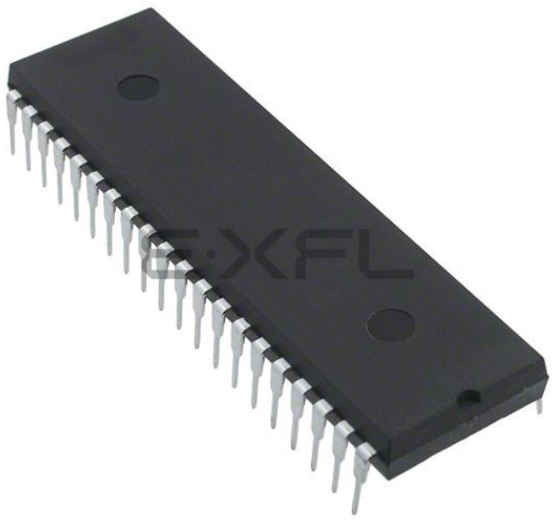
Microchip Technology - ATMEGA323-8PI Datasheet



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	8MHz
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)	4.5V ~ 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/atmega323-8pi

The General Purpose Register File

Figure 7 shows the structure of the 32 general purpose working registers in the CPU.

Figure 7. AVR CPU General Purpose Working Registers

	7 0	Addr.	
	R0	$00	
	R1	$01	
	R2	$02	
	…		
	R13	$0D	
General	R14	$0E	
Purpose	R15	$0F	
Working	R16	$10	
Registers	R17	$11	
	…		
	R26	$1A	X-register Low Byte
	R27	$1B	X-register High Byte
	R28	$1C	Y-register Low Byte
	R29	$1D	Y-register High Byte
	R30	$1E	Z-register Low Byte
	R31	$1F	Z-register High Byte

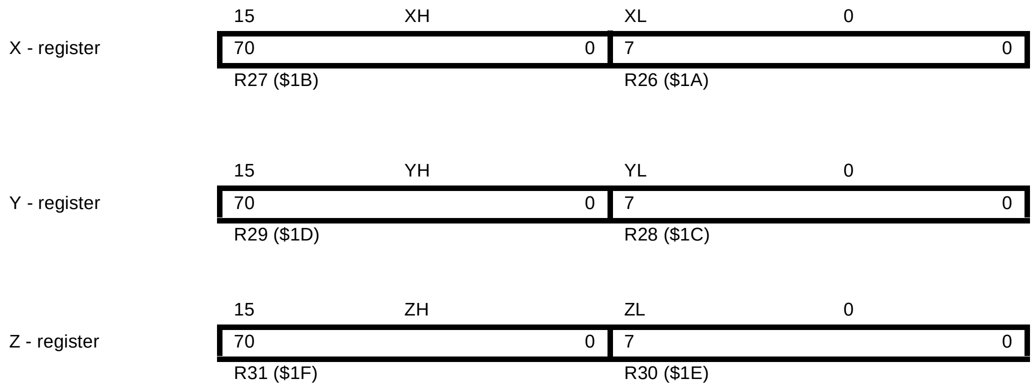
Most register operating instructions in the instruction set have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 7, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-registers can be set to index any register in the file.

The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as:

Figure 8. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after Power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.

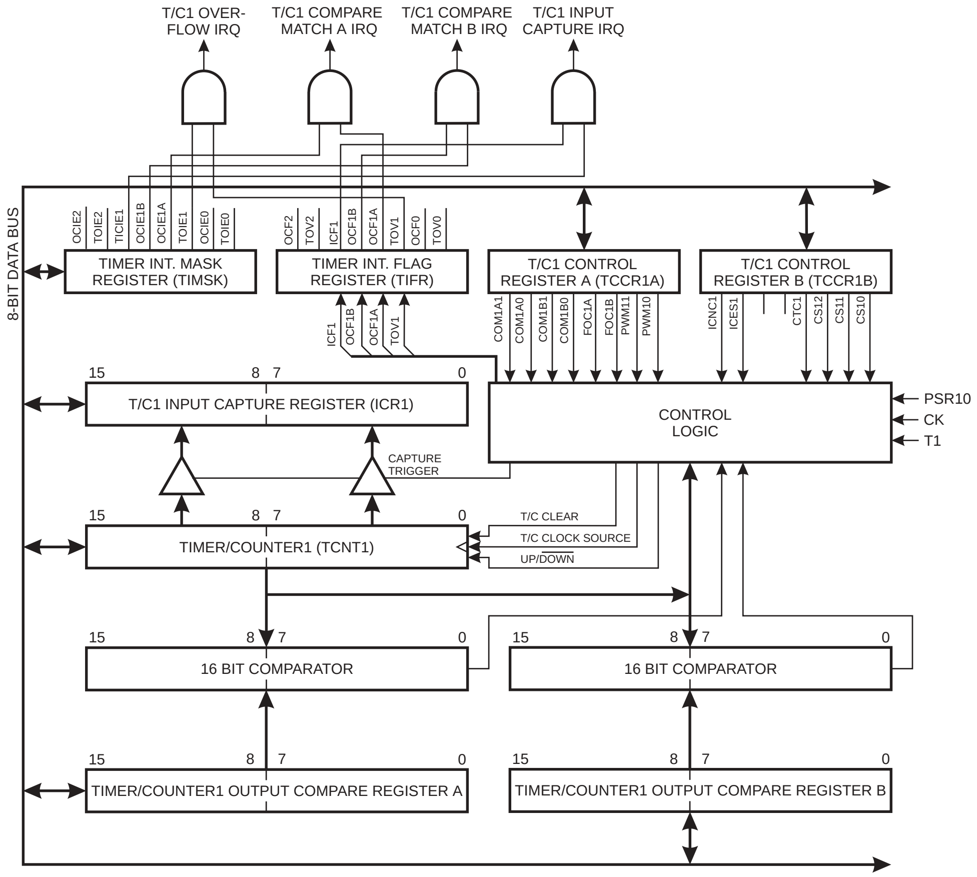
Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.

During asynchronous operation, the synchronization of the Interrupt Flags for the asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The Output Compare Pin is changed on the timer clock and is not synchronized to the processor clock.

16-bit Timer/Counter1

Figure 36 shows the block diagram for Timer/Counter1.

Figure 36. Timer/Counter1 Block Diagram



The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in section “Timer/Counter1 Control

When the prescaler is set to divide by 8, the timer will count like this:

... **|** C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 **|** C, C, C, C, C, C, C, C **|** 0, 0, 0, 0, 0, 0, 0, 0 **|**1,1,1,1,1,1,1,1**|**...

In PWM mode, this bit has a different function. If the CTC1 bit is cleared in PWM mode, the Timer/Counter1 acts as an up/down counter. If the CTC1 bit is set (one), the Timer/Counter wraps when it reaches the TOP value. Refer to page 60 for a detailed description.

- **Bits 2..0 – CS12, CS11, CS10: Clock Select1, Bit 2, 1, and 0**

The Clock Select1 bits 2, 1, and 0 define the prescaling source of Timer/Counter1.

Table 19. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is Stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T1, Falling Edge
1	1	1	External Pin T1, Rising Edge

The Stop condition provides a Timer Enable/Disable function. The prescaled modes are scaled directly from the CK Oscillator clock. If the external pin modes are used for Timer/Counter1, transitions on PB1/(T1) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

Timer/Counter1 – TCNT1H and TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	**MSB**								**TCNT1H**
$2C ($4C)								**LSB**	**TCNT1L**
	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	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the high and Low Bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP). This temporary register is also used when accessing OCR1A, OCR1B, and ICR1. If the main program and also interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program and interrupt routines.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter uses the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity Mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the *USART Stop Bit Select* (USBS) bit. The Receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even}	Parity bit using even parity
P_{odd}	Parity bit using odd parity
d_n	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

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 TXC 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 TXC Flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that is 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. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers. When the function writes to the UCSRC Register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out UBRRH, r17
  out UBRRL, r16
  ; Enable Receiver and Transmitter
  ldi r16, (1<<RXEN)|(1<<TXEN)
  out UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
  out UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable Receiver and Transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

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

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the Baud and Control Registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

USART Register Description

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C) Read	RXB[7:0]								UDR (Read)
$0C ($2C) Write	TXB[7:0]								UDR (Write)
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	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The Receive Buffer consists of a two level FIFO. The FIFO will change its state whenever the Receive Buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC: USART Receive Complete**

This flag bit is one when there are unread data 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, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit is set one when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (character size) in a frame the Receiver and Transmitter uses.

Table 35. Character Size

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOL: Clock Polarity**

This bit is used for synchronous mode only. Set this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Falling XCK Edge	Rising XCK Edge
1	Rising XCK Edge	Falling XCK Edge

USART Baud Rate Registers – UBRRL and UBRRHs

Bit	15	14	13	12	11	10	9	8	
$20 ($40)	URSEL	–	–	–	UBRR[11:8]				UBRRH
$09 ($29)	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	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 UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/UCSRC Registers" on page 92 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be set to zero when UBRRH is written.

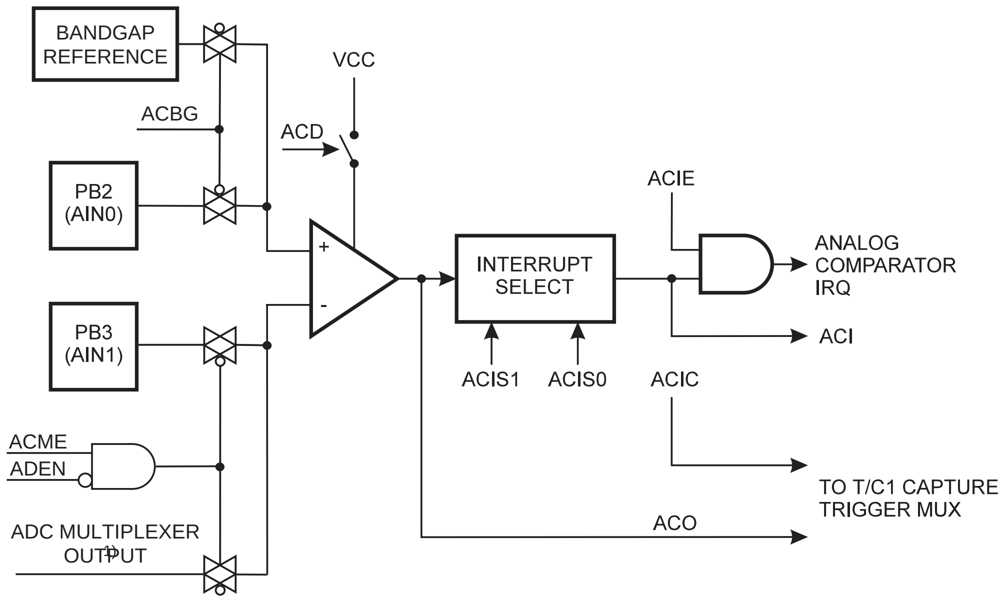
Table 37. Miscellaneous States

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+W	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+W or	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	X	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

The Analog Comparator

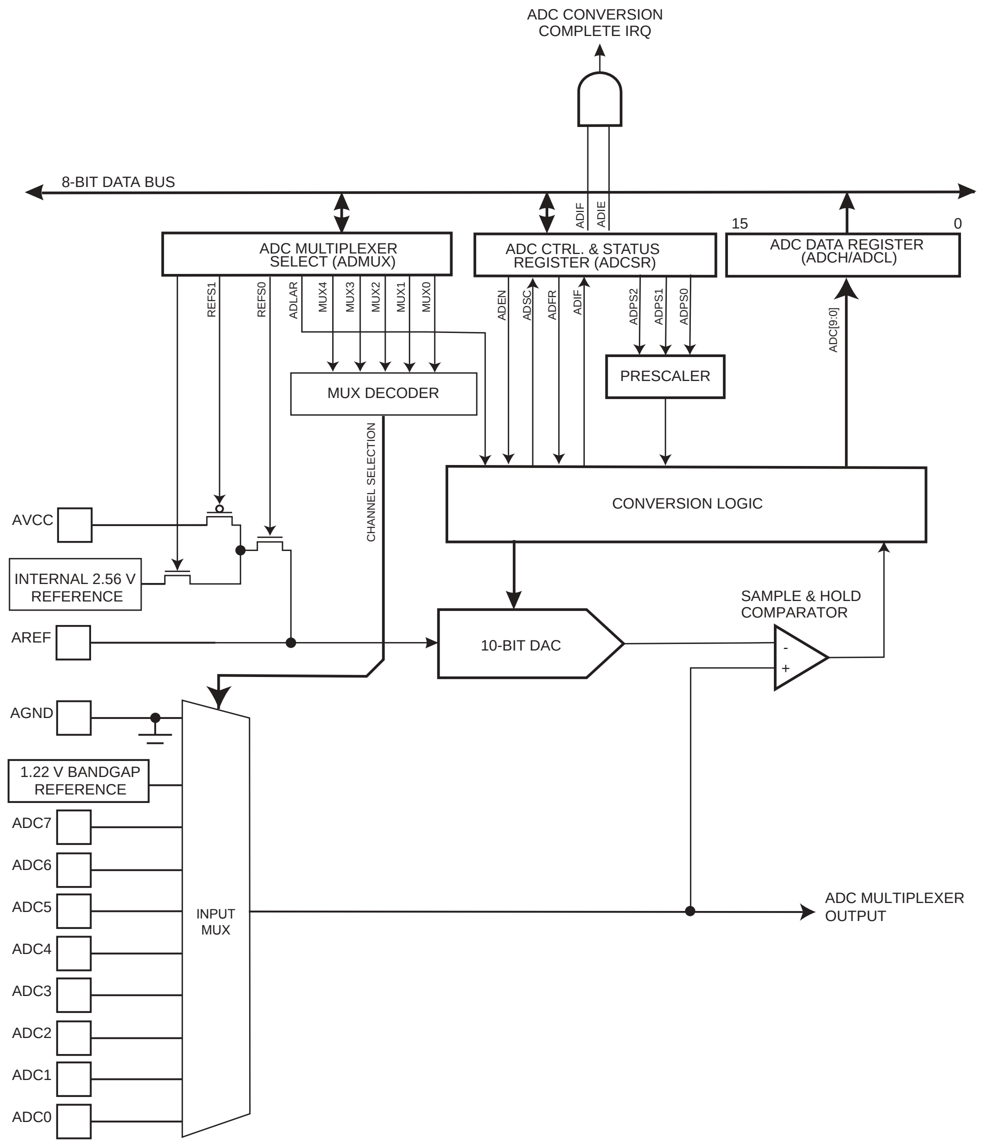
The Analog Comparator compares the input values on the positive pin PB2 (AIN0) and negative pin PB3 (AIN1). When the voltage on the positive pin PB2 (AIN0) is higher than the voltage on the negative pin PB3 (AIN1), the Analog Comparator Output, ACO, is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 59.

Figure 59. Analog Comparator Block Diagram



Note: See Figure 60 on page 128.

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.

keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a conversion by setting the ADSC bit in ADCSR, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. In certain situations, the ADC needs more clock cycles to initialization and minimize offset errors. Extended conversions take 25 ADC clock cycles and occur as the first conversion after the ADC is switched on (ADEN in ADCSR is set). Additionally, when changing voltage reference, the user may improve accuracy by disregarding the first conversion result after the reference or MUX setting was changed.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an extended conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge. In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. Using Free Running mode and an ADC clock frequency of 200 kHz gives the lowest conversion time with a maximum resolution, 65 µs, equivalent to 15 kSPS. For a summary of conversion times, see Table 43.

Figure 62. ADC Timing Diagram, Extended Conversion (Single Conversion Mode)

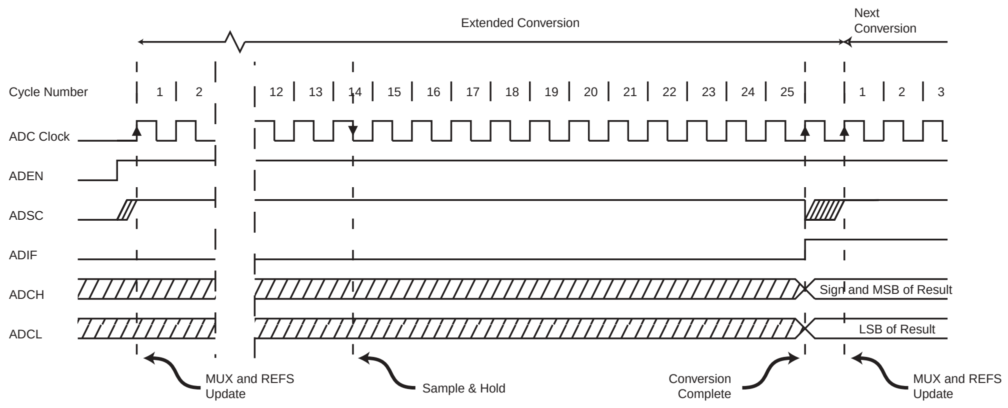
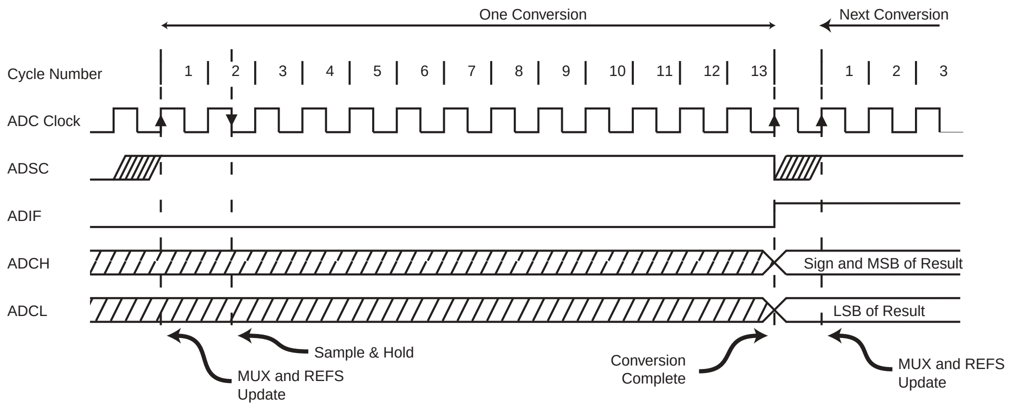


Figure 63. ADC Timing Diagram, Single Conversion



ADC Characteristics – Preliminary Data

Symbol	Parameter	Condition	Min[(2)]	Typ	Max[(3)]	Units
	Resolution	Single-ended Conversion		10		Bits
	Absolute accuracy	V_{REF} = 4V ADC clock = 200 kHz		1	2	LSB
	Absolute accuracy	V_{REF} = 4V ADC clock = 1 MHz		4		LSB
	Absolute accuracy	V_{REF} = 4V ADC clock = 2 MHz		16		LSB
	Integral Non-linearity	V_{REF} > 2V		0.5		LSB
	Differential Non-linearity	V_{REF} > 2V		0.5		LSB
	Zero Error (Offset)	V_{REF} > 2V		1		LSB
	Conversion Time	Free Running Conversion	65		260	µs
	Clock Frequency		50		200	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3[(2)]		V_{CC} + 0.3[(3)]	V
V_{REF}	Reference Voltage		2 V		AV_{CC}	V
VINT	Internal Voltage Reference		2.35	2.56	2.77	V
V_{BG}	Bandgap Voltage Reference		1.12	1.22	1.32	V
R_{REF}	Reference Input Resistance		6	10	13	kΩ
V_{IN}	Input Voltage		AGND		AREF	V
R_{AIN}	Analog Input Resistance			100		MΩ

Notes:
1. Values are guidelines only. Actual values are TBD.
2. Minimum for AVCC is 2.7V.
3. Maximum for AVCC is 5.5V.

I/O Ports

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

Port A

Port A is an 8-bit bi-directional I/O port with optional internal pull-ups.

Three I/O Memory address locations are allocated for Port A, one each for the Data Register – PORTA, $1B($3B), Data Direction Register – DDRA, $1A($3A) and the Port A Input Pins – PINA, $19($39). The Port A Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port A output buffers can sink 20 mA and thus drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Port A has an alternate function as analog inputs for the ADC. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

During Power-down mode, the Schmitt Trigger of the digital input is disconnected. This allows analog signals that are close to $V_{CC}/2$ to be present during Power-down without causing excessive power consumption.

The Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

The Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

The Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port A Input Pins address – PINA – is not a register, and this address enables access to the physical value on each Port A pin. When reading PORTA the Port A Data Latch is read, and when reading PINA, the logical values present on the pins are read.

Figure 84. Port D Schematic Diagram (Pin PD7)

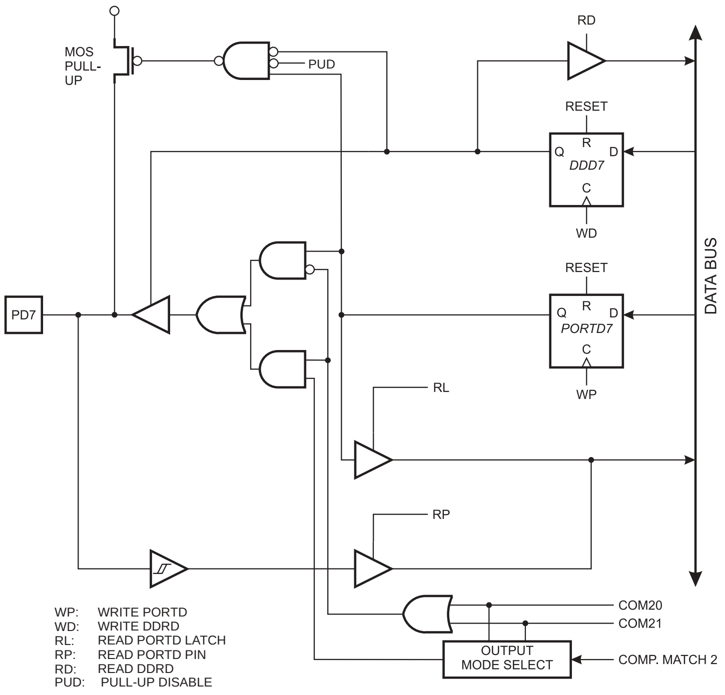


Table 58. ATmega323 Boundary-Scan Order (Continued)

Bit Number	Signal Name	Module
104	SIG_PRIVATE27	Private Section 2 (Cont.)
103	SIG_PRIVATE28	
102	SIG_PRIVATE29	
101	SIG_PRIVATE30	
100	SIG_PRIVATE31	
99	SIG_PRIVATE32	
98	SIG_PRIVATE33	
97	SIG_PRIVATE34	
96	SIG_PRIVATE35	
95	SIG_PRIVATE36	
94	SIG_PRIVATE37	
93	SIG_PRIVATE38	
92	SIG_PRIVATE39	
91	SIG_PRIVATE40	
90	SIG_PRIVATE41	
89	SIG_PRIVATE42	
88	PB0.Data	Port B
87	PB0.Control	
86	PB0.PuLLup_Disable	
85	PB1.Data	
84	PB1.Control	
83	PB1.PuLLup_Disable	
82	PB2.Data	
81	PB2.Control	
80	PB2.PuLLup_Disable	
79	PB3.Data	
78	PB3.Control	
77	PB3.PuLLup_Disable	
76	PB4.Data	
75	PB4.Control	
74	PB4.PuLLup_Disable	

Table 58. ATmega323 Boundary-Scan Order (Continued)

Bit Number	Signal Name	Module
11	PA3.Data	Port A
10	PA3.Control	
9	PA3.PuLLup_Disable	
8	PA2.Data	
7	PA2.Control	
6	PA2.PuLLup_Disable	
5	PA1.Data	
4	PA1.Control	
3	PA1.PuLLup_Disable	
2	PA0.Data	
1	PA0.Control	
0	PA0.PuLLup_Disable	

Boundary-Scan Description Language Files

Boundary-Scan Description Language (BSDL) files describe Boundary-Scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-Scan Data Register are included in this description. A BSDL file for ATmega323 is available.

Figure 94. Programming the Flash Waveforms

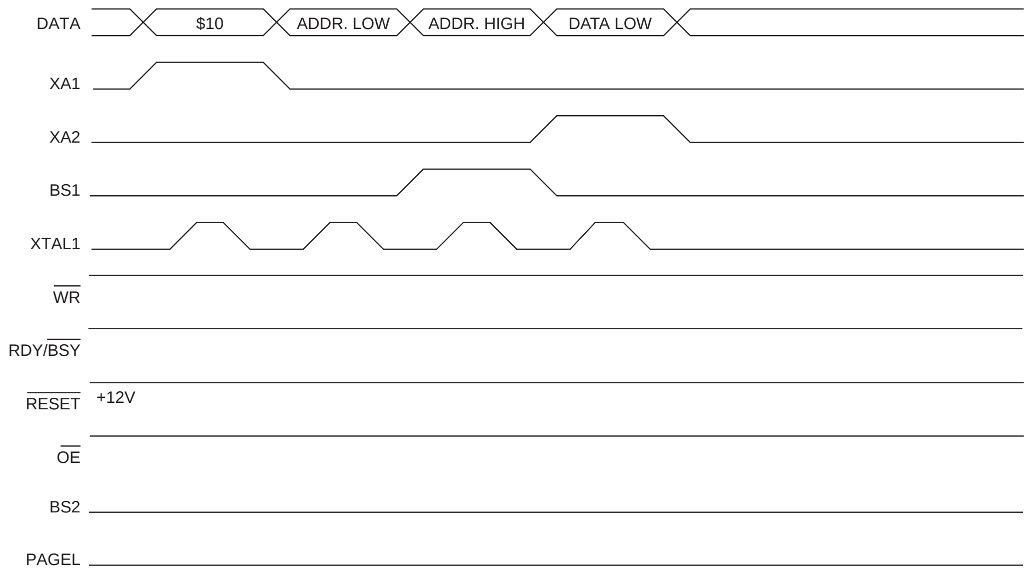
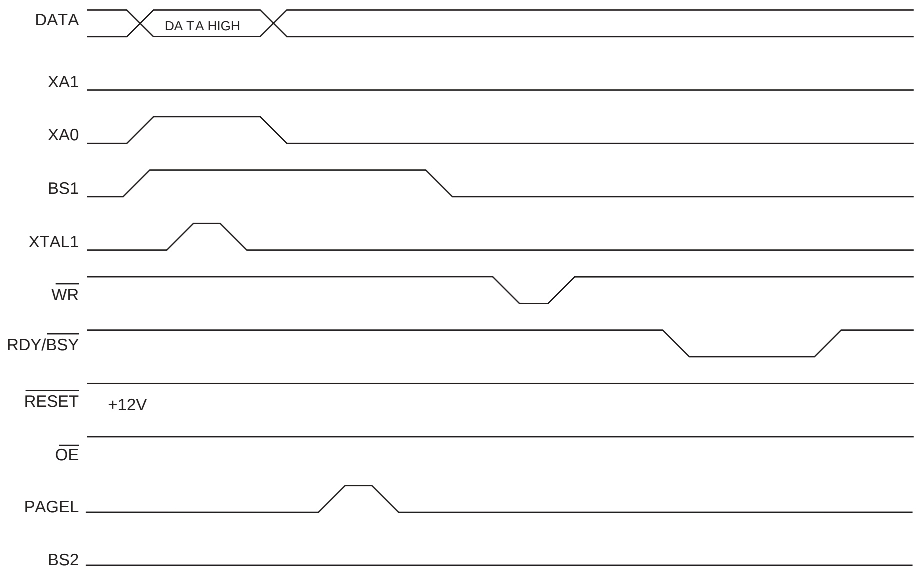


Figure 95. Programming the Flash Waveforms (continued)



Programming the EEPROM

The programming algorithm for the EEPROM Data memory is as follows (refer to “Programming the Flash” on page 190 for details on Command, Address and Data loading):

1. A: Load Command “0001 0001”.
2. H: Load Address High Byte ($00 - $03)
3. B: Load Address Low Byte ($00 - $FF)
4. C: Load Data Low Byte ($00 - $FF)

K: Write Data Low Byte

1. Set BS1 to “0”. This selects low data.
2. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{\text{BSY}}$ goes low.

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•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

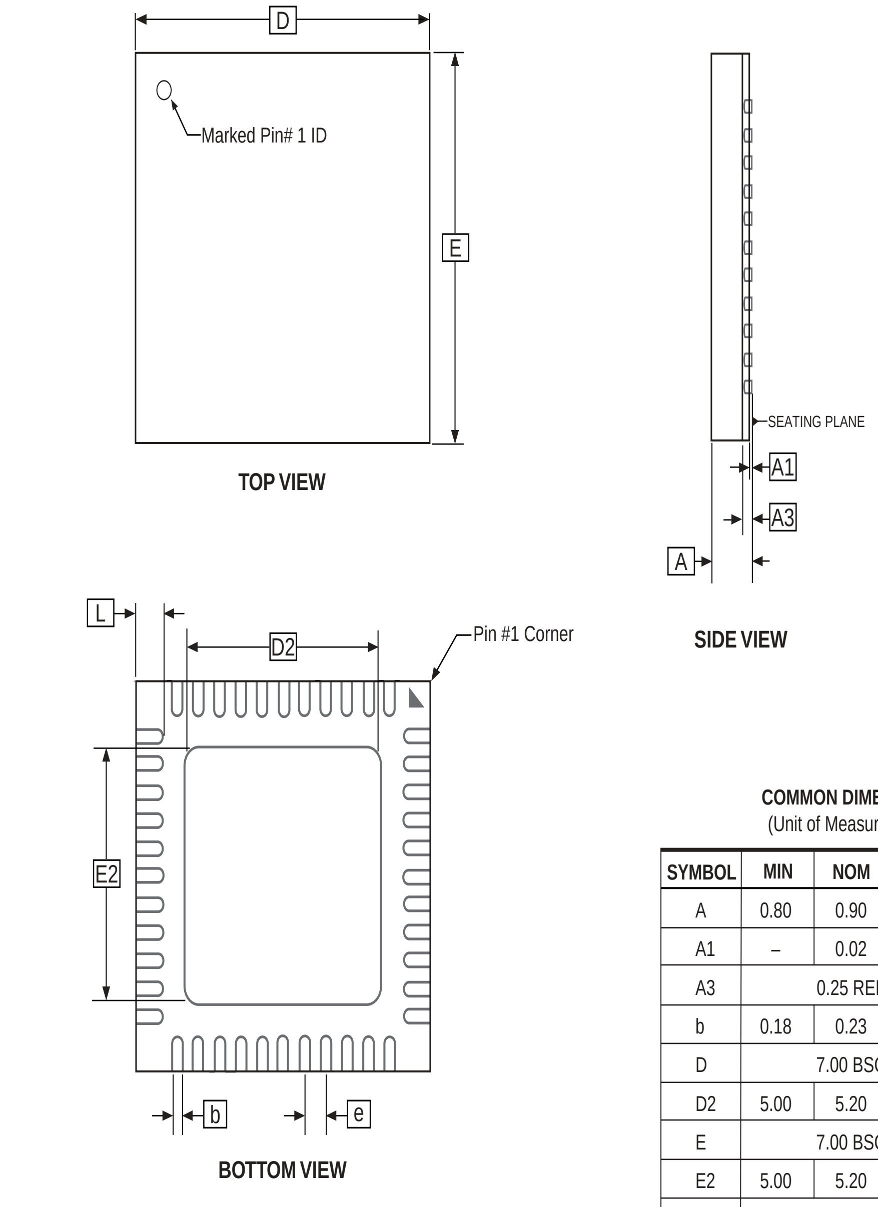
When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value $FF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire 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}.

Packaging Information

44A



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.