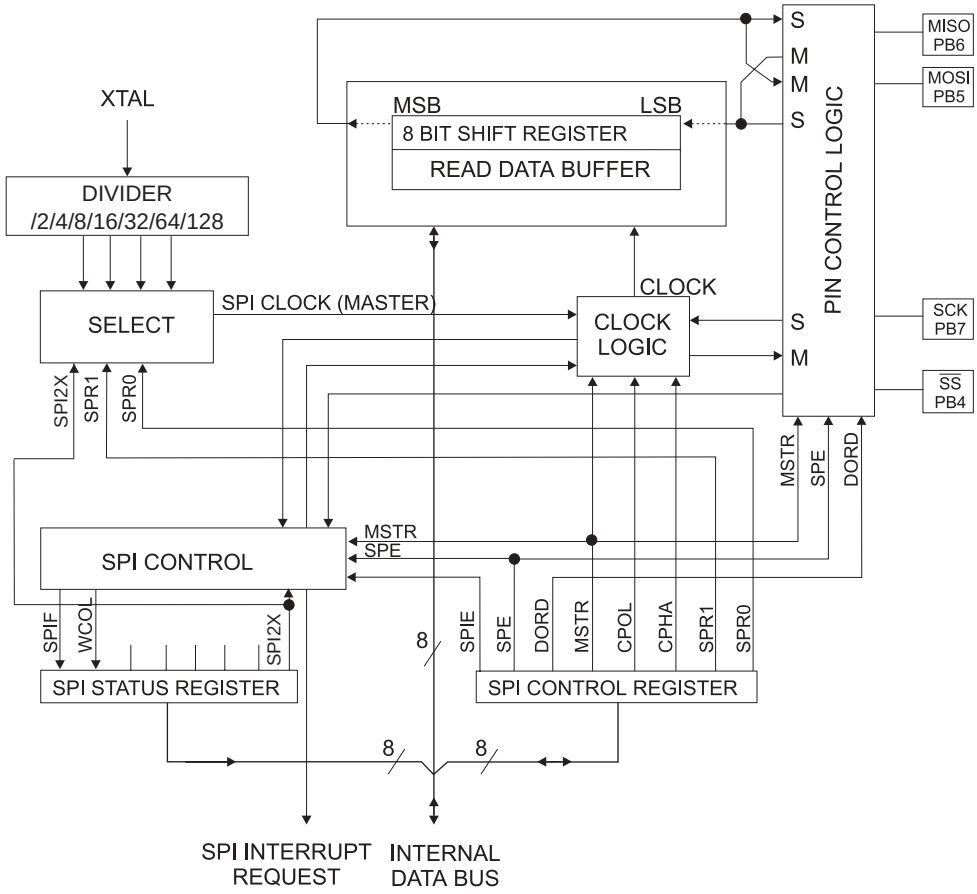
Notes: 1. In PWM mode, these bits have a different function. Refer to Table 15 for a description.
2. n = 0 or 2

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega323 and peripheral devices or between several AVR devices. The ATmega323 SPI includes the following features:

- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

Figure 41. SPI Block Diagram



The interconnection between Master and Slave CPUs with SPI is shown in Figure 42. The PB7(SCK) pin is the clock output in the Master mode and the clock input in the Slave mode. Writing to the SPI Data Register of the Master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the Slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Slave Select input, PB4($\overline{SS}$), is set low to select an individual Slave SPI device. The two Shift Registers in the Master and the Slave can be considered as one distributed 16-bit circular Shift Register. This is shown in Figure 42. When data is shifted from the Master to the Slave, data is also shifted in the opposite direction, simultaneously. During one shift cycle, data in the Master and the Slave is interchanged.

Table 29 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 29. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).
BAUD Baud rate (in bits per second, bps)
f_{OSC} System Oscillator clock frequency
UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)
Some examples of UBRR values for some system clock frequency are found in Table 36 (see page 99).

Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

External Clock

External clocking is used by the Synchronous Slave modes of operation. The description in this section refers to Figure 46 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

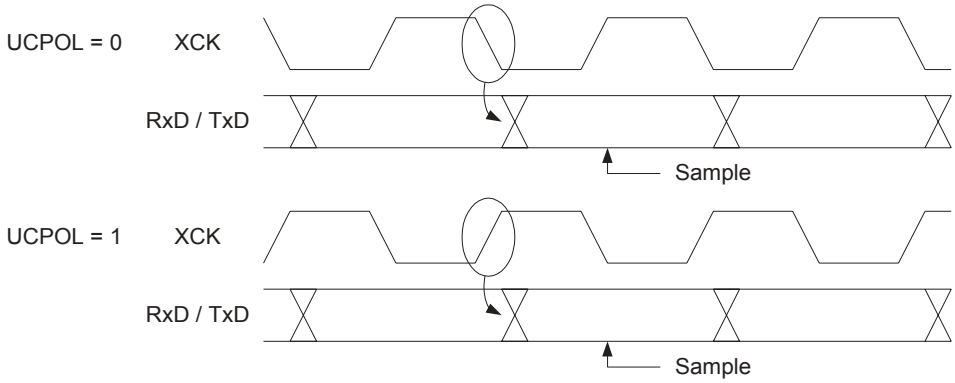
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCK pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 47. Synchronous Mode XCK Timing



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 47 shows, when UCPOL is zero the data will be changed at falling XCK edge and sampled at rising XCK edge. If UCPOL is set, the data will be changed at rising XCK edge and sampled at falling XCK edge.

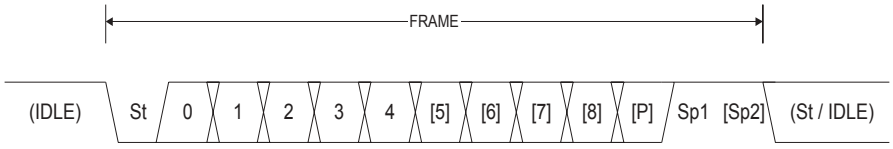
Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accept all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even, or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to a idle (high) state. Figure 48 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 48. Frame Formats



St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

Empty Interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) Flag bit is set one when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the Transmit Buffer. The TXC Flag bit is automatically cleared when a Transmit Complete Interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter Receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the Transmit Complete Interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the Transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register does not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD pin.

Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR, and PE Status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and PE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example(1)

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get status and ninth bit, then data from buffer
  in    r18, UCSRA
  in    r17, UCSRB
  in    r16, UDR
  ; If error, return -1
  andi  r18,(1<<FE)|(1<<DOR)|(1<<PE)
  breq  USART_ReceiveNoError
  ldi   r17, HIGH(-1)
  ldi   r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the ninth bit, then return
  lsr   r17
  andi  r17, 0x01
  ret
```

C Code Example(1)

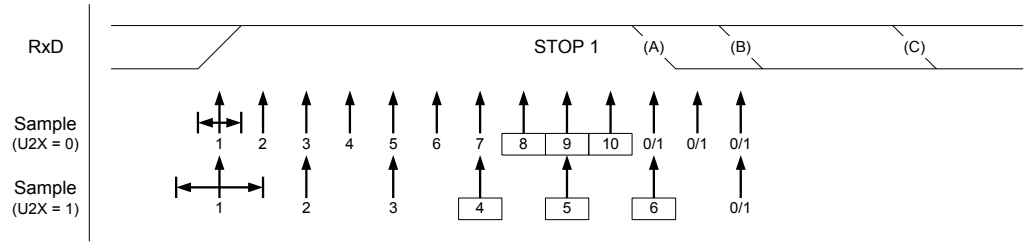
```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) ) {};
  /* Get status and ninth bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<PE) )
    return -1;
  /* Filter the ninth bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

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

The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process act as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the Receiver only uses the first stop bit of a frame.

Figure 51 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 51. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For normal speed mode, the first low level sample can be at point marked (A) in Figure 51. For double speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

Asynchronous Operational Range

The operational range of the Receiver is dependent of the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have exact base frequency, the Receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal Receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5 to 10 bit).

S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.

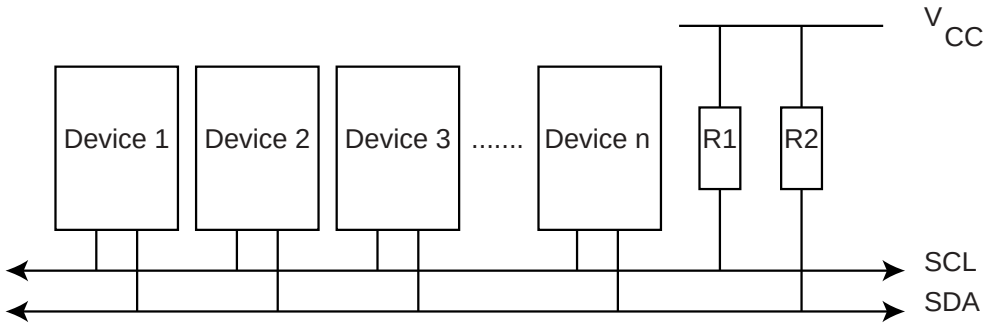
S_F First sample number used for majority voting. S_F = 8 for Normal Speed and S_F = 4 for Double Speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for Normal Speed and S_M = 5 for Double Speed mode.

Two-wire Serial Interface (Byte Oriented)

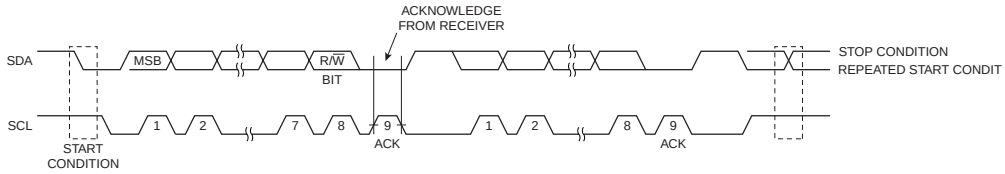
The Two-wire Serial Interface supports bi-directional serial communication. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. Various communication configurations can be designed using this bus. Figure 52 shows a typical Two-wire Serial Bus configuration. Any device connected to the bus can be Master or Slave. Note that all AVR devices connected to the bus must be powered to allow any bus operation.

Figure 52. Two-wire Serial Bus Configuration



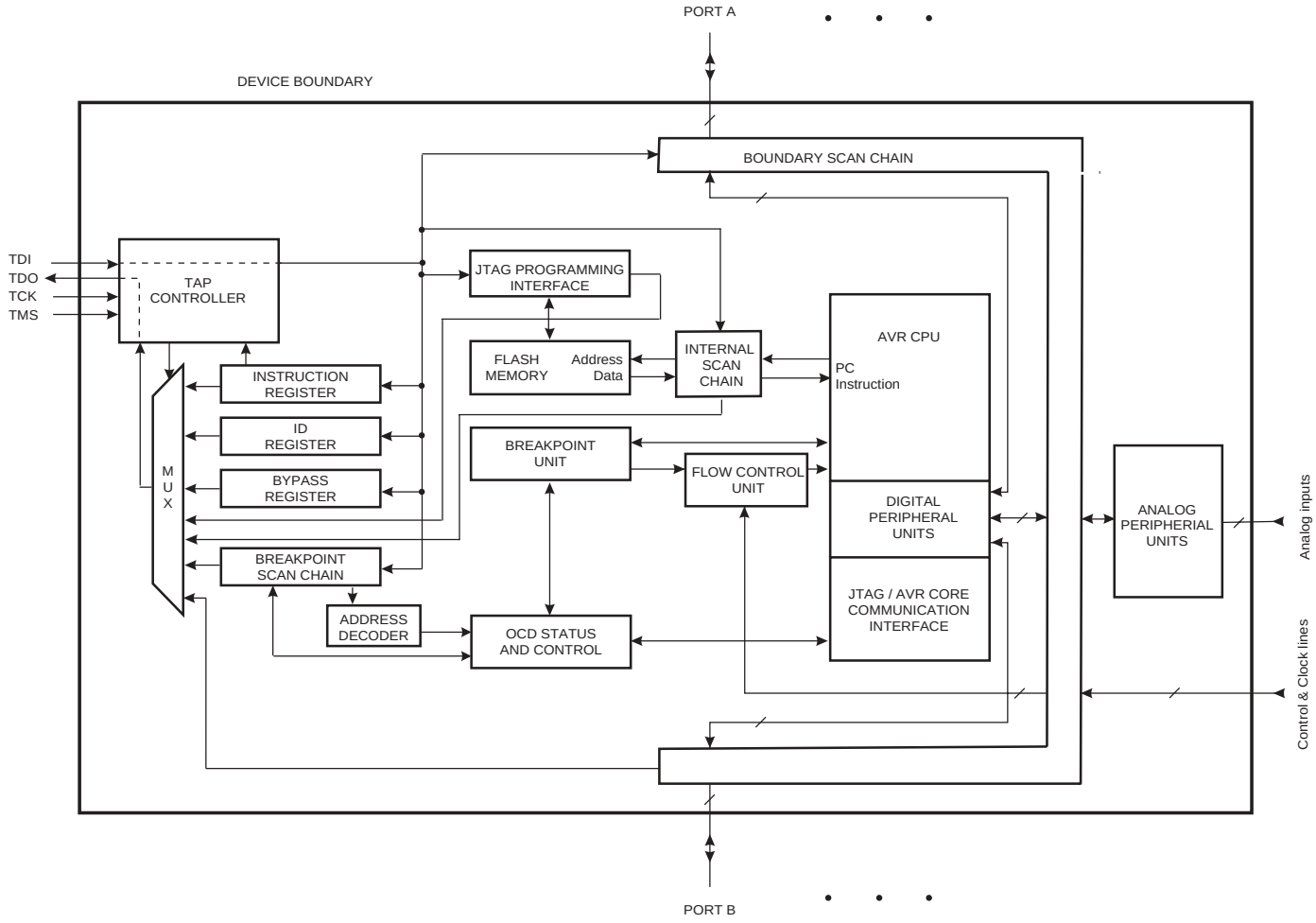
The Two-wire Serial Interface supports Master/Slave and Transmitter/Receiver operation at up to 400 kHz bus clock rate. The Two-wire Serial Interface has hardware support for 7-bit addressing. When the Two-wire Serial Interface is enabled (TWEN in TWCR is set), a glitch filter is enabled for the input signals from the pins PC0 (SCL) and PC1 (SDA), and the output from these pins is slew-rate controlled. The Two-wire Serial Interface is byte oriented. The operation of the Two-wire Serial Bus is shown as a pulse diagram in Figure 53, including the START and STOP conditions and generation of ACK signal by the bus Receiver.

Figure 53. Two-wire Serial Bus Timing Diagram



The block diagram of the Two-wire Serial Interface is shown in Figure 54.

Figure 85. Block Diagram



ATmega323 Boundary-Scan Order

Table 64 shows the Scan order between TDI and TDO when the Boundary-Scan chain is selected as data path. Bit 0 is the LSB; The first bit scanned in, and the first bit scanned out. The scan order follows the pinout order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 89, PXn.Data corresponds to FF0, PXn.Control corresponds to FF1, and PXn. Pullup_dissable corresponds to FF2. Bit 2, 3, 4, and 5 of Port C is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled

Table 58. ATmega323 Boundary-Scan Order

Bit Number	Signal Name	Module
131	SIG_PRIVATE0	Private Section 1
130	SIG_PRIVATE1	
129	SIG_PRIVATE2	
128	SIG_PRIVATE3	
127	SIG_PRIVATE4	Private Section 2
126	SIG_PRIVATE5	
125	SIG_PRIVATE6	
124	SIG_PRIVATE7	
123	SIG_PRIVATE8	
122	SIG_PRIVATE9	
121	SIG_PRIVATE10	
120	SIG_PRIVATE11	
119	SIG_PRIVATE12	
118	SIG_PRIVATE13	
117	SIG_PRIVATE14	
116	SIG_PRIVATE15	
115	SIG_PRIVATE16	
114	SIG_PRIVATE17	
113	SIG_PRIVATE18	
112	SIG_PRIVATE19	
111	SIG_PRIVATE20	
110	SIG_PRIVATE21	
109	SIG_PRIVATE22	
108	SIG_PRIVATE23	
107	SIG_PRIVATE24	
106	SIG_PRIVATE25	
105	SIG_PRIVATE26	

Perform a Page Write

To execute page write, set up the address in the Z-pointer, write “00101” to the five LSB in 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 Z14:Z7. During this operation, Z6:Z0 must be zero to ensure that the page is written correctly. It is recommended that the interrupts are disabled during the page write operation.

Consideration while Updating the Boot Loader Section

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit 11 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 bit 11 to protect the Boot Loader software from any internal software changes.

Wait for SPM Instruction to Complete

Though the CPU is halted during Page Write, Page Erase or Lock bit write, for future compatibility, the user software must poll for SPM complete by reading the SPMCR Register and loop until the SPMEN bit is cleared after a programming operation. See “Assembly code example for a Boot Loader” on page 185 for a code example.

Instruction Word Read after Page Erase, Page Write, and Lock Bit Write

To ensure proper instruction pipelining after programming action (Page Erase, Page Write, or Lock bit write), the SPM instruction must be followed with the sequence (.dw $FFFF - NOP) as shown below:

```
spm
.dw $FFFF
nop
```

If not, the instruction following SPM might fail. It is not necessary to add this sequence when the SPM instruction only loads the temporary buffer.

Avoid Reading the Application Section During Self-programming

During Self-programming (either Page Erase or Page Write), the user software should not read the application section. The user software itself must prevent addressing this section during the Self-programming operations. This implies that interrupts must be disabled or moved to the Boot Loader section. Before addressing the application section after the programming is completed, for future compatibility, the user software must write”10001” to the five LSB in SPMCR and execute SPM within four clock cycles. Then the user software should verify that the ASB bit is cleared. See “Assembly code example for a Boot Loader” on page 185 for an example. Though the ASB and ASRE bits have no special function in this device, it is important for future code compatibility that they are treated as described above.

Boot Loader Lock Bits

ATmega323 has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU.
- To only protect the Boot Loader Flash section from a software update by the MCU.
- To only protect application Flash section from a software update by the MCU.
- Allowing software update in the entire Flash.

See Table 61 for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can only be cleared by a chip erase command.

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 $0001 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within five CPU cycles after the BLBSET and SPMEN bits are set in SPMCR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no SPM, or LPM, instruction is executed within four, respectively five, CPU cycles. When BLBSET and SPMEN are cleared, LPM will work as described in "Constant Addressing Using the LPM and SPM Instructions" on page 16 and 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 $0000 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within five cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse Low bits will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	BODLEVEL	BODEN	–	–	CKSEL3	CKSEL2	CKSEL1	CKSEL0

Similarly, when reading the Fuse High bits, load $0003 in the Z-pointer. When an LPM instruction is executed within five cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse High bits will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	OCDEN	JTAGEN	SPIEN	–	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST

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

In all cases, the read value of unused bit positions are undefined.

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 (EEWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCR Register. If EEPROM writing is performed inside an interrupt routine, the user software should disable that interrupt before checking the EEWE status bit.

Addressing the Flash During Self-programming

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

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

Z15 always ignored

Z14:Z7 page select, for page erase and page write

Z6:Z1 word select, for filling temp buffer (must be zero during page write operation)

Z0 should be zero for all SPM commands, byte select for the LPM instruction.

The only operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation.

Table 63. Lock Bit Protection Modes

Memory Lock Bits			
LB mode	**LB2**	**LB1**	**Protection Type**
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. Program the Fuse bits before programming the Lock bits.

Fuse Bits

The ATmega323 has 13 Fuse bits, divided in two groups. The Fuse High bits are OCDEN, JTAGEN, SPIEN, EESAVE, BOOTSZ1..0, and BOOTRST, and the Fuse Low bits are BODLEVEL, BODEN, and CKSEL3..0. All Fuses are accessible in Parallel Programming mode and when programming via the JTAG interface. In Serial Programming mode, all but the SPIEN Fuse is accessible.

- When the OCDEN Fuse is programmed, the On-chip debug system is enabled if the JTAGEN Fuse is programmed. If the JTAGEN Fuse is unprogrammed, the OCDEN Fuse has no visible effect.Never ship a product with the OCDEN Fuse programmed. Regardless of the setting of Lock bits and the JTAGEN Fuse, a programmed OCDEN Fuse enables some parts of the clock system be running in all sleep modes. This may increase the power consumption. Default value is unprogrammed (“1”).
- When the JTAGEN Fuse is programmed, the JTAG interface is enabled on port C pins PC5..2. Default value is programmed (“0”).
- If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.
- When the SPIEN Fuse is programmed (“0”), Serial Program and Data Downloading are enabled. Default value is programmed (“0”). The SPIEN Fuse is not accessible in SPI Serial Programming mode.
- When EESAVE is programmed, the EEPROM Memory is preserved through the Chip Erase cycle. Default value is unprogrammed (“1”). The EESAVE Fuse bit can not be programmed if any of the Lock bits are programmed.
- BOOTSZ1..0 select the size and start address of the Boot Flash section according to Table on page 177. Default value is “11” (both unprogrammed).
- When BOOTRST is programmed (“0”), the Reset Vector is set to the start address of the Boot Flash section, as selected by the BOOTSZ Fuses according to Table 59 on page 177. If the BOOTRST is unprogrammed (“1”), the Reset Vector is set to address $0000. Default value is unprogrammed (“1”).
- The BODLEVEL Fuse selects the Brown-out Detection Level and changes the Start-up times, according to Table 5 on page 26 and Table 6 on page 27, respectively. Default value is unprogrammed (“1”).

Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address $00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address $01 and address $02 to read the second and third signature bytes, respectively.

Reading the Calibration Byte

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address $00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

Figure 115. Idle Supply Current vs. V_{CC}, Device Clocked by External 32kHz Crystal

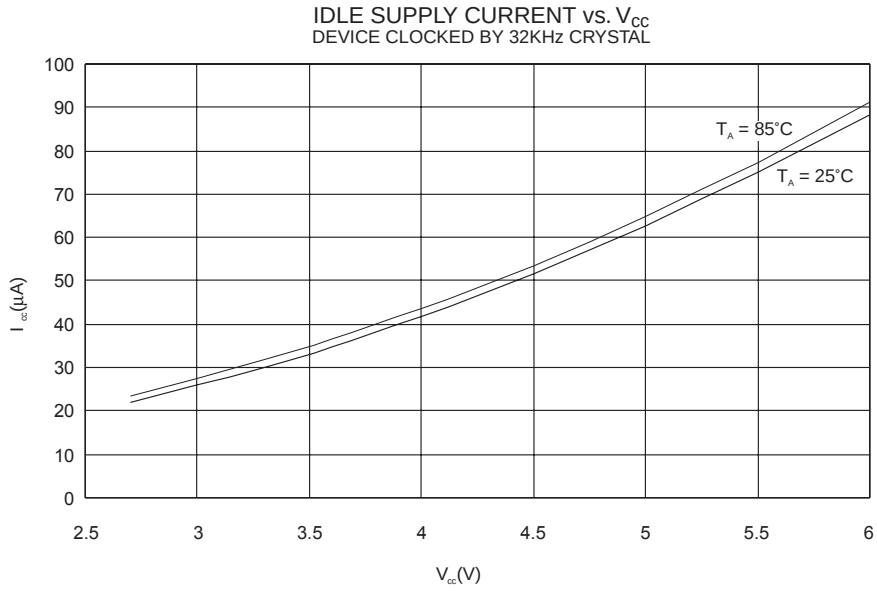


Figure 116. Power-down Supply Current vs. V_{CC}

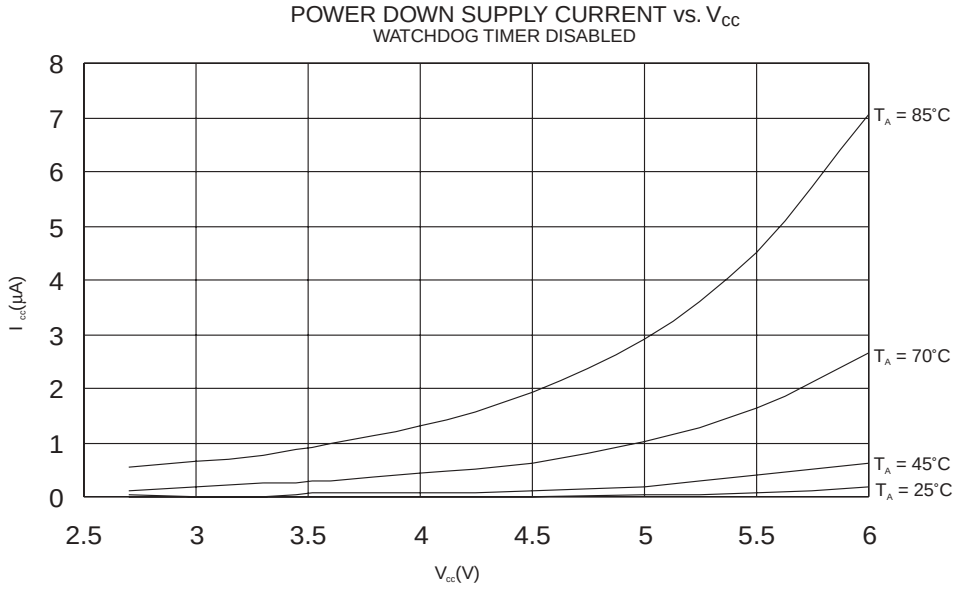


Figure 125. Pull-Up Resistor Current vs. Input Voltage

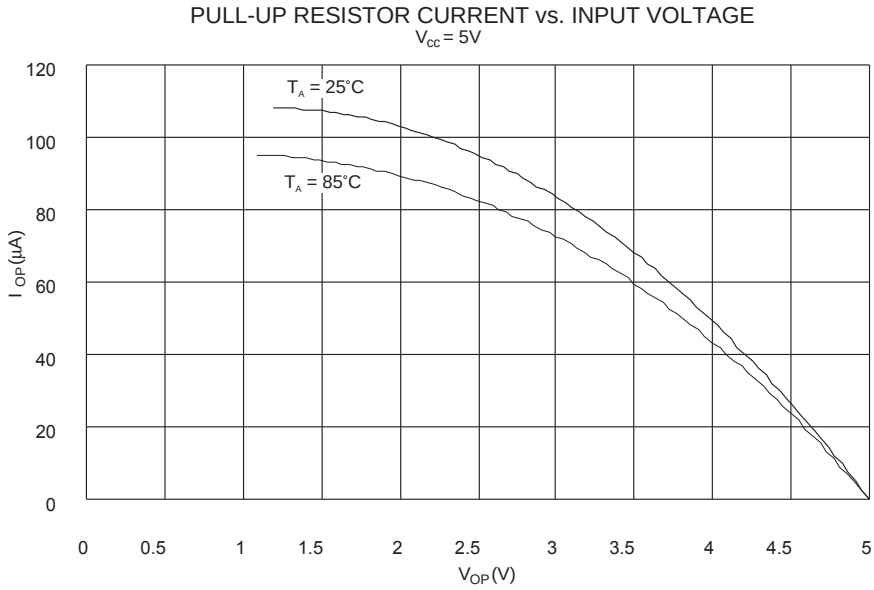
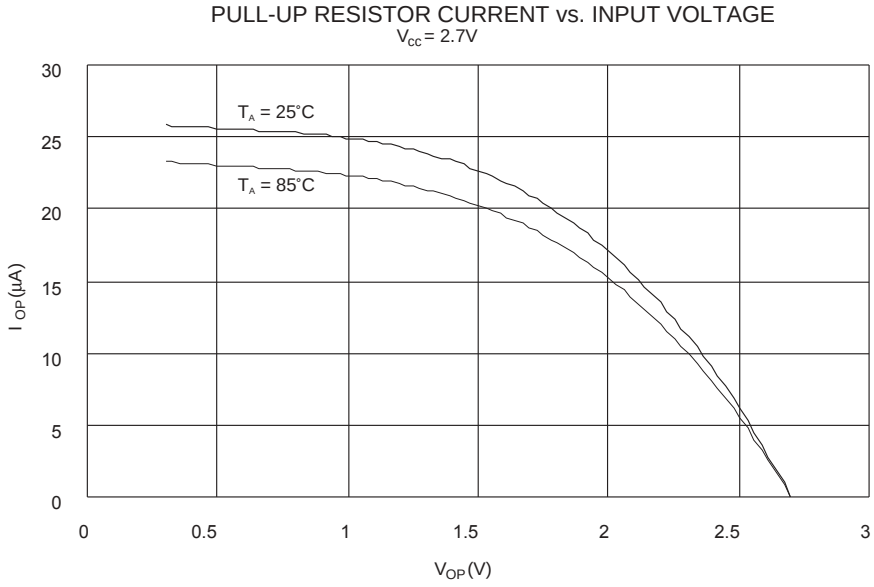


Figure 126. Pull-Up Resistor Current vs. Input Voltage



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	page 21
$3E ($5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	page 22
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 22
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								page 47
$3B ($5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	page 33
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	page 34
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	page 36
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	page 36
$37 ($57)	SPMCR	–	ASB	–	ASRE	BLBSET	PGWRT	PGERS	SPMEN	page 183
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	page 104
$35 ($55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	page 37
$34 ($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	page 30
$33 ($53)	TCCR0	FOC0	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	page 47
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								page 49
$31 ($51)	OSCCAL	Oscillator Calibration Register								page 41
	OCRD	On-chip Debug Register								page 161
$30 ($50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	page 45
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM11	PWM10	page 56
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	–	CTC1	CS12	CS11	CS10	page 57
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								page 58
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								page 58
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								page 59
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								page 59
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								page 59
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								page 59
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								page 60
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								page 60
$25 ($45)	TCCR2	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	page 47
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								page 49
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								page 49
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	page 52
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	page 64
$20 ($40)	UBRRH	URSEL	–	–	–	UBRR[11:8]				page 98
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	page 97
$1F ($3F)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	page 66
$1E ($3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	page 66
$1D ($3D)	EEDR	EEPROM Data Register								page 66
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	page 67
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	page 137
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	page 137
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	page 137
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 139
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 139
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 139
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	page 146
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	page 146
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	page 146
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	page 151
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	page 151
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	page 151
$0F ($2F)	SPDR	SPI Data Register								page 73
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	page 72
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 71
$0C ($2C)	UDR	USART I/O Data Register								page 94
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	page 94
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	page 96
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								page 98
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	page 125
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	page 132
$06 ($26)	ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 133
$05 ($25)	ADCH	ADC Data Register High Byte								page 134
$04 ($24)	ADCL	ADC Data Register Low Byte								page 134
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								page 106
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	page 107