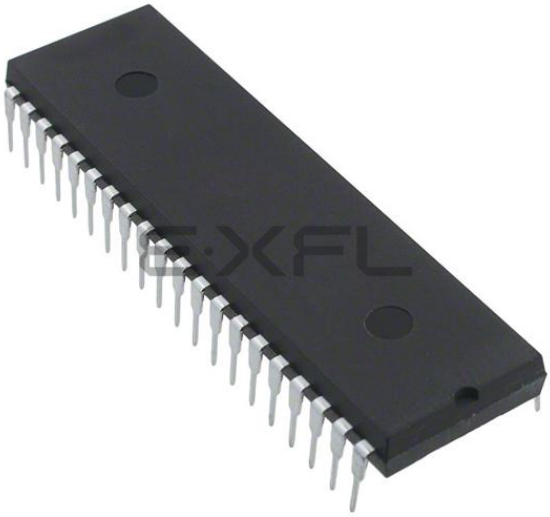


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

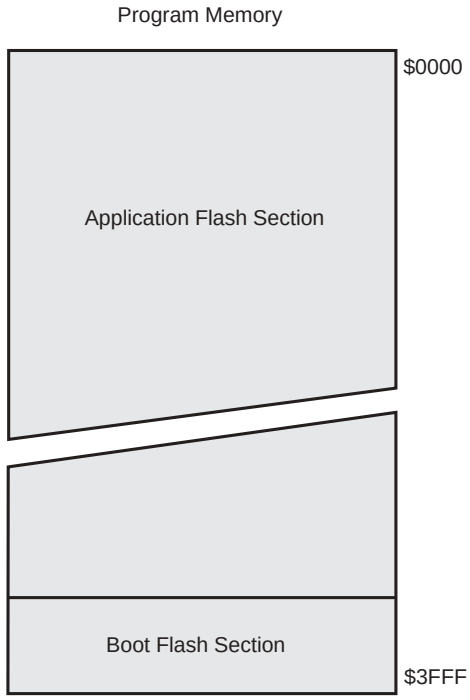
| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 4MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 32 |
| Program Memory Size | 32KB (16K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C |
| Mounting Type | Through Hole |
| Package / Case | 40-DIP (0.600", 15.24mm) |
| Supplier Device Package | 40-PDIP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atmega323l-4pi |

The 2K bytes data SRAM can be easily accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

**Figure 6.** Memory Maps



### Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The maximum start-up time is TBD. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODEN Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit, the user must always allow the reference to start up before the output from the Analog Comparator is used. The bandgap reference uses typically 10 µA, and to reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

### Interrupt Handling

The ATmega323 has two 8-bit Interrupt Mask Control Registers: GICR – General Interrupt Control Register and TIMSK – Timer/Counter Interrupt Mask Register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction – RETI – is executed.

When the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

If an interrupt condition occurs while the corresponding interrupt enable bit is cleared (zero), the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.

If one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared (zero), the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set (one), and will be executed by order of priority.

Note that external level interrupt does not have a flag, and will only be remembered for as long as the interrupt condition is present.

Note that the Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

### Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the Program Vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter (14 bits) is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-flag in SREG is set. When AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

## Calibrated Internal RC Oscillator

The calibrated internal Oscillator provides a fixed 1.0 MHz (nominal) clock at 5V and 25°C. This clock may be used as the system clock. See the section “Clock Options” on page 6 for information on how to select this clock as the system clock. This Oscillator can be calibrated by writing the calibration byte to the OSCCAL Register. When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For details on how to use the pre-programmed calibration value, see the section “Calibration Byte” on page 188.

### Oscillator Calibration Register – OSCCAL

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $31 ($51) | CAL7 | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 | OSCCAL |
| 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 | |

- **Bits 7..0 – CAL7..0: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the internal Oscillator to remove process variations from the Oscillator frequency. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the internal Oscillator. Writing $FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0 MHz, thus tuning to other values is not guaranteed.

**Table 11.** Internal RC Oscillator Frequency Range

| OSCCAL Value | Min Frequency (MHz) | Max Frequency (MHz) |
|---|---|---|
| $00 | 0.5 | 1.0 |
| $7F | 0.7 | 1.5 |
| $FF | 1.0 | 2.0 |

### Special Function IO Register – SFIOR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $30 ($50) | – | – | – | – | ACME | PUD | PSR2 | PSR10 | SFIOR |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7..4 – Res: Reserved Bits**

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

- **Bit 3 – ACME: Analog Comparator Multiplexer Enable**

When this bit is set (one) and the ADC is switched off (ADEN in ADCSR is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is cleared (zero), AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see “Analog Comparator Multiplexed Input” on page 126.

- **Bit 2 – PUD: Pull-up Disable**

When this bit is set (one), all pull-ups on all ports are disabled. If the bit is cleared (zero), the pull-ups can be individually enabled as described in the chapter “I/O Ports” on page 137.

**TCNT1 Timer/Counter1 Write**

When the CPU writes to the High Byte TCNT1H, the written data is placed in the TEMP Register. Next, when the CPU writes the Low Byte TCNT1L, this byte of data is combined with the byte data in the TEMP Register, and all 16-bits are written to the TCNT1 Timer/Counter1 Register simultaneously. Consequently, the High Byte TCNT1H must be accessed first for a full 16-bit register write operation.

**TCNT1 Timer/Counter1 Read**

When the CPU reads the Low Byte TCNT1L, the data of the Low Byte TCNT1L is sent to the CPU and the data of the High Byte TCNT1H is placed in the TEMP Register. When the CPU reads the data in the High Byte TCNT1H, the CPU receives the data in the TEMP Register. Consequently, the Low Byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

**Timer/Counter1 Output Compare Register – OCR1AH and OCR1AL**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $2B ($4B) | **MSB** | | | | | | | | **OCR1AH** |
| $2A ($4A) | | | | | | | | **LSB** | **OCR1AL** |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Timer/Counter1 Output Compare Register – OCR1BH and OCR1BL**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $29 ($49) | **MSB** | | | | | | | | **OCR1BH** |
| $28 ($48) | | | | | | | | **LSB** | **OCR1BL** |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

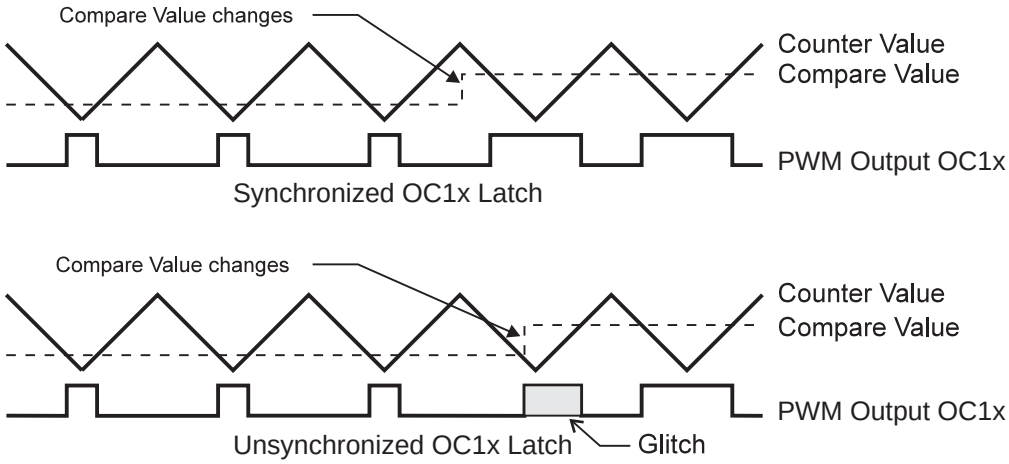
The Output Compare Registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare Registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status Register. A software write to the Timer/Counter Register blocks compare matches in the next Timer/Counter clock cycle. This prevents immediate interrupts when initializing the Timer/Counter.

A Compare Match will set the Compare Interrupt Flag in the CPU clock cycle following the compare event.

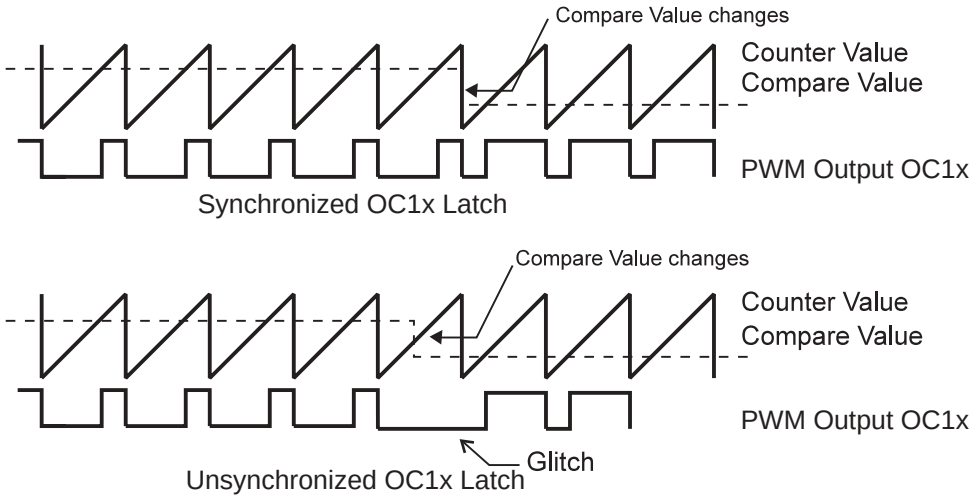
Since the Output Compare Registers – OCR1A and OCR1B – are 16-bit registers, a temporary register TEMP is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the High Byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP Register. When the CPU writes the Low Byte, OCR1AL or OCR1BL, the TEMP Register is simultaneously written to OCR1AH or OCR1BH. Consequently, the High Byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

**Figure 38.** Effects of Unsynchronized OCR1 Latching.




Note: x = A or B

**Figure 39.** Effects of Unsynchronized OCR1 Latching in Overflow Mode




Note: X = A or B

During the time between the write and the latch operation, a read from OCR1A or OCR1B will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A/B.

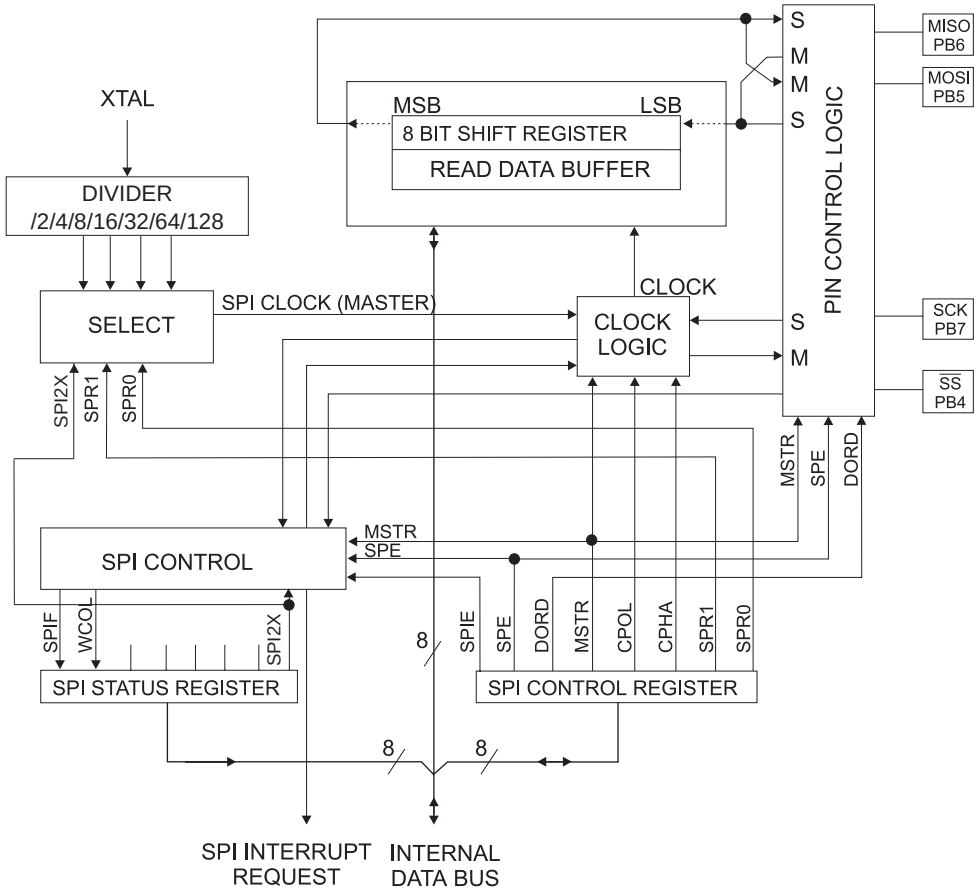
When the OCR1X contains $0000 or TOP, and the up/down PWM mode is selected, the output OC1A/OC1B is updated to low or high on the next compare match according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 23. In overflow PWM mode, the output OC1A/OC1B is held low or high only when the Output Compare Register contains TOP.

## Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega323 and peripheral devices or between several AVR devices. The ATmega323 SPI includes the following features:

- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

**Figure 41.** SPI Block Diagram




The interconnection between Master and Slave CPUs with SPI is shown in Figure 42. The PB7(SCK) pin is the clock output in the Master mode and the clock input in the Slave mode. Writing to the SPI Data Register of the Master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the Slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Slave Select input, PB4($\overline{SS}$), is set low to select an individual Slave SPI device. The two Shift Registers in the Master and the Slave can be considered as one distributed 16-bit circular Shift Register. This is shown in Figure 42. When data is shifted from the Master to the Slave, data is also shifted in the opposite direction, simultaneously. During one shift cycle, data in the Master and the Slave is interchanged.

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared (zero) by first reading the SPI Status Register with WCOL set (one), and then accessing the SPI Data Register.

- **Bit 5..1 – Res: Reserved Bits**

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

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is set (one) the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 27). This means that the minimum SCK period will be 2 CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{ck}/4$ or lower.

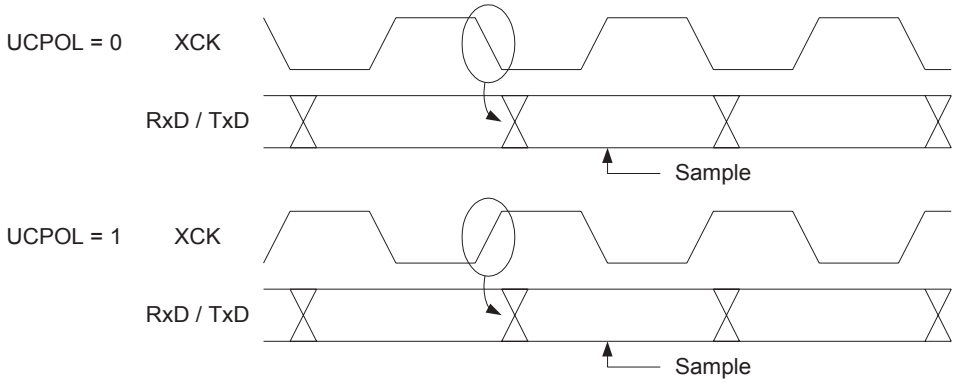
The SPI interface on the ATmega323 is also used for Program memory and EEPROM downloading or uploading. See page 197 for Serial Programming and verification.

**The SPI Data Register – SPDR**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $0F ($2F) | MSB | | | | | | | LSB | SPDR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | X | X | X | X | X | X | X | X | Undefined |

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

**Figure 47.** Synchronous Mode XCK Timing




The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 47 shows, when UCPOL is zero the data will be changed at falling XCK edge and sampled at rising XCK edge. If UCPOL is set, the data will be changed at rising XCK edge and sampled at falling XCK edge.

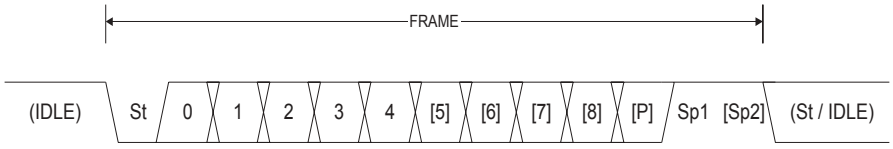
## Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accept all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even, or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to a idle (high) state. Figure 48 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

**Figure 48.** Frame Formats




St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

### Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the Low Byte of the character written to UDR. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in Registers R17:R16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Copy ninth bit from r17 to TXB8
  cbi   UCSRB,TXB8
  sbrc  r17,0
  sbi   UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE))) ) {};
  /* Copy ninth bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. I.e., only the TXB8 bit of the UCSRB Register is used after initialization.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

### Transmitter Flags and Interrupts

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

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

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRB is set, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register

## Multi-processor Communication Mode

Setting the Multi-Processor Communication mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contain data or address information. If the Receiver is set up for frames with 9 data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication mode enables several Slave MCUs to receive data from a Master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular Slave MCU has been addressed, it will receive the following data frames as normal, while the other Slave MCUs will ignore the received frames until another address frame is received.

### Using MPCM

For an MCU to act as a Master MCU, it can use a 9-bit character frame format (UCSZ = 7). The ninth bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The Slave MCUs must in this case be set to use a 9-bit character frame format.

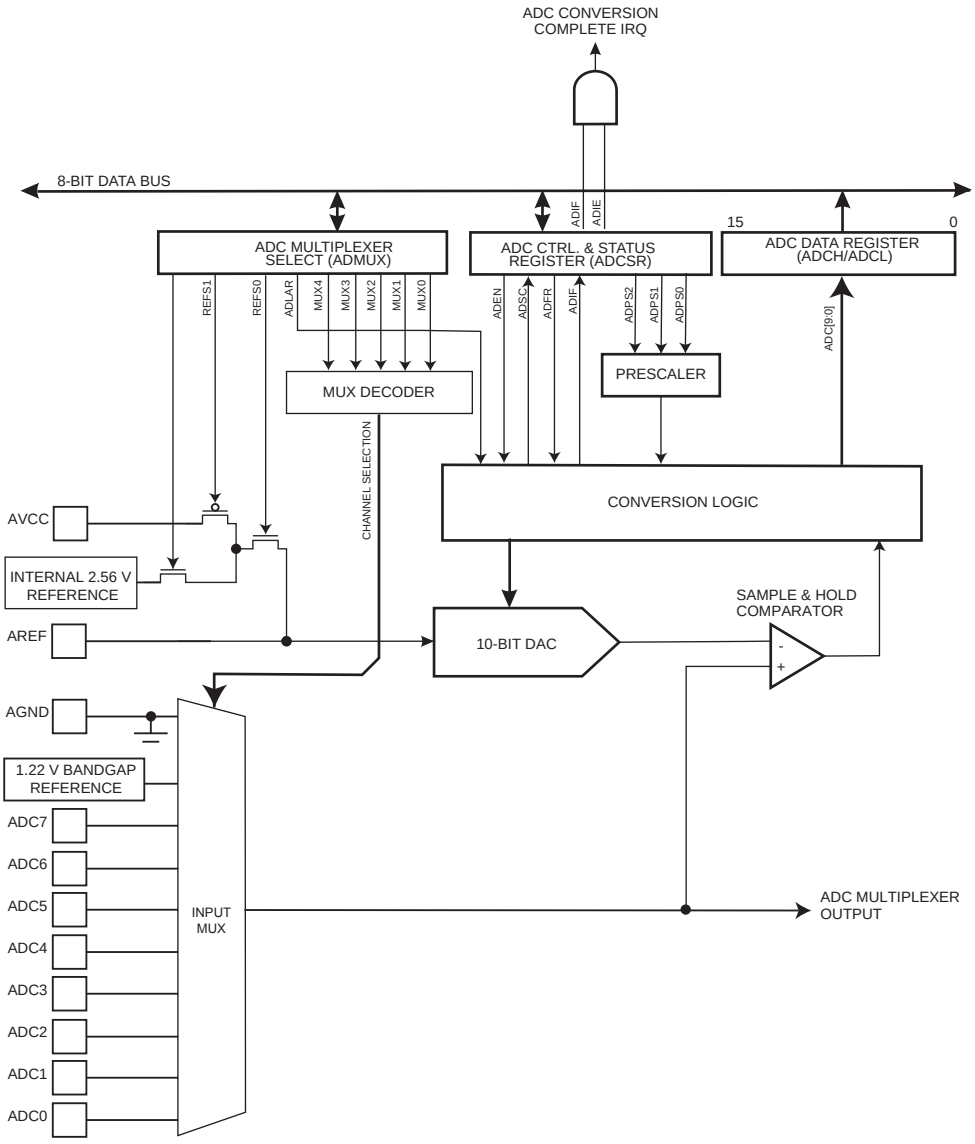
The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor Communication mode (MPCM in UCSRA is set).
2. The Master MCU sends an address frame, and all Slaves Receive and read this frame. In the Slave MCUs, the RXC Flag in UCSRA will be set as normal.
3. Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from Master. The process then repeats from 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the Transmitter and Receiver uses the same character size setting. If 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.

**Figure 60.** Analog to Digital Converter Block Schematic



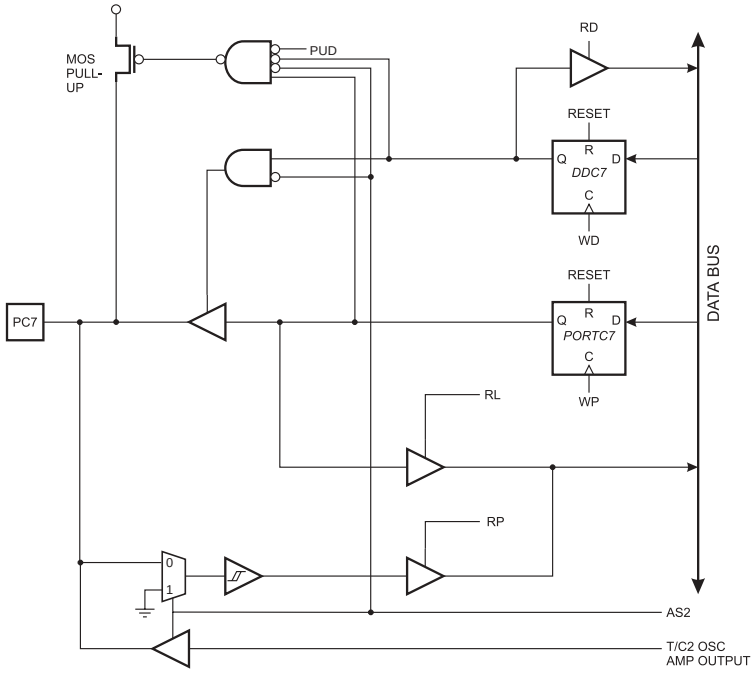

## Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents AGND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the eight ADC input pins ADC7..0, as well as AGND and a fixed bandgap voltage reference of nominally 1.22 V ($V_{BG}$), can be selected as single ended inputs to the ADC.

The ADC can operate in two modes – Single Conversion and Free Running mode. In Single Conversion mode, each conversion will have to be initiated by the user. In Free Running mode, the ADC is constantly sampling and updating the ADC Data Register. The ADFR bit in ADCSR selects between the two available modes.

**Figure 78.** Port C Schematic Diagram (Pins PC7)




WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
AS2: ASYNCH SELECT T/C2
PUD: PULL-UP DISABLE

### Port D As General Digital I/O

PDn, General I/O pin: The DDDn bit in the DDRD Register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin the MOS pull up resistor is activated. To switch the pull up resistor off the PDn has to be cleared (zero), the pin has to be configured as an output pin, or the PUD bit has to be set. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Table 54.** DDDn Effects on Port D Pins

| DDDn | PORTDn | PUD (in SFIOR) | I/O | Pull Up | Comment |
|---|---|---|---|---|---|
| 0 | 0 | X | Input | No | Tri-state (Hi-Z) |
| 0 | 1 | 0 | Input | Yes | PDn will Source Current if Ext. Pulled Low. |
| 0 | 1 | 1 | Input | No | Tri-state (Hi-Z) |
| 1 | 0 | X | Output | No | Push-pull Zero Output |
| 1 | 1 | X | Output | No | Push-pull One Output |

Note: n: 7,6…0, pin number.

### Alternate Functions of Port D

- **OC2 – Port D, Bit 7**

OC2, Timer/Counter2 Output Compare Match output: The PD7 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDD7 set (one)) to serve this function. See the timer description on how to enable this function. The OC2 pin is also the output pin for the PWM mode timer function.

- **ICP – Port D, Bit 6**

ICP – Input Capture Pin: The PD6 pin can act as an Input Capture Pin for Timer/Counter1. The pin has to be configured as an input (DDD6 cleared(zero)) to serve this function. See the timer description on how to enable this function.

- **OC1A – Port D, Bit 5**

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. See the timer description on how to enable this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **OC1B – Port D, Bit 4**

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. See the timer description on how to enable this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

**Figure 82.** Port D Schematic Diagram (Pins PD4 and PD5)

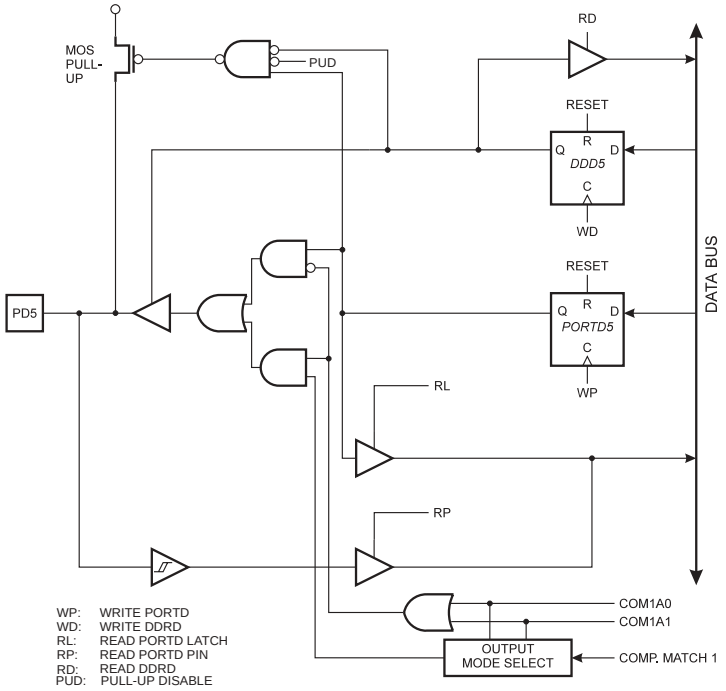
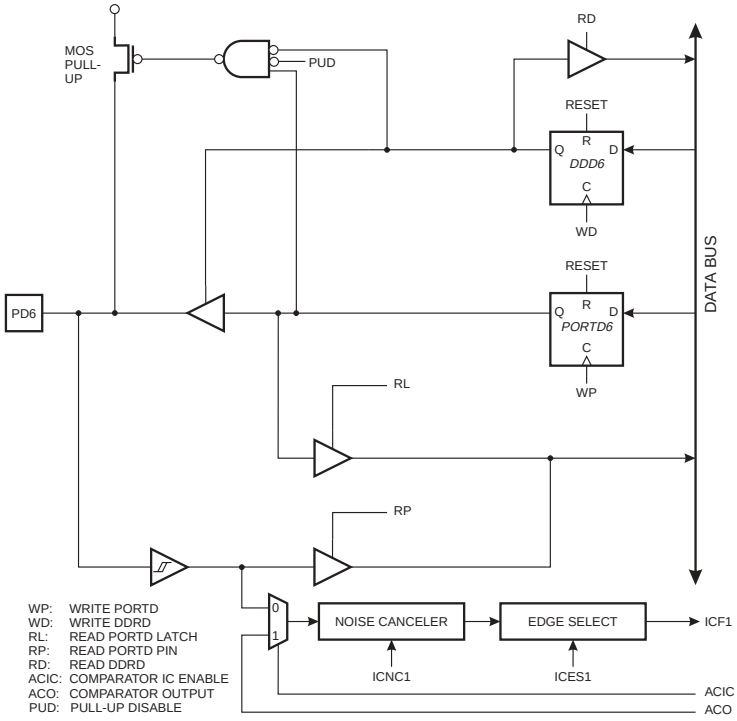



**Figure 83.** Port D Schematic Diagram (Pin PD6)



mentation and JTAG instructions are therefore irrelevant for the user of the On-chip Debug system.

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the On-chip debug system to work. The disabling of the On-chip debug system when any Lock bits are set is a security feature. Otherwise, the On-chip debug system would have provided a back-door into a secured device.

The AVR Studio enables the user to fully control execution of programs on an AVR device with On-chip Debug capability, AVR In-Circuit Emulator, or the built-in AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly programs assembled with Atmel Corporation's AVR Assembler and C programs compiled with 3rd party vendors' compilers.

AVR Studio runs under Microsoft® Windows® 95/98/2000 and Microsoft Windows NT®.

For a full description of the AVR Studio, please refer to the **AVR Studio User Guide**. Only highlights are presented in this document.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have up to two Data memory Break Points, alternatively combined as a mask (range) Break Point.

## On-chip Debug Specific JTAG Instructions

The On-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction opcode listed for reference.

**PRIVATE0; $8** Private JTAG instruction for accessing On-chip debug system.

**PRIVATE1; $9** Private JTAG instruction for accessing On-chip debug system.

**PRIVATE2; $A** Private JTAG instruction for accessing On-chip debug system.

**PRIVATE3; $B** Private JTAG instruction for accessing On-chip debug system.

## ATmega323 Boundary-Scan Order

Table 64 shows the Scan order between TDI and TDO when the Boundary-Scan chain is selected as data path. Bit 0 is the LSB; The first bit scanned in, and the first bit scanned out. The scan order follows the pinout order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 89, PXn.Data corresponds to FF0, PXn.Control corresponds to FF1, and PXn. Pullup_dissable corresponds to FF2. Bit 2, 3, 4, and 5 of Port C is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled

**Table 58.** ATmega323 Boundary-Scan Order

| Bit Number | Signal Name | Module |
|---|---|---|
| 131 | SIG_PRIVATE0 | Private Section 1 |
| 130 | SIG_PRIVATE1 | |
| 129 | SIG_PRIVATE2 | |
| 128 | SIG_PRIVATE3 | |
| 127 | SIG_PRIVATE4 | Private Section 2 |
| 126 | SIG_PRIVATE5 | |
| 125 | SIG_PRIVATE6 | |
| 124 | SIG_PRIVATE7 | |
| 123 | SIG_PRIVATE8 | |
| 122 | SIG_PRIVATE9 | |
| 121 | SIG_PRIVATE10 | |
| 120 | SIG_PRIVATE11 | |
| 119 | SIG_PRIVATE12 | |
| 118 | SIG_PRIVATE13 | |
| 117 | SIG_PRIVATE14 | |
| 116 | SIG_PRIVATE15 | |
| 115 | SIG_PRIVATE16 | |
| 114 | SIG_PRIVATE17 | |
| 113 | SIG_PRIVATE18 | |
| 112 | SIG_PRIVATE19 | |
| 111 | SIG_PRIVATE20 | |
| 110 | SIG_PRIVATE21 | |
| 109 | SIG_PRIVATE22 | |
| 108 | SIG_PRIVATE23 | |
| 107 | SIG_PRIVATE24 | |
| 106 | SIG_PRIVATE25 | |
| 105 | SIG_PRIVATE26 | |

3. The Serial Programming instructions will not work if the communication is out of synchronization. When in sync. the second byte ($53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the $53 did not echo back, give SCK a positive pulse and issue a new Programming Enable command. If the $53 is not seen within 32 attempts, there is no functional device connected.
4. If a chip erase is performed (must be done to erase the Flash), wait $2 \cdot t_{WD\_FLASH}$ after the instruction, give $\overline{\text{RESET}}$ a positive pulse, and start over from Step 2. See Table 68 for the $t_{WD\_FLASH}$ figure.
5. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program memory Page instruction. The Program memory Page is stored by loading the Write Program memory Page instruction with the 8 MSB of the address. If polling is not used, the user must wait at least $t_{WD\_FLASH}$ before issuing the next page. (Please refer to Table 68). Accessing the Serial Programming interface before the Flash write operation completes can result in incorrect programming.
6. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM Memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least $t_{WD\_EEPROM}$ before issuing the next byte. (Please refer to Table 68). In a chip erased device, no $FFs in the data file(s) need to be programmed.
7. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.
8. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
9. Power-off sequence (if needed):
   Set XTAL1 to “0” (if a crystal is not used).
   Set $\overline{\text{RESET}}$ to “1”.
   Turn $V_{CC}$ power off

**Data Polling Flash**

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value $FF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value $FF, so when programming this value, the user will have to wait for at least $t_{WD\_FLASH}$ before programming the next page. As a Chip Erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped. See Table 68 $t_{WD\_FLASH}$.

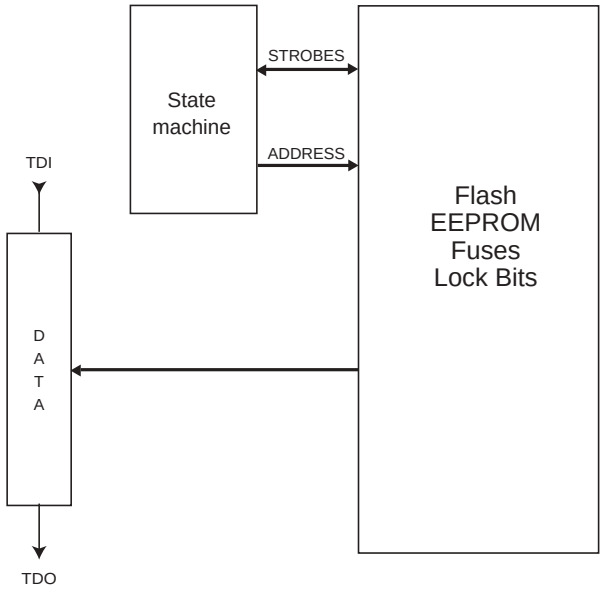
**Data Polling EEPROM**

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value $FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value $FF, but the user should have the following in mind: As a Chip Erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped. This does not apply if the EEPROM is re-programmed without Chip Erasing the device. In this case, data polling cannot be used for the value $FF, and the user will have to wait at least $t_{WD\_EEPROM}$ before programming the next byte. See Table 68 for $t_{WD\_EEPROM}$.

## Virtual Flash Page Read Register

The Virtual Flash Page Read Register is a virtual scan chain with length equal to the number of bits in one Flash page plus eight, 1,032 in total. Internally the Shift Register is 8-bit, and the data are automatically transferred from the Flash data page byte-by-byte. The first eight cycles are used to transfer the first byte to the internal Shift Register, and the bits that are shifted out during these eight cycles should be ignored. Following this initialization, data are shifted out starting with the LSB of the instruction with page address 0 and ending with the MSB of the instruction with page address 3F. This provides an efficient way to read one full Flash page to verify programming.

**Figure 105.** Virtual Flash Page Read Register




## Programming algorithm

All references below of type “1a”, “1b”, and so on, refer to Table 71.

### Entering programming mode

1. Enter JTAG instruction AVR_RESET and shift 1 in the Reset Register.
2. Enter instruction PROG_ENABLE and shift 1010_0011_0111_0000 in the Programming Enable Register.

### Leaving Programming Mode

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by using no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the programming Enable Register.
4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset Register.

If PROG_ENABLE instruction is not followed by the AVR_RESET instruction, the following algorithm should be used:

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by using no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the Programming Enable Register.
4. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the Programming Enable Register.
5. Wait until the selected Oscillator has started before applying more commands.

6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

**Programming the EEPROM**

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load address using programming instructions 4b and 4c.
4. Load data using programming instructions 4d.
5. Write the data using programming instruction 4e.
6. Poll for EEPROM write complete using programming instruction 4f, or wait for $t_{WLRH}$ (refer to Table 67 on page 196).
7. Repeat steps 3 to 6 until all data have been programmed.

**Reading the EEPROM**

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM read using programming instruction 5a.
3. Load address using programming instructions 5b and 5c.
4. Read data using programming instruction 5d.
5. Repeat steps 3 and 4 until all data have been read.

**Programming the Fuses**

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data High Byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse, a “1” will unprogram the fuse.
4. Write Fuse High Byte using programming instruction 6c.
5. Poll for Fuse write complete using programming instruction 6d, or wait for $t_{WLRH}$ (refer to Table 67 on page 196).
6. Load data Low Byte using programming instructions 6e. A “0” will program the fuse, a “1” will unprogram the fuse.
7. Write Fuse Low Byte using programming instruction 6f.
8. Poll for Fuse write complete using programming instruction 6g, or wait for $t_{WLRH}$ (refer to Table 67 on page 196).

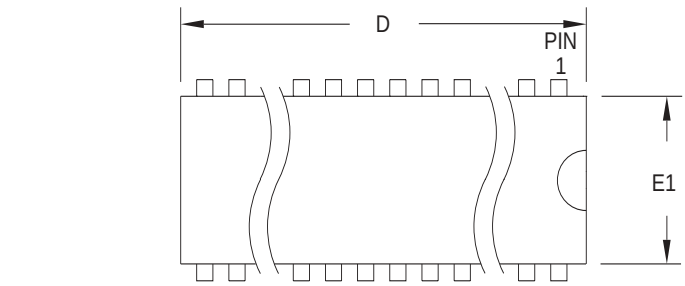
**Programming the Lock Bits**

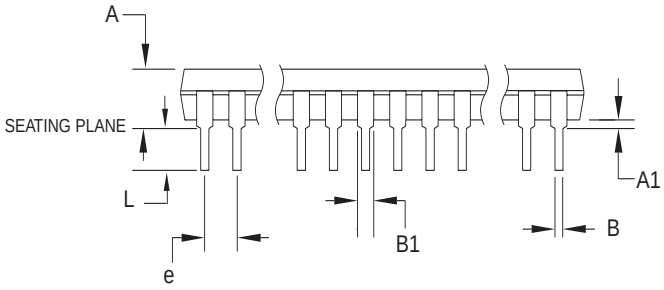
1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for $t_{WLRH}$ (refer to Table 67 on page 196).

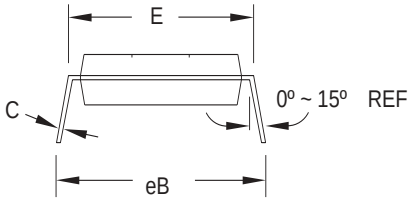
**Reading the Fuses and Lock Bits**

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

## 40P6




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

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | – | – | 4.826 | |
| A1 | 0.381 | – | – | |
| D | 52.070 | – | 52.578 | Note 2 |
| E | 15.240 | – | 15.875 | |
| E1 | 13.462 | – | 13.970 | Note 2 |
| B | 0.356 | – | 0.559 | |
| B1 | 1.041 | – | 1.651 | |
| L | 3.048 | – | 3.556 | |
| C | 0.203 | – | 0.381 | |
| eB | 15.494 | – | 17.526 | |
| e | 2.540 TYP | | | |

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. 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. |
|---|---|---|---|
| 2325 Orchard Parkway<br>San Jose, CA 95131 | **40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP) | 40P6 | B |