**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 | 16KB (8K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C |
| 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/atmega163l-4ai |

The 1,024 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 at the beginning of the Program memory. 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

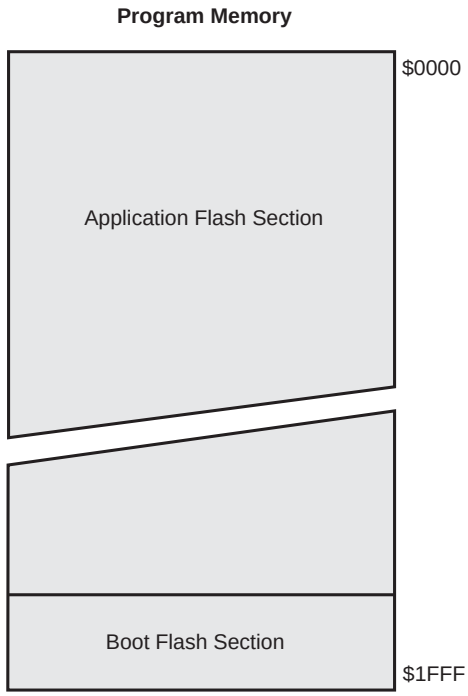


**Figure 24.** Reset Logic

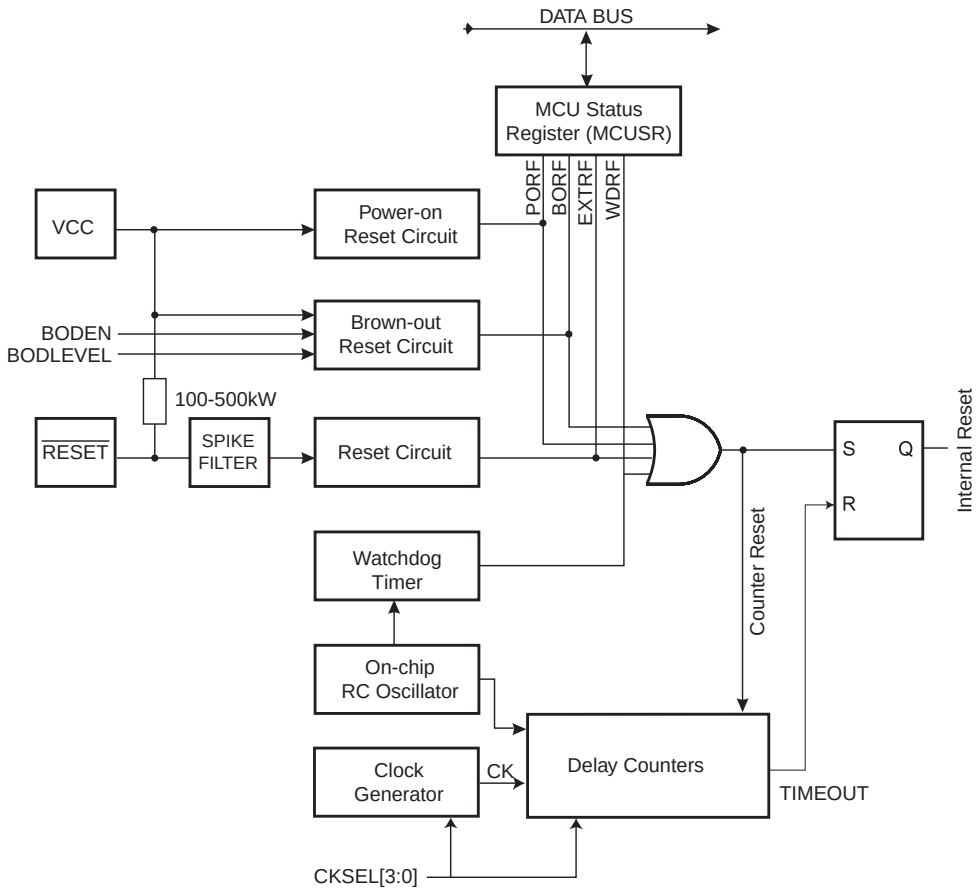



**Table 4.** Reset Characteristics ($V_{CC}$ = 5.0V)

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{POT}$ | Power-on Reset Threshold Voltage (rising) | | 1.0 | 1.4 | 1.8 | V |
| | Power-on Reset Threshold Voltage (falling)(1) | | 0.4 | 0.6 | 0.8 | V |
| $V_{RST}$ | $\overline{\text{RESET}}$ Pin Threshold Voltage | | – | – | 0.85 $V_{CC}$ | V |
| $V_{BOT}$ | Brown-out Reset Threshold Voltage | (BODLEVEL = 1) | 2.4 | 2.7 | 3.2 | V |
| | | (BODLEVEL = 0) | 3.5 | 4.0 | 4.5 | |

Notes: 1. The Power-on Reset will not work unless the supply voltage has been below $V_{POT}$ (falling).

When the prescaler is set to divide by eight, 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 48 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 14.** 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 |

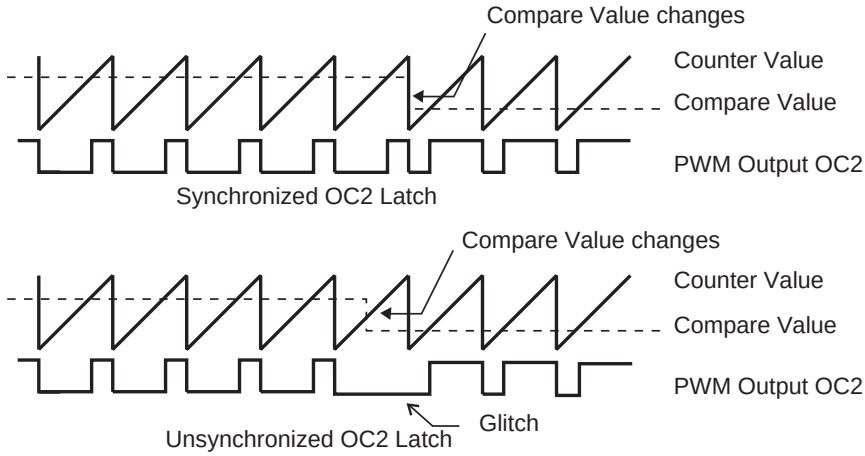
The Stop condition provides a Timer Enable/Disable function. The prescaled modes are scaled directly from the CK Oscillator clock. If the external pin modes are used for Timer/Counter1, transitions on PB1/(T1) will clock the counter even if the pin is configured as an output. This feature can give the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

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




During the time between the write and the latch operation, a read from OCR2 will read the contents of the temporary location. This means that the most recently written value always will read out of OCR2.

When the Output Compare Register contains $00 or $FF, and the up/down PWM mode is selected, the output PD7(OC2) is updated to low or high on the next compare match according to the settings of COM21/COM20. This is shown in Table 22. In overflow PWM mode, the output PD7(OC2) is held low or high only when the Output Compare Register contains $FF.

**Table 22.** PWM Outputs OCR2 = $00 or $FF

| COM21 | COM20 | OCR2 | Output OC2 |
|---|---|---|---|
| 1 | 0 | $00 | L |
| 1 | 0 | $FF | H |
| 1 | 1 | $00 | H |
| 1 | 1 | $FF | L |

In up/down PWM mode, the Timer Overflow Flag – TOV2, is set when the counter changes direction at $00. In overflow PWM mode, the Timer Overflow Flag is set as in normal Timer/Counter mode. The Timer Overflow Interrupt operates exactly as in normal Timer/Counter mode, i.e., it is executed when TOV2 is set provided that Timer Overflow Interrupt and Global Interrupts are enabled. This also applies to the Timer Output Compare Flag and Interrupt.

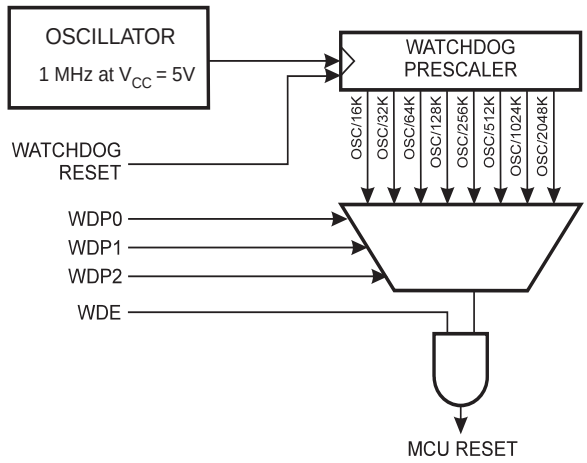
The frequency of the PWM will be Timer Clock Frequency divided by 510.

## Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical value at $V_{CC}$ = 5V. See characterization data for typical values at other $V_{CC}$ levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 23 on page 61. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega163 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 28.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

**Figure 40.** Watchdog Timer




### The Watchdog Timer Control Register – WDTCR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $21 ($41) | – | – | – | WDTOE | WDE | WDP2 | WDP1 | WDP0 | WDTCR |
| Read/Write | R | R | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and will always read as zero.

- **Bit 4 – WDTOE: Watchdog Turn-off Enable**

This bit must be set (one) when the WDE bit is cleared. Otherwise, the Watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled Watchdog Timer, the following procedure must be followed:

### The EEPROM Control Register – EECR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $1C ($3C) | – | – | – | – | EERIE | EEMWE | EEWE | EERE | EECR |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

When the I bit in SREG and EERIE are set (one), the EEPROM Ready Interrupt is enabled. When cleared (zero), the interrupt is disabled. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared (zero).

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is not essential):

1. Wait until EEWE becomes zero.
2. Write new EEPROM address to EEAR (optional).
3. Write new EEPROM data to EEDR (optional).
4. Write a logical one to the EEMWE bit in EECR.
5. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

**Caution:** An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during the four last steps to avoid these problems.

When the write access time (see Table 24) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for four cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR Register. The EEPROM read access takes one instruction, and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is not possible to set the EERE bit, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 24 lists the typical programming time for EEPROM access from the CPU

**Table 24.** EEPROM Programming Time.

| Symbol | Number of Calibrated RC Oscillator Cycles | Min Programmingn Time | Max Programming Time |
|---|---|---|---|
| EEPROM write (from CPU) | 2048 | 1.9 ms | 3.8 ms |

## Preventing EEPROM Corruption

During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using the EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

EEPROM data corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low $V_{CC}$ Reset Protection circuit can be used. If a Reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply is voltage is sufficient.
2. Keep the AVR core in Power-down Sleep Mode during periods of low $V_{CC}$. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the EEPROM Registers from unintentional writes.
3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory can not be updated by the CPU unless the boot loader software supports writing to the Flash and the Boot Lock bits are configured so that writing to the Flash memory from CPU is allowed. See “Boot Loader Support” on page 134 for details.

### The Two-wire Serial Interface (Slave) Address Register – TWAR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $02 ($22) | TWA6 | TWA5 | TWA4 | TWA3 | TWA2 | TWA1 | TWA0 | TWGCE | TWAR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |

- **Bits 7..1 – TWA: Two-wire Serial Interface (Slave) Address Register**

These seven bits constitute the slave address of the Two-wire Serial Bus unit.

- **Bit 0 – TWGCE: Two-wire Serial Interface General Call Recognition Enable Bit**

This bit enables, if set, the recognition of the General Call given over the Two-wire Serial Bus.

The TWAR should be loaded with the 7-bit slave address (in the seven most significant bits of TWAR) to which the Two-wire Serial Interface will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. The LSB of TWAR is used to enable recognition of the general call address ($00). There is an associated address comparator that looks for the slave address (or generall call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

## Two-wire Serial Interface Modes

The Two-wire Serial Interface can operate in four different modes:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfer in each mode of operation is shown in Figure 52 to Figure 55. These figures contain the following abbreviations:

S: START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\overline{A}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

In Figure 52 to Figure 55, circles are used to indicate that the Two-wire Serial Interface Interrupt Flag is set. The numbers in the circles show the status code held in TWSR. At these points, actions must be taken by the application to continue or complete the Two-wire Serial Bus transfer. The Two-wire Serial Bus transfer is suspended until the Two-wire Serial Interface Interrupt Flag is cleared by software.

The Two-wire Serial Interface Interrupt Flag is not automatically cleared by hardware when executing the interrupt routine. Software has to clear the flag to continue the Two-wire transfer. Also note that the Two-wire Serial Interface starts execution as soon as this bit is cleared, so that all access to TWAR, TWDR, and TWSR must have been completed before clearing this flag.

```
                              ; ACK should be returned after receiving first
                              ; data byte
wait12:in      r16,TWCR       ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT     ; data has been received and ACK returned
      rjmp     wait12

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

      in       r16, TWDR      ; Input received data from TWDR.
      nop                     ;<do something with received data>
      ldi      r16, (1<<TWINT) | (1<<TWEN)
      out      TWCR, r16      ; Clear TWINT bit in TWCR to start reception of
                              ; data. Not setting TWEA causes NACK to be
                              ; returned after reception of next data byte
wait13:in      r16,TWCR       ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT     ; data has been received and NACK returned
      rjmp     wait13

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

      in       r16, TWDR        ; Input received data from TWDR.
      nop                       ;<do something with received data>
      ldi      r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out      TWCR, r16        ; Clear TWINT bit in TWCR to start reception of
                                ; data. Setting TWEA causes TWI unit to enter
                                ; not addressed slave mode with reckognition of
                                ; own SLA
  ;<Wait for next data transmission or do something else>
```

**Table 35.** Status Codes for Slave Transmitter Mode

| 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 | |
| $A8 | Own SLA+R has been received; ACK has been returned | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $B0 | Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $B8 | Data byte in TWDR has been transmitted; ACK has been received | Load data byte or | X | 0 | 1 | 0 | Last data byte will be transmitted and NOT ACK should be received |
| | | Load data byte | X | 0 | 1 | 1 | Data byte will be transmitted and ACK should be received |
| $C0 | Data byte in TWDR has been transmitted; NOT ACK has been received | No TWDR action or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | No TWDR action or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” |
| | | No TWDR action or | 1 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free |
| | | No TWDR action | 1 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free |
| $C8 | Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received | No TWDR action or | 0 | 0 | 1 | 0 | Switched to the not addressed Slave mode; no recognition of own SLA or GCA |
| | | No TWDR action or | 0 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” |
| | | No TWDR action or | 1 | 0 | 1 | 0 | Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free |
| | | No TWDR action | 1 | 0 | 1 | 1 | Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free |

**Figure 55.** Formats and States in the Slave Transmitter Mode

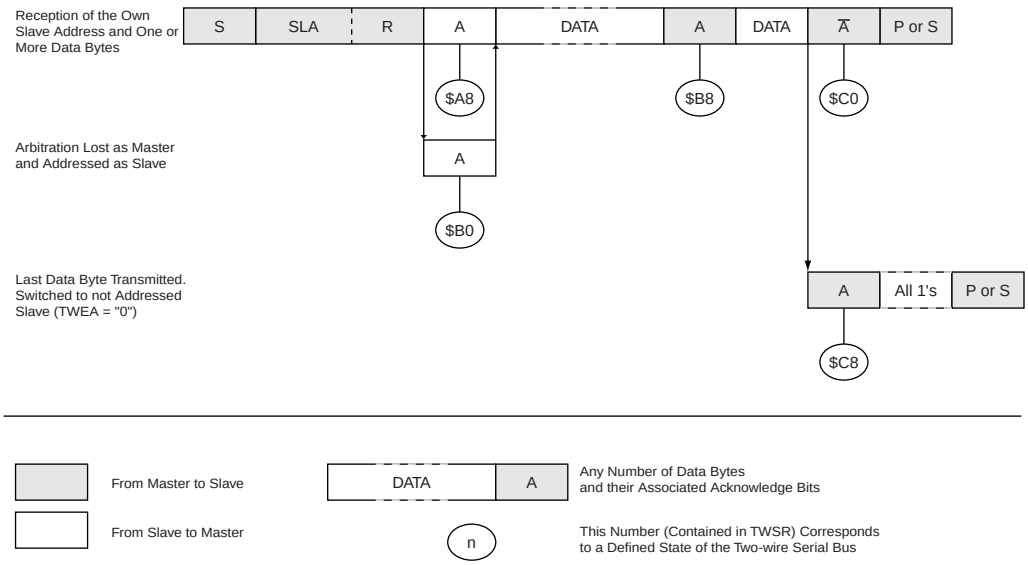
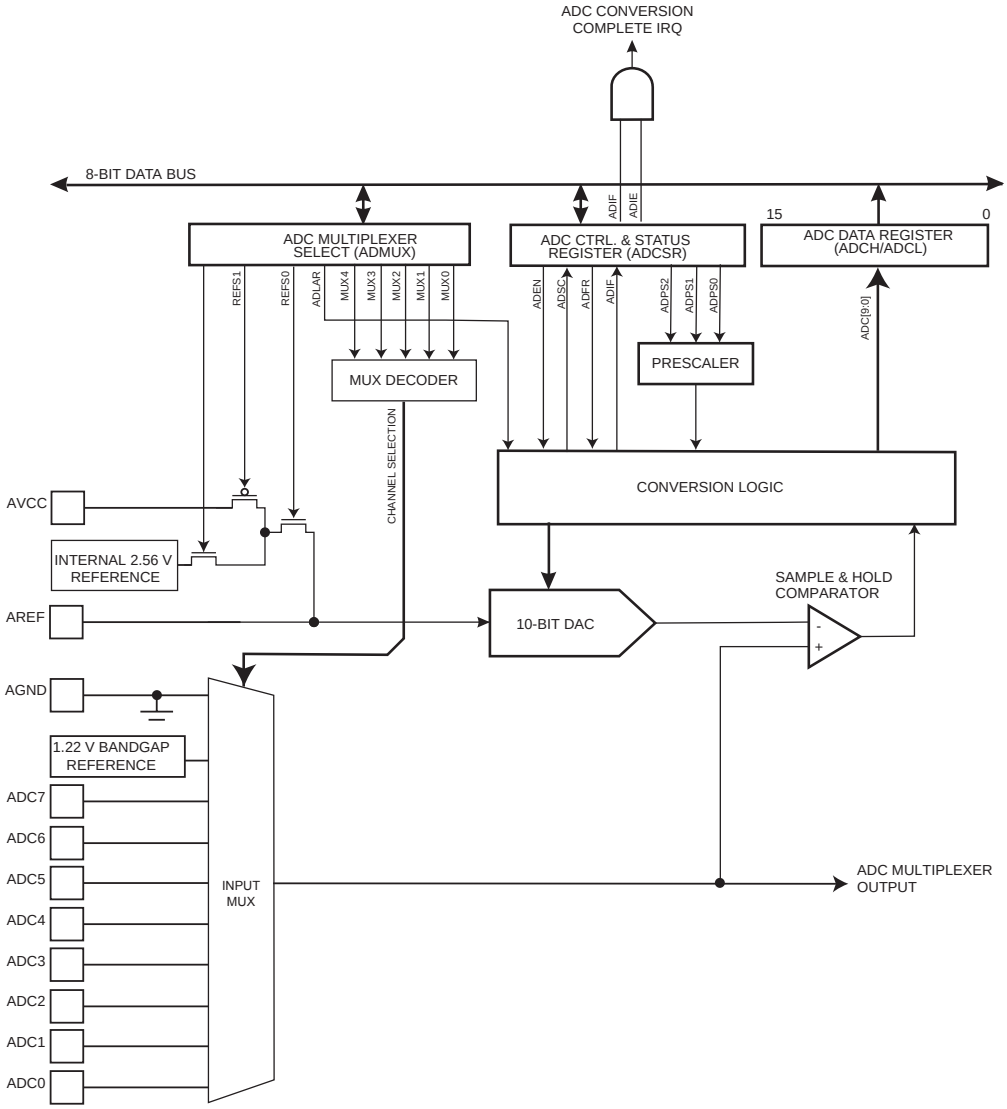


**Figure 57.** Analog to Digital Converter Block Schematic




## 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 1.22V ($V_{BG}$), 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.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSR. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not

**Table 53.** Boot Lock Bit0 Protection Modes (Application Section)(1)

| BLB0 mode | BLB02 | BLB01 | Protection |
|---|---|---|---|
| 1 | 1 | 1 | No restrictions for SPM, LPM accessing the Application section |
| 2 | 1 | 0 | SPM is not allowed to write to the Application section |
| 3 | 0 | 0 | SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section |
| 4 | 0 | 1 | LPM executing from the Boot Loader section is not allowed to read from the Application section |

Note: 1. "1" means unprogrammed, "0" means programmed

**Table 54.** Boot Lock Bit1 Protection Modes (Boot Loader Section)(1)

| BLB1 mode | BLB12 | BLB11 | Protection |
|---|---|---|---|
| 1 | 1 | 1 | No restrictions for SPM, LPM accessing the Boot Loader section |
| 2 | 1 | 0 | SPM is not allowed to write to the Boot Loader section |
| 3 | 0 | 0 | SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If code is executed from Boot section, the interrupts are disabled when BLB12 is programmed. |
| 4 | 0 | 1 | LPM executing from the Application section is not allowed to read from the Boot Loader section. If code is executed from Boot section, the interrupts are disabled when BLB12 is programmed. |

Note: 1. "1" means unprogrammed, "0" means programmed

**Setting the Boot Loader Lock Bits by SPM**

To set the Boot Loader Lock bits, write the desired data to R0, write "00001001" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The only accessible Lock bits are the Boot Lock bits that may prevent the Application and Boot Loader section from any software update by the MCU.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **R0** | **1** | **1** | **BLB12** | **BLB11** | **BLB02** | **BLB01** | **1** | **1** |

If bits 5..2 in R0 are cleared (zero), the corresponding Boot Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCR.

**Reading the Fuse and Lock Bits from Software**

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with $0001 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within five CPU cycles after the BLBSET and SPMEN bits are set in SPMCR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no SPM, or LPM, instruction is executed within four, respectively five, CPU cycles. When BLBSET and SPMEN are cleared, LPM will work as described in "Constant Addressing Using The LPM and SPM Instructions" on page 15 and in the Instruction set Manual.

```
sbrc    temp1, SPMEN
rjmp    Wait_spm
ret
```

## Program and Data Memory Lock Bits

The ATmega163 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 55. The Lock bits can only be erased to “1” with the Chip Erase command.

**Table 55.** Lock Bit Protection Modes

| Memory Lock Bits | | | |
|---|---|---|---|
| **LB mode** | **LB1** | **LB2** | **Protection Type** |
| 1 | 1 | 1 | No memory lock features enabled for Parallel and Serial Programming. |
| 2 | 0 | 1 | Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1) |
| 3 | 0 | 0 | Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1) |
| **BLB0 mode** | **BLB01** | **BLB02** | |
| 1 | 1 | 1 | No restrictions for SPM, LPM accessing the Application section. |
| 2 | 0 | 1 | SPM is not allowed to write to the Application section. |
| 3 | 0 | 0 | SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. |
| 4 | 1 | 0 | LPM executing from the Boot Loader section is not allowed to read from the Application section. |
| **BLB1 mode** | **BLB11** | **BLB12** | |
| 1 | 1 | 1 | No restrictions for SPM, LPM accessing the Boot Loader section. |
| 2 | 0 | 1 | SPM is not allowed to write to the Boot Loader section. |
| 3 | 0 | 0 | SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If code executed from the Boot Section, the interrupts are disabled when BLB12 is programmed. |
| 4 | 1 | 0 | LPM executing from the Application section is not allowed to read from the Boot Loader section. If code executed from the Boot Section, the interrupts are disabled when BLB12 is programmed. |

Note: 1. Program the Fuse bits before programming the Lock bits.

3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give WR a negative pulse. This starts the Chip Erase. RDY/BSY goes low.
5. Wait until RDY/BSY goes high before loading a new command.

**Programming the Flash**

The Flash is organized as 128 pages of 128 bytes each. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

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

B. Load Address Low Byte

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

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data Low Byte ($00 - $FF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Latch Data Low Byte

1. Set BS1 to “0”. This selects Low Data Byte.
2. Give PAGEL a positive pulse. This latches the data Low Byte. (See Figure 82 for signal waveforms)

E. Load Data High Byte

1. Set BS1 to “1”. This selects High Data Byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data High Byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the data byte.

F. Latch Data High Byte

1. Set BS1 to “1”. This selects High Data Byte.
2. Give PAGEL a positive pulse. This latches the data High Byte.

G. Repeat B through F 64 times to fill the page buffer.

To address a page in the Flash, seven bits are needed (128 pages). The five most significant bits are read from address high byte as described in section “H” below. The two least significant page address bits however, are the two most significant bits (bit7 and bit6) of the latest loaded address low byte as described in section “B”.

H. Load Address High byte

1. 1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address High Byte ($00 - $1F).
4. Give XTAL1 a positive pulse. This loads the address High Byte.

I. Program Page

1. Give WR a negative pulse. This starts programming of the entire page of data. RDY/BSYgoes low.
2. Wait until RDY/BSY goes high.
   (See Figure 83 for signal waveforms)

J. End Page Programming

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

K. Repeat A through J 128 times or until all data has been programmed.

**Figure 82.** Programming the Flash Waveforms

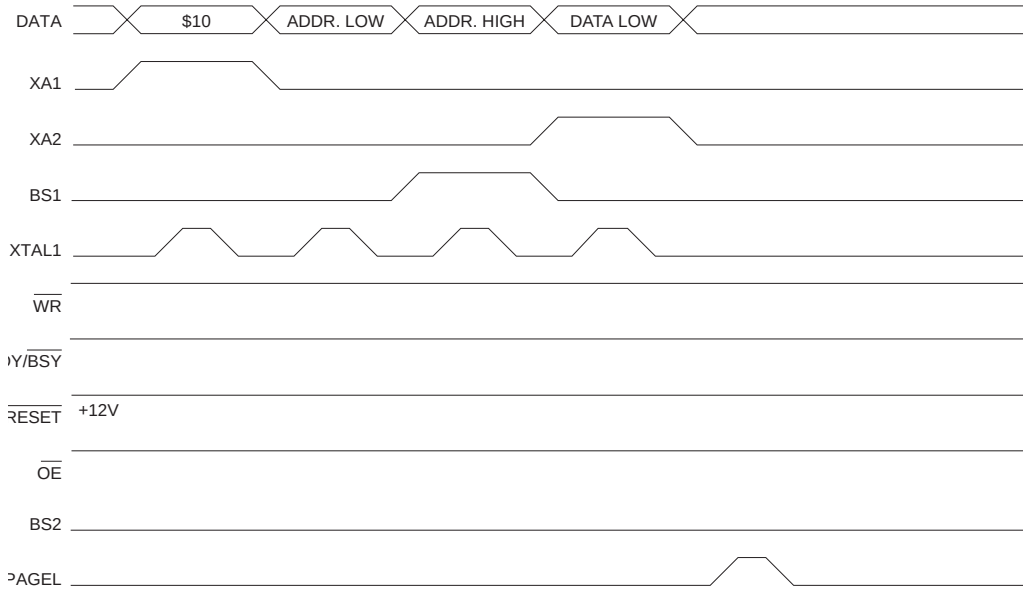
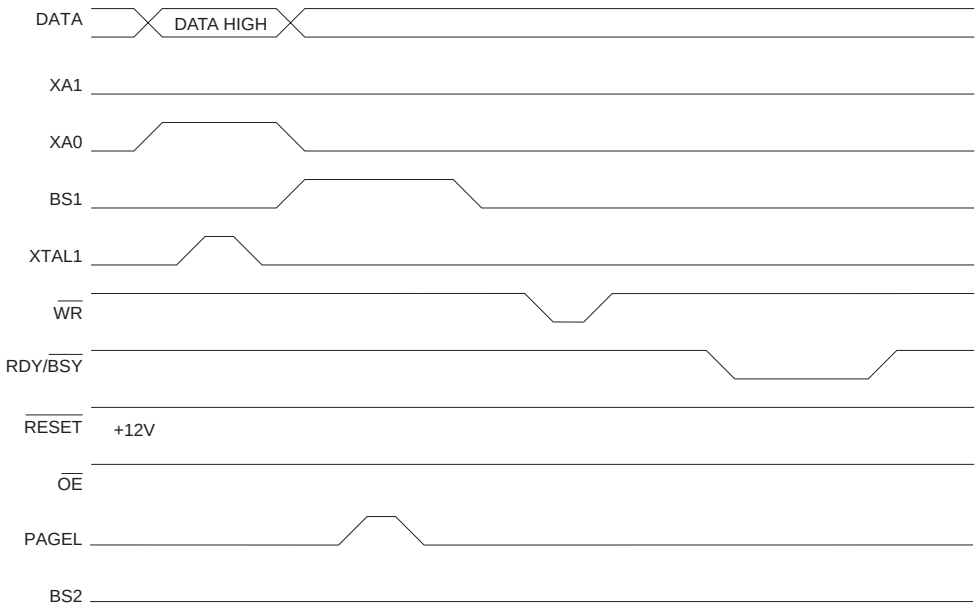


**Figure 83.** Programming the Flash Waveforms (continued)




**Programming the EEPROM**

The programming algorithm for the EEPROM Data Memory is as follows (refer to "Programming the Flash" on page 147 for details on Command, Address and Data loading):

1. A: Load Command "0001 0001".
2. H: Load Address High Byte ($00 - $01)
3. B: Load Address Low Byte ($00 - $FF)
4. E: Load Data Low Byte ($00 - $FF)

L: Write Data Low Byte

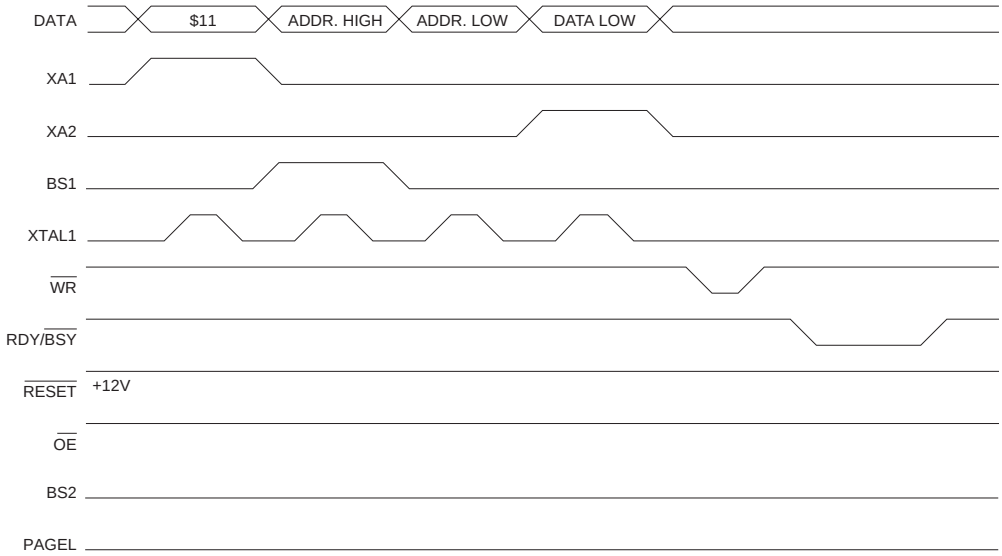
1. Set BS to "0". This selects low data.
2. Give WR a negative pulse. This starts programming of the data byte. RDY/BSY goes low.
3. Wait until to RDY/BSY goes high before programming the next byte. (See Figure 84 for signal waveforms)

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

- The command needs only be loaded once when writing or reading multiple memory locations.
- Address high byte needs only be loaded before programming a new 256 word page in the EEPROM.
- Skip writing the data value $FF, that is the contents of the entire EEPROM after a Chip Erase.

These considerations also applies to Flash, EEPROM and Signature bytes reading.

**Figure 84.** Programming the EEPROM Waveforms




**Reading the Flash**

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 147 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
2. H: Load Address High Byte ($00 - $1F).
3. B: Load Address Low Byte ($00 - $FF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
5. Set BS to “1”. The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

**Reading the EEPROM**

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 147 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. H: Load Address High Byte ($00 - $01).
3. B: Load Address ($00 - $FF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

**Programming the Fuse Low Bits**

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 147 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
   Bit 7 = BODLEVEL Fuse bit
   Bit 6 = BODEN Fuse bit
   Bit 5 = SPIEN Fuse bit
   Bit 3..0 = CKSEL3..0 Fuse bits
   Bit 4 = “1”. This bit is reserved and should be left unprogrammed (“1”).
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

## External Clock Drive

**Table 63.** External Clock Drive

| Symbol | Parameter | $V_{CC}$ = 2.7V to 5.5V | | $V_{CC}$ = 4.0V to 5.5V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | 4 | 0 | 8 | MHz |
| $t_{CLCL}$ | Clock Period | 250 | | 125 | | ns |
| $t_{CHCX}$ | High Time | 100 | | 50 | | ns |
| $t_{CLCX}$ | Low Time | 100 | | 50 | | ns |
| $t_{CLCH}$ | Rise Time | | 1.6 | | 0.5 | µs |
| $t_{CHCL}$ | Fall Time | | 1.6 | | 0.5 | µs |

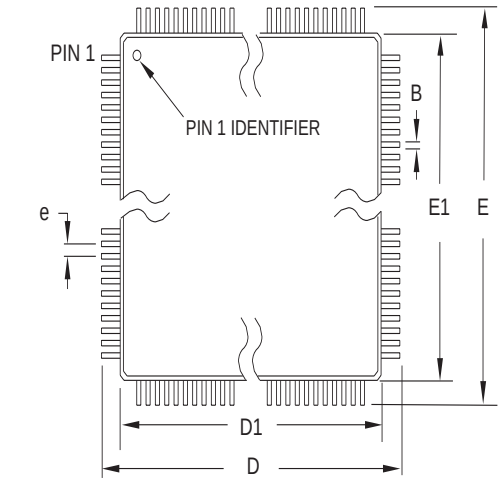
**Table 64.** External RC Oscillator, typical frequencies

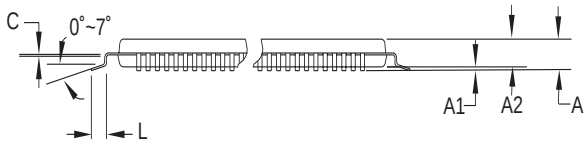
| R [kΩ] | C [pF] | f |
|---|---|---|
| 100 | 70 | 100 kHz |
| 31.5 | 20 | 1.0 MHz |
| 6.5 | 20 | 4.0 MHz |

Note: R should be in the range 3kΩ - 100kΩ, and C should be at least 20pF. The C values given in the table includes pin capacitance. This will vary with package type.

## Packaging Information

### 44A




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

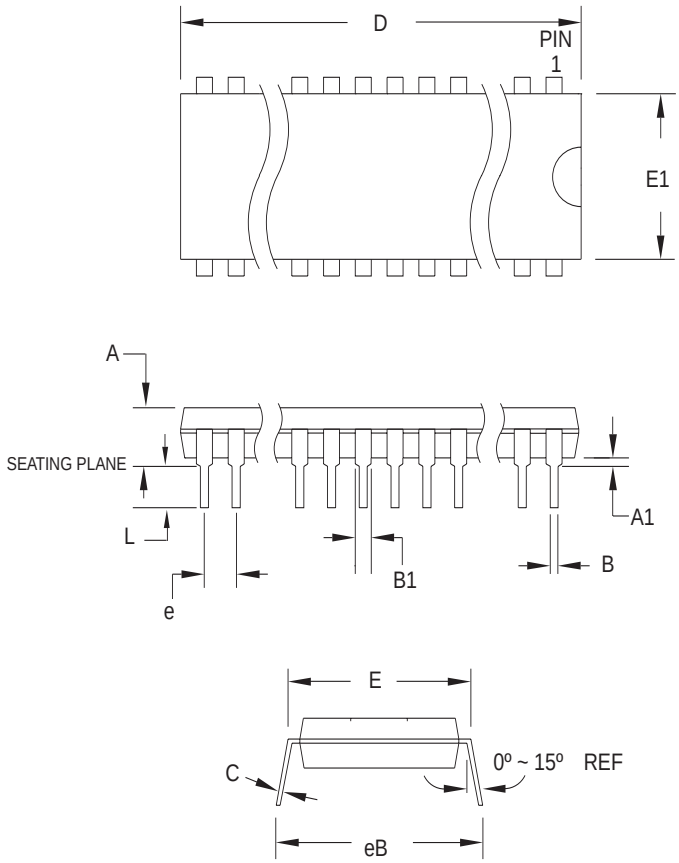
| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | – | – | 1.20 | |
| A1 | 0.05 | – | 0.15 | |
| A2 | 0.95 | 1.00 | 1.05 | |
| D | 11.75 | 12.00 | 12.25 | |
| D1 | 9.90 | 10.00 | 10.10 | Note 2 |
| E | 11.75 | 12.00 | 12.25 | |
| E1 | 9.90 | 10.00 | 10.10 | Note 2 |
| B | 0.30 | – | 0.45 | |
| C | 0.09 | – | 0.20 | |
| L | 0.45 | – | 0.75 | |
| e | 0.80 TYP | | | |

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

| | TITLE | DRAWING NO. | REV. |
|---|---|---|---|
| 2325 Orchard Parkway<br>San Jose, CA 95131 | **44A,** 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP) | 44A | B |

## 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 |