Microchip Technology - ATMEGA165V-8AI Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

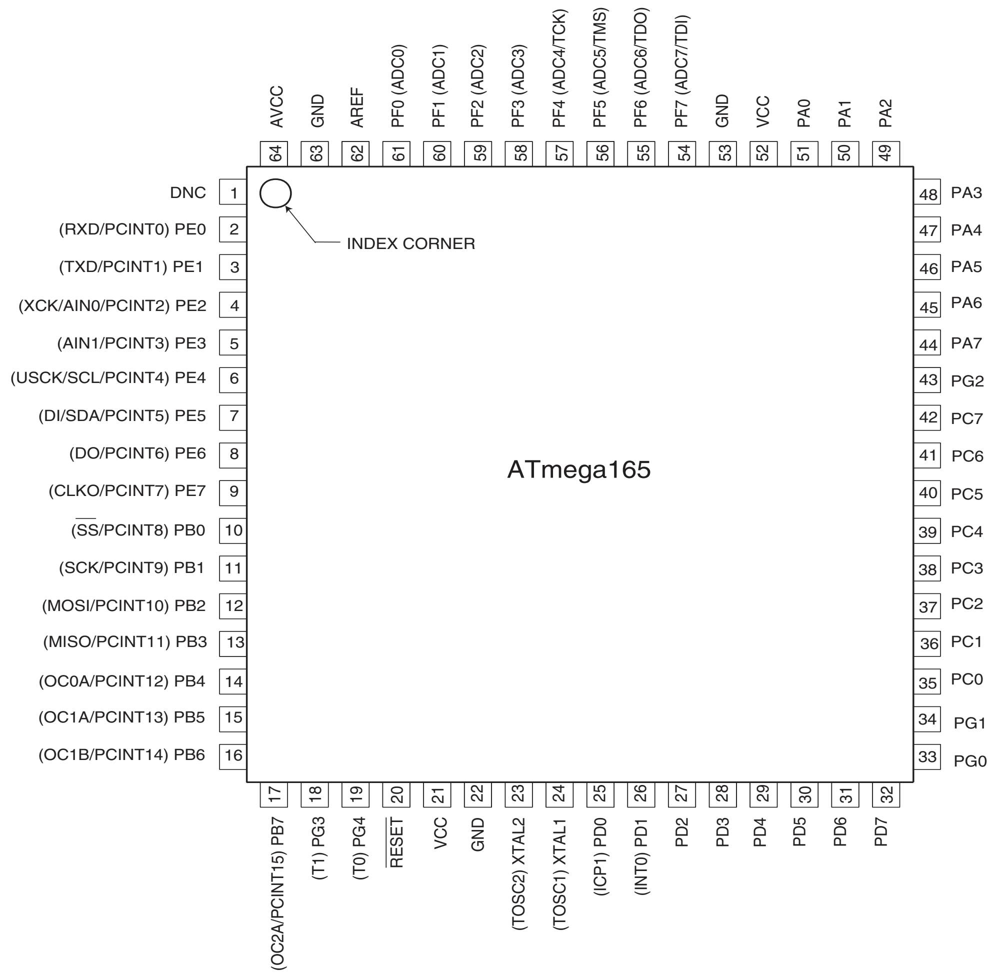
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	8MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	54
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega165v-8ai

Pin Configurations

Figure 1. Pinout ATmega165



Note: The large center pad underneath the QFN/MLF packages is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.

Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Table 13. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

Switching Time

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

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

Standby Mode

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

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

	Active Clock Domains					Oscillators		Wake-up Sources					
Sleep Mode	**clk_{CPU}**	**clk_{FLASH}**	**clk_{IO}**	**clk_{ADC}**	**clk_{ASY}**	**Main Clock Source Enabled**	**Timer Osc Enabled**	**INT0 and Pin Change**	**USI Start Condition**	**Timer2**	**SPM/ EEPROM Ready**	**ADC**	**Other I/O**
Idle			X	X	X	X	X[2]	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X[2]	X	X	
Power-down								X[3]	X				
Power-save					X		X	X[3]	X	X			
Standby[1]						X		X[3]	X				

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

Power Reduction Register

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

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

Power Reduction Register - PRR

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

- **Bit 7..4 - Res: Reserved bits**

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

- **Bit 3 - PRTIM1: Power Reduction Timer/Counter1**

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

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

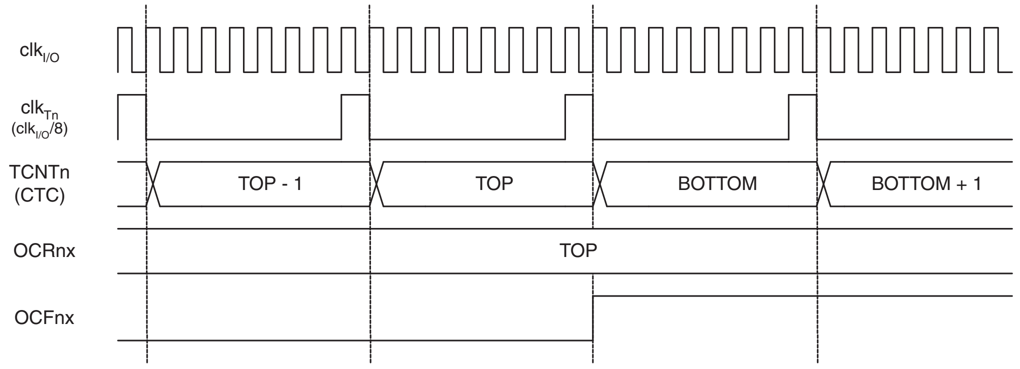
```
Address   Labels Code                     Comments
0x0000    RESET: ldi     r16,high(RAMEND) ; Main program start
0x0001           out     SPH,r16          ; Set Stack Pointer to top of RAM
0x0002           ldi     r16,low(RAMEND)
0x0003           out     SPL,r16
0x0004           sei                      ; Enable interrupts
0x0005           <instr>  xxx
;
.org 0x1C02
0x1C02           jmp     EXT_INT0         ; IRQ0 Handler
0x1C04           jmp     PCINT0           ; PCINT0 Handler
...              ...     ...              ;
0x1C2C           jmp     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:

```
Address   Labels Code               Comments
.org 0x0002
0x0002           jmp     EXT_INT0         ; IRQ0 Handler
0x0004           jmp     PCINT0           ; PCINT0 Handler
...              ...     ...              ;
0x002C           jmp     SPM_RDY          ; Store Program Memory Ready Handler
;
.org 0x1C00
0x1C00    RESET: ldi     r16,high(RAMEND) ; Main program start
0x1C01           out     SPH,r16          ; Set Stack Pointer to top of RAM
0x1C02           ldi     r16,low(RAMEND)
0x1C03           out     SPL,r16
0x1C04           sei                      ; Enable interrupts
0x1C05           <instr>  xxx
```

Figure 37 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode.

Figure 37. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)



8-bit Timer/Counter Register Description

Timer/Counter Control Register A – TCCR0A

Bit	7	6	5	4	3	2	1	0	
	FOC0A	**WGM00**	**COM0A1**	**COM0A0**	**WGM01**	**CS02**	**CS01**	**CS00**	**TCCR0**
Read/Write	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 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM00 bit specifies a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate compare match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bit 6, 3 – WGM01:0: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 42 and "Modes of Operation" on page 80.

Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using "C", the compiler handles the 16-bit access.

Assembly Code Examples[1]

```
...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in  r16,TCNT1L
in  r17,TCNT1H
...
```

C Code Examples[1]

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

Note: 1. See "About Code Examples" on page 6.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 60 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 60. Timer/Counter Timing Diagram, no Prescaling

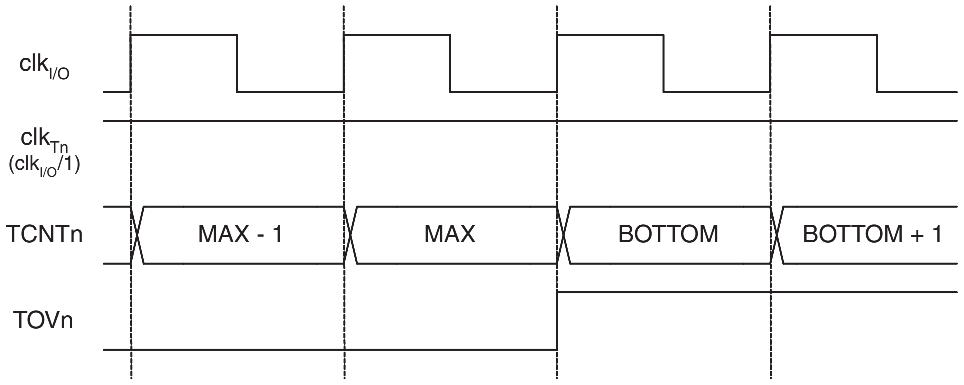


Figure 61 shows the same timing data, but with the prescaler enabled.

Figure 61. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

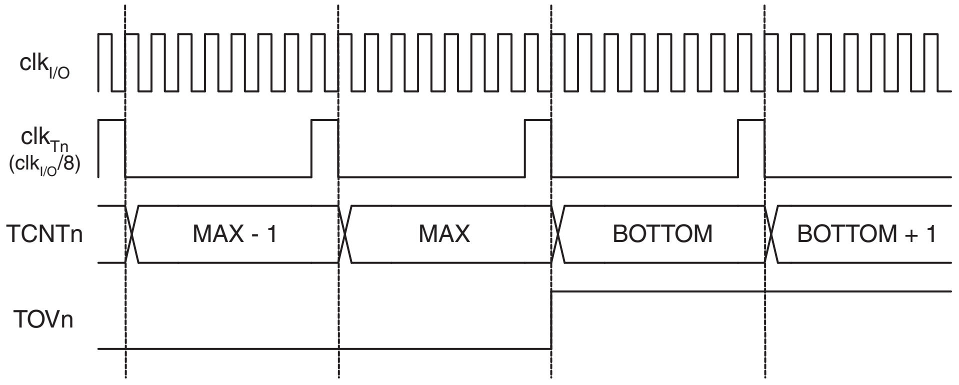


Figure 62 shows the setting of OCF2A in all modes except CTC mode.

Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register 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 do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD pin.

•Bit 5 – UDRE: USART Data Register Empty

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the Transmitter is ready.

• Bit 4 – FE: Frame Error

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

• Bit 3 – DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. 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. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

• Bit 2 – UPE: USART Parity Error

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

• Bit 1 – U2X: Double the USART Transmission Speed

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

• Bit 0 – MPCM: Multi-processor Communication Mode

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 165.

Start Condition Detector

The start condition detector is shown in Figure 81. The SDA line is delayed (in the range of 50 to 300 ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in Two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the Power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the Oscillator start-up time set by the CKSEL Fuses (see "Clock Systems and their Distribution" on page 23) must also be taken into the consideration. Refer to the USISIF bit description on page 182 for further details.

Clock speed considerations.

Maximum frequency for SCL and SCK is f_{CK} /4. This is also the maximum data transmit and receieve rate in both two- and three-wire mode. In two-wire slave mode the Two-wire Clock Control Unit will hold the SCL low until the slave is ready to receive more data. This may reduce the actual data rate in two-wire mode.

Alternative USI Usage

When the USI unit is not used for serial communication, it can be set up to do alternative tasks due to its flexible design.

Half-duplex Asynchronous Data Transfer

By utilizing the Shift Register in Three-wire mode, it is possible to implement a more compact and higher performance UART than by software only.

4-bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will generate an increment.

12-bit Timer/Counter

Combining the USI 4-bit counter and Timer/Counter0 allows them to be used as a 12-bit counter.

Edge Triggered External Interrupt

By setting the counter to maximum value (F) it can function as an additional external interrupt. The Overflow Flag and Interrupt Enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

USI Register Descriptions

USI Data Register – USIDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	USIDR
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	

The USI uses no buffering of the Serial Register, i.e., when accessing the Data Register (USIDR) the Serial Register is accessed directly. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending of the USICS1..0 bits setting. The shift operation can be controlled by an external clock edge, by a Timer/Counter0 Compare Match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM1..0 = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the Shift Register.

The output pin in use, DO or SDA depending on the wire mode, is connected via the output latch to the most significant bit (bit 7) of the Data Register. The output latch is open

JTAG Interface and On-chip Debug System

Features

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard**
- **Debugger Access to:**
  - **–All Internal Peripheral Units**
  - **–Internal and External RAM**
  - **–The Internal Register File**
  - **–Program Counter**
  - **–EEPROM and Flash Memories**
- **Extensive On-chip Debug Support for Break Conditions, Including**
  - **–AVR Break Instruction**
  - **–Break on Change of Program Memory Flow**
  - **–Single Step Break**
  - **–Program Memory Break Points on Single Address or Address Range**
  - **–Data Memory Break Points on Single Address or Address Range**
- **Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface**
- **On-chip Debugging Supported by AVR Studio®**

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- Testing PCBs by using the JTAG Boundary-scan capability
- Programming the non-volatile memories, Fuses and Lock bits
- On-chip debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-scan Chain can be found in the sections "Programming via the JTAG Interface" on page 266 and "IEEE 1149.1 (JTAG) Boundary-scan" on page 212, respectively. The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 97 shows a block diagram of the JTAG interface and the On-chip Debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (Shift Register) between the TDI – input and TDO – output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the Data Registers used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for serial programming via the JTAG interface. The Internal Scan Chain and Break Point Scan Chain are used for On-chip debugging only.

Test Access Port – TAP

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port – TAP. These pins are:

- TMS: Test mode select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK.
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains).
- TDO: Test Data Out. Serial output data from Instruction Register or Data Register.

On-chip Debug Related Register in I/O Memory

On-chip Debug Register – OCDR

Bit	7	6	5	4	3	2	1	0	
	MSB/IDRD							LSB	OCDR
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	

The OCDR Register provides a communication channel from the running program in the microcontroller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an internal flag; I/O Debug Register Dirty – IDRD – is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information.

In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

Refer to the debugger documentation for further information on how to use this register.

Using the JTAG Programming Capabilities

Programming of AVR parts via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI, and TDO. These are the only pins that need to be controlled/observed to perform JTAG programming (in addition to power pins). It is not required to apply 12V externally. The JTAGEN Fuse must be programmed and the JTD bit in the MCUCR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- Flash programming and verifying.
- EEPROM programming and verifying.
- Fuse programming and verifying.
- Lock bit programming and verifying.

The Lock bit security is exactly as in parallel programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section "Programming via the JTAG Interface" on page 266.

Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993.
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; 0xF

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic "0" into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

Boundary-scan Related Register in I/O Memory

MCU Control Register – MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	JTD	**–**	**–**	**PUD**	**–**	**–**	**IVSEL**	**IVCE**	**MCUCR**
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – JTD: JTAG Interface Disable**

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

If the JTAG interface is left unconnected to other JTAG circuitry, the JTD bit should be set to one. The reason for this is to avoid static current at the TDO pin in the JTAG interface.

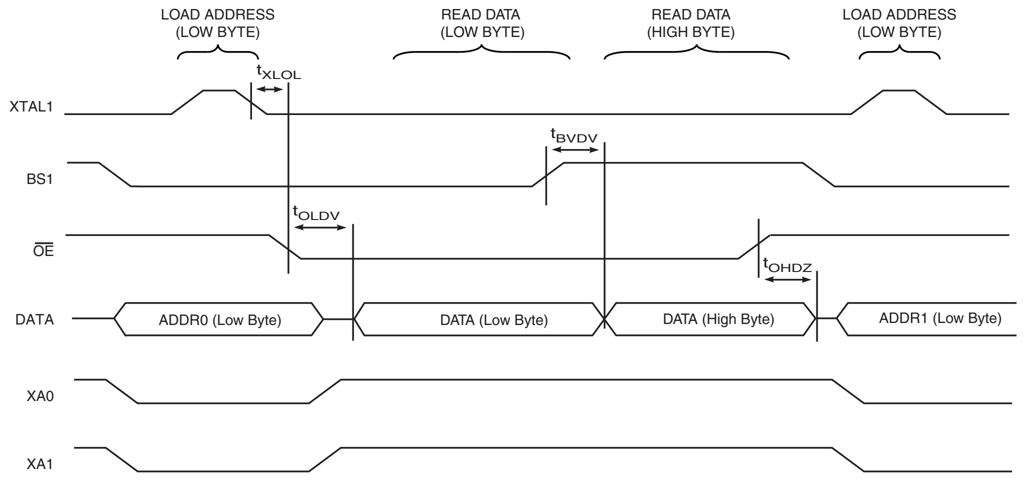
Table 88. Boundary-scan Signals for the ADC[1]

Signal Name	Direction as Seen from the ADC	Description	Recommen-ded Input when not in Use	Output Values when Recommended Inputs are Used, and CPU is not Using the ADC
COMP	Output	Comparator Output	0	0
ACLK	Input	Clock signal to differential amplifier implemented as Switch-cap filters	0	0
ACTEN	Input	Enable path from differential amplifier to the comparator	0	0
ADCBGEN	Input	Enable Band-gap reference as negative input to comparator	0	0
ADCEN	Input	Power-on signal to the ADC	0	0
AMPEN	Input	Power-on signal to the differential amplifier	0	0
DAC_9	Input	Bit 9 of digital value to DAC	1	1
DAC_8	Input	Bit 8 of digital value to DAC	0	0
DAC_7	Input	Bit 7 of digital value to DAC	0	0
DAC_6	Input	Bit 6 of digital value to DAC	0	0
DAC_5	Input	Bit 5 of digital value to DAC	0	0
DAC_4	Input	Bit 4 of digital value to DAC	0	0
DAC_3	Input	Bit 3 of digital value to DAC	0	0
DAC_2	Input	Bit 2 of digital value to DAC	0	0
DAC_1	Input	Bit 1 of digital value to DAC	0	0
DAC_0	Input	Bit 0 of digital value to DAC	0	0
EXTCH	Input	Connect ADC channels 0 - 3 to by-pass path around differential amplifier	1	1
GNDEN	Input	Ground the negative input to comparator when true	0	0

Table 90. ATmega165 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
132	PB0.Control	Port B
131	PB0.Pull-up_Enable	
130	PB1.Data	
129	PB1.Control	
128	PB1.Pull-up_Enable	
127	PB2.Data	
126	PB2.Control	
125	PB2.Pull-up_Enable	
124	PB3.Data	
123	PB3.Control	
122	PB3.Pull-up_Enable	
121	PB4.Data	
120	PB4.Control	
119	PB4.Pull-up_Enable	
118	PB5.Data	
117	PB5.Control	
116	PB5.Pull-up_Enable	
115	PB6.Data	
114	PB6.Control	
113	PB6.Pull-up_Enable	
112	PB7.Data	
111	PB7.Control	
110	PB7.Pull-up_Enable	
109	PG3.Data	Port G
108	PG3.Control	
107	PG3.Pull-up_Enable	
106	PG4.Data	
105	PG4.Control	
104	PG4.Pull-up_Enable	
103	PG5	(Observe Only)
102	RSTT	Reset Logic (Observe-only)
101	RSTHV	
100	EXTCLKEN	Enable signals for main Clock/Oscillators
99	OSCON	
98	RCOSCEN	
97	OSC32EN	

Figure 119. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[(1)]



Note: 1. The timing requirements shown in Figure 117 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 112. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	μA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to XTAL1 high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	μs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[(1)]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[(2)]	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns

Programming the Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 112 on page 259).

Reading the Fuses and Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse/Lock bit read using programming instruction 8a.
3. To read all Fuses and Lock bits, use programming instruction 8e.
   To only read Fuse High byte, use programming instruction 8b.
   To only read Fuse Low byte, use programming instruction 8c.
   To only read Lock bits, use programming instruction 8d.

Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address 0x00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address 0x01 and address 0x02 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 0x00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

Pin Thresholds and hysteresis

Figure 171. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as “1”)

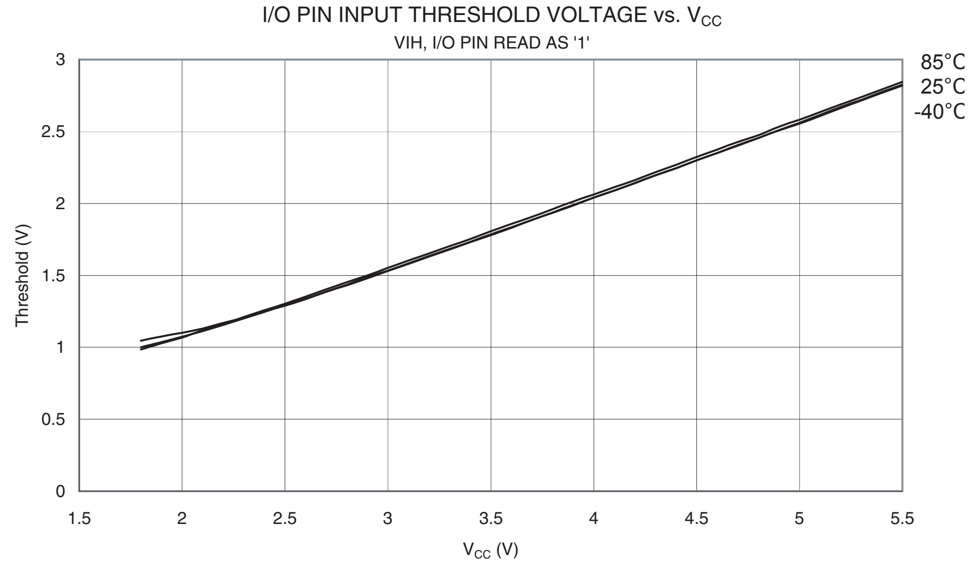
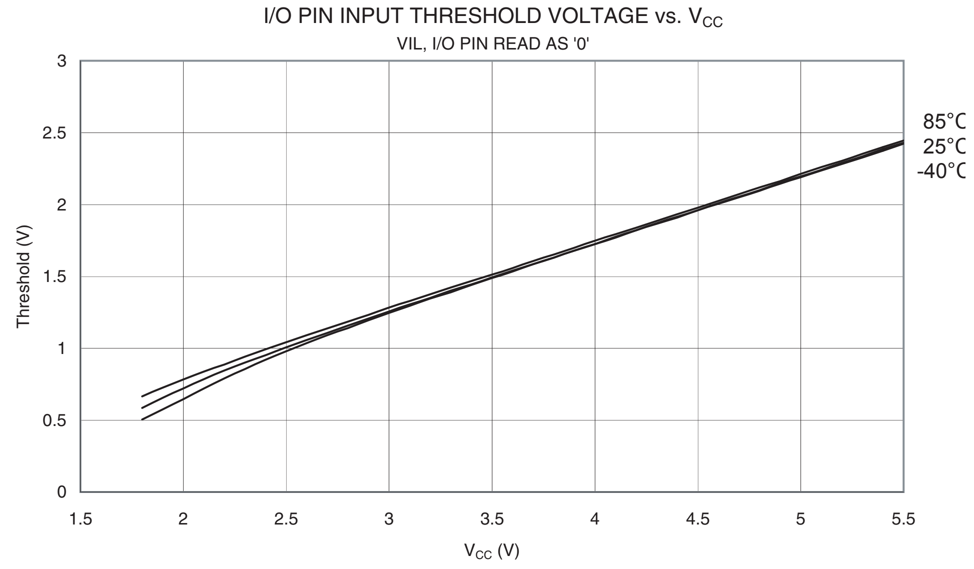


Figure 172. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as “0”)



Current Consumption of Peripheral Units

Figure 187. Brownout Detector Current vs. V_{CC}

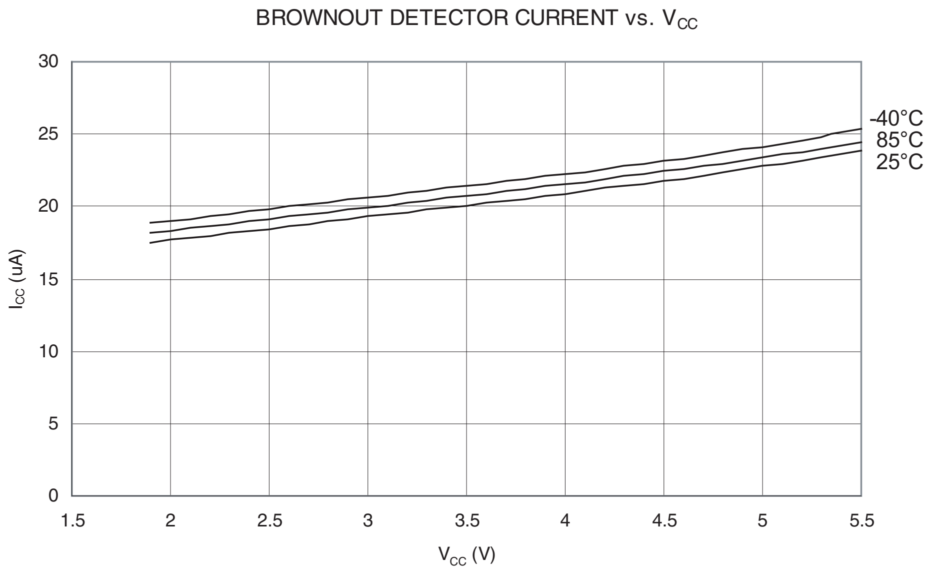


Figure 188. ADC Current vs. V_{CC} (AREF = AVCC)

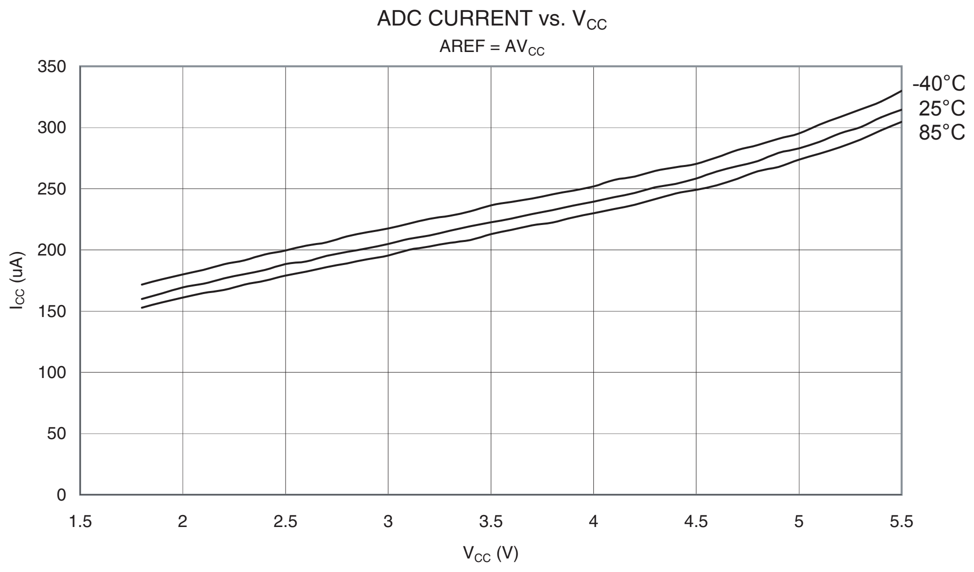
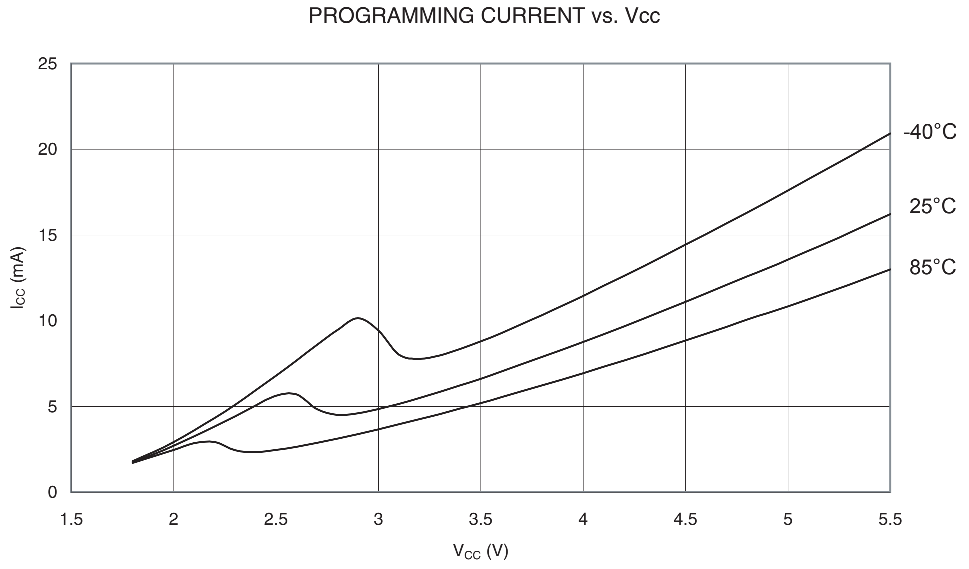
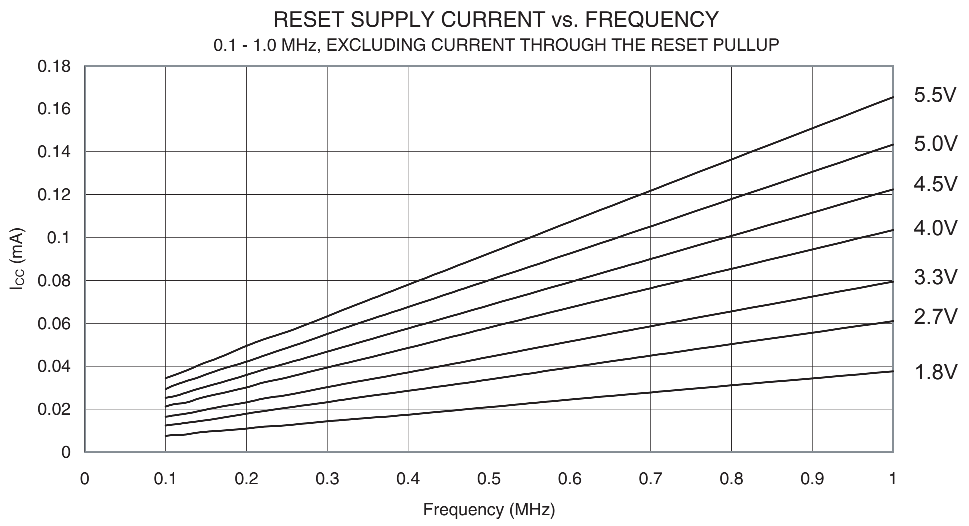


Figure 193. Programming Current vs. V_{CC}



Current Consumption in Reset and Reset Pulsewidth

Figure 194. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current Through The Reset Pull-up)


RESET SUPPLY CURRENT vs. FREQUENCY
0.1 - 1.0 MHz, EXCLUDING CURRENT THROUGH THE RESET PULLUP
ICC (mA)
Frequency (MHz)
5.5V
5.0V
4.5V
4.0V
3.3V
2.7V
1.8V