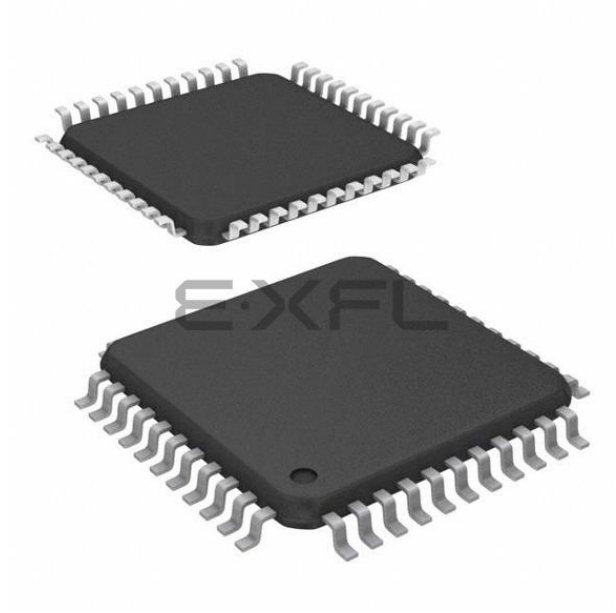


Welcome to E-XFL.COM

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	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega323l-4ac

Table 6 shows the start-up times from Reset. When the CPU wakes up from Power-down or Power-save, only the clock counting part of the start-up time is used. The Watchdog Oscillator is used for timing the Real Time part of the start-up time. The number WDT Oscillator cycles used for each time-out is shown in Table 7.

The frequency of the Watchdog Oscillator is voltage dependent as shown in the Electrical Characteristics section. The device is shipped with CKSEL = “0010” (Internal RC Oscillator, slowly rising power).

Table 7. Number of Watchdog Oscillator Cycles

BODLEVEL(1)	V_{CC} Condition	Time-out	Number of Cycles
Unprogrammed	2.7V	30 µs	8
Unprogrammed	2.7V	130 µs	32
Unprogrammed	2.7V	4.2 ms	1K
Unprogrammed	2.7V	67 ms	16K
Programmed	4.0V	10 µs	8
Programmed	4.0V	35 µs	32
Programmed	4.0V	5.8 ms	4K
Programmed	4.0V	92 ms	64K

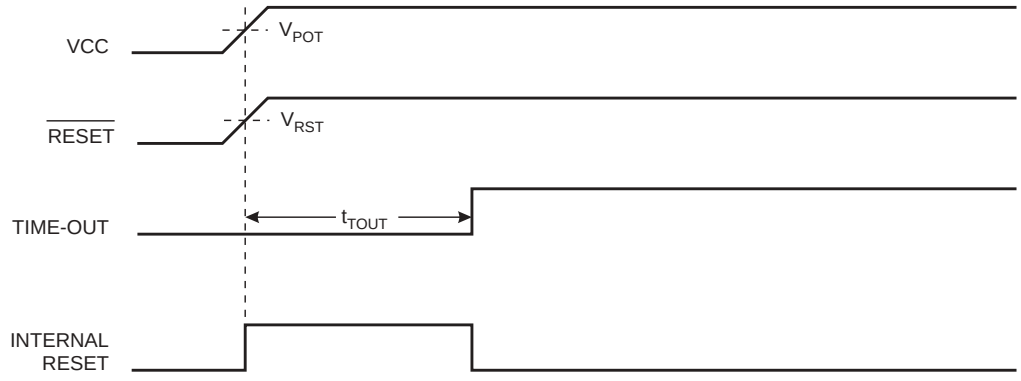
Note: 1. The BODLEVEL Fuse can be used to select start-up times even if the Brown-out Detection is disabled (BODEN Fuse unprogrammed).

Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 5. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes a delay counter, which determines the delay, for which the device is kept in RESET after V_{CC} rise. The time-out period of the delay counter can be defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in Table 6. The RESET signal is activated again, without any delay, when the V_{CC} decreases below detection level.

Figure 25. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}



- **Bit 2 – TOV1: Timer/Counter1 Overflow Flag**

The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logic one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

- **Bit 1– OCF0: Output Compare Flag 0**

The OCF0 bit is set (one) when a Compare Match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register 0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE0 (Timer/Counter0 Compare Match Interrupt Enable), and the OCF0 are set (one), the Timer/Counter0 Compare Match Interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter0 changes counting direction at $00.

External Interrupts

The External Interrupts are triggered by the INT0, INT1, and INT2 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..2 pins are configured as outputs. This feature provides a way of generating a software interrupt. The External Interrupts can be triggered by a falling or rising edge or a low level (INT2 is only an edge triggered interrupt). This is set up as indicated in the specification for the MCU Control Register – MCUCR and MCU Control and Status Register – MCUCSR. When the External Interrupt is enabled and is configured as level triggered (only INT0/INT1), the interrupt will trigger as long as the pin is held low.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$37 ($57)	**SE**	**SM2**	**SM1**	**SM0**	**ISC11**	**ISC10**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SE: Sleep Enable**

The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to set the Sleep Enable (SE) bit just before the execution of the SLEEP instruction.

Table 16. PWM Outputs OCRn = $00 or $FF[1]

COMn1	COMn0	OCRn	Output PWMn
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

Note: 1. n = 0 or 2
In overflow PWM mode, the table above is only valid for OCRn = $FF.

In up/down PWM mode, the Timer Overflow Flag, TOV0 or TOV2, is set when the counter advances from $00. In overflow PWM mode, the Timer Overflow Flag is set as in normal Timer/Counter mode. Timer Overflow Interrupt0 and 2 operate exactly as in normal Timer/Counter mode, i.e., they are executed when TOV0 or TOV2 are set provided that Timer Overflow Interrupt and Global Interrupts are enabled. This does also apply to the Timer Output Compare Flag and interrupt.

Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
$22 ($22)	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
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 – AS2: Asynchronous Timer/Counter2**

When AS2 is cleared (zero), Timer/Counter2 is clocked from the internal system clock, CK. When AS2 is set (one), Timer/Counter2 is clocked from the TOSC1 pin. Pins PC6 and PC7 are connected to a crystal Oscillator and cannot be used as general I/O pins. When the value of this bit is changed, the contents of TCNT2, OCR2, and TCCR2 might be corrupted.

- **Bit 2 – TCN2UB: Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set (one). When TCNT2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

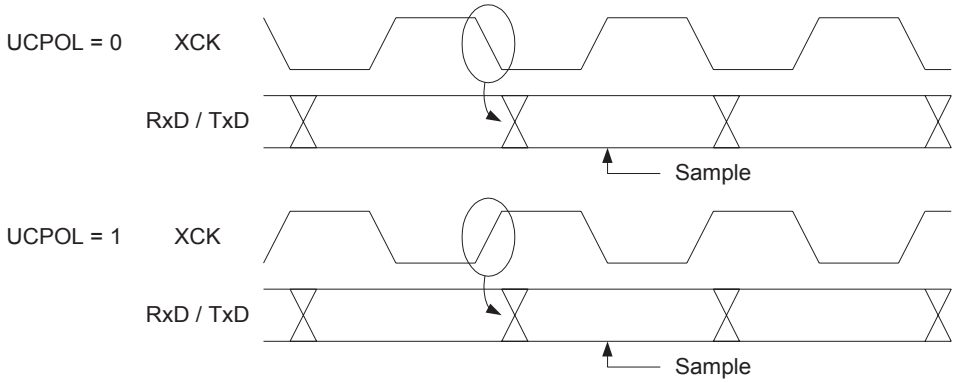
- **Bit 1 – OCR2UB: Output Compare Register2 Update Busy**

When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set (one). When OCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that OCR2 is ready to be updated with a new value.

- **Bit 0 – TCR2UB: Timer/Counter Control Register2 Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2 is written, this bit becomes set (one). When TCCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that TCCR2 is ready to be updated with a new value.

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.

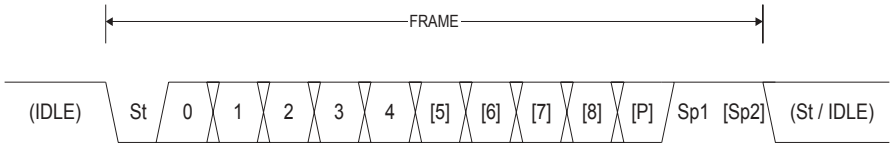
Frame Formats

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.

Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR, and PE Status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and PE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example(1)

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get status and ninth bit, then data from buffer
  in    r18, UCSRA
  in    r17, UCSRB
  in    r16, UDR
  ; If error, return -1
  andi  r18,(1<<FE)|(1<<DOR)|(1<<PE)
  breq  USART_ReceiveNoError
  ldi   r17, HIGH(-1)
  ldi   r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the ninth bit, then return
  lsr   r17
  andi  r17, 0x01
  ret
```

C Code Example(1)

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) ) {};
  /* Get status and ninth bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<PE) )
    return -1;
  /* Filter the ninth bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. The example code assumes that the part specific header file is included.

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

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

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

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

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

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

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

;<send more data bytes if needed>

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

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

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

Analog to Digital Converter

Features

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

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

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

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

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

Figure 69. Port B Schematic Diagram (Pin PB2)

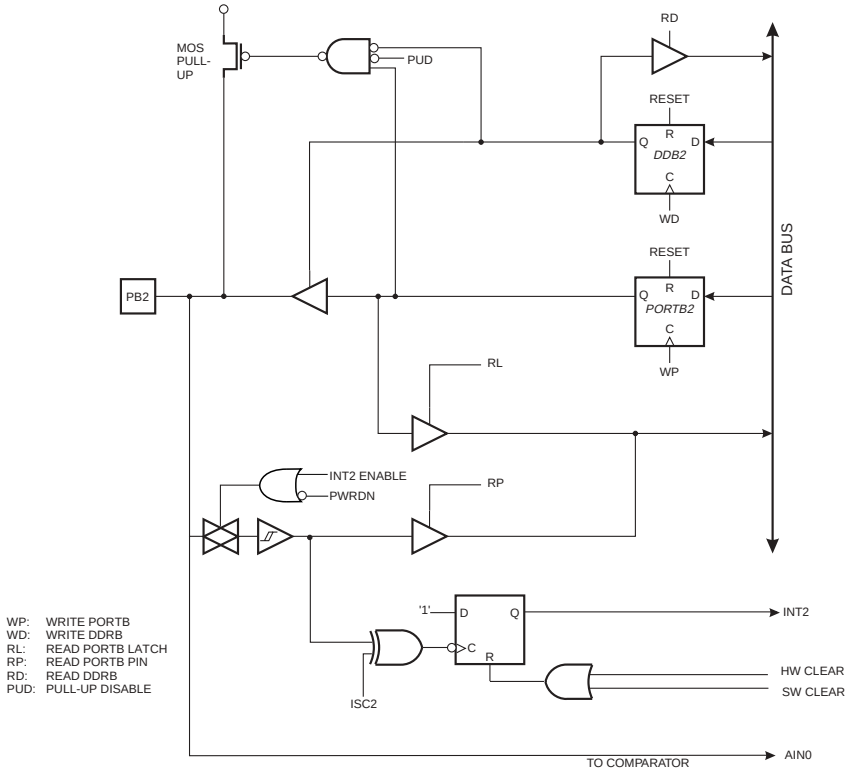


Figure 70. Port B Schematic Diagram (Pin PB3)

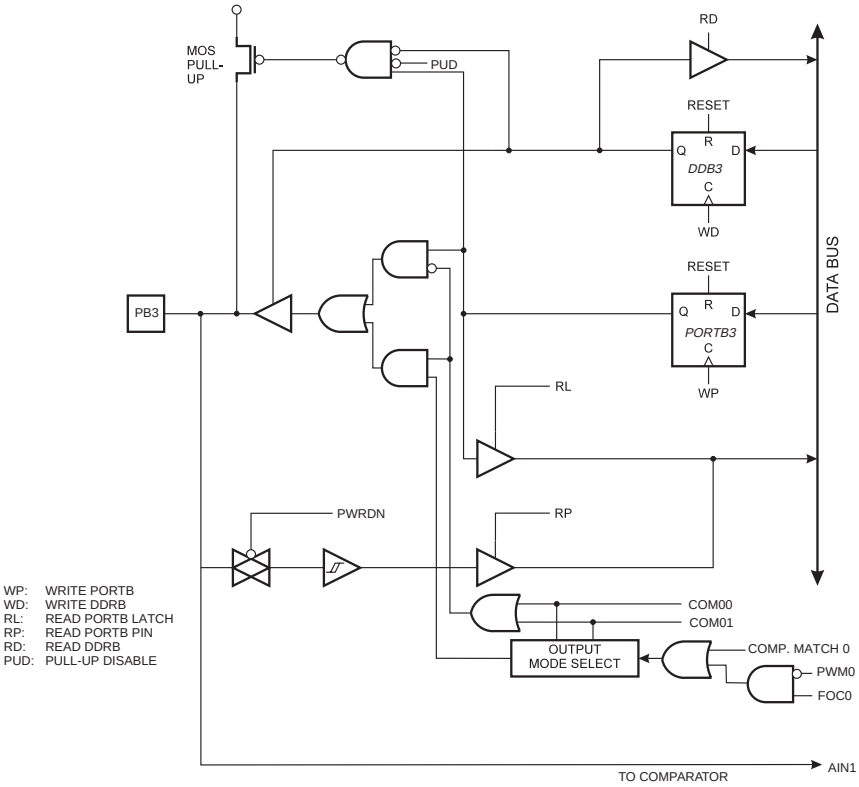


Table 57. Boundary-Scan signals for the ADC (Continued)

Signal Name	Type of scan cell	Recommended input when not in use
SIG_PRIVATE18	General Scan Cell	0
SIG_PRIVATE19	General Scan Cell	0
SIG_PRIVATE20	General Scan Cell	0
SIG_PRIVATE21	General Scan Cell	1
SIG_PRIVATE22	General Scan Cell	0
SIG_PRIVATE23	General Scan Cell	0
SIG_PRIVATE24	General Scan Cell	0
SIG_PRIVATE25	General Scan Cell	1
SIG_PRIVATE26	General Scan Cell	0
SIG_PRIVATE27	General Scan Cell	0
SIG_PRIVATE28	General Scan Cell	0
SIG_PRIVATE29	General Scan Cell	0
SIG_PRIVATE30	General Scan Cell	0
SIG_PRIVATE31	General Scan Cell	0
SIG_PRIVATE32	General Scan Cell	0
SIG_PRIVATE33	General Scan Cell	0
SIG_PRIVATE34	General Scan Cell	1
SIG_PRIVATE35	General Scan Cell	0
SIG_PRIVATE36	General Scan Cell	0
SIG_PRIVATE37	General Scan Cell	0
SIG_PRIVATE38	General Scan Cell	1
SIG_PRIVATE39	General Scan Cell	1
SIG_PRIVATE40	General Scan Cell	0
SIG_PRIVATE41	General Scan Cell	0
SIG_PRIVATE42	General Scan Cell	0
SIG_PRIVATE43	Observe Only	X
SIG_PRIVATE44	Observe Only	X
SIG_PRIVATE45	Observe Only	X
SIG_PRIVATE46	Observe Only	X

Figure 93. Parallel Programming

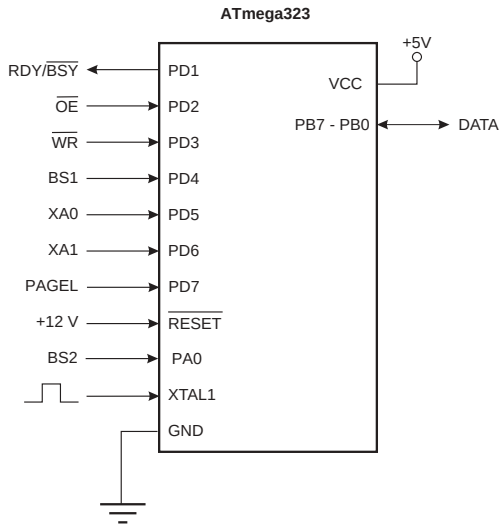


Table 64. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is Busy Programming, 1: Device is Ready for New Command
$\overline{OE}$	PD2	I	Output Enable (Active Low)
$\overline{WR}$	PD3	I	Write Pulse (Active Low)
BS1	PD4	I	Byte Select 1 (“0” Selects Low Byte, “1” Selects High Byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program Memory Page Load
BS2	PA0	I	Byte Select 2 (“0” Selects Low Byte, “1” Selects 2 nd High Byte)
DATA	PB7 - 0	I/O	Bidirectional Data Bus (Output When $\overline{OE}$ is Low)

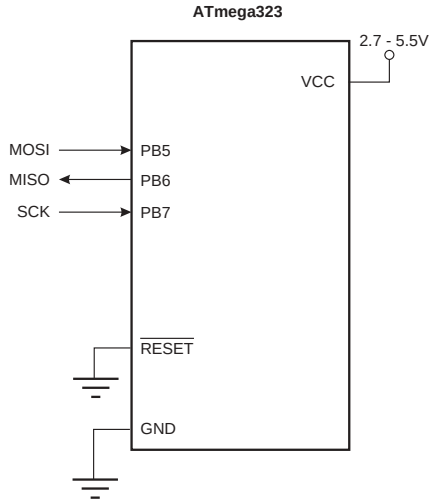
Table 65. XA1 and XA0 Coding

XA1	XA0	Action When XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or Low Address Byte Determined by BS1)
0	1	Load Data (High or Low Data Byte for Flash Determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Serial Downloading

Both the Flash and EEPROM Memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Figure 98. Serial Programming and Verify



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 $FF.

The Program and EEPROM Memory arrays have separate address spaces:

$0000 to $3FFF for Program memory and $0000 to $03FF for EEPROM Memory.

The device can be clocked by any clock option during Low Voltage Serial Programming. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles

High: > 2 CPU clock cycles

Serial Programming Algorithm

When writing serial data to the ATmega323, data is clocked on the rising edge of SCK.

When reading data from the ATmega323, data is clocked on the falling edge of SCK. See Figure 99, Figure 100, and Table 70 for timing details.

To program and verify the ATmega323 in the Serial Programming mode, the following sequence is recommended (See four byte instruction formats in Table 69):

1. Power-up sequence:

   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. The $\overline{\text{RESET}}$ and SCK are set to “0”. In accordance with the setting of CKSEL Fuses, apply a crystal/resonator, external clock or RC network, or let the device run on the internal RC Oscillator. In some systems, the programmer can not guarantee that SCK is held low during Power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two XTAL1 cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable Serial Programming by sending the Programming Enable serial instruction to pin MOSI/PB5.

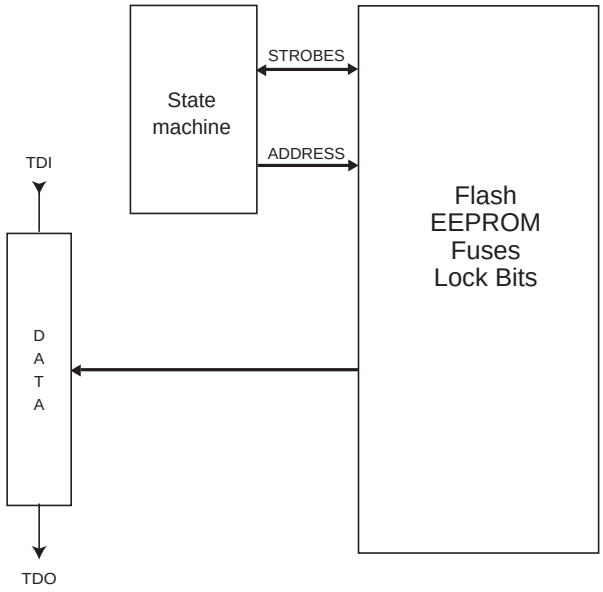
Table 71. JTAG Programming Instruction Set

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for chip erase complete	0110011_10000000	xxxxxox_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address High Byte	0000111_00aaaaaa	xxxxxxx_xxxxxxxx	
2c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
2d. Load Data Low Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2g. Write Page	0110111_00000000 0110101_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Poll for Page Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_00aaaaaa	xxxxxxx_xxxxxxxx	
3c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	 Low Byte High Byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_000000aa	xxxxxxx_xxxxxxxx	
4c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Write EEPROM byte	0110011_00000000 0110001_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Poll for Byte Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_000000aa	xxxxxxx_xxxxxxxx	
5c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data High Byte	0010011_IH11GFED	xxxxxxx_xxxxxxxx	(3)

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.

Performing Chip Erase

1. Enter JTAG instruction PROG_COMMANDS.
2. Start chip erase using programming instruction 1a.
3. Poll for chip erase complete using programming instruction 1b, or wait for t_{WLRH_CE} (refer to Table 67 on page 196).

Programming the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load address using programming instructions 2b and 2c.
4. Load data using programming instructions 2d, 2e and 2f.
5. Repeat step 3 and 4 for all 64 instruction words in the page.
6. Write the page using programming instruction 2g.
7. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH_FLASH} (refer to Table 67 on page 196).
8. Repeat steps 3 to 7 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load the page address using programming instructions 2b and 2c. The 6 LSB are used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGELOAD.
5. Load the entire page by shifting in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page.
6. Enter JTAG instruction PROG_COMMANDS.
7. Write the page using programming instruction 2g.
8. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH_FLASH} (refer to Table 67 on page 196).
9. Repeat steps 3 to 8 until all data have been programmed.

Reading the Flash

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

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load the page address using programming instructions 3b and 3c. The 6 LSB are used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page by shifting out all instruction words in the page, starting with the LSB of the instruction with page address 0 and ending with the MSB of the instruction with page address 3F. Remember that the first eight bits should be ignored.

communicate at full speed (400 kHz) with other ATmega323 devices, as well as any other device with a proper t_{LOW} acceptance margin.

Figure 107. Two-wire Serial Bus timing

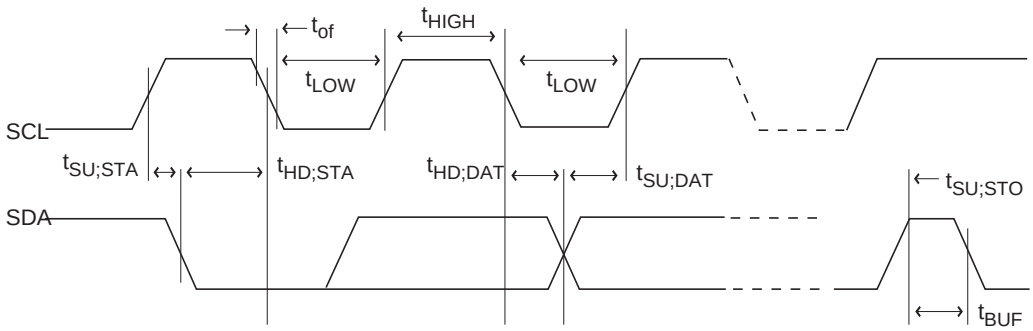


Figure 121. Analog Comparator Offset voltage vs. Common Mode Voltage

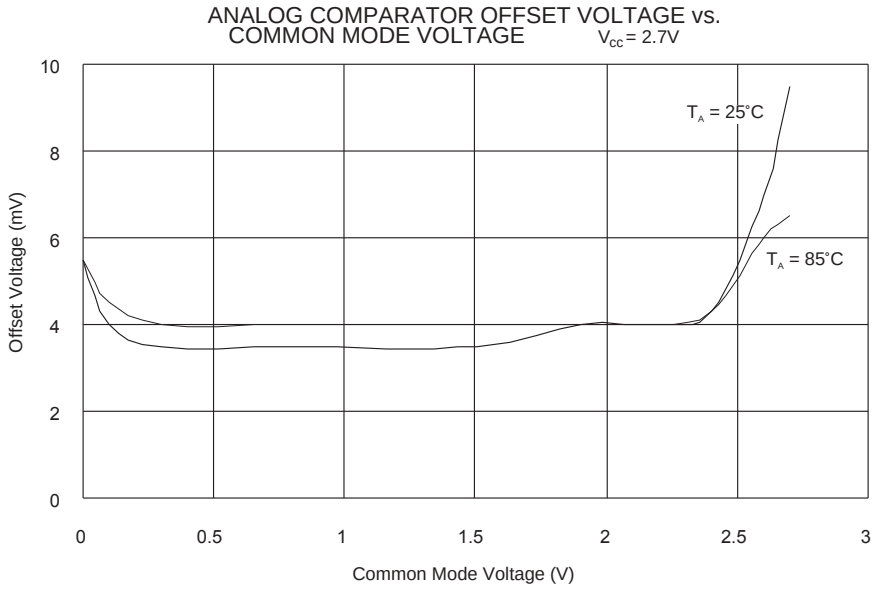


Figure 122. Analog Comparator Input Leakage Current

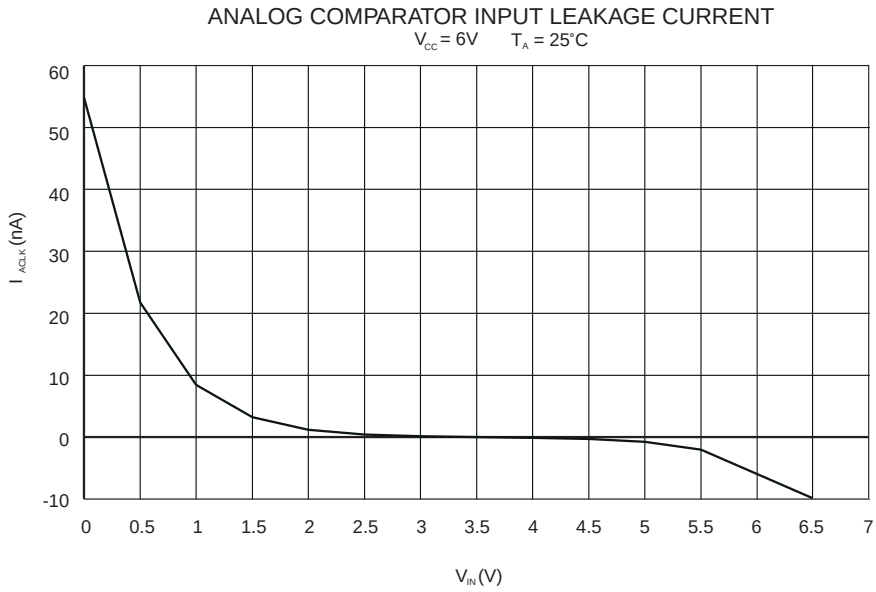


Figure 125. Pull-Up Resistor Current vs. Input Voltage

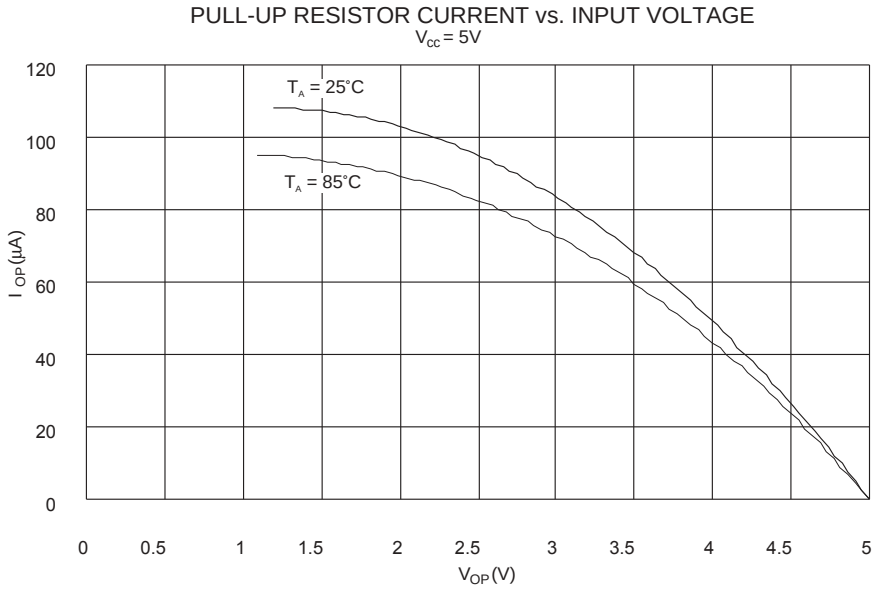
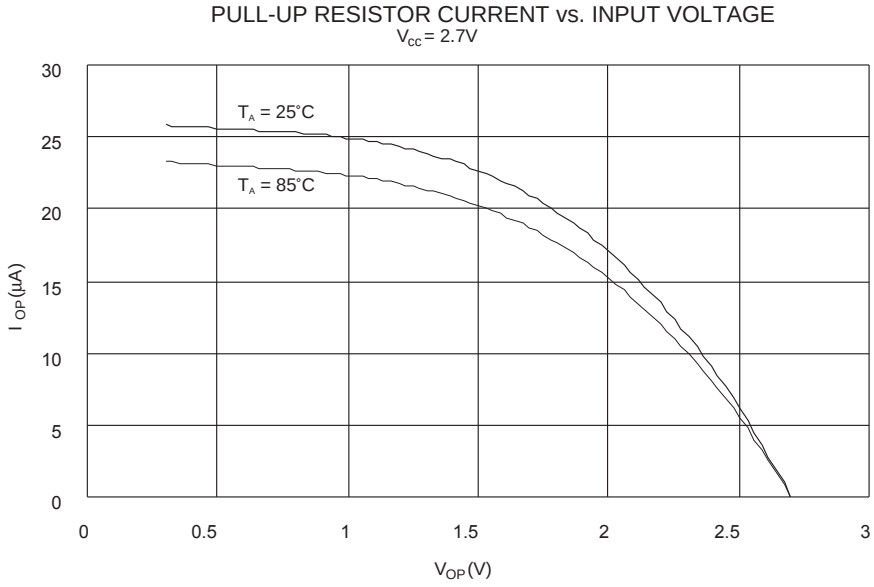
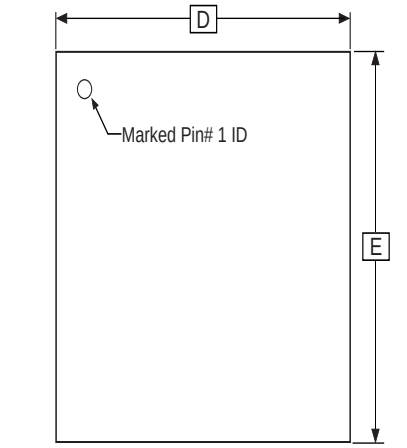


Figure 126. Pull-Up Resistor Current vs. Input Voltage

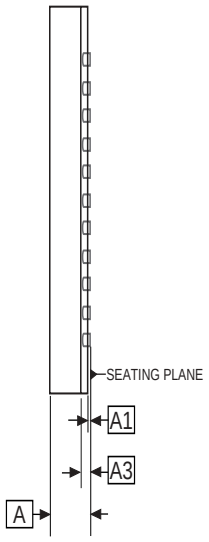


Packaging Information

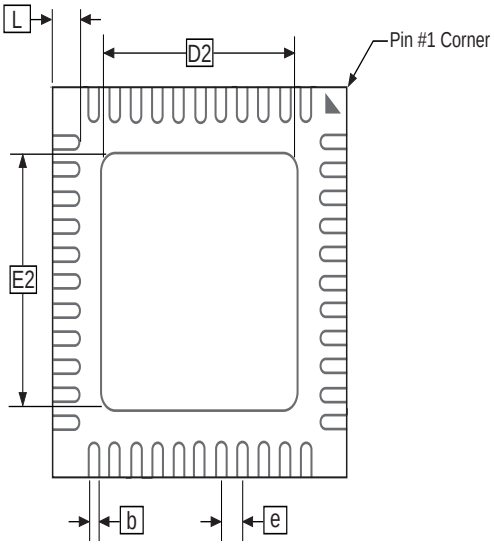
44A



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.25 REF		
b	0.18	0.23	0.30	
D		7.00 BSC		
D2	5.00	5.20	5.40	
E		7.00 BSC		
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.35	0.55	0.75	

Notes: 1. JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-1.

01/15/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44M1**, 44-pad, 7 x 7 x 1.0 mm Body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)	44M1	C

3. TWI is Speed Limited in Slave Mode

When the Two-wire Serial Interface operates in Slave mode, frames may be undetected if the CPU frequency is less than 64 times the bus frequency.

Problem Fix/Workaround

Ensure that the CPU frequency is at least 64 times the TWI bus frequency.

2. Problems with UBRR Settings

The baud rate corresponding to the previous UBRR setting is used for the first transmitted/received bit when either UBRRH or UBRRL is written. This will disturb communication if the UBRR is changed from a very high to a very low baud rate setting, as the internal baud rate counter will have to count down to zero before using the new setting.

In addition, writing to UBRRL incorrectly clears the UBRRH setting.

Problem Fix/Workaround

UBRRH must be written after UBRRL because setting UBRRL clears UBRRH. By doing an additional dummy write to UBRRH, the baud rate is set correctly. The following is an example on how to set UBRR. UBRRH is updated first for upward compatibility with corrected devices.

```
ldi r17, HIGH(baud)
ldi r16, LOW(baud)
out UBRRH, r17        ; Added for upward compatibility
out UBRRL, r16        ; Set new UBRRL, UBRRH incorrectly cleared
out UBRRH, r17        ; Set new UBRRH
out UBRRH, r17        ; Loads the baud rate counter with new (correct) value
```

1. Missing OverRun Flag and Fake Frame Error in USART

When the USART has received three characters without any of them been read, the USART FIFO is full. If the USART detects the start bit of a fourth character, the Data OverRun (DOR) Flag will be set for the third character. However, if a read from the USART Data Register is performed just after the start bit of the fourth byte is received, a Frame Error is generated for character three. If the USART Data Register is read between the reception of the first data bit and the end of the fourth character, the Data OverRun Flag of character three will be lost.

Problem Fix/Workaround

The user should design the application to never completely fill the USART FIFO. If this is not possible, the user must use a high-level protocol to be able to detect if any characters were lost and request a retransmission if this happens.

The following is not errata for ATmega323, all revisions. However, a proposal for solving problems regarding the JTAG instruction IDCODE is presented below.

IDCODE masks data from TDI input

The public but optional JTAG instruction IDCODE is not implemented correctly according to IEEE1149.1; a logic one is scanned into the shift register instead of the TDI input while shifting the Device ID Register. Hence, captured data from the preceding devices in the boundary scan chain are lost and replaced by all-ones, and data to succeeding devices are replaced by all-ones during Update-DR.

If ATmega323 is the only device in the scan chain, the problem is not visible.

Datasheet Change Log for ATmega323

This document contains a log on the changes made to the datasheet for ATmega323.

Changes from Rev. 1457F – 09/02 to Rev. 1457G – 09/03

1. **Removed “Preliminary” from the .**
2. **Updated “The Test Access Port – TAP” on page 158 regarding JTAGEN.**
3. **Updated description for the JTD bit on page 30.**
4. **Added extra information regarding the JTAGEN interface to “Fuse Bits” on page 187.**
5. **Updated some values in “Electrical Characteristics” on page 213.**
5. **Added a proposal for solving problems regarding the JTAG instruction IDCODE in “Errata for ATmega323 Rev. B” on page 239.**

Changes from Rev. 1457E – 11/01 to Rev. 1457F – 09/02

1. **Added watermark: “Not recommended for new designs. Use ATmega32”.**
2. **Added “Errata for ATmega323 Rev. B” on page 239.**