**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 | 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) | 4.5V ~ 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/atmega323-8ai |

### The Stack Pointer – SP

The ATmega323 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the ATmega323 Data memory has $860 locations, 12 bits are used.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $3E ($5E) | – | – | – | – | SP11 | SP10 | SP9 | SP8 | SPH |
| $3D ($5D) | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SPL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | 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 Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

## Reset and Interrupt Handling

The ATmega323 provides nineteen different interrupt sources. These interrupts and the separate Reset Vector, each have a separate Program Vector in the Program memory space. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be disabled when Boot Lock bits BLB02 or BLB12 are set. See the section “Boot Loader Support” on page 177 for details

The lowest addresses in the Program memory space are automatically defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in Table 3. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0, etc. The Interrupt Vectors can be moved to the start of the boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). See the GICR description on page 33 for details..

**Table 3.** Reset and Interrupt Vectors

| Vector No. | Program Address(2) | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $000(1) | RESET | External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset |
| 2 | $002 | INT0 | External Interrupt Request 0 |
| 3 | $004 | INT1 | External Interrupt Request 1 |
| 4 | $006 | INT2 | External Interrupt Request 2 |
| 5 | $008 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 6 | $00A | TIMER2 OVF | Timer/Counter2 Overflow |
| 7 | $00C | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 8 | $00E | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 9 | $010 | TIMER1 COMPB | Timer/Counter1 Compare Match B |

### The EEPROM Control Register – EECR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $1C ($3C) | – | – | – | – | EERIE | EEMWE | EEWE | EERE | EECR |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | |

- **Bits 7..4 – Res: Reserved Bits**

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

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

When the I bit in SREG and EERIE are set (one), the EEPROM Ready Interrupt is enabled. When cleared (zero), the interrupt is disabled. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared (zero).

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is not essential):

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

**Caution:** An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during the 4 last steps to avoid these problems.

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

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR Register. The EEPROM read access takes one instruction, and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for four cycles before the next instruction is executed.

### The Two-wire Serial Interface (Slave) Address Register – TWAR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $02 ($22) | TWA6 | TWA5 | TWA4 | TWA3 | TWA2 | TWA1 | TWA0 | TWGCE | TWAR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |

- **Bits 7..1 – TWA: Two-wire Serial Interface (Slave) Address Register**

These seven bits constitute the slave address of the Two-wire Serial Bus unit.

- **Bit 0 – TWGCE: Two-wire Serial Interface General Call Recognition Enable bit**

This bit enables, if set, the recognition of the General Call given over the Two-wire Serial Bus.

The TWAR should be loaded with the 7-bit slave address (in the seven most significant bits of TWAR) to which the Two-wire Serial Interface will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. The LSB of TWAR is used to enable recognition of the general call address ($00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

## Two-wire Serial Interface Modes

The Two-wire Serial Interface can operate in four different modes:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfer in each mode of operation is shown in Figure 55 to Figure 58. These figures contain the following abbreviations:

S: START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\overline{A}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

In Figure 55 to Figure 58, circles are used to indicate that the Two-wire Serial Interface Interrupt Flag is set. The numbers in the circles show the status code held in TWSR. At these points, actions must be taken by the application to continue or complete the Two-wire Serial Bus transfer. The Two-wire Serial Bus transfer is suspended until the Two-wire Serial Interface Interrupt Flag is cleared by software.

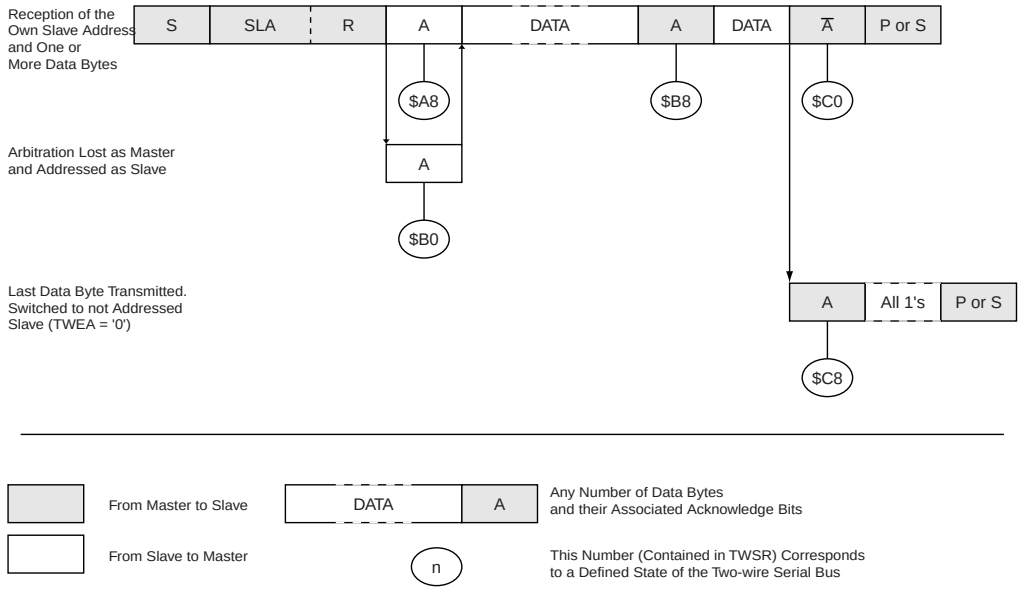
The Two-wire Serial Interface Interrupt Flag is not automatically cleared by hardware when executing the interrupt routine. Software has to clear the flag to continue the Two-wire transfer. Also note that the Two-wire Serial Interface starts execution as soon as this bit is cleared, so that all access to TWAR, TWDR, and TWSR must have been completed before clearing this flag.

```
        ldi   r16, 0xc8        ; Load SLA+W into TWDR Register
        out   TWDR, r16
        ldi   r16, (1<<TWINT) | (1<<TWEN)
        out   TWCR, r16        ; Clear TWINT bit in TWCR to start transmission
                               ; of address

wait2:  in    r16, TWCR        ; Wait for TWINT Flag set. This indicates that
        sbrs  r16, TWINT       ; SLA+W has been transmitted, and ACK/NACK has
        rjmp  wait2            ; been received

        in    r16, TWSR        ; Check value of TWI Status Register. If status
        cpi   r16, MT_SLA_ACK  ; different from MT_SLA_ACK, go to ERROR
        brne  ERROR

        ldi   r16, 0x33        ; Load data (here, data=0x33) into TWDR
                               ; Register
        out   TWDR, r16
        ldi   r16, (1<<TWINT) | (1<<TWEN)
        out   TWCR, r16        ; Clear TWINT bit in TWCR to start transmission
                               ; of data

wait3:  in    r16, TWCR        ; Wait for TWINT Flag set. This indicates that
        sbrs  r16, TWINT       ; data has been transmitted, and ACK/NACK has
        rjmp  wait3            ; been received

        in    r16, TWSR        ; Check value of TWI Status Register. If status
        cpi   r16, MT_DATA_ACK ; different from MT_DATA_ACK, go to ERROR
        brne  ERROR

        ldi   r16, 0x44        ; Load data (here, data = 0x44) into TWDR
                               ; Register
        out   TWDR, r16
        ldi   r16, (1<<TWINT) | (1<<TWEN)
        out   TWCR, r16        ; Clear TWINT bit in TWCR to start transmission
                               ; of data

;<send more data bytes if needed>

wait4:  in    r16, TWCR        ; Wait for TWINT Flag set. This indicates that
        sbrs  r16, TWINT       ; data has been transmitted, and ACK/NACK has
        rjmp  wait4            ; been received

        in    r16, TWSR        ; Check value of TWI Status Register. If status
        cpi   r16, MT_DATA_ACK ; different from MT_DATA_ACK, go to ERROR
        brne  ERROR

        ldi   r16, (1<<TWINT) | (1<<TWSTO) | (1<<TWEN)
        out   TWCR, r16        ; Transmit STOP condition
```

**Table 40.** Miscellaneous States

| Status Code (TWSR) | Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware | Application Software Response | | | | | Next Action Taken by Two-wire Serial Interface Hardware |
|---|---|---|---|---|---|---|---|
| | | To/from TWDR | To TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| $A8 | Own SLA+R has been received; ACK has been returned | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $B0 | Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $B8 | Data byte in TWDR has been transmitted; ACK has been received | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $C0 | Data byte in TWDR has been transmitted; NOT ACK has been received | No TWDR action or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | No TWDR action or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1" |
| | | No TWDR action or | 1 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free |
| | | No TWDR action | 1 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free |
| $C8 | Last data byte in TWDR has been transmitted (TWEA = "0"); ACK has been received | No TWDR action or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | No TWDR action or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1" |
| | | No TWDR action or | 1 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free |
| | | No TWDR action | 1 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free |

**Figure 58.** Formats and States in the Slave Transmitter Mode



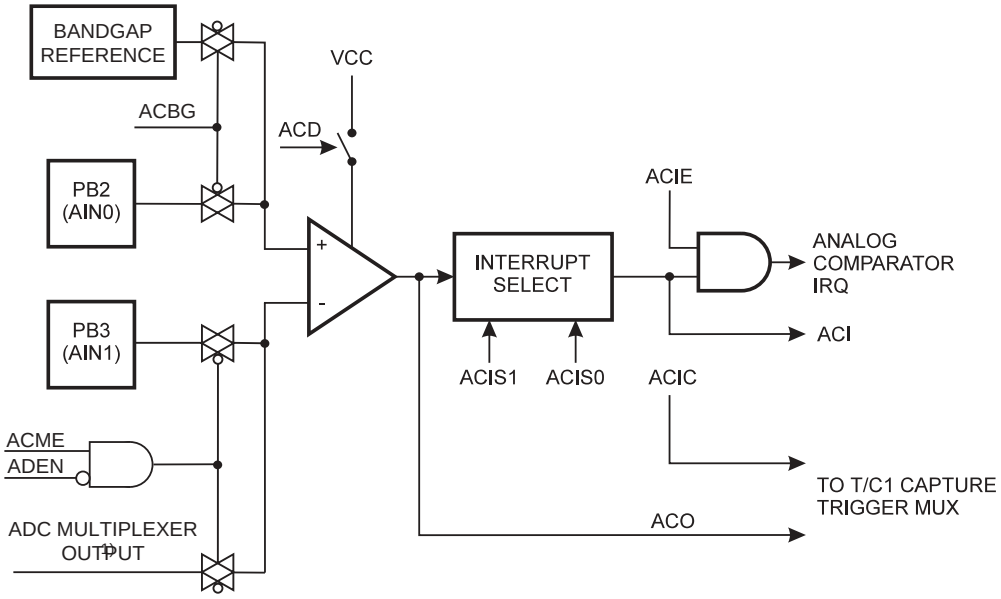

## Assembly Code Example – Slave Transmitter Mode

```
; Part specific include file and TWI include file must be included.
; <Initialize registers, including TWAR, TWBR and TWCR>

     ldi   r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
     out   TWCR, r16          ; Enable TWI in Slave Transmitter mode

; <Receive START condition and SLA+R>

wait14:in  r16,TWCR           ; Wait for TWINT Flag set. This indicates that
     sbrs  r16, TWINT         ; SLA+R has been received, and ACK/NACK has
     rjmp  wait14             ; been returned

     in    r16, TWSR          ; Check value of TWI Status Register. If status
     cpi   r16, ST_SLA_ACK    ; different from ST_SLA_ACK, go to ERROR
     brne  ERROR

     ldi   r16, 0x33          ; Load data(here, data=0x33)into TWDR Register
     out   TWDR, r16
     ldi   r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
     out   TWCR, r16          ; Clear TWINT bit in TWCR to start transmission
                              ; of data. Setting TWEA indicates that ACK
                              ; should be received when transfer finished
   ; <Send more data bytes if needed>
wait15:in  r16,TWCR           ; Wait for TWINT Flag set. This indicates that
     sbrs  r16, TWINT         ; data has been transmitted, and ACK/NACK has
     rjmp  wait15             ; been received

     in    r16, TWSR          ; Check value of TWI Status Register. If status
     cpi   r16, ST_DATA_ACK   ; different from ST_DATA_ACK, go to ERROR
     brne  ERROR
```

## The Analog Comparator

The Analog Comparator compares the input values on the positive pin PB2 (AIN0) and negative pin PB3 (AIN1). When the voltage on the positive pin PB2 (AIN0) is higher than the voltage on the negative pin PB3 (AIN1), the Analog Comparator Output, ACO, is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 59.

**Figure 59.** Analog Comparator Block Diagram




Note: See Figure 60 on page 128.

# Analog to Digital Converter

## Features

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **Up to 76 kSPS at 8-bit Resolution**
- **Eight Multiplexed Single Ended Input Channels**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - $V_{CC}$ ADC Input Voltage Range**
- **Selectable 2.56V ADC Reference Voltage**
- **Free Run or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The ATmega323 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows each pin of Port A to be used as input for the ADC.

The ADC contains a Sample and Hold Amplifier which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 60.

The ADC has two separate analog supply voltage pins, AVCC and AGND. AGND must be connected to GND, and the voltage on AVCC must not differ more than ±0.3V from $V_{CC}$. See the paragraph ADC Noise Canceling Techniques on how to connect these pins.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The 2.56V reference may be externally decoupled at the AREF pin by a capacitor for better noise performance. See “Internal Voltage Reference” on page 31 for a description of the internal voltage reference.

**Table 46.** Input Channel Selections (Continued)

| MUX4..0 | Single-ended Input |
|---|---|
| 01000..11101 | Reserved |
| 11110 | 1.22V ($V_{BG}$) |
| 11111 | 0V (AGND) |

### The ADC Control and Status Register – ADCSR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $06 ($26) | ADEN | ADSC | ADFR | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 | ADCSR |
| 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 – ADEN: ADC Enable**

Writing a logical “1” to this bit enables the ADC. By clearing this bit to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, a logical “1” must be written to this bit to start each conversion. In Free Running mode, a logical “1” must be written to this bit to start the first conversion. The first time ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, an extended conversion will precede the initiated conversion. This extended conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. When a extended conversion precedes a real conversion, ADSC will stay high until the real conversion completes. Writing a 0 to this bit has no effect.

- **Bit 5 – ADFR: ADC Free Running Select**

When this bit is set (one) the ADC operates in Free Running mode. In this mode, the ADC samples and updates the Data Registers continuously. Clearing this bit (zero) will terminate Free Running mode.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set (one) when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set (one). ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSR, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is set (one) and the I-bit in SREG is set (one), the ADC Conversion Complete Interrupt is activated.

## Scanning Multiple Channels

Since change of analog channel always is delayed until a conversion is finished, the Free Running mode can be used to scan multiple channels without interrupting the converter. Typically, the ADC Conversion Complete interrupt will be used to perform the channel shift. However, the user should take the following fact into consideration:

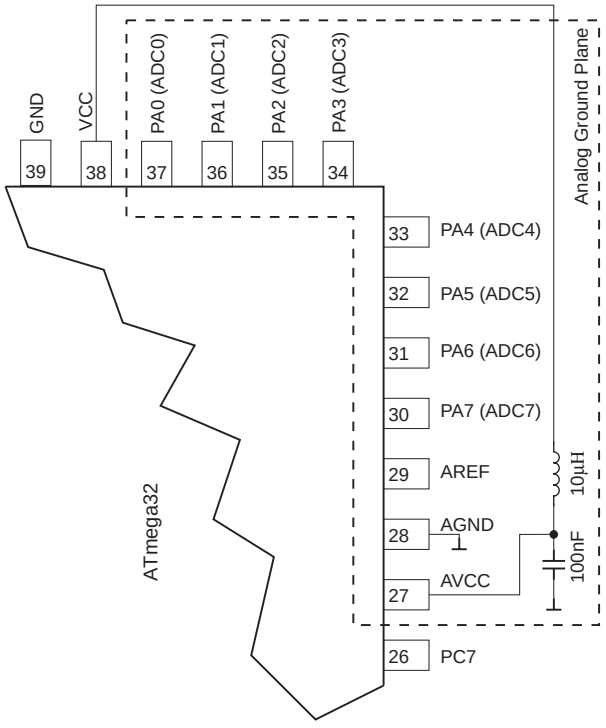
The interrupt triggers once the result is ready to be read. In Free Running mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started, and the old setting is used.

## ADC Noise Canceling Techniques

Digital circuitry inside and outside the ATmega323 generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. The analog part of the ATmega323 and all analog components in the application should have a separate analog ground plane on the PCB. This ground plane is connected to the digital ground plane via a single point on the PCB.
2. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
3. The AVCC pin on the ATmega323 should be connected to the digital $V_{CC}$ supply voltage via an LC network as shown in Figure 65.
4. Use the ADC noise canceler function to reduce induced noise from the CPU.
5. If some Port A pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

**Figure 65.** ADC Power Connections



## ADC Characteristics – Preliminary Data

| Symbol | Parameter | Condition | Min[2] | Typ | Max[3] | Units |
|---|---|---|---|---|---|---|
| | Resolution | Single-ended Conversion | | 10 | | Bits |
| | Absolute accuracy | $V_{REF}$ = 4V<br>ADC clock = 200 kHz | | 1 | 2 | LSB |
| | Absolute accuracy | $V_{REF}$ = 4V<br>ADC clock = 1 MHz | | 4 | | LSB |
| | Absolute accuracy | $V_{REF}$ = 4V<br>ADC clock = 2 MHz | | 16 | | LSB |
| | Integral Non-linearity | $V_{REF}$ > 2V | | 0.5 | | LSB |
| | Differential Non-linearity | $V_{REF}$ > 2V | | 0.5 | | LSB |
| | Zero Error (Offset) | $V_{REF}$ > 2V | | 1 | | LSB |
| | Conversion Time | Free Running Conversion | 65 | | 260 | µs |
| | Clock Frequency | | 50 | | 200 | kHz |
| $AV_{CC}$ | Analog Supply Voltage | | $V_{CC}$ - 0.3[2] | | $V_{CC}$ + 0.3[3] | V |
| $V_{REF}$ | Reference Voltage | | 2 V | | $AV_{CC}$ | V |
| VINT | Internal Voltage Reference | | 2.35 | 2.56 | 2.77 | V |
| $V_{BG}$ | Bandgap Voltage Reference | | 1.12 | 1.22 | 1.32 | V |
| $R_{REF}$ | Reference Input Resistance | | 6 | 10 | 13 | kΩ |
| $V_{IN}$ | Input Voltage | | AGND | | AREF | V |
| $R_{AIN}$ | Analog Input Resistance | | | 100 | | MΩ |

Notes: 1. Values are guidelines only. Actual values are TBD.
2. Minimum for AVCC is 2.7V.
3. Maximum for AVCC is 5.5V.

**Figure 80.** Port D Schematic Diagram (Pin PD1)

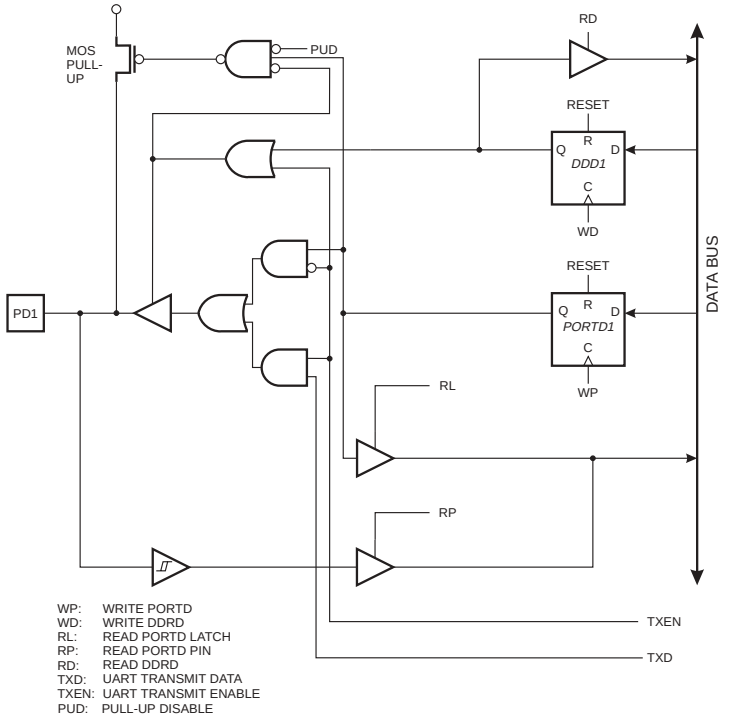
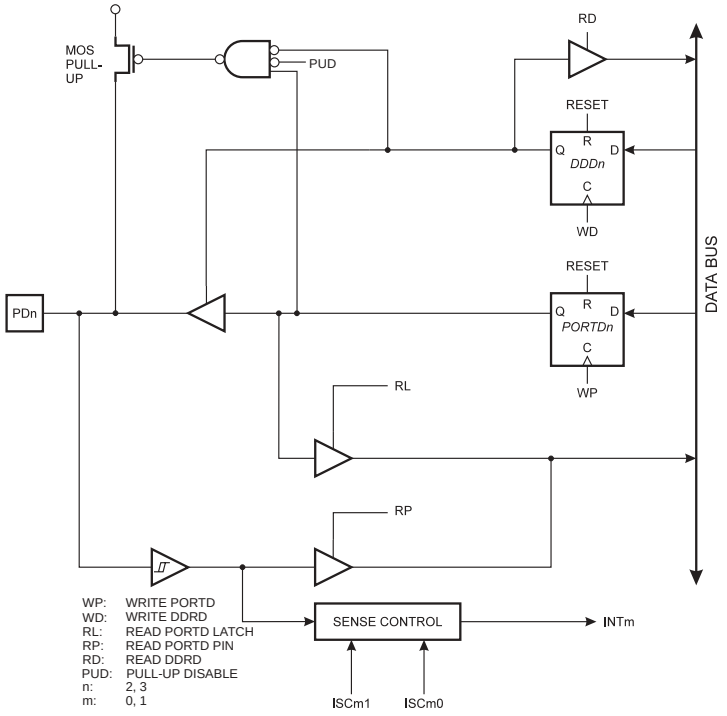



**Figure 81.** Port D Schematic Diagram (Pins PD2 and PD3)



# JTAG Interface and the On-chip Debug System

## Features

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-Scan Capabilities According to the 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**
  - **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 202 and “IEEE 1149.1 (JTAG) Boundary-Scan” on page 164**,** respectively. The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 85 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.

Of the Data Registers, the ID-Register, Bypass Register, and the Boundary-Scan Chain are 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.

**Table 58.** ATmega323 Boundary-Scan Order (Continued)

| Bit Number | Signal Name | Module |
|---|---|---|
| 44 | PD5.Data | Port D |
| 43 | PD5.Control | |
| 42 | PD5.PuLLup_Disable | |
| 41 | PD6.Data | |
| 40 | PD6.Control | |
| 39 | PD6.PuLLup_Disable | |
| 38 | PD7.Data | |
| 37 | PD7.Control | |
| 36 | PD7.PuLLup_Disable | |
| 35 | PC0.Data | Port C |
| 34 | PC0.Control | |
| 33 | PC0.PuLLup_Disable | |
| 32 | PC1.Data | |
| 31 | PC1.Control | |
| 30 | PC1.PuLLup_Disable | |
| 29 | PC6.Data | |
| 28 | PC6.Control | |
| 27 | PC6.PuLLup_Disable | |
| 26 | PC7.Data | |
| 25 | PC7.Control | |
| 24 | PC7.PuLLup_Disable | |
| 23 | PA7.Data | Port A |
| 22 | PA7.Control | |
| 21 | PA7.PuLLup_Disable | |
| 20 | PA6.Data | |
| 19 | PA6.Control | |
| 18 | PA6.PuLLup_Disable | |
| 17 | PA5.Data | |
| 16 | PA5.Control | |
| 15 | PA5.PuLLup_Disable | |
| 14 | PA4.Data | |
| 13 | PA4.Control | |
| 12 | PA4.PuLLup_Disable | |

**Table 61.** Boot Lock Bit0 Protection Modes (Application Section)(1)

| BLB0 Mode | BLB02 | BLB01 | Protection |
|---|---|---|---|
| 1 | 1 | 1 | No restrictions for SPM or LPM accessing the Application section. |
| 2 | 1 | 0 | SPM is not allowed to write to the Application section. |
| 3 | 0 | 0 | SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section. |
| 4 | 0 | 1 | LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section. |

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

**Table 62.** Boot Lock Bit1 Protection Modes (Boot Loader Section)(1)

| BLB1 mode | BLB12 | BLB11 | Protection |
|---|---|---|---|
| 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. “1” means unprogrammed, “0” means programmed

### Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits, write the desired data to R0, write “00001001” 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** |

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 SPMEN are set in SPMCR.

**Table 67.** Parallel Programming Characteristics, $T_A$ = 25°C ± 10%, $V_{CC}$ = 5 V ± 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 |
| $t_{XHXL}$ | XTAL1 Pulse Width High | 67 | | | ns |
| $t_{XLDX}$ | Data and Control Hold after XTAL1 Low | 67 | | | ns |
| $t_{XLWL}$ | XTAL1 Low to $\overline{WR}$ Low | 67 | | | ns |
| $t_{BVPH}$ | BS1 Valid before PAGEL High | 67 | | | ns |
| $t_{PHPL}$ | PAGEL Pulse Width High | 67 | | | ns |
| $t_{PLBX}$ | BS1 Hold after PAGEL Low | 67 | | | ns |
| $t_{PLWL}$ | PAGEL Low to $\overline{WR}$ Low | 67 | | | ns |
| $t_{BVWL}$ | BS1 Valid to $\overline{WR}$ Low | 67 | | | ns |
| $t_{RHBX}$ | BS1 Hold after RDY/$\overline{BSY}$ High | 67 | | | ns |
| $t_{WLWH}$ | $\overline{WR}$ Pulse Width Low | 67 | | | ns |
| $t_{WLRL}$ | $\overline{WR}$ Low to RDY/$\overline{BSY}$ Low | 0 | | 2.5 | µs |
| $t_{WLRH}$(1) | $\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1) | 1 | 1.5 | 1.9 | ms |
| $t_{WLRH\_CE}$(2) | $\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2) | 16 | 23 | 30 | ms |
| $t_{WLRH\_FLASH}$(3) | $\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Write Flash(3) | 8 | 12 | 15 | ms |
| $t_{XLOL}$ | XTAL1 Low to $\overline{OE}$ Low | 67 | | | ns |
| $t_{OLDV}$ | $\overline{OE}$ Low to DATA Valid | | 20 | | ns |
| $t_{OHDZ}$ | $\overline{OE}$ High to DATA Tri-stated | | | 20 | ns |

Notes: 1. $t_{WLRH}$ is valid for the Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. tWLRH_CE is valid for the Chip Erase command.
3. tWLRH_FLASH is valid for the Write Flash command.

### Programming Times for Non-volatile Memory

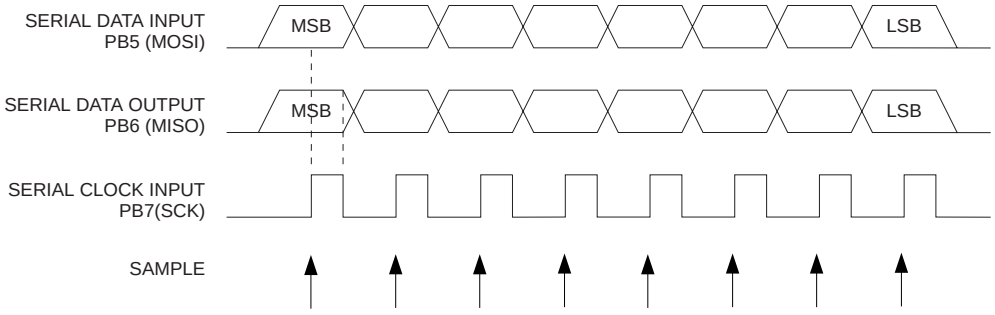
The internal RC Oscillator is used to control programming time when programming or erasing Flash, EEPORM, Fuses and Lock bits. During Parallel or Serial Programming, the device is in reset, and this Oscillator runs at its initial, uncalibrated frequency, which may vary from 0.5 MHz to 1.0 MHz. In software it is possible to calibrate this Oscillator to 1.0 MHz (see “Calibrated Internal RC Oscillator” on page 41). Consequently, programming times will be shorter and more accurate when programming or erasing non-volatile memory from software, using SPM or the EEPROM interface. See Table 68 for a summary of programming times.

**Table 68.** Maximum Programming Times for Non-volatile Memory

| Operation | Symbol | Number of RC Oscillator Cycles | Parallel/Serial Programming | | Self-programming (1) |
|---|---|---|---|---|---|
| | | | 2.7V | 5.0V | |
| Chip Erase | $t_{WD\_CE}$ | 16K | 32 ms | 30 ms | 17 ms |
| Flash Write | $t_{WD\_FLASH}$ | 8K | 16 ms | 15 ms | 8.5 ms |
| EEPROM Write(2) | $t_{WD\_EEPROM}$ | 2K | 4 ms | 3.8 ms | 2.2 ms |
| Fuse/Lock bit write | $t_{WD\_FUSE}$ | 1K | 2 ms | 1.9 ms | 1.1 ms |

Notes: 1. Includes variation over voltage and temperature after RC Oscillator has been calibrated to 1.0 MHz
2. Parallel EEPROM programming takes 1K cycles

**Figure 99.** Serial Programming Waveforms



## DC Characteristics (Continued)

$T_A$ = -40°C to 85°C, $V_{CC}$ = 2.7V to 5.5V (Unless Otherwise Noted)

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $I_{CC}$ | Power Supply Current | Active 4 MHz, $V_{CC}$ = 3V (ATmega323L) | | | 5 | mA |
| | | Active 8 MHz, $V_{CC}$ = 5V (ATmega323) | | | 15 | mA |
| | | Idle 4 MHz, $V_{CC}$ = 3V (ATmega323L) | | | 2.5 | mA |
| | | Idle 8 MHz, $V_{CC}$ = 5V (ATmega323) | | | 8 | mA |
| | Power-down mode(5) | WDT enabled, $V_{CC}$ = 3V | | 9 | 15.0 | µA |
| | | WDT disabled, $V_{CC}$ = 3V | | <1 | 4.0 | µA |
| $V_{ACIO}$ | Analog Comparator Input Offset Voltage | $V_{CC}$ = 5V<br>$V_{in}$ = $V_{CC}$/2 | | | 40 | mV |
| $I_{ACLK}$ | Analog Comparator Input Leakage Current | $V_{CC}$ = 5V<br>$V_{in}$ = $V_{CC}$/2 | -50 | | 50 | nA |
| $t_{ACID}$ | Analog Comparator Initialization Delay | $V_{CC}$ = 2.7V<br>$V_{CC}$ = 4.0V | | 750<br>500 | | ns |

Note:
1. “Max” means the highest value where the pin is guaranteed to be read as low
2. “Min” means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20mA at Vcc = 5V, 10mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed – PDIP Package:
   1] The sum of all IOL, for all ports, should not exceed 200 mA.
   2] The sum of all IOL, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0 - B7,C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA – TQFP Package:
   1] The sum of all IOL, for all ports, should not exceed 400 mA.
   2] The sum of all IOL, for ports A0 - A7, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all IOL, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all IOL, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all IOL, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all IOL, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOL, for ports D4 - D7, should not exceed 100 mA
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (3mA at Vcc = 5V, 1.5mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed – PDIP Package:
   1] The sum of all IOH, for all ports, should not exceed 200 mA.
   2] The sum of all IOH, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0 - B7,C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA – TQFP Package:
   1] The sum of all IOH, for all ports, should not exceed 400 mA.
   2] The sum of all IOH, for ports A0 - A7, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all IOH, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all IOH, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all IOH, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all IOH, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOH, for ports D4 - D7, should not exceed 100 mA
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum $V_{CC}$ for Power-down is 2.5V.

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

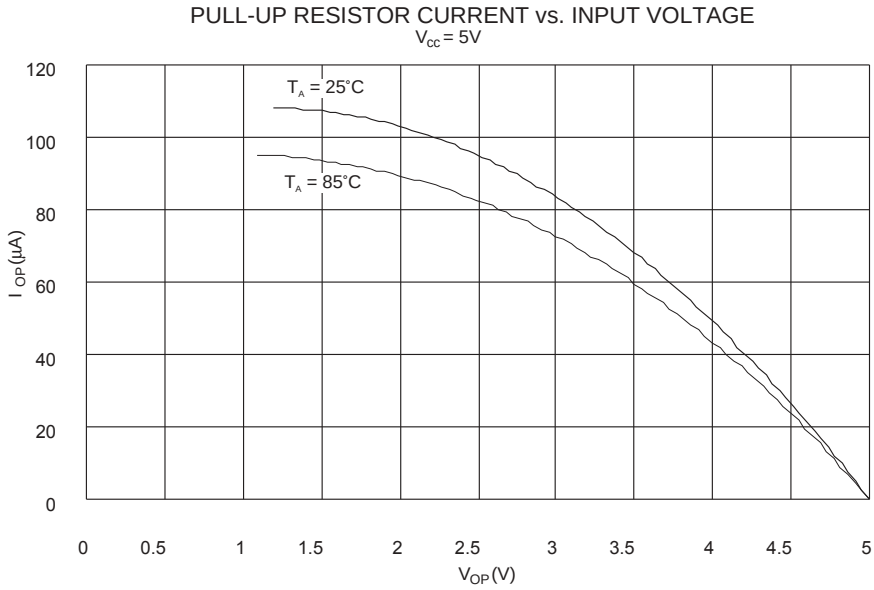
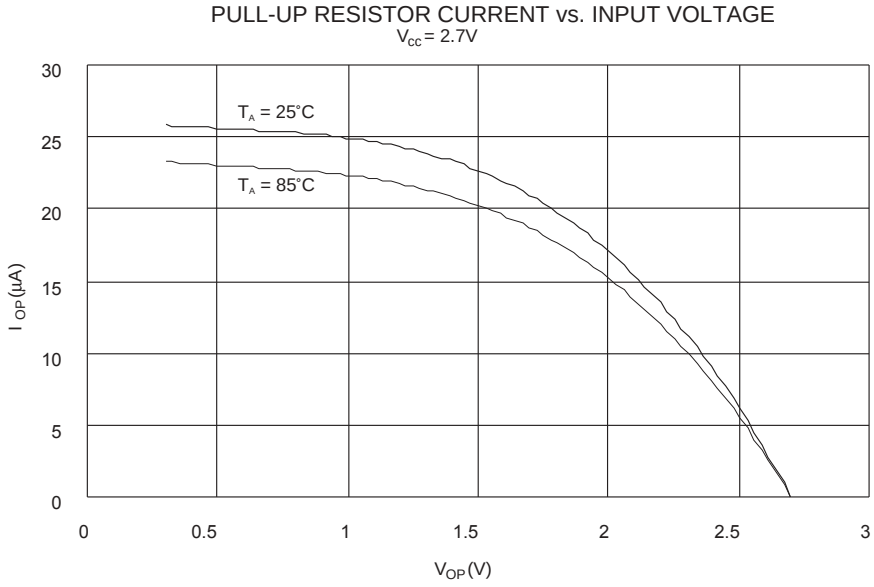



**Figure 126.** Pull-Up Resistor Current vs. Input Voltage



| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| CLH | | Clear Half Carry Flag in SREG | H ← 0 | H | 1 |
| NOP | | No Operation | | None | 1 |
| SLEEP | | Sleep | (see specific descr. for Sleep function) | None | 1 |
| WDR | | Watchdog Reset | (see specific descr. for WDR/timer) | None | 1 |
| BREAK | | Break | For On-chip Debug Only | None | N/A |