**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 | 20MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 32 |
| 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 ~ 105°C (TA) |
| 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/atmega644p-20aq |

# 2. Configuration Summary

The table below compares the device series of feature and pin compatible devices, providing a seamless migration path.

**Table 2-1. Configuration Summary and Device Comparison**

| Features | ATmega164/V | ATmega324/V | ATmega644/V |
|---|---|---|---|
| Pin Count | 40/44 | 40/44 | 40/44 |
| Flash (Bytes) | 16K | 32K | 64K |
| SRAM (Bytes) | 1K | 2K | 4K |
| EEPROM (Bytes) | 512 | 1K | 2K |
| General Purpose I/O Lines | 32 | 32 | 32 |
| SPI | 1 | 1 | 1 |
| TWI ($I^2C$) | 1 | 1 | 1 |
| USART | 2 | 2 | 2 |
| ADC | 10-bit 15ksps | 10-bit 15ksps | 10-bit 15ksps |
| ADC Channels | 8 | 8 | 8 |
| Analog Comparator | 1 | 1 | 1 |
| 8-bit Timer/Counters | 2 | 2 | 2 |
| 16-bit Timer/Counters | 1 | 1 | 1 |
| PWM channels | 6 | 6 | 6 |
| Packages | PDIP<br>TQFP<br>VQFN/QFN | PDIP<br>TQFP<br>VQFN/QFN | PDIP<br>TQFP<br>VQFN/QFN |

# 3. Ordering Information

| Speed [MHz][3] | Power Supply [V] | Ordering Code[2] | Package[1] | Operational Range |
|---|---|---|---|---|
| 10 | 1.8 - 5.5 | ATmega644PV-10AU<br>ATmega644PV-10AUR[4]<br>ATmega644PV-10PU<br>ATmega644PV-10MU<br>ATmega644PV-10MUR[4] | 44A<br>44A<br>40P6<br>44M1<br>44M1 | Industrial<br>(-40°C to 85°C) |
| 20 | 2.7 - 5.5 | ATmega644P-20AU<br>ATmega644P-20AUR[4]<br>ATmega644P-20PU<br>ATmega644P-20MU<br>ATmega644P-20MUR[4] | 44A<br>44A<br>40P6<br>44M1<br>44M1 | Industrial<br>(-40°C to 85°C) |
| 10 | 1.8 - 5.5 | ATmega644PV-10AN<br>ATmega644PV-10ANR[4]<br>ATmega644PV-10PN<br>ATmega644PV-10MN<br>ATmega644PV-10MNR[4] | 44A<br>44A<br>40P6<br>44M1<br>44M1 | Industrial<br>(-40°C to 105°C) |
| 20 | 2.7 - 5.5 | ATmega644P-20AN<br>ATmega644P-20ANR[4]<br>ATmega644P-20PN<br>ATmega644P-20MN<br>ATmega644P-20MNR[4] | 44A<br>44A<br>40P6<br>44M1<br>44M1 | Industrial<br>(-40°C to 105°C) |

**Note:**

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Refer to *Speed Grades* for Speed vs. $V_{CC}$
4. Tape & Reel.

The Timer/Counter2 can be clocked both synchronously and asynchronously in Power-save mode. If Timer/Counter2 is not using the asynchronous clock, the Timer/Counter Oscillator is stopped during sleep. If Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Even if the synchronous clock is running in Power-save, this clock is only available for Timer/Counter2.

## 11.8. Standby Mode

When the SM[2:0] bits are written to '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.

## 11.9. Extended Standby Mode

When the SM[2:0] bits are written to '111' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-Save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles.

## 11.10. 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 corresponding bit in the 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. In all other sleep modes, the clock is already stopped.

**Related Links**

PRR0 on page 67

## 11.11. Minimizing Power Consumption

There are several possibilities to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

### 11.11.1. Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion.

**Related Links**

Analog-to-Digital Converter on page 304

# 12. SCRST - System Control and Reset

## 12.1. 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 an Absolute Jump instruction (JMP) to the reset handling routine for . 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 the next section shows the reset logic.

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 the *System Clock and Clock Options* chapter.

**Related Links**

System Clock and Clock Options on page 43

## 12.2. Reset Sources

The device has the following sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is less than 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 System Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog System Reset mode is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage $V_{CC}$ is less than 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* for details.

### 14.1.2.6. Pin Change Mask Register 0

**Name:** PCMSK0
**Offset:** 0x6B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PCINT7 | PCINT6 | PCINT5 | PCINT4 | PCINT3 | PCINT2 | PCINT1 | PCINT0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

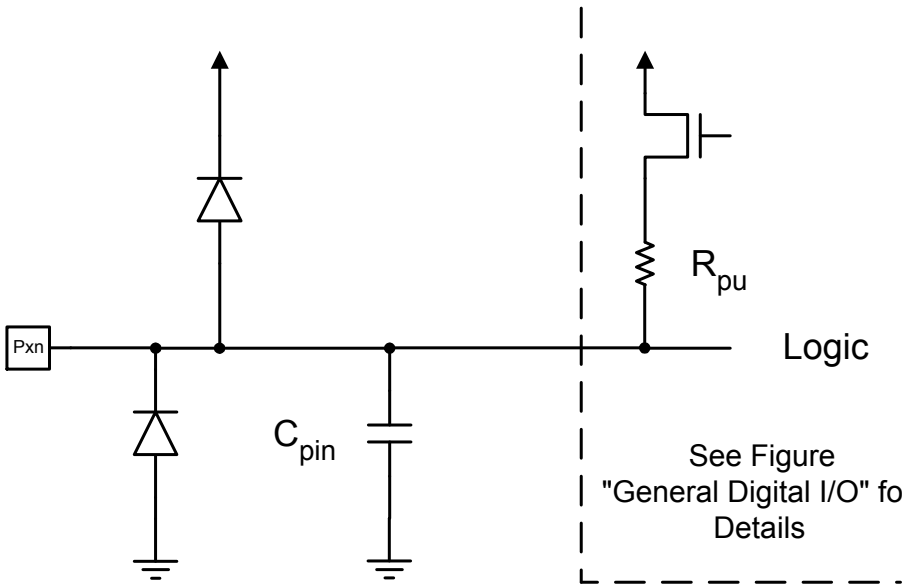
**Bits 7:0 – PCINTn: Pin Change Enable Mask [n = 7:0]**
Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

# 15. I/O-Ports

## 15.1. Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both $V_{CC}$ and Ground as indicated in the following figure.

**Figure 15-1. I/O Pin Equivalent Schematic**




All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing '1' to a bit in the PINx Register will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

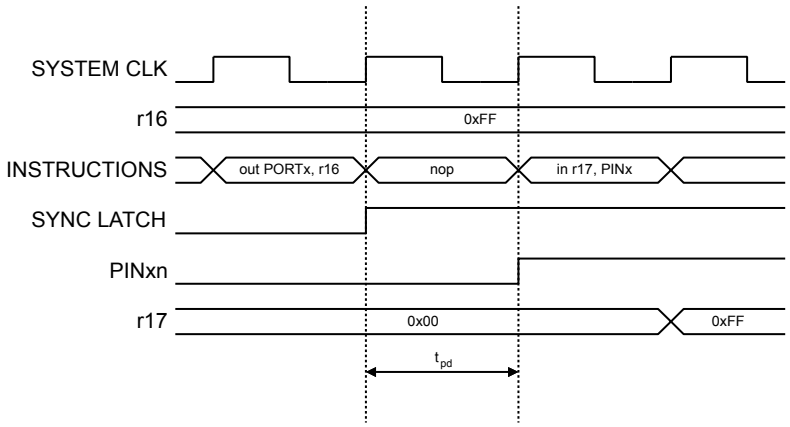
Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer to the individual module sections for a full description of the alternate functions.

Enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

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 the following figure. 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 15-4. 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.

**Assembly Code Example(1)**

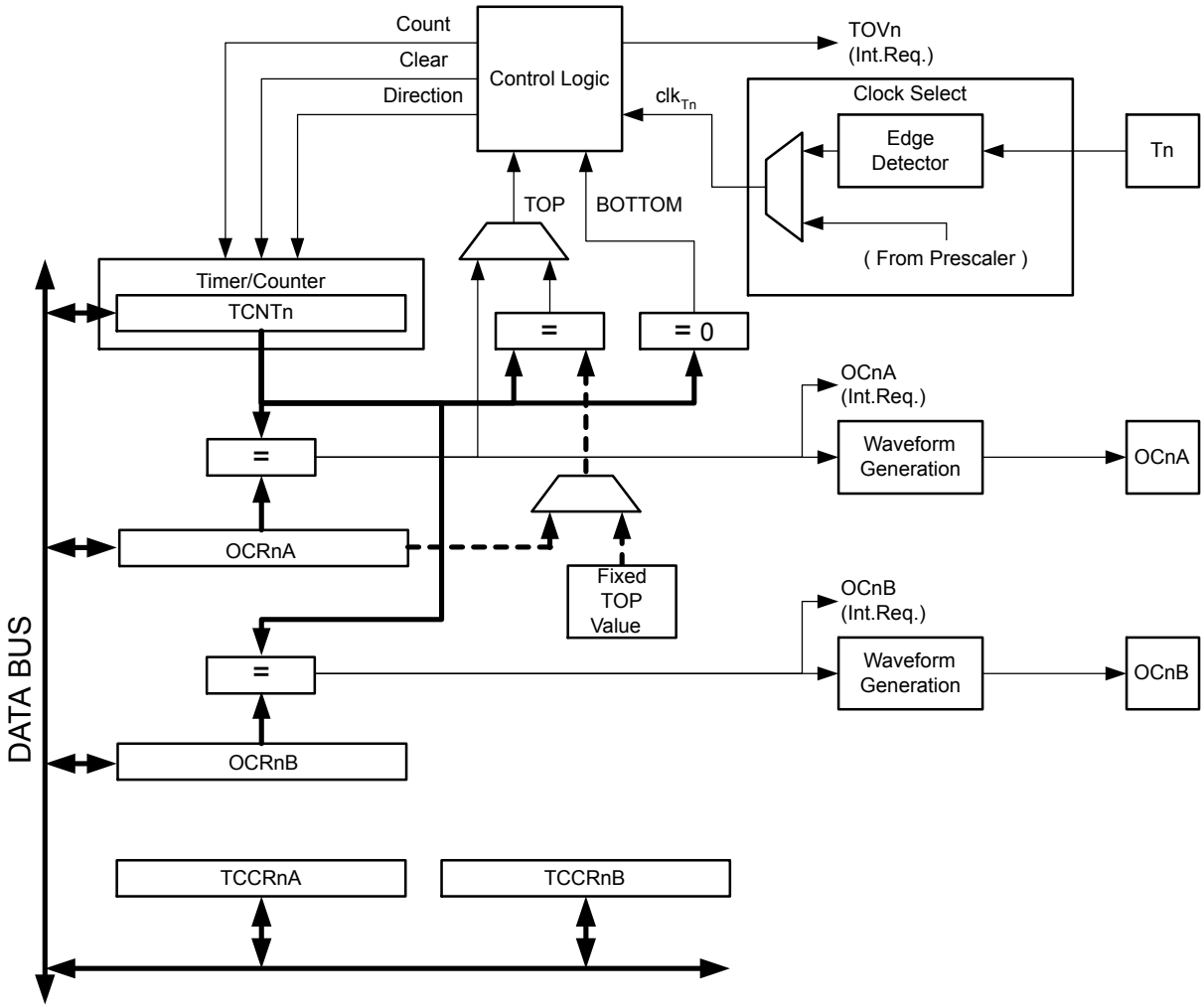
```
...
 ; Define pull-ups and set outputs high
 ; Define directions for port pins
 ldi r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
 ldi r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
 out PORTB,r16
 out DDRB,r17
 ; Insert nop for synchronization
 nop
 ; Read port pins
 in r16,PINB
 ...
```

**Note:** 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

**C Code Example**

```
unsigned char i;
 ...
 /* Define pull-ups and set outputs high */
 /* Define directions for port pins */
 PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
 DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
 /* Insert nop for synchronization*/
 __no_operation();
 /* Read port pins */
```

**Figure 16-1. 8-bit Timer/Counter Block Diagram**




### 16.2.1. Definitions

Many register and bit references in this section are written in general form:

- n=0 represents the Timer/Counter number
- x=A,B represents the Output Compare Unit A or B

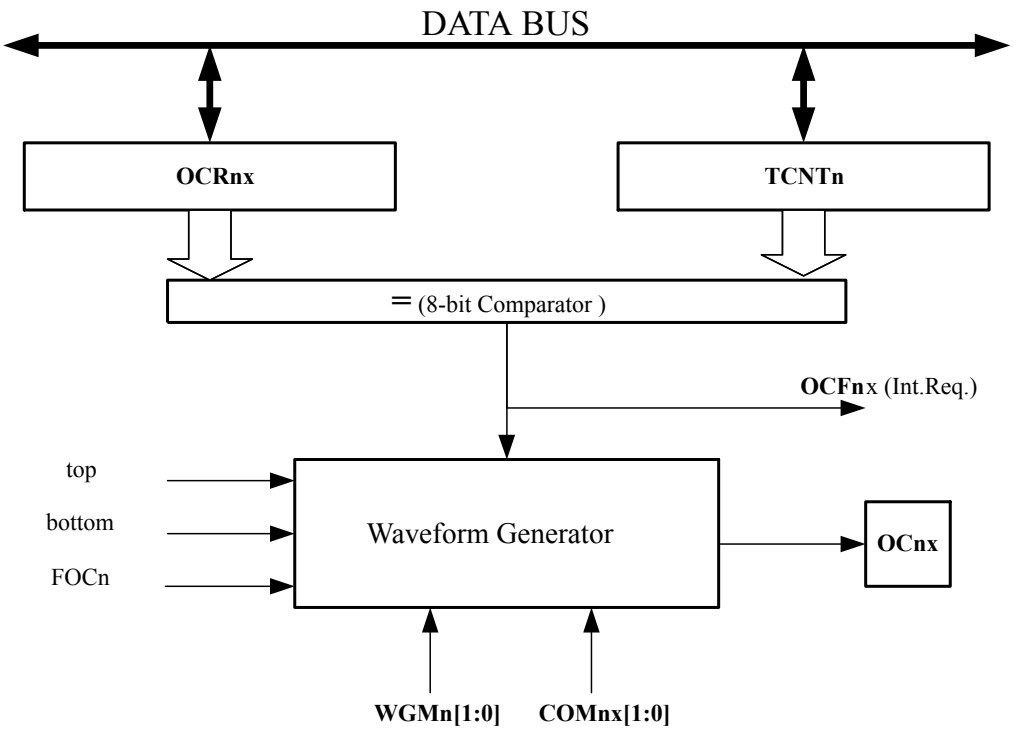
However, when using the register or bit definitions in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value.

The following definitions are used throughout the section:

**Table 16-1. Definitions**

| Constant | Description |
|---|---|
| BOTTOM | The counter reaches the BOTTOM when it becomes zero (0x00 for 8-bit counters, or 0x0000 for 16-bit counters). |

**Figure 16-3. Output Compare Unit, Block Diagram**




**Note:** The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. When double buffering is enabled, the CPU has access to the OCR0x Buffer Register. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The double buffering is disabled for the normal and Clear Timer on Compare (CTC) modes of operation, and the CPU will access the OCR0x directly.

### 16.5.1. Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a '1' to the Force Output Compare (TCCR0C.FOC0x) bit. Forcing compare match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the TCCR0A.COM0x[1:0] bits define whether the OC0x pin is set, cleared or toggled).

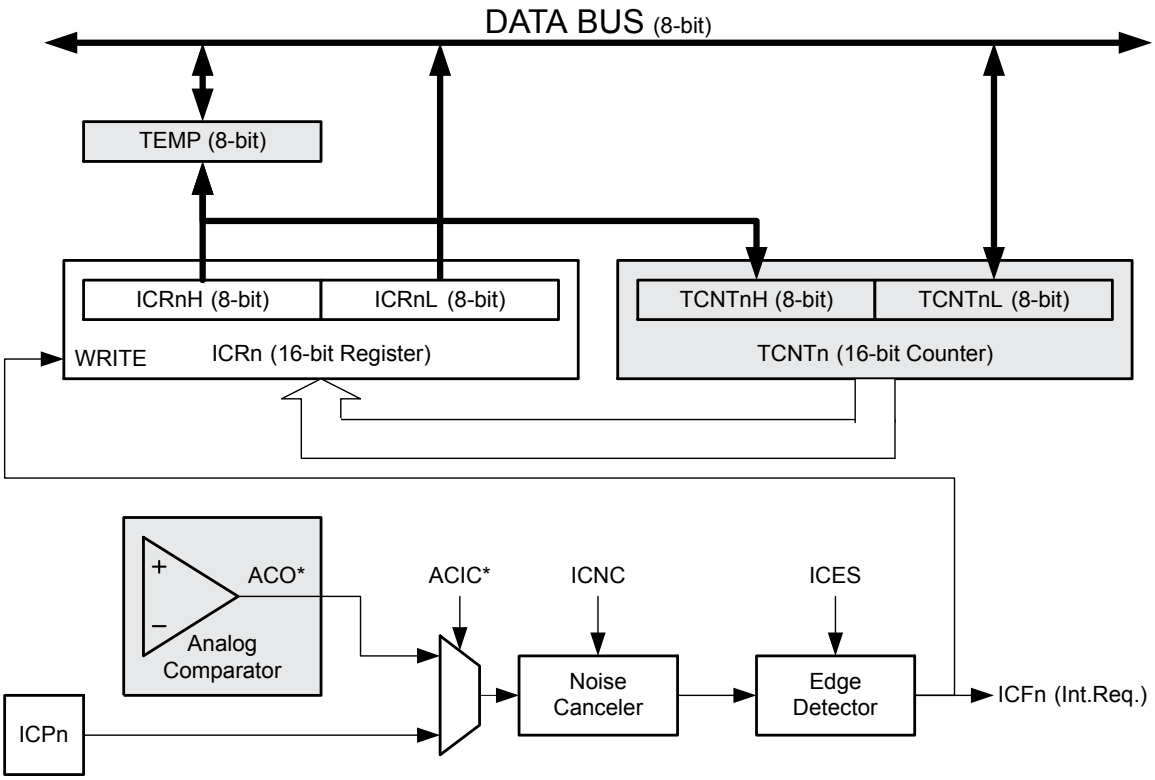
### 16.5.2. Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

### 16.5.3. Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

**Figure 17-3. Input Capture Unit Block Diagram for TC1**




**Note:** The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B).

When a change of the logic level (an event) occurs on the Input Capture pin (ICP1), or alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered: the 16-bit value of the counter (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF) is set at the same system clock cycle as the TCNT1 value is copied into the ICR1 Register. If enabled (TIMSK1.ICIE=1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF Flag can be cleared by software by writing '1' to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read form ICR1L, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter’s TOP value. In these cases the Waveform Generation mode bits (WGM1[3:0]) must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register, the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

See also Accessing 16-bit Registers.

### 17.9.1. Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in

In Fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Writing the COM1x[1:0] bits to 0x2 will produce an inverted PWM and a non-inverted PWM output can be generated by writing the COM1x[1:0] to 0x3. The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{\mathrm{OCnxPWM}} = \frac{f_{\mathrm{clk\_I/O}}}{N \cdot (1 + \mathrm{TOP})}$$

**Note:**

- The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x registers represents special cases when generating a PWM waveform output in the Fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output which is controlled by COM1x[1:0]).

A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OC1A to toggle its logical level on each compare match (COM1A[1:0]=0x1). This applies only if OCR1A is used to define the TOP value (WGM1[3:0]=0xF). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the Fast PWM mode.

### 17.12.4. Phase Correct PWM Mode

The Phase Correct Pulse Width Modulation or Phase Correct PWM modes (WGM1[3:0]= 0x1, 0x2, 0x3, 0xA, and 0xB) provide a high resolution, phase correct PWM waveform generation option. The Phase Correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the Phase Correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{\mathrm{PCPWM}} = \frac{\log(\mathrm{TOP}+1)}{\log(2)}$$

In Phase Correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0]= 0x1, 0x2, or 0x3), the value in ICR1 (WGM1[3:0]=0xA), or the value in OCR1A (WGM1[3:0]=0xB). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the Phase Correct PWM mode is shown below, using OCR1A or ICR1 to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The

| Mode | WGM13 | WGM12 (CTC1)[1] | WGM11 (PWM11)[1] | WGM10 (PWM10)[1] | Timer/ Counter Mode of Operation | TOP | Update of OCR1x at | TOV1 Flag Set on |
|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 0 | 1 | 1 | PWM, Phase Correct, 10-bit | 0x03FF | TOP | BOTTOM |
| 4 | 0 | 1 | 0 | 0 | CTC | OCR1A | Immediate | MAX |
| 5 | 0 | 1 | 0 | 1 | Fast PWM, 8-bit | 0x00FF | BOTTOM | TOP |
| 6 | 0 | 1 | 1 | 0 | Fast PWM, 9-bit | 0x01FF | BOTTOM | TOP |
| 7 | 0 | 1 | 1 | 1 | Fast PWM, 10-bit | 0x03FF | BOTTOM | TOP |
| 8 | 1 | 0 | 0 | 0 | PWM, Phase and Frequency Correct | ICR1 | BOTTOM | BOTTOM |
| 9 | 1 | 0 | 0 | 1 | PWM, Phase and Frequency Correct | OCR1A | BOTTOM | BOTTOM |
| 10 | 1 | 0 | 1 | 0 | PWM, Phase Correct | ICR1 | TOP | BOTTOM |
| 11 | 1 | 0 | 1 | 1 | PWM, Phase Correct | OCR1A | TOP | BOTTOM |
| 12 | 1 | 1 | 0 | 0 | CTC | ICR1 | Immediate | MAX |
| 13 | 1 | 1 | 0 | 1 | Reserved | - | - | - |
| 14 | 1 | 1 | 1 | 0 | Fast PWM | ICR1 | BOTTOM | TOP |
| 15 | 1 | 1 | 1 | 1 | Fast PWM | OCR1A | BOTTOM | TOP |

**Note:**

1. The CTC1 and PWM1[1:0] bit definition names are obsolete. Use the WGM1[3:0] definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

| Status Code (TWSRb) Prescaler Bits are 0 | Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware | Application SofTWARne Response | | | | | Next Action Taken by TWI Hardware |
|---|---|---|---|---|---|---|---|
| | | To/from TWDRn | To TWCRn | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0xC8 | Last data byte in TWDRn has been transmitted (TWEA = “0”);<br>ACK has been received | No TWDRn action | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode;<br>no recognition of own SLA or GCA |
| | | | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode;<br>own SLA will be recognized;<br>GCA will be recognized if TWGCE = “1” |
| | | | 1 | 0 | 1 | 0 | Switched to the not addressed Slave mode;<br>no recognition of own SLA or GCA;<br>a START condition will be transmitted when the bus<br>becomes free |
| | | | 1 | 0 | 1 | 1 | Switched to the not addressed Slave mode;<br>own SLA will be recognized;<br>GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free |

| Status Code (TWSR) Prescaler Bits are 0 | Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware | Application SofTWARne Response | | | | | Next Action Taken by TWI Hardware |
|---|---|---|---|---|---|---|---|
| | | To/from TWDRn | To TWCRn | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0xA0 | A STOP condition or repeated START condition has been received while still addressed as Slave | No action | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” |
| | | | 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 |
| | | | 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 |

### 23.9.4. TWI Data Register

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

**Name:** TWDR
**Offset:** 0xBB
**Reset:** 0xFF
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TWD7 | TWD6 | TWD5 | TWD4 | TWD3 | TWD2 | TWD1 | TWD0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – TWDn: TWI Data**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire Serial Bus.

### 25.8.1. ADC Multiplexer Selection Register

**Name:** ADMUX
**Offset:** 0x7C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:6 – REFSn: Reference Selection [n = 1:0]**
These bits select the voltage reference for the ADC. 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 25-3. ADC Voltage Reference Selection**

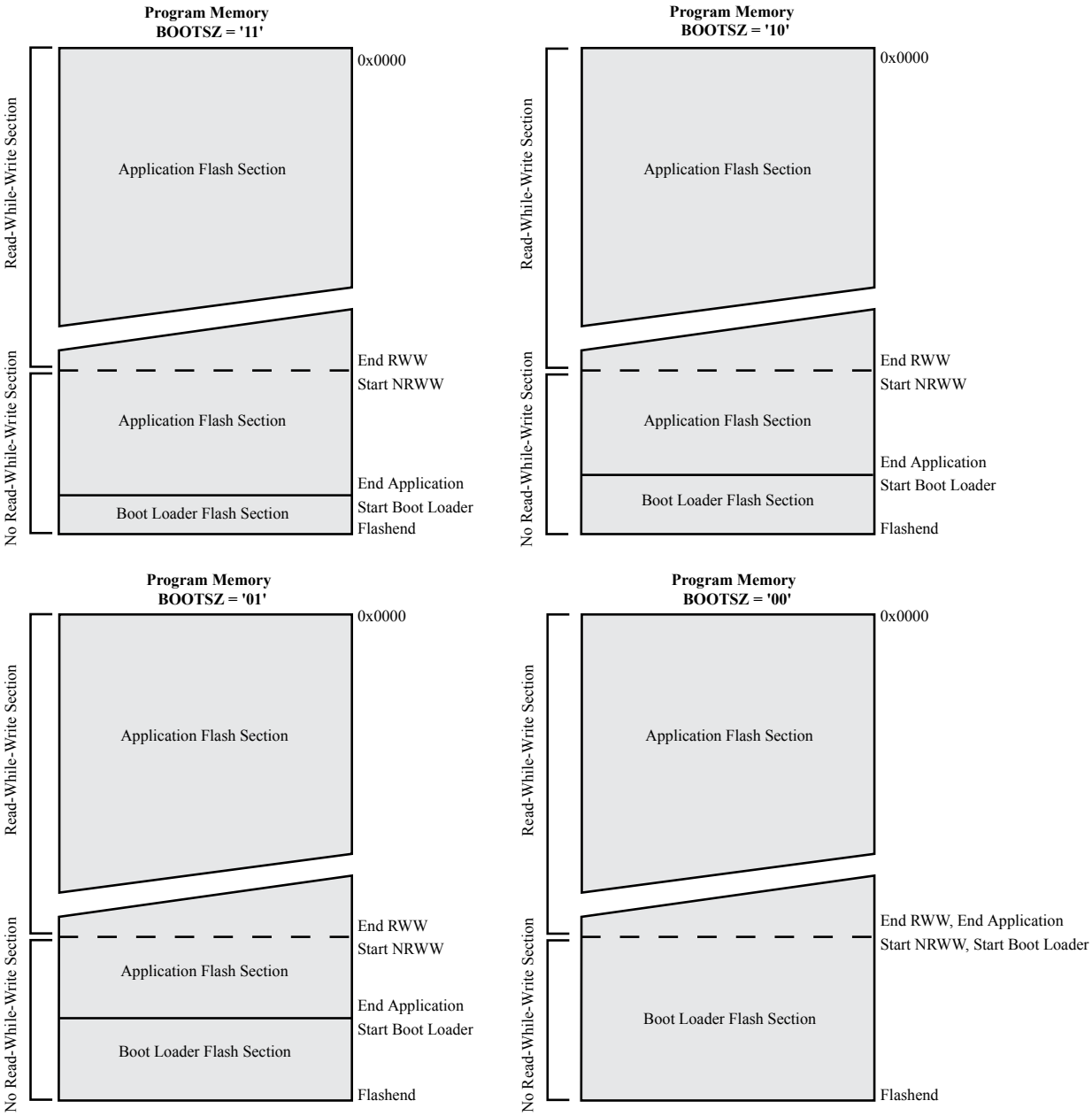
| REFS[1:0] | Voltage Reference Selection |
|---|---|
| 00 | AREF, Internal $V_{ref}$ turned off |
| 01 | $AV_{CC}$ with external capacitor at AREF pin |
| 10 | Internal 1.1V Voltage Reference with external capacitor at AREF pin |
| 11 | Internal 2.56V Voltage Reference with external capacitor at AREF pin |

**Note:** If differential channels are selected, only 2.56V should be used as Internal Voltage Reference.

**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 the ADCL and ADCH.

**Bits 4:0 – MUXn: Analog Channel and Gain Selection Bits [n = 4:0]**
The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. 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).

**Figure 27-2. Memory Sections**




## 27.5. Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. The fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

| Command Byte | Command Executed |
|---|---|
| 0010 0000 | Write Lock bits |
| 0001 0000 | Write Flash |
| 0001 0001 | Write EEPROM |
| 0000 1000 | Read Signature Bytes and Calibration byte |
| 0000 0100 | Read Fuse and Lock bits |
| 0000 0010 | Read Flash |
| 0000 0011 | Read EEPROM |

## 28.8. Parallel Programming

### 28.8.1. Enter Programming Mode

The following algorithm puts the device in Parallel (High-voltage) Programming mode:

1. Set Prog_enable pins listed in *Pin Values Used to Enter Programming Mode* of Signal Names section “0x0000”, $\overline{\text{RESET}}$ pin to 0V and $V_{CC}$ to 0V.
2. Apply 4.5 - 5.5V between $V_{CC}$ and GND.
   Ensure that $V_{CC}$ reaches at least 1.8V within the next 20µs.
3. Wait 20 - 60µs, and apply 11.5 - 12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Wait at least 300µs before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing $\overline{\text{RESET}}$ pin to 0V.

If the rise time of the $V_{CC}$ is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set Prog_enable pins listed in *Pin Values Used to Enter Programming Mode* of Signal Names section to “0000”, $\overline{\text{RESET}}$ pin to 0V and $V_{CC}$ to 0V.
2. Apply 4.5 - 5.5V between $V_{CC}$ and GND.
3. Monitor $V_{CC}$, and as soon as $V_{CC}$ reaches 0.9 - 1.1V, apply 11.5 - 12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Wait until $V_{CC}$ actually reaches 4.5 - 5.5V before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing $\overline{\text{RESET}}$ pin to 0V.

### 28.8.2. Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256byte EEPROM. This consideration also applies to Signature bytes reading.

### 28.8.3. Chip Erase

The Chip Erase will erase the Flash, the SRAM and the EEPROM memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

**Note:** The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

### 28.8.4. Programming the Flash

The Flash is organized in pages as number of Words in a Page and number of Pages in the Flash. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

**Step A. Load Command “Write Flash”**

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

**Step B. Load Address Low Byte**

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

**Step C. Load Data Low Byte**

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

**Step D. Load Data High Byte**

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

**Step E. Latch Data**

1. Set BS1 to “1”. This selects high data byte.

**Figure 30-23. I/O Pin Output Voltage vs. Source Current ($V_{CC}$ = 3V)**

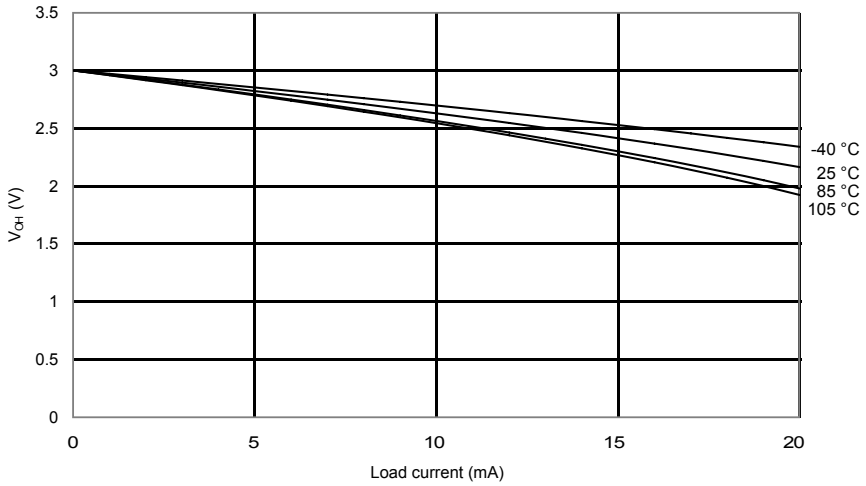



**Figure 30-24. I/O Pin Output Voltage vs. Source Current ($V_{CC}$ = 5V)**