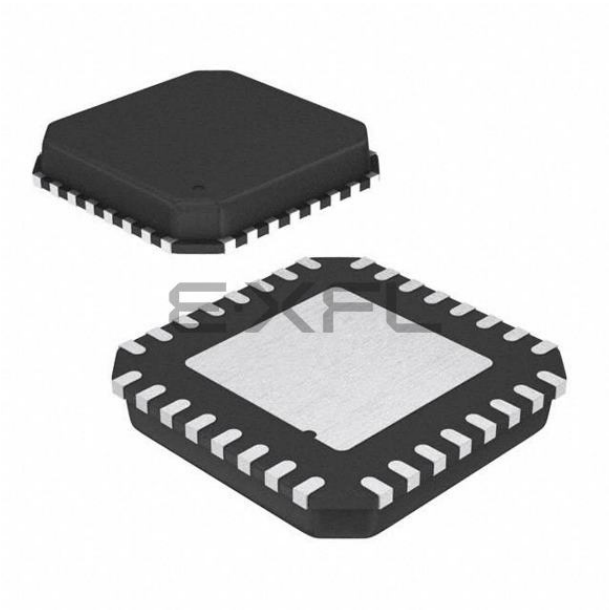




## What is "Embedded - Microcontrollers"?

"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 | 20MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 23 |
| Program Memory Size | 8KB (4K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 1K 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 | 32-VFQFN Exposed Pad |
| Supplier Device Package | 32-VQFN (5x5) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atmega88-20mi |

| Assembly Code Example |
| --- |

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEPE
  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 EEMPE
  sbi  EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi  EECR,EEPE
  ret
```

| C Code Example |
| --- |

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

### 6.1.4 Asynchronous Timer Clock – $clk_{ASY}$

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external clock or an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

### 6.1.5 ADC Clock – $clk_{ADC}$

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

## 6.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

**Table 6-1.** Device Clocking Options Select[(1)]

| Device Clocking Option | CKSEL3..0 |
|---|---|
| Low Power Crystal Oscillator | 1111 - 1000 |
| Full Swing Crystal Oscillator | 0111 - 0110 |
| Low Frequency Crystal Oscillator | 0101 - 0100 |
| Internal 128 kHz RC Oscillator | 0011 |
| Calibrated Internal RC Oscillator | 0010 |
| External Clock | 0000 |
| Reserved | 0001 |

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

### 6.2.1 Default Clock Source

The device is shipped with internal RC oscillator at 8.0MHz and with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock. The startup time is set to maximum and time-out period enabled. (CKSEL = "0010", SUT = "10", CKDIV8 = "0"). The default setting ensures that all users can make their desired clock source setting using any available programming interface.

### 6.2.2 Clock Startup Sequence

Any clock source needs a sufficient $V_{CC}$ to start oscillating and a minimum number of oscillating cycles before it can be considered stable.

To ensure sufficient $V_{CC}$, the device issues an internal reset with a time-out delay ($t_{TOUT}$) after the device reset is released by all other reset sources. ”System Control and Reset” on page 43 describes the start conditions for the internal reset. The delay ($t_{TOUT}$) is timed from the Watchdog Oscillator and the number of cycles in the delay is set by the SUTx and CKSELx fuse bits. The selectable delays are shown in Table 6-2. The frequency of the Watchdog Oscillator is voltage

### 8.0.5 Brown-out Detection

ATmega48/88/168 has an On-chip Brown-out Detection (BOD) circuit for monitoring the $V_{CC}$ level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

**Table 8-2.** BODLEVEL Fuse Coding[1]

| BODLEVEL 2..0 Fuses | Min $V_{BOT}$ | Typ $V_{BOT}$ | Max $V_{BOT}$ | Units |
|---|---|---|---|---|
| 111 | BOD Disabled | | | |
| 110 | 1.7[2] | 1.8 | 2.0[2] | V |
| 101 | 2.5[2] | 2.7 | 2.9[2] | |
| 100 | 4.1[2] | 4.3 | 4.5[2] | |
| 011 | Reserved | | | |
| 010 | | | | |
| 001 | | | | |
| 000 | | | | |

Notes: 1. $V_{BOT}$ may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-Out Reset will occur before $V_{CC}$ drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 110 and BODLEVEL = 101 for ATmega48V/88V/168V, and BODLEVEL = 101 and BODLEVEL = 101 for ATmega48/88/168.

2. Min/Max values applicable for ATmega48.

**Table 8-3.** Brown-out Characteristics

| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $V_{HYST}$ | Brown-out Detector Hysteresis | | 50 | | mV |
| $t_{BOD}$ | Min Pulse Width on Brown-out Reset | | 2 | | µs |

When the BOD is enabled, and $V_{CC}$ decreases to a value below the trigger level ($V_{BOT-}$ in Figure 8-5), the Brown-out Reset is immediately activated. When $V_{CC}$ increases above the trigger level ($V_{BOT+}$ in Figure 8-5), the delay counter starts the MCU after the Time-out period $t_{TOUT}$ has expired.

The BOD circuit will only detect a drop in $V_{CC}$ if the voltage stays below the trigger level for longer than $t_{BOD}$ given in Table 8-1.

### 11.1.2 External Interrupt Mask Register – EIMSK

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | INT1 | INT0 | EIMSK |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7..2 – Res: Reserved Bits**

These bits are unused bits in the ATmega48/88/168, and will always read as zero.

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

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

- **Bit 0 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

### 11.1.3 External Interrupt Flag Register – EIFR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | INTF1 | INTF0 | EIFR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7..2 – Res: Reserved Bits**

These bits are unused bits in the ATmega48/88/168, and will always read as zero.

- **Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

non-PWM modes refer to Table 13-1 on page 128. For fast PWM mode refer to Table 13-2 on page 128, and for phase correct and phase and frequency correct PWM refer to Table 13-3 on page 129.

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

## 13.8 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (See Section "13.7" on page 117.)

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 125.

### 13.8.1 Normal Mode

The simplest mode of operation is the *Normal mode* (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

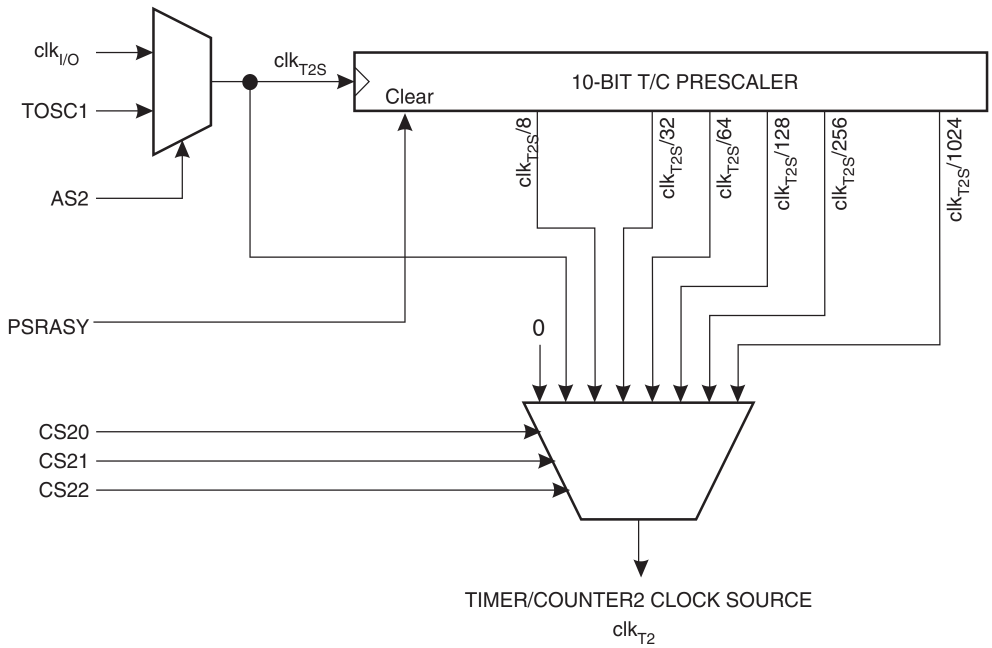
### 13.8.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 13-6. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

## 15.10 Timer/Counter Prescaler

**Figure 15-12.** Prescaler for Timer/Counter2




The clock source for Timer/Counter2 is named $clk_{T2S}$. $clk_{T2S}$ is by default connected to the main system I/O clock $clk_{IO}$. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a Real Time Counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port C. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. Applying an external clock source to TOSC1 is not recommended.

For Timer/Counter2, the possible prescaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, $clk_{T2S}$ as well as 0 (stop) may be selected. Setting the PSRASY bit in GTCCR resets the prescaler. This allows the user to operate with a predictable prescaler.

### 15.10.1 General Timer/Counter Control Register – GTCCR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TSM | – | – | – | – | – | PSRASY | PSRSYNC | GTCCR |
| Read/Write | R/W | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 1 – PSRASY: Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set. Refer to the description of the "Bit 7 – TSM: Timer/Counter Synchronization Mode" on page 137 for a description of the Timer/Counter Synchronization mode.

## 17.4 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter.

### 17.6.3 Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXCn) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXENn = 0), the receive buffer will be flushed and consequently the RXCn bit will become zero.

When the Receive Complete Interrupt Enable (RXCIEn) in UCSRnB is set, the USART Receive Complete interrupt will be executed as long as the RXCn Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDRn in order to clear the RXCn Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

### 17.6.4 Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FEn), Data OverRun (DORn) and Parity Error (UPEn). All can be accessed by reading UCSRnA. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSRnA must be read before the receive buffer (UDRn), since reading the UDRn I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRnA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FEn) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FEn Flag is zero when the stop bit was correctly read (as one), and the FEn Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FEn Flag is not affected by the setting of the USBSn bit in UCSRnC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRnA.

The Data OverRun (DORn) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. If the DORn Flag is set there was one or more serial frame lost between the frame last read from UDRn, and the next frame read from UDRn. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. The DORn Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPEn) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPEn bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see ”Parity Bit Calculation” on page 173 and ”Parity Checker” on page 181.

### 17.6.5 Parity Checker

The Parity Checker is active when the high USART Parity mode (UPMn1) bit is set. Type of Parity Check to be performed (odd or even) is selected by the UPMn0 bit. When enabled, the Parity Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (UPEn) Flag can then be read by software to check if the frame had a Parity Error.

# 18. USART in SPI Mode

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation. The Master SPI Mode (MSPIM) has the following features:

- **Full Duplex, Three-wire Synchronous Data Transfer**
- **Master Operation**
- **Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)**
- **LSB First or MSB First Data Transfer (Configurable Data Order)**
- **Queued Operation (Double Buffered)**
- **High Resolution Baud Rate Generator**
- **High Speed Operation (fXCKmax = fCK/2)**
- **Flexible Interrupt Generation**

## 18.1 Overview

Setting both UMSELn1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

## 18.2 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRRn setting can therefore be calculated using the same equations, see Table 18-1:

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREn) Flag and the Receive Complete (RXCn) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16 and the data received will be available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

**Table 19-5.** Status Codes for Slave Receiver Mode

| Status Code (TWSR) Prescaler Bits are 0 | Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware | Application Software Response | | | | | Next Action Taken by TWI Hardware |
|---|---|---|---|---|---|---|---|
| | | To/from TWDR | To TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0x60 | Own SLA+W has been received; ACK has been returned | No TWDR action or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | No TWDR action | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x68 | Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned | No TWDR action or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | No TWDR action | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x70 | General call address has been received; ACK has been returned | No TWDR action or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | No TWDR action | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x78 | Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned | No TWDR action or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | No TWDR action | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x80 | Previously addressed with own SLA+W; data has been received; ACK has been returned | Read data byte or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | Read data byte | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x88 | Previously addressed with own SLA+W; data has been received; NOT ACK has been returned | Read data byte or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | Read data byte or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” |
| | | Read data byte 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 |
| | | Read data byte | 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 |
| 0x90 | Previously addressed with general call; data has been received; ACK has been returned | Read data byte or | X | 0 | 1 | 0 | Data byte will be received and NOT ACK will be returned |
| | | Read data byte | X | 0 | 1 | 1 | Data byte will be received and ACK will be returned |
| 0x98 | Previously addressed with general call; data has been received; NOT ACK has been returned | Read data byte or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | Read data byte or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” |
| | | Read data byte 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 |
| | | Read data byte | 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 |
| 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 |

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

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

This bit is set 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. 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 ADCSRA, 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 written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

**Table 21-4.** ADC Prescaler Selections

| ADPS2 | ADPS1 | ADPS0 | Division Factor |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | 128 |

# 25. Memory Programming

## 25.1 Program And Data Memory Lock Bits

The ATmega88/168 provides six Lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 25-2. The Lock bits can only be erased to "1" with the Chip Erase command.The ATmega48 has no separate Boot Loader section. The SPM instruction is enabled for the whole Flash if the SELFPRGEN fuse is programmed ("0"), otherwise it is disabled.

**Table 25-1.** Lock Bit Byte[(1)]

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

Notes: 1. "1" means unprogrammed, "0" means programmed
2. Only on ATmega88/168.

**Table 25-2.** 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)] |

Notes: 1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. "1" means unprogrammed, "0" means programmed

**Table 25-14.** Parallel Programming Characteristics, $V_{CC}$ = 5V ± 10% (Continued)

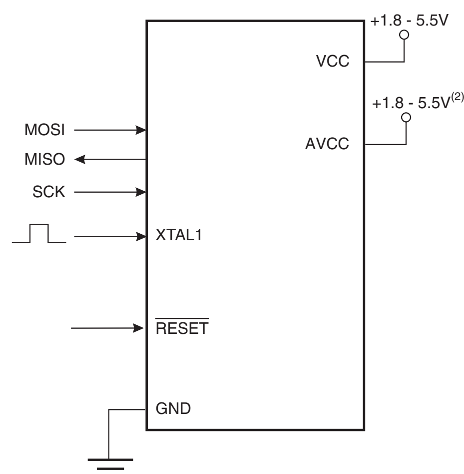
| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $t_{BVDV}$ | BS1 Valid to DATA valid | 0 | | 250 | ns |
| $t_{OLDV}$ | $\overline{OE}$ Low to DATA Valid | | | 250 | ns |
| $t_{OHDZ}$ | $\overline{OE}$ High to DATA Tri-stated | | | 250 | ns |

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

## 25.8 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 25-15 on page 296, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

**Figure 25-10.** Serial Programming and Verify[1]




Notes: 1. If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.
2. $V_{CC}$ - 0.3V < $AV_{CC}$ < $V_{CC}$ + 0.3V, however, $AV_{CC}$ should always be within 1.8 - 5.5V

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for $f_{ck}$ < 12 MHz, 3 CPU clock cycles for $f_{ck}$ >= 12 MHz

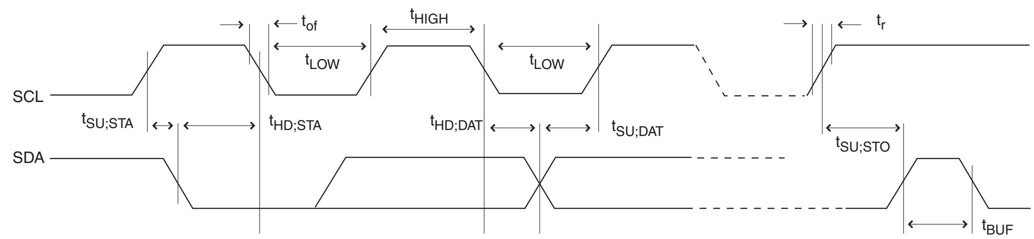
High: > 2 CPU clock cycles for $f_{ck}$ < 12 MHz, 3 CPU clock cycles for $f_{ck}$ >= 12 MHz

**Table 26-2.** 2-wire Serial Bus Requirements (Continued)

| Symbol | Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|---|
| $C_i$[1] | Capacitance for each I/O Pin | | – | 10 | pF |
| $f_{SCL}$ | SCL Clock Frequency | $f_{CK}$[4] > max($16f_{SCL}$, 250kHz)[5] | 0 | 400 | kHz |
| Rp | Value of Pull-up resistor | $f_{SCL} \le 100$ kHz | $\frac{V_{CC} - 0{,}4V}{3mA}$ | $\frac{1000ns}{C_b}$ | Ω |
| | | $f_{SCL} > 100$ kHz | $\frac{V_{CC} - 0{,}4V}{3mA}$ | $\frac{300ns}{C_b}$ | Ω |
| $t_{HD;STA}$ | Hold Time (repeated) START Condition | $f_{SCL} \le 100$ kHz | 4.0 | – | µs |
| | | $f_{SCL} > 100$ kHz | 0.6 | – | µs |
| $t_{LOW}$ | Low Period of the SCL Clock | $f_{SCL} \le 100$ kHz[6] | 4.7 | – | µs |
| | | $f_{SCL} > 100$ kHz[7] | 1.3 | – | µs |
| $t_{HIGH}$ | High period of the SCL clock | $f_{SCL} \le 100$ kHz | 4.0 | – | µs |
| | | $f_{SCL} > 100$ kHz | 0.6 | – | µs |
| $t_{SU;STA}$ | Set-up time for a repeated START condition | $f_{SCL} \le 100$ kHz | 4.7 | – | µs |
| | | $f_{SCL} > 100$ kHz | 0.6 | – | µs |
| $t_{HD;DAT}$ | Data hold time | $f_{SCL} \le 100$ kHz | 0 | 3.45 | µs |
| | | $f_{SCL} > 100$ kHz | 0 | 0.9 | µs |
| $t_{SU;DAT}$ | Data setup time | $f_{SCL} \le 100$ kHz | 250 | – | ns |
| | | $f_{SCL} > 100$ kHz | 100 | – | ns |
| $t_{SU;STO}$ | Setup time for STOP condition | $f_{SCL} \le 100$ kHz | 4.0 | – | µs |
| | | $f_{SCL} > 100$ kHz | 0.6 | – | µs |
| $t_{BUF}$ | Bus free time between a STOP and START condition | $f_{SCL} \le 100$ kHz | 4.7 | – | µs |
| | | $f_{SCL} > 100$ kHz | 1.3 | – | µs |

Notes:
1. In ATmega48/88/168, this parameter is characterized and not 100% tested.
2. Required only for $f_{SCL} > 100$ kHz.
3. $C_b$ = capacitance of one bus line in pF.
4. $f_{CK}$ = CPU clock frequency
5. This requirement applies to all ATmega48/88/168 2-wire Serial Interface operation. Other devices connected to the 2-wire Serial Bus need only obey the general $f_{SCL}$ requirement.
6. The actual low period generated by the ATmega48/88/168 2-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus $f_{CK}$ must be greater than 6 MHz for the low time requirement to be strictly met at $f_{SCL}$ = 100 kHz.
7. The actual low period generated by the ATmega48/88/168 2-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus the low time requirement will not be strictly met for $f_{SCL} > 308$ kHz when $f_{CK}$ = 8 MHz. Still, ATmega48/88/168 devices connected to the bus may communicate at full speed (400 kHz) with other ATmega48/88/168 devices, as well as any other device with a proper $t_{LOW}$ acceptance margin.

**Figure 26-4.** 2-wire Serial Bus Timing




## 26.7 SPI Timing Characteristics

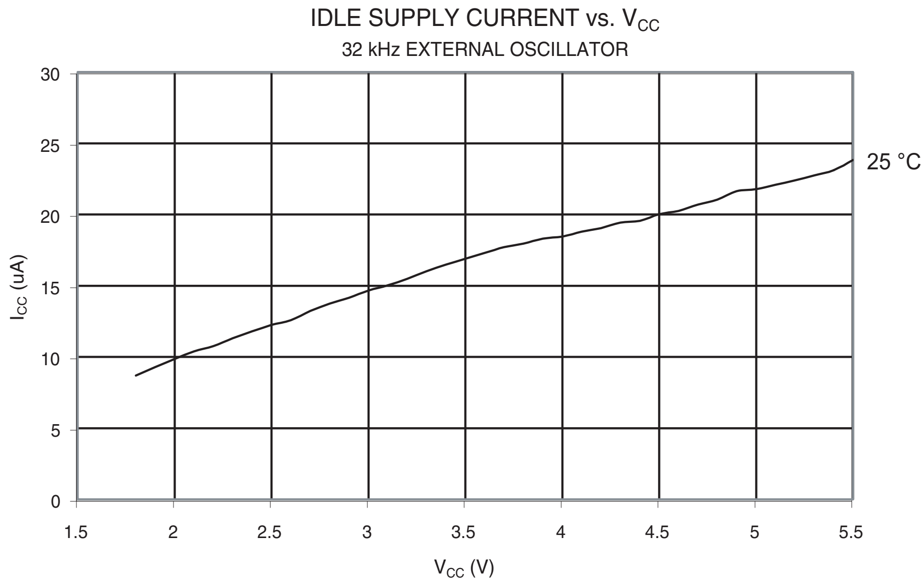
See Figure 26-5 and Figure 26-6 for details.

**Table 26-3.** SPI Timing Parameters

| | Description | Mode | Min | Typ | Max | |
|---|---|---|---|---|---|---|
| 1 | SCK period | Master | | See Table 16-4 | | |
| 2 | SCK high/low | Master | | 50% duty cycle | | |
| 3 | Rise/Fall time | Master | | 3.6 | | |
| 4 | Setup | Master | | 10 | | |
| 5 | Hold | Master | | 10 | | |
| 6 | Out to SCK | Master | | $0.5 \bullet t_{sck}$ | | |
| 7 | SCK to out | Master | | 10 | | |
| 8 | SCK to out high | Master | | 10 | | |
| 9 | $\overline{SS}$ low to out | Slave | | 15 | | ns |
| 10 | SCK period | Slave | $4 \bullet t_{ck}$ | | | |
| 11 | SCK high/low(1) | Slave | $2 \bullet t_{ck}$ | | | |
| 12 | Rise/Fall time | Slave | | | 1600 | |
| 13 | Setup | Slave | 10 | | | |
| 14 | Hold | Slave | $t_{ck}$ | | | |
| 15 | SCK to out | Slave | | 15 | | |
| 16 | SCK to $\overline{SS}$ high | Slave | 20 | | | |
| 17 | $\overline{SS}$ high to tri-state | Slave | | 10 | | |
| 18 | $\overline{SS}$ low to SCK | Slave | 20 | | | |

Note:
1. In SPI Programming mode the minimum SCK high/low period is:
   - 2 $t_{CLCL}$ for $f_{CK}$ < 12 MHz
   - 3 $t_{CLCL}$ for $f_{CK}$ > 12 MHz
2. All DC Characteristics contained in this datasheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.

**Figure 27-12.** Idle Supply Current vs. $V_{CC}$ (32 kHz External Oscillator)




## 27.3 Supply Current of I/O modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the Power Reduction Register. See "Power Reduction Register" on page 39 for details.

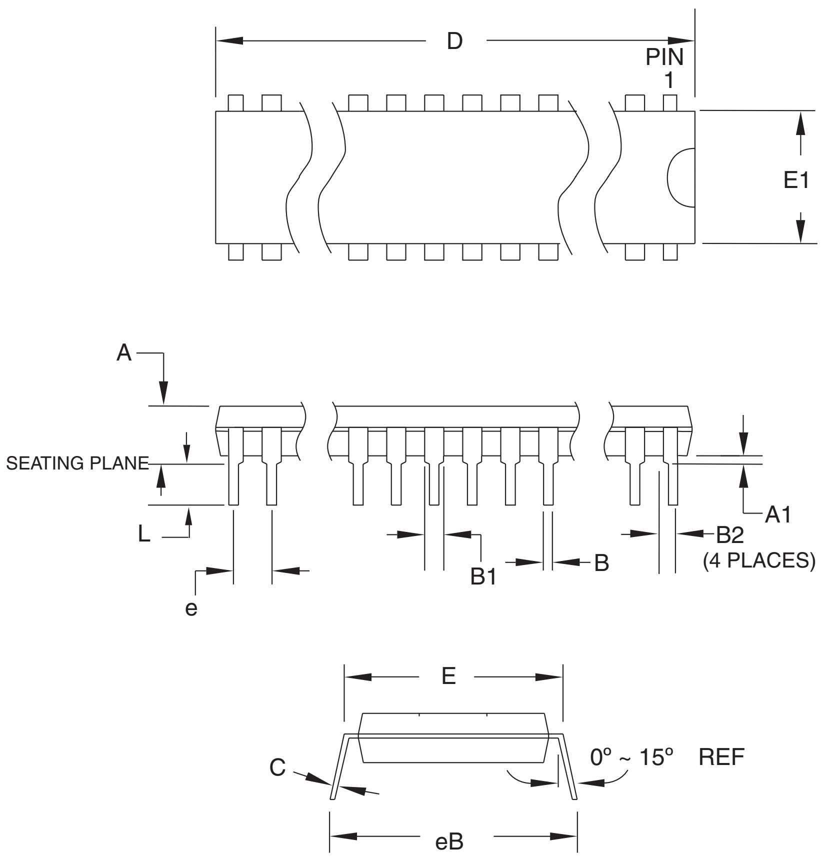
**Table 27-1.** Additional Current Consumption for the different I/O modules (absolute values)

| PRR bit | Typical numbers | | |
|---|---|---|---|
| | $V_{CC}$ = 2V, F = 1MHz | $V_{CC}$ = 3V, F = 4MHz | $V_{CC}$ = 5V, F = 8MHz |
| PRUSART0 | 8.0 uA | 51 uA | 220 uA |
| PRTWI | 12 uA | 75 uA | 315 uA |
| PRTIM2 | 11 uA | 72 uA | 300 uA |
| PRTIM1 | 5.0 uA | 32 uA | 130 uA |
| PRTIM0 | 4.0 uA | 24 uA | 100 uA |
| PRSPI | 15 uA | 95 uA | 400 uA |
| PRADC | 12 uA | 75 uA | 315 uA |

# 28. Register Summary

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| (0xFF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xED) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC6) | UDR0 | USART I/O Data Register | | | | | | | | 187 |
| (0xC5) | UBRR0H | | | | | USART Baud Rate Register High | | | | 191 |
| (0xC4) | UBRR0L | USART Baud Rate Register Low | | | | | | | | 191 |
| (0xC3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC2) | UCSR0C | UMSEL01 | UMSEL00 | UPM01 | UPM00 | USBS0 | UCSZ01 /UDORD0 | UCSZ00 / UCPHA0 | UCPOL0 | 189/203 |
| (0xC1) | UCSR0B | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | UCSZ02 | RXB80 | TXB80 | 188 |
| (0xC0) | UCSR0A | RXC0 | TXC0 | UDRE0 | FE0 | DOR0 | UPE0 | U2X0 | MPCM0 | 187 |

## 31.2 28P3




**COMMON DIMENSIONS**
(Unit of Measure = mm)

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | – | – | 4.5724 | |
| A1 | 0.508 | – | – | |
| D | 34.544 | – | 34.798 | Note 1 |
| E | 7.620 | – | 8.255 | |
| E1 | 7.112 | – | 7.493 | Note 1 |
| B | 0.381 | – | 0.533 | |
| B1 | 1.143 | – | 1.397 | |
| B2 | 0.762 | – | 1.143 | |
| L | 3.175 | – | 3.429 | |
| C | 0.203 | – | 0.356 | |
| eB | – | – | 10.160 | |
| e | 2.540 TYP | | | |

Note: 1. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

| | TITLE | DRAWING NO. | REV. |
|---|---|---|---|
| ATMEL 2325 Orchard Parkway San Jose, CA 95131 | **28P3**, 28-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP) | 28P3 | B |

The Asynchronous oscillator does not stop when entering power down mode. This leads to higher power consumption than expected.

**Problem fix / Workaround**

Manually disable the asynchronous timer before entering power down.

## 32.2 Errata ATmega88

The revision letter in this section refers to the revision of the ATmega88 device.

### 32.2.1 Rev. A

- **Writing to EEPROM does not work at low Operating Voltages**
- **Part may hang in reset**

**1. Writing to EEPROM does not work at low operating voltages**

Writing to the EEPROM does not work at low voltages.

**Problem Fix/Workaround**

Do not write the EEPROM at voltages below 4.5 Volts.
This will be corrected in rev. B.

**2. Part may hang in reset**

Some parts may get stuck in a reset state when a reset signal is applied when the internal reset state-machine is in a specific state. The internal reset state-machine is in this state for approximately 10 ns immediately before the part wakes up after a reset, and in a 10 ns window when altering the system clock prescaler. The problem is most often seen during In-System Programming of the device. There are theoretical possibilities of this happening also in run-mode. The following three cases can trigger the device to get stuck in a reset-state:

- Two succeeding resets are applied where the second reset occurs in the 10ns window before the device is out of the reset-state caused by the first reset.

- A reset is applied in a 10 ns window while the system clock prescaler value is updated by software.

- Leaving SPI-programming mode generates an internal reset signal that can trigger this case.

The two first cases can occur during normal operating mode, while the last case occurs only during programming of the device.

**Problem Fix/Workaround**

The first case can be avoided during run-mode by ensuring that only one reset source is active. If an external reset push button is used, the reset start-up time should be selected such that the reset line is fully debounced during the start-up time.

The second case can be avoided by not using the system clock prescaler.

The third case occurs during In-System programming only. It is most frequently seen when using the internal RC at maximum frequency.

If the device gets stuck in the reset-state, turn power off, then on again to get the device out of this state.

### 32.2.2 Rev. D

No errata.