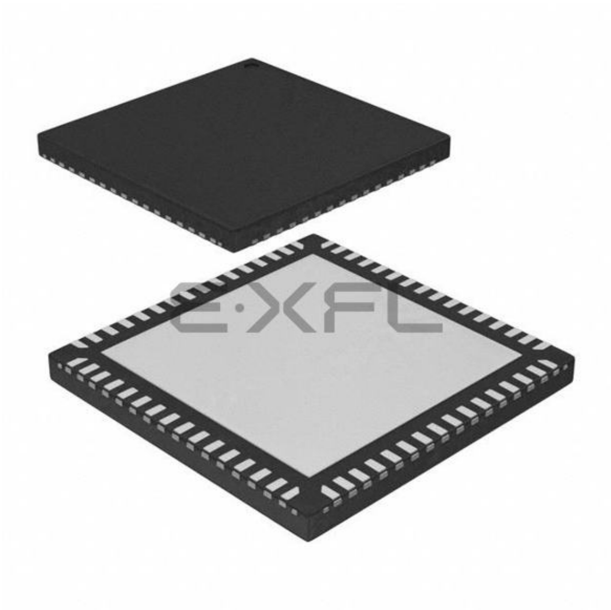


Welcome to E-XFL.COM

## 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 | Active |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 16MHz |
| Connectivity | SPI, UART/USART, USI |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 53 |
| Program Memory Size | 64KB (32K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 4K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-VFQFN Exposed Pad |
| Supplier Device Package | 64-QFN (9x9) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atmega645-16mur |

**Table 8-1.** EEPROM Programming Time

| Symbol | Number of Calibrated RC Oscillator Cycles | Typical Programming Time |
|---|---|---|
| EEPROM write (from CPU) | 27,072 | 3.4ms |

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out EEARH, r18
  out EEARL, r17
  ; Write data (r16) to Data Register
  out EEDR,r16
  ; Write logical one to EEMWE
  sbi EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and Data Registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

# 11. System Control and Reset

## 11.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 11-1 on page 42 shows the reset logic. Table 28-4 on page 301 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 27.

## 11.2 Reset Sources

The Atmel ATmega325/3250/645/6450 has five sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold ($V_{POT}$).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage $V_{CC}$ is below the Brown-out Reset threshold ($V_{BOT}$) and the Brown-out Detector is enabled.
- JTAG AVR Reset. The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. Refer to the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 224 for details.

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 4K 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 0x3802/0x7802
0x3804/0x7804   jmp    EXT_INT0       ; IRQ0 Handler
0x3806/0x7806   jmp    PCINT0         ; PCINT0 Handler
...             ...    ...            ;
0x1C2C          jmp    SPM_RDY        ; Store Program Memory Ready Handler
```

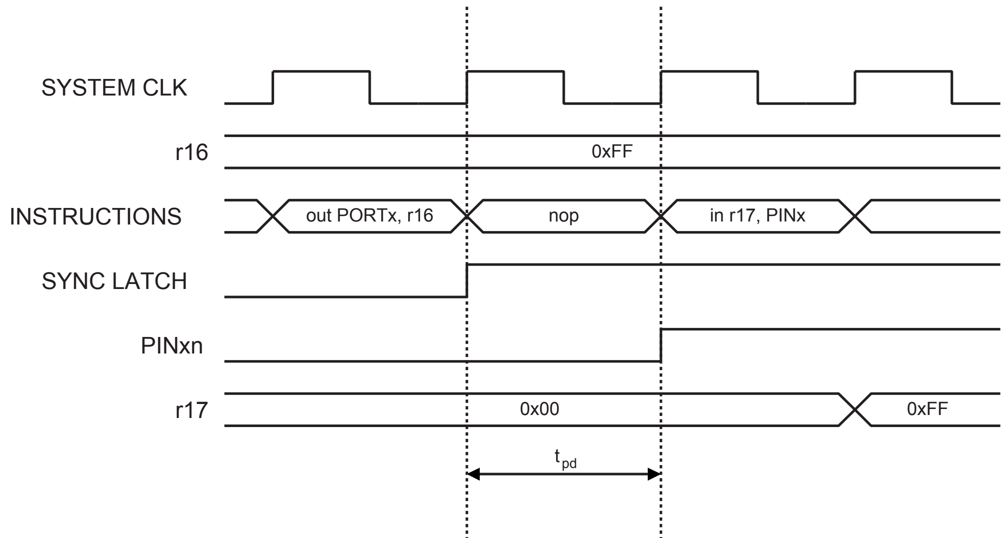
When the BOOTRST Fuse is programmed and the Boot section size set to 4K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
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 0x3800/0x7800
0x3800/0x7801RESET:ldir16,high(RAMEND); Main program start
0x3801/0x7801   out    SPH,r16      ; Set Stack Pointer to top of RAM
0x3802/0x7802   ldi    r16,low(RAMEND)
0x3803/0x7803   out    SPL,r16
0x3804/0x7804   sei                   ; Enable interrupts
0x3805/0x7805   <instr>  xxx
```

Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 14-4. The out instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

**Figure 14-4.** Synchronization when Reading a Software Assigned Pin Value

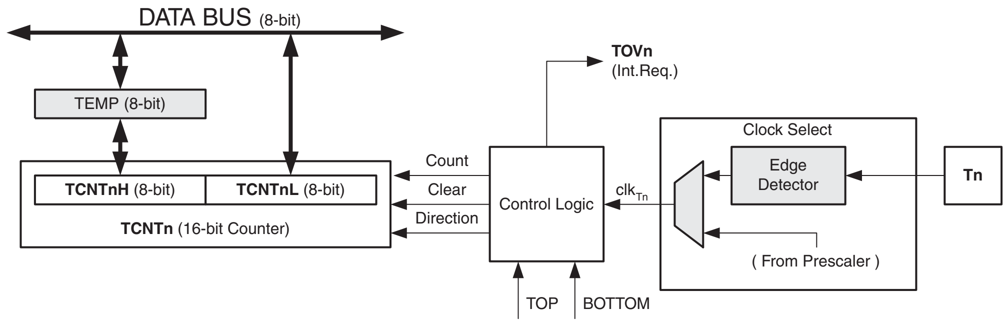



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

## 17.5 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 17-2 shows a block diagram of the counter and its surroundings.

**Figure 17-2.** Counter Unit Block Diagram




Signal description (internal signals):

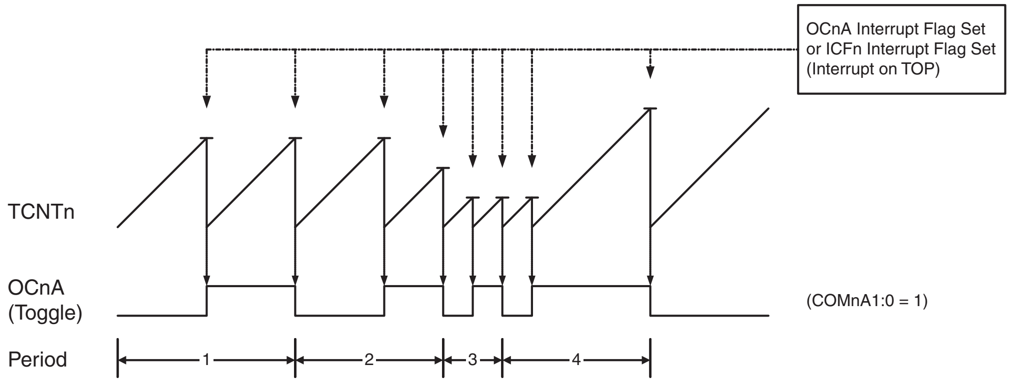
| | |
|---|---|
| **Count** | Increment or decrement TCNT1 by 1. |
| **Direction** | Select between increment and decrement. |
| **Clear** | Clear TCNT1 (set all bits to zero). |
| **$clk_{T1}$** | Timer/Counter clock. |
| **TOP** | Signalize that TCNT1 has reached maximum value. |
| **BOTTOM** | Signalize that TCNT1 has reached minimum value (zero). |

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* ($clk_{T1}$). The $clk_{T1}$ can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether $clk_{T1}$ is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the *Waveform Generation mode* bits (WGM13:0) located in the *Timer/Counter Control Registers* A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 114.

**Figure 17-6.** CTC Mode, Timing Diagram




An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{\text{clk\_I/O}}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

### 17.9.3 Fast PWM Mode

The *fast Pulse Width Modulation* or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

# 18. 8-bit Timer/Counter2 with PWM and Asynchronous Operation

## 18.1 Features

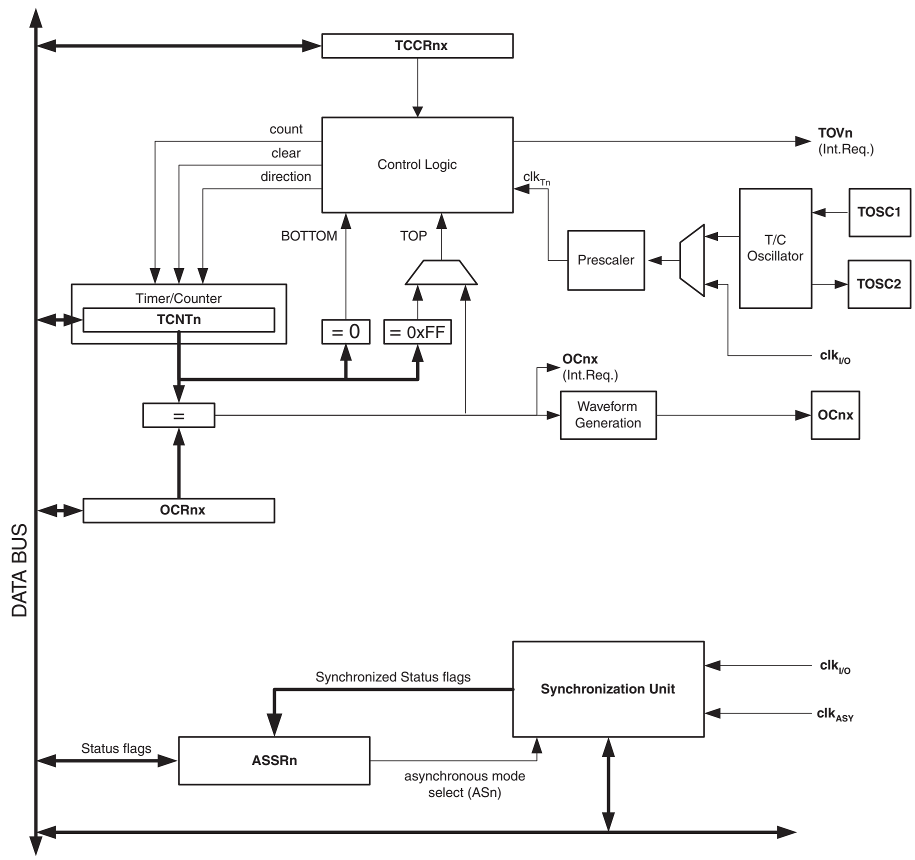
Timer/Counter2 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- **Single Compare Unit Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV2 and OCF2A)**
- **Allows Clocking from External 32kHz Watch Crystal Independent of the I/O Clock**

## 18.2 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 18-1. For the actual placement of I/O pins, refer to "Pinout ATmega3250/6450" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "Register Description" on page 143.

**Figure 18-1.** 8-bit Timer/Counter Block Diagram



- **Bit 5:4 – COM2A1:0: Compare Match Output Mode A**

These bits control the Output Compare pin (OC2A) behavior. If one or both of the COM2A1:0 bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to OC2A pin must be set in order to enable the output driver.

When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM21:0 bit setting. Table 18-3 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

**Table 18-3.** Compare Output Mode, non-PWM Mode

| COM2A1 | COM2A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC2A disconnected. |
| 0 | 1 | Toggle OC2A on compare match. |
| 1 | 0 | Clear OC2A on compare match. |
| 1 | 1 | Set OC2A on compare match. |

Table 18-4 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

**Table 18-4.** Compare Output Mode, Fast PWM Mode[(1)]

| COM2A1 | COM2A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC2A disconnected. |
| 0 | 1 | Reserved |
| 1 | 0 | Clear OC2A on compare match, set OC2A at BOTTOM, (non-inverting mode). |
| 1 | 1 | Set OC2A on compare match, clear OC2A at BOTTOM, (inverting mode). |

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 136 for more details.

Table 18-5 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode.

**Table 18-5.** Compare Output Mode, Phase Correct PWM Mode[(1)]

| COM2A1 | COM2A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC2A disconnected. |
| 0 | 1 | Reserved |
| 1 | 0 | Clear OC2A on compare match when up-counting. Set OC2A on compare match when counting down. |
| 1 | 1 | Set OC2A on compare match when up-counting. Clear OC2A on compare match when counting down. |

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 138 for more details.

# 23. Analog to Digital Converter

## 23.1 Features

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **± 2 LSB Absolute Accuracy**
- **13µs - 260µs Conversion Time (50kHz to 1MHz ADC clock)**
- **Up to 76.9kSPS at Maximum Resolution (200kHz ADC clock)**
- **Eight Multiplexed Single Ended Input Channels**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - $V_{CC}$ ADC Input Voltage Range**
- **Selectable 1.1V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The Atmel ATmega325/3250/645/6450 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port F. The single-ended voltage inputs refer to 0V (GND).

The ADC contains a Sample and Hold circuit 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 23-1.

The ADC has a separate analog supply voltage pin, AVCC. $AV_{CC}$ must not differ more than ± 0.3V from $V_{CC}$. See the paragraph “ADC Noise Canceler” on page 207 on how to connect this pin.

Internal reference voltages of nominally 1.1V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

The Power Reduction ADC bit, PRADC, in “Power Reduction Register” on page 37 must be disabled by writing a logical zero to be able to use the ADC module.

## 23.8 Register Description

### 23.8.1 ADMUX – ADC Multiplexer Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x7C) | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 | ADMUX |
| 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 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 23-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

**Table 23-3.** Voltage Reference Selections for ADC

| REFS1 | REFS0 | Voltage Reference Selection |
|---|---|---|
| 0 | 0 | AREF, Internal Vref turned off |
| 0 | 1 | AVCC with external capacitor at AREF pin |
| 1 | 0 | Reserved |
| 1 | 1 | Internal 1.1V Voltage Reference with external capacitor at AREF pin |

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see “ADCL and ADCH – The ADC Data Register” on page 216.

- **Bits 4:0 – MUX4:0: Analog Channel Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 23-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

# 24. JTAG Interface and On-chip Debug System

## 24.1 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®**

## 24.2 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 284 and "IEEE 1149.1 (JTAG) Boundary-scan" on page 224, respectively. The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 24-1 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.

## 24.3 TAP – Test Access Port

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).

**Table 25-7.** ATmega325/645 Boundary-scan Order, 64-pin (Continued)

| Bit Number | Signal Name | Module |
|---|---|---|
| 157 | PE0.Data | Port E |
| 156 | PE0.Control | |
| 155 | PE0.Pull-up_Enable | |
| 154 | PE1.Data | |
| 153 | PE1.Control | |
| 152 | PE1.Pull-up_Enable | |
| 151 | PE2.Data | |
| 150 | PE2.Control | |
| 149 | PE2.Pull-up_Enable | |
| 148 | PE3.Data | |
| 147 | PE3.Control | |
| 146 | PE3.Pull-up_Enable | |
| 145 | PE4.Data | |
| 144 | PE4.Control | |
| 143 | PE4.Pull-up_Enable | |
| 142 | PE5.Data | |
| 141 | PE5.Control | |
| 140 | PE5.Pull-up_Enable | |
| 139 | PE6.Data | |
| 138 | PE6.Control | |
| 137 | PE6.Pull-up_Enable | |
| 136 | PE7.Data | |
| 135 | PE7.Control | |
| 134 | PE7.Pull-up_Enable | |

# 27. Memory Programming

## 27.1 Program And Data Memory Lock Bits

The Atmel ATmega325/3250/645/6450 provides six Lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 27-2. The Lock bits can only be erased to "1" with the Chip Erase command.

**Table 27-1.** Lock Bit Byte[1]

| Lock Bit Byte | Bit No | Description | Default Value |
|---|---|---|---|
| | 7 | – | 1 (unprogrammed) |
| | 6 | – | 1 (unprogrammed) |
| BLB12 | 5 | Boot Lock bit | 1 (unprogrammed) |
| BLB11 | 4 | Boot Lock bit | 1 (unprogrammed) |
| BLB02 | 3 | Boot Lock bit | 1 (unprogrammed) |
| BLB01 | 2 | Boot Lock bit | 1 (unprogrammed) |
| LB2 | 1 | Lock bit | 1 (unprogrammed) |
| LB1 | 0 | Lock bit | 1 (unprogrammed) |

Note: 1. "1" means unprogrammed, "0" means programmed

**Table 27-2.** Lock Bit Protection Modes[1][2]

| Memory Lock Bits | | | Protection Type |
|---|---|---|---|
| **LB Mode** | **LB2** | **LB1** | |
| 1 | 1 | 1 | No memory lock features enabled. |
| 2 | 1 | 0 | Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.[1] |
| 3 | 0 | 0 | Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Boot Lock bits and Fuse bits are locked in both Serial and Parallel Programming mode.[1] |
| **BLB0 Mode** | **BLB02** | **BLB01** | |
| 1 | 1 | 1 | No restrictions for SPM or LPM accessing the Application section. |
| 2 | 1 | 0 | SPM is not allowed to write to the Application section. |
| 3 | 0 | 0 | SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section. |
| 4 | 0 | 1 | LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section. |

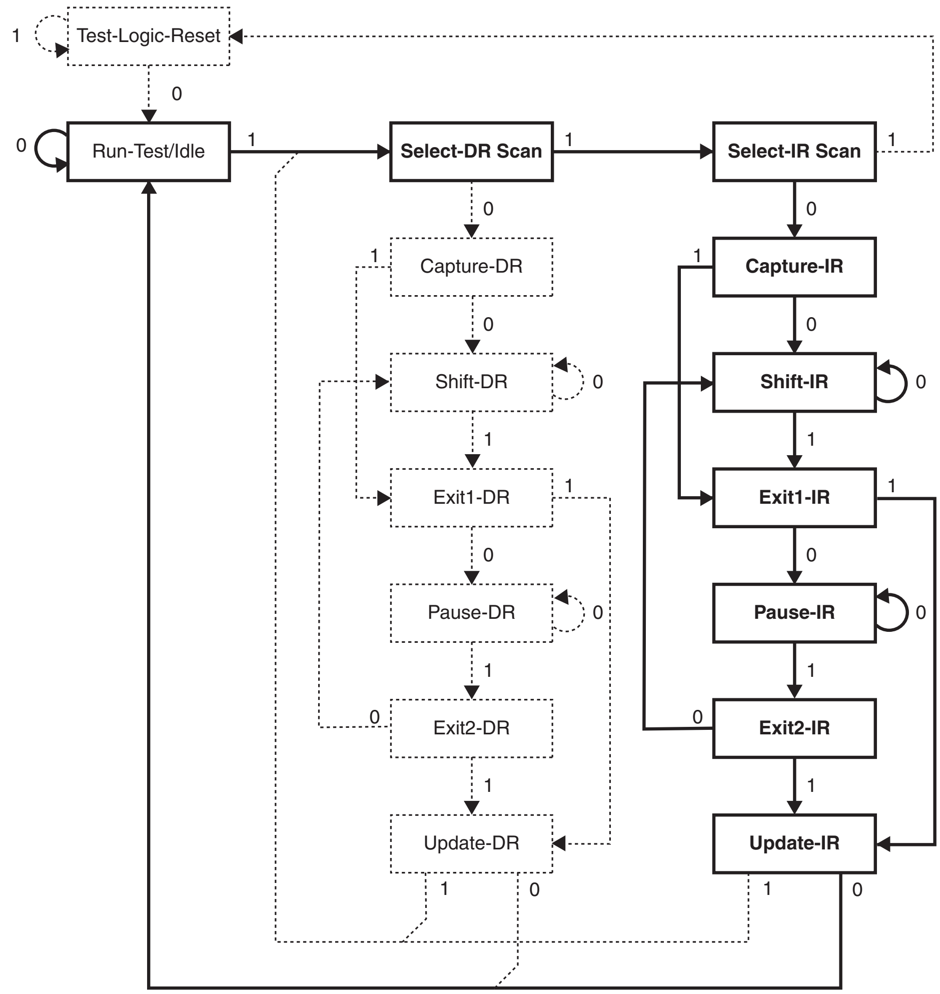
### 27.8.1 Programming Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. The JTAG instructions useful for programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences. The state machine sequence for changing the instruction word is shown in Figure 27-13.

**Figure 27-13.** State Machine Sequence for Changing the Instruction Word




### 27.8.2 AVR_RESET (0xC)

The AVR specific public JTAG instruction for setting the AVR device in the Reset mode or taking the device out from the Reset mode. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic "one" in the Reset Chain. The output from this chain is not latched.

The active states are:

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

## 28.7 ADC

**Table 28-7.** ADC Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Resolution | Single Ended Conversion | | 10 | | Bits |
| | | Differential Conversion | | 8 | | Bits |
| | Absolute accuracy (Including INL, DNL, quantization error, gain and offset error) | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz | | 2 | 2.5 | LSB |
| | | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 1MHz | | 4.5 | | LSB |
| | | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz Noise Reduction Mode | | 2 | | LSB |
| | | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 1 MHz Noise Reduction Mode | | 4.5 | | LSB |
| | Integral Non-Linearity (INL) | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz | | 0.5 | | LSB |
| | Differential Non-Linearity (DNL) | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz | | 0.25 | | LSB |
| | Gain Error | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz | | 2 | | LSB |
| | Offset Error | Single Ended Conversion $V_{REF}$ = 4V, $V_{CC}$ = 4V, ADC clock = 200kHz | | 2 | | LSB |
| | Conversion Time | Free Running Conversion | 13 | | 260 | µs |
| | Clock Frequency | Single Ended Conversion | 50 | | 1000 | kHz |
| AVCC | Analog Supply Voltage | | $V_{CC}$ - 0.3 | | $V_{CC}$ + 0.3 | V |
| $V_{REF}$ | Reference Voltage | Single Ended Conversion | 1.0 | | AVCC | V |
| | | Differential Conversion | 1.0 | | AVCC - 0.5 | V |
| $V_{IN}$ | Pin Input Voltage | Single Ended Channels | GND | | $V_{REF}$ | V |
| | | Differential Channels | GND | | AVCC | V |
| | Input Range | Single Ended Channels | GND | | $V_{REF}$ | V |
| | | Differential Channels[(1)] | -0.85$V_{REF}$ | | $V_{REF}$ | V |
| | Input Bandwidth | Single Ended Channels | | 38,5 | | kHz |
| | | Differential Channels | | 4 | | kHz |

**Table 28-7.** ADC Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{INT}$ | Internal Voltage Reference | | 1.0 | 1.1 | 1.2 | V |
| $R_{REF}$ | Reference Input Resistance | | | 32 | | kΩ |
| $R_{AIN}$ | Analog Input Resistance | | | 100 | | MΩ |

Note: 1. Voltage difference between channels

**Figure 29-19.** Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC}$ = 2.7V)

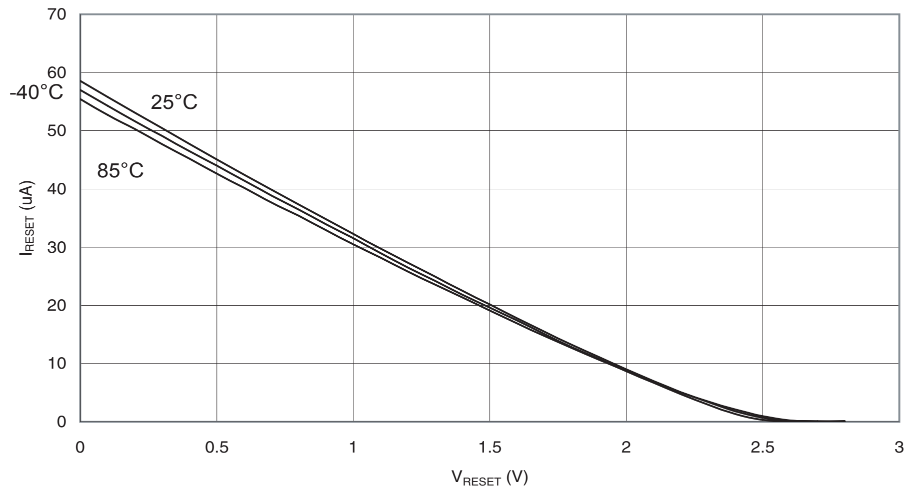
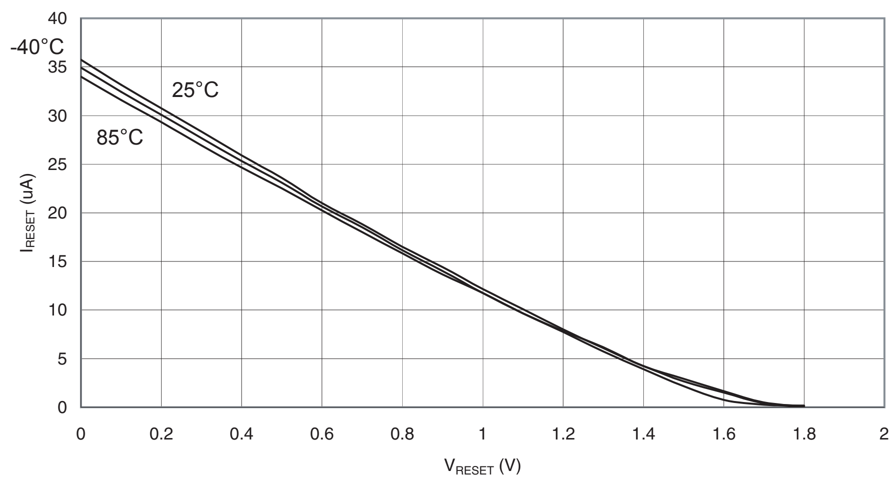



**Figure 29-20.** Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC}$ = 1.8V)



## 29.8 Pin Driver Strength

**Figure 29-21.** I/O Pin Source Current vs. Output Voltage, Ports A, C, D, E, F, G, H, J ($V_{CC}$ = 5V)

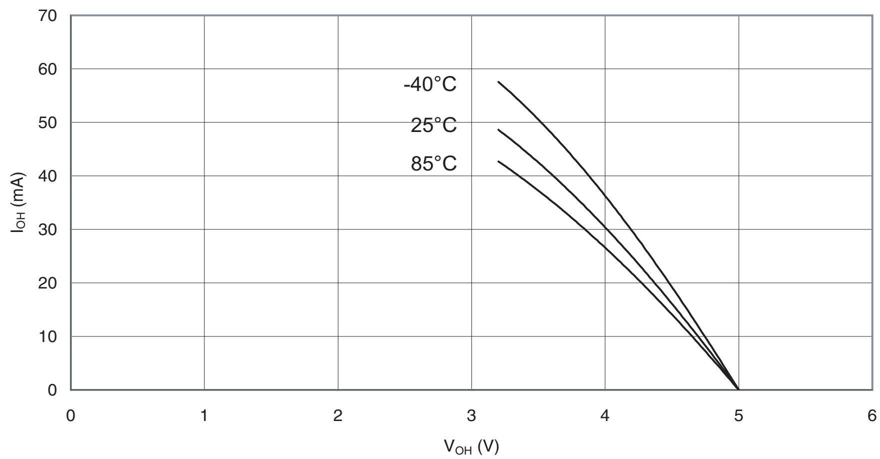



**Figure 29-22.** I/O Pin Source Current vs. Output Voltage, Ports A, C, D, E, F, G, H, J ($V_{CC}$ = 2.7V)

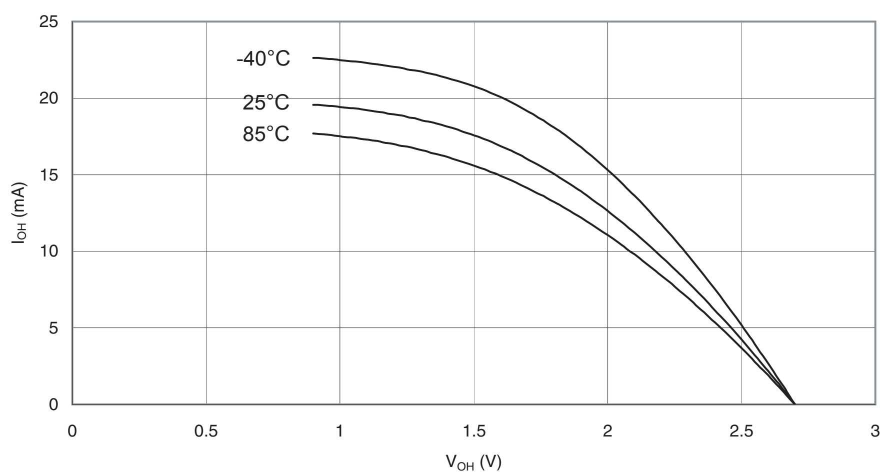
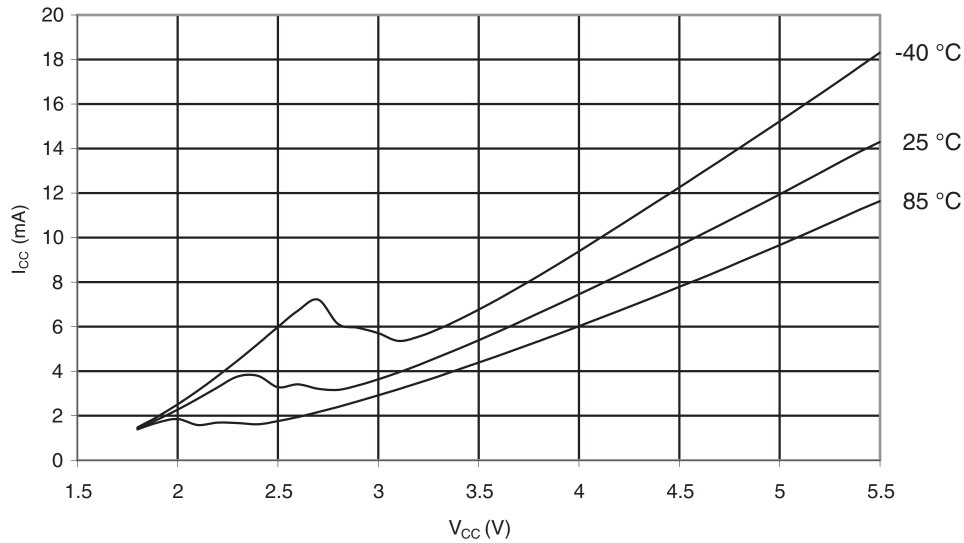
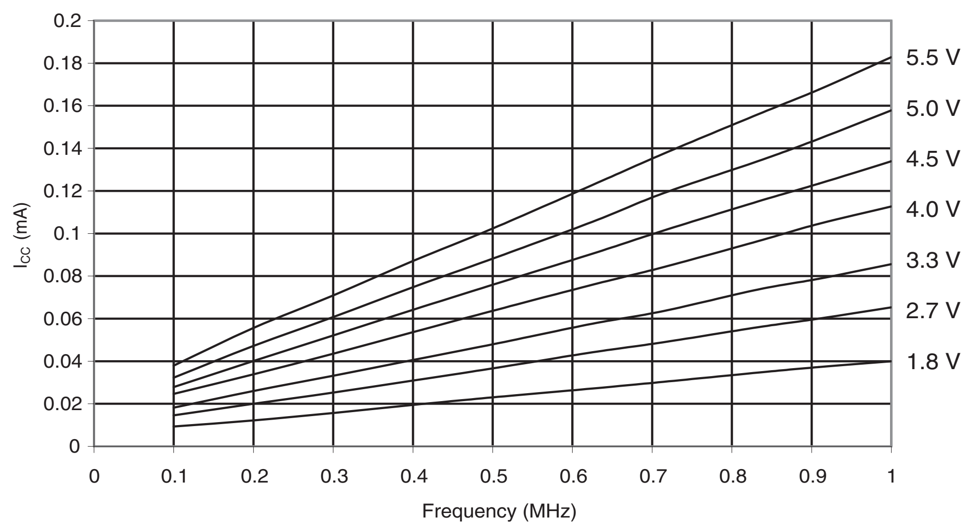


**Figure 29-55.** Programming Current vs. $V_{CC}$




## 29.13 Current Consumption in Reset and Reset Pulsewidth

**Figure 29-56.** Reset Supply Current vs. $V_{CC}$ (0.1 - 1.0MHz, Excluding Current Through The Reset Pull-up)



| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| BRTC | k | Branch if T Flag Cleared | if (T = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRVS | k | Branch if Overflow Flag is Set | if (V = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRVC | k | Branch if Overflow Flag is Cleared | if (V = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRIE | k | Branch if Interrupt Enabled | if ( I = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRID | k | Branch if Interrupt Disabled | if ( I = 0) then PC ← PC + k + 1 | None | 1/2 |
| **BIT AND BIT-TEST INSTRUCTIONS** | | | | | |
| SBI | P,b | Set Bit in I/O Register | I/O(P,b) ← 1 | None | 2 |
| CBI | P,b | Clear Bit in I/O Register | I/O(P,b) ← 0 | None | 2 |
| LSL | Rd | Logical Shift Left | Rd(n+1) ← Rd(n), Rd(0) ← 0 | Z,C,N,V | 1 |
| LSR | Rd | Logical Shift Right | Rd(n) ← Rd(n+1), Rd(7) ← 0 | Z,C,N,V | 1 |
| ROL | Rd | Rotate Left Through Carry | Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7) | Z,C,N,V | 1 |
| ROR | Rd | Rotate Right Through Carry | Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0) | Z,C,N,V | 1 |
| ASR | Rd | Arithmetic Shift Right | Rd(n) ← Rd(n+1), n=0..6 | Z,C,N,V | 1 |
| SWAP | Rd | Swap Nibbles | Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0) | None | 1 |
| BSET | s | Flag Set | SREG(s) ← 1 | SREG(s) | 1 |
| BCLR | s | Flag Clear | SREG(s) ← 0 | SREG(s) | 1 |
| BST | Rr, b | Bit Store from Register to T | T ← Rr(b) | T | 1 |
| BLD | Rd, b | Bit load from T to Register | Rd(b) ← T | None | 1 |
| SEC | | Set Carry | C ← 1 | C | 1 |
| CLC | | Clear Carry | C ← 0 | C | 1 |
| SEN | | Set Negative Flag | N ← 1 | N | 1 |
| CLN | | Clear Negative Flag | N ← 0 | N | 1 |
| SEZ | | Set Zero Flag | Z ← 1 | Z | 1 |
| CLZ | | Clear Zero Flag | Z ← 0 | Z | 1 |
| SEI | | Global Interrupt Enable | I ← 1 | I | 1 |
| CLI | | Global Interrupt Disable | I ← 0 | I | 1 |
| SES | | Set Signed Test Flag | S ← 1 | S | 1 |
| CLS | | Clear Signed Test Flag | S ← 0 | S | 1 |
| SEV | | Set Twos Complement Overflow. | V ← 1 | V | 1 |
| CLV | | Clear Twos Complement Overflow | V ← 0 | V | 1 |
| SET | | Set T in SREG | T ← 1 | T | 1 |
| CLT | | Clear T in SREG | T ← 0 | T | 1 |
| SEH | | Set Half Carry Flag in SREG | H ← 1 | H | 1 |
| CLH | | Clear Half Carry Flag in SREG | H ← 0 | H | 1 |
| **DATA TRANSFER INSTRUCTIONS** | | | | | |
| MOV | Rd, Rr | Move Between Registers | Rd ← Rr | None | 1 |
| MOVW | Rd, Rr | Copy Register Word | Rd+1:Rd ← Rr+1:Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd ← K | None | 1 |
| LD | Rd, X | Load Indirect | Rd ← (X) | None | 2 |
| LD | Rd, X+ | Load Indirect and Post-Inc. | Rd ← (X), X ← X + 1 | None | 2 |
| LD | Rd, - X | Load Indirect and Pre-Dec. | X ← X - 1, Rd ← (X) | None | 2 |
| LD | Rd, Y | Load Indirect | Rd ← (Y) | None | 2 |
| LD | Rd, Y+ | Load Indirect and Post-Inc. | Rd ← (Y), Y ← Y + 1 | None | 2 |
| LD | Rd, - Y | Load Indirect and Pre-Dec. | Y ← Y - 1, Rd ← (Y) | None | 2 |
| LDD | Rd,Y+q | Load Indirect with Displacement | Rd ← (Y + q) | None | 2 |
| LD | Rd, Z | Load Indirect | Rd ← (Z) | None | 2 |
| LD | Rd, Z+ | Load Indirect and Post-Inc. | Rd ← (Z), Z ← Z+1 | None | 2 |
| LD | Rd, -Z | Load Indirect and Pre-Dec. | Z ← Z - 1, Rd ← (Z) | None | 2 |
| LDD | Rd, Z+q | Load Indirect with Displacement | Rd ← (Z + q) | None | 2 |
| LDS | Rd, k | Load Direct from SRAM | Rd ← (k) | None | 2 |
| ST | X, Rr | Store Indirect | (X) ← Rr | None | 2 |
| ST | X+, Rr | Store Indirect and Post-Inc. | (X) ← Rr, X ← X + 1 | None | 2 |
| ST | - X, Rr | Store Indirect and Pre-Dec. | X ← X - 1, (X) ← Rr | None | 2 |
| ST | Y, Rr | Store Indirect | (Y) ← Rr | None | 2 |
| ST | Y+, Rr | Store Indirect and Post-Inc. | (Y) ← Rr, Y ← Y + 1 | None | 2 |
| ST | - Y, Rr | Store Indirect and Pre-Dec. | Y ← Y - 1, (Y) ← Rr | None | 2 |
| STD | Y+q,Rr | Store Indirect with Displacement | (Y + q) ← Rr | None | 2 |
| ST | Z, Rr | Store Indirect | (Z) ← Rr | None | 2 |
| ST | Z+, Rr | Store Indirect and Post-Inc. | (Z) ← Rr, Z ← Z + 1 | None | 2 |
| ST | -Z, Rr | Store Indirect and Pre-Dec. | Z ← Z - 1, (Z) ← Rr | None | 2 |
| STD | Z+q,Rr | Store Indirect with Displacement | (Z + q) ← Rr | None | 2 |
| STS | k, Rr | Store Direct to SRAM | (k) ← Rr | None | 2 |
| LPM | | Load Program Memory | R0 ← (Z) | None | 3 |
| LPM | Rd, Z | Load Program Memory | Rd ← (Z) | None | 3 |
| LPM | Rd, Z+ | Load Program Memory and Post-Inc | Rd ← (Z), Z ← Z+1 | None | 3 |
| SPM | | Store Program Memory | (Z) ← R1:R0 | None | - |