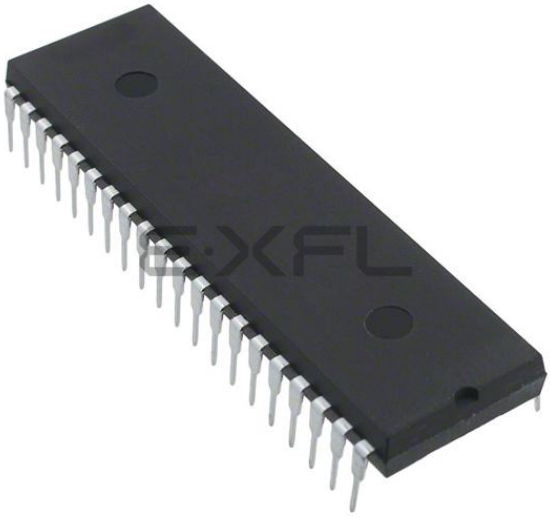


**Welcome to E-XFL.COM**

## 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 | 0°C ~ 70°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/atmega163l-4pc |

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

## The ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the Register File are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. ATmega163 also provides a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the Instruction Set section for a detailed description.

## The In-System Self-Programmable Flash Program Memory

The ATmega163 contains 16K bytes On-chip In-System Self-Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 8K x 16. The Flash Program memory space is divided in two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 1,000 write/erase cycles. The ATmega163 Program Counter (PC) is 13 bits wide, thus addressing the 8,192 Program Memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail on page 134. See also page 154 for a detailed description on Flash data serial downloading.

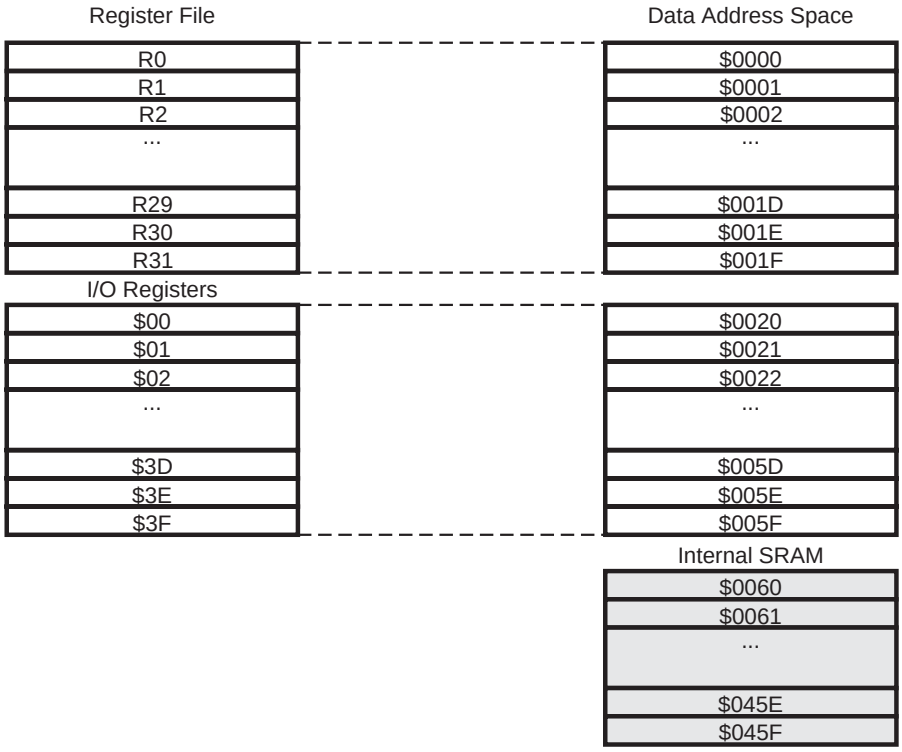
Constant tables can be allocated within the entire Program Memory address space (see the LPM – Load Program Memory instruction description).

See also page 12 for the different Program Memory Addressing modes.

## The SRAM Data Memory

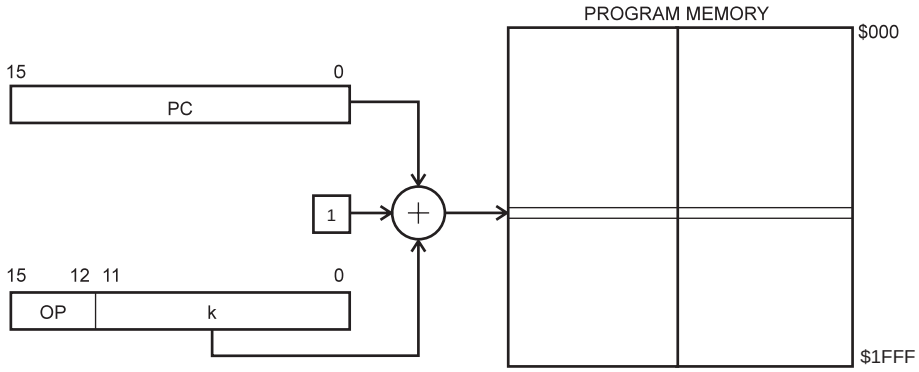
Figure 9 shows how the ATmega163 SRAM Memory is organized.

**Figure 9.** SRAM Organization



**Relative Program Addressing, RJMP and RCALL**

**Figure 20.** Relative Program Memory Addressing




Program execution continues at address PC + k + 1.
The relative address k is from -2,048 to 2,047.

## The EEPROM Data Memory

The ATmega163 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on page 62 specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For the SPI data downloading, see page 154 for a detailed description.

## Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the main Oscillator for the chip. No internal clock division is used.

Figure 21 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

**Figure 21.** The Parallel Instruction Fetches and Instruction Executions

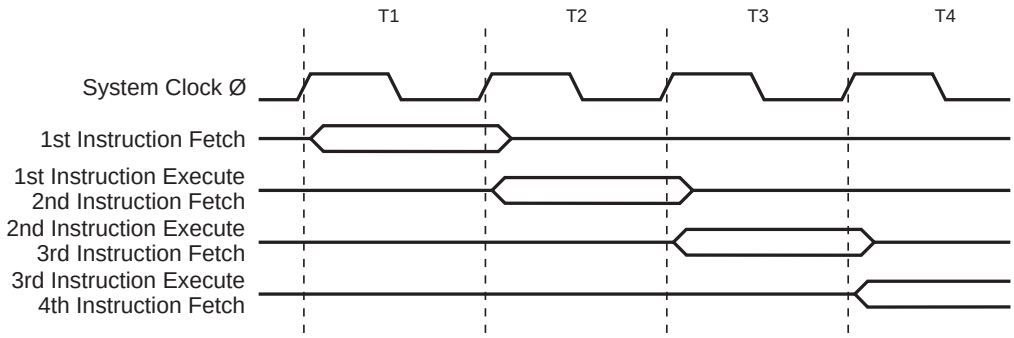



Figure 22 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

### The Stack Pointer – SP

The ATmega163 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the ATmega163 data memory has $460 locations, 11 bits are used.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $3E ($5E) | – | – | – | – | – | SP10 | SP9 | SP8 | SPH |
| $3D ($5D) | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SPL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | 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 | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

## Reset and Interrupt Handling

The ATmega163 provides 17 different interrupt sources. These interrupts and the separate Reset Vector, each have a separate Program Vector in the Program Memory space. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the Status Register in order to enable the interrupt.

The lowest addresses in the Program Memory space are automatically defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in Table 3. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0, etc.

**Table 3.** Reset and Interrupt Vectors

| Vector No. | Program Address | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $000(1) | RESET | External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset |
| 2 | $002 | INT0 | External Interrupt Request 0 |
| 3 | $004 | INT1 | External Interrupt Request 1 |
| 4 | $006 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 5 | $008 | TIMER2 OVF | Timer/Counter2 Overflow |
| 6 | $00A | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 7 | $00C | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 8 | $00E | TIMER1 COMPB | Timer/Counter1 Compare Match B |
| 9 | $010 | TIMER1 OVF | Timer/Counter1 Overflow |
| 10 | $012 | TIMER0 OVF | Timer/Counter0 Overflow |
| 11 | $014 | SPI, STC | Serial Transfer Complete |
| 12 | $016 | UART, RXC | UART, Rx Complete |

**Figure 24.** Reset Logic

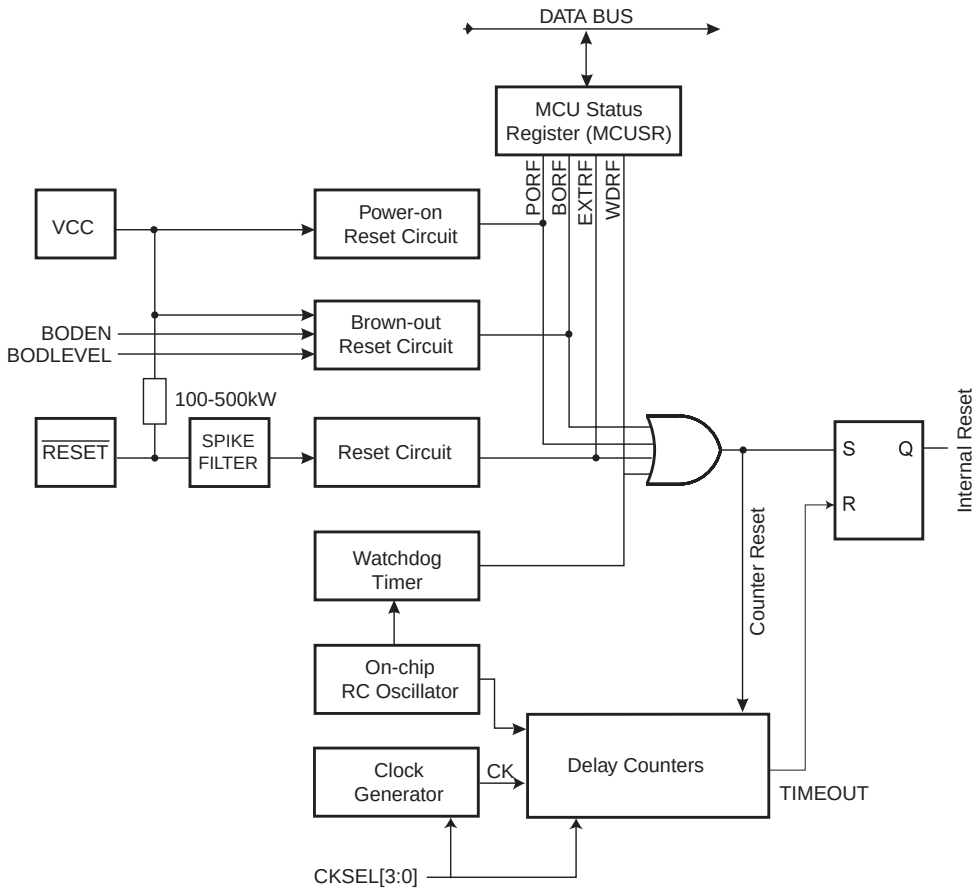
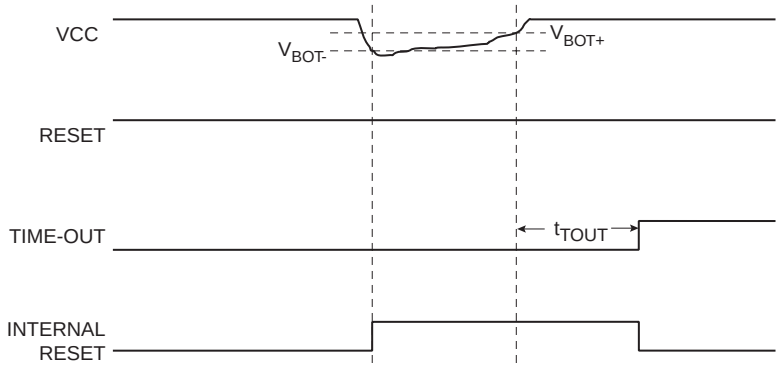



**Table 4.** Reset Characteristics ($V_{CC}$ = 5.0V)

| 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).

**Figure 28.** Brown-out Reset During Operation

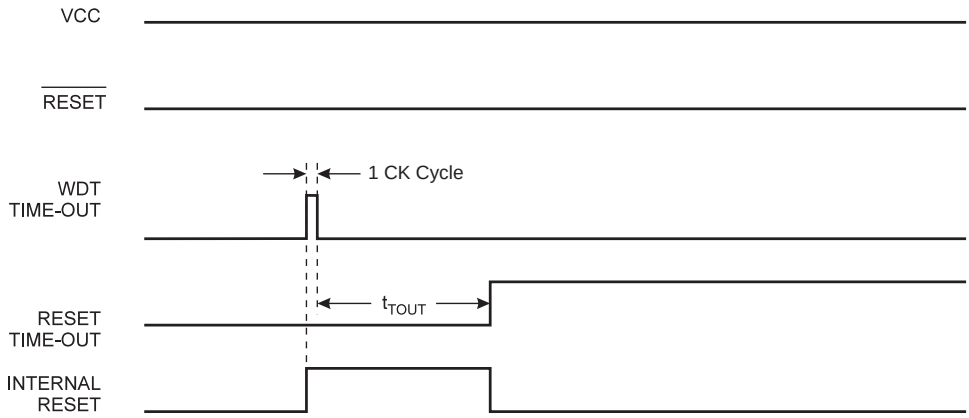



The hysteresis on $V_{BOT}$: $V_{BOT+} = V_{BOT} + 25$ mV, $V_{BOT-} = V_{BOT} - 25$ mV

### Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out Period $t_{TOUT}$. Refer to page 60 for details on operation of the Watchdog Timer.

**Figure 29.** Watchdog Reset During Operation




### MCU Status Register – MCUSR

The MCU Status Register provides information on which reset source caused an MCU Reset.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $34 ($54) | – | – | – | – | WDRF | BORF | EXTRF | PORF | MCUSR |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | See Bit Description | | | | |

- **Bits 7..4 – Res: Reserved Bits**

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

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the Flag.

**MCU Control Register – MCUCR**

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $35 ($55) | – | SE | SM1 | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| Read/Write | R | 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 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and always reads as zero.

- **Bit 6 – 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.

- **Bits 5, 4 – SM1/SM0: Sleep Mode Select Bits 1 and 0**

These bits select between the three available sleep modes as shown in Table 7.

**Table 7.** Sleep Mode Select

| SM1 | SM0 | Sleep Mode |
|---|---|---|
| 0 | 0 | Idle |
| 0 | 1 | ADC Noise Reduction |
| 1 | 0 | Power-down |
| 1 | 1 | Power-save |

- **Bits 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-Flag and the corresponding interrupt mask in the GIMSK are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 8. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

**Table 8.** Interrupt 1 Sense Control

| ISC11 | ISC10 | Description |
|---|---|---|
| 0 | 0 | The low level of INT1 generates an interrupt request. |
| 0 | 1 | Any logical change on INT1 generates an interrupt request. |
| 1 | 0 | The falling edge of INT1 generates an interrupt request. |
| 1 | 1 | The rising edge of INT1 generates an interrupt request. |

## Calibrated Internal RC Oscillator

The calibrated internal Oscillator provides a fixed 1 MHz (nominal) clock at 5V and 25°C. This clock may be used as the system clock. See the section “Clock Options” on page 5 for information on how to select this clock as the system clock. This Oscillator can be calibrated by writing the calibration byte to the OSCCAL Register. When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. At 5V and 25°C, the pre-programmed calibration byte gives a frequency within ± 1% of the nominal frequency. For details on how to use the pre-programmed calibration value, see “Calibration Byte” on page 144.

### Oscillator Calibration Register – OSCCAL

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $31 ($51) | CAL7 | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Writing the calibration byte to this address will trim the internal Oscillator to remove process variations from the Oscillator frequency. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the internal Oscillator. Writing $FF to the register gives the highest available frequency.

The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write operation may fail. Note that the Oscillator is intended for calibration to 1.0MHz, thus tuning to other values is not guaranteed.

**Table 10.** Internal RC Oscillator Frequency Range.

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

### Special Function I/O Register – SFIOR

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

- **Bit 7..4 – Res: Reserved Bits**

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

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

When this bit is set (one) and the ADC is switched off (ADEN in ADCSR is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is cleared (zero), AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see “Analog Comparator Multiplexed Input” on page 104.

clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

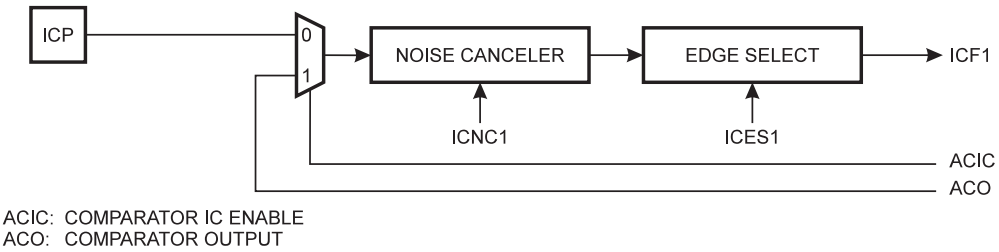
The 16-bit Timer/Counter1 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B (OCR1A and OCR1B) as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions includes optional clearing of the counter on Compare A Match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as an 8-, 9-, or 10-bit Pulse Width Modulator (PWM). In this mode the counter and the OCR1A/OCR1B Registers serve as a dual glitch-free stand-alone PWM with centered pulses. Alternatively, the Timer/Counter1 can be configured to operate at twice the speed in PWM mode, but without centered pulses. Refer to page 48 for a detailed description of this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register – ICR1, triggered by an external event on the Input Capture Pin – ICP. The actual capture event settings are defined by the Timer/Counter1 Control Register – TCCR1B. In addition, the Analog Comparator can be set to trigger the Input Capture. Refer to the section, “The Analog Comparator” on page 102, for details on this. The ICP pin logic is shown in Figure 34.

**Figure 34.** ICP Pin Schematic Diagram




If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over four samples, and all four must be equal to activate the Capture Flag.

**Assembly Code Example – Slave Transmitter Mode**

```
; Part specific include file and TWI include file must be included.
; <Initialize registers, including TWAR, TWBR and TWCR>

      ldi   r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out   TWCR, r16                          ; Enable TWI in Slave
Transmitter Mode

; <Receive START condition and SLA+R>

wait14:in   r16,TWCR       ; Wait for TWINT flag set. This indicates that
      sbrs  r16, TWINT     ; SLA+R has been received, and ACK/NACK has
      rjmp  wait14         ; been returned

      in    r16, TWSR      ; Check value of TWI Status Register. If status
      cpi   r16, ST_SLA_ACK; different from ST_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<<TWEA) | (1<<TWEN)
      out   TWCR, r16      ; Clear TWINT bit in TWCR to start transmission of
                           ; data. Setting TWEA indicates that ACK should be
                           ; received when transfer finished
; <Send more data bytes if needed>
wait15: in       r16,TWCR  ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT  ; data has been transmitted, and ACK/NACK has
      rjmp     wait15      ; been received

      in       r16, TWSR   ; Check value of TWI Status Register. If status
      cpi      r16, ST_DATA_ACK ; different from ST_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<<TWEA) | (1<<TWEN)
      out      TWCR, r16   ; Clear TWINT bit in TWCR to start transmission of
                           ; data. Setting TWEA indicates that ACK should be
                           ; received when transfer finished
wait16:in      r16,TWCR    ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT  ; data has been transmitted, and ACK/NACK has
      rjmp     wait16      ; been received

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

      ldi      r16, 0x55   ; Load data (here, data = 0x55) 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. Not setting TWEA indicates that NACK should
```

```
.equ    ST_DATA_NACK    =$C0    ;Data byte has been tramsmitted and NACK
                                ;received
.equ    ST_LAST_DATA    =$C8    ;Last byte in I2DR has been transmitted (TWEA =
                                ;'0'), ACK has been received

;***** Slave Receiver staus codes *****
.equ    SR_SLA_ACK      =$60    ;SLA+R has been received and ACK returned
.equ    SR_ARB_LOST_SLA_ACK=$68;Arbitration lost in SLA+R/W as Master. Own
                                ;SLA+R has been received and ACK returned
.equ    SR_GCALL_ACK    =$70    ;Generall call has been received and ACK
                                ;returned
.equ    SR_ARB_LOST_GCALL_ACK=$78;Arbitration lost in SLA+R/W as Master.
                                ;General Call has been received and ACK
                                ;returned
.equ    SR_DATA_ACK     =$80    ;Previously addressed with own SLA+W. Data byte
                                ;has been received and ACK returned
.equ    SR_DATA_NACK    =$88    ;Previously addressed with own SLA+W. Data byte
                                ;has been received and NACK returned
.equ    SR_GCALL_DATA_ACK=$90;Previously addressed with General Call.Data
                                ;byte has been received and ACK returned
.equ    SR_GCALL_DATA_NACK=$98;Previously addressed with General Call. Data
                                ;byte has been received and NACK returned
.equ    SR_STOP         =$A0    ;A STOP condition or repeated START condition
                                ;has been received while still addressed as a
                                ;slave
;***** Miscellanous States *****
.equ    NO_INFO         =$F8    ;No relevant state information; TWINT = '0'
.equ    BUS_ERROR       =$00    ;Bus error due to illegal START or STOP
                                ;condition
```

## Analog Comparator Multiplexed Input

It is possible to select any of the PA7..0 (ADC7..0) pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in SFIOR) is set (one) and the ADC is switched off (ADEN in ADCSR is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 38. If ACME is cleared (zero) or ADEN is set (one), PB3 (AIN1) is applied to the negative input to the Analog Comparator.

**Table 38.** Analog Comparator Multiplexed Input

| ACME | ADEN | MUX2..0 | Analog Comparator Negative Input |
|---|---|---|---|
| 0 | x | xxx | AIN1 |
| 1 | 1 | xxx | AIN1 |
| 1 | 0 | 000 | ADC0 |
| 1 | 0 | 001 | ADC1 |
| 1 | 0 | 010 | ADC2 |
| 1 | 0 | 011 | ADC3 |
| 1 | 0 | 100 | ADC4 |
| 1 | 0 | 101 | ADC5 |
| 1 | 0 | 110 | ADC6 |
| 1 | 0 | 111 | ADC7 |

# Analog to Digital Converter

## Feature List

- **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 ATmega163 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 57.

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 perfomance. See “Internal Voltage Reference” on page 29 for a description of the internal voltage reference.

### The ADC Multiplexer Selection Register – ADMUX

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $07 ($27) | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 | ADMUX |
| 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, 6 – REFS1..0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 17. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set). The user should disregard the first conversion result after changing these bits to obtain maximum accuracy. The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

**Table 40.** Voltage Reference Selections for ADC

| REFS1 | REFS0 | Voltage Reference Selection |
|---|---|---|
| 0 | 0 | AREF, Internal Vref turned off |
| 0 | 1 | AVCC with external capacitor at AREF pin |
| 1 | 0 | Reserved |
| 1 | 1 | Internal 2.56V Voltage Reference with external capacitor at AREF pin |

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. If ADLAR is cleared, the result is right adjusted. If ADLAR is set, the result is left adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see “The ADC Data Register – ADCL and ADCH” on page 112.

- **Bits 4..0 – MUX4..MUX0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 41 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set).

**Table 41.** Input Channel Selections

| MUX4..0 | Single-ended Input |
|---|---|
| 00000 | ADC0 |
| 00001 | ADC1 |
| 00010 | ADC2 |
| 00011 | ADC3 |
| 00100 | ADC4 |
| 00101 | ADC5 |
| 00110 | ADC6 |
| 00111 | ADC7 |

## Port D

Port D is an 8 bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for Port D, one each for the Data Register – PORTD, $12($32), Data Direction Register – DDRD, $11($31) and the Port D Input Pins – PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. Some Port D pins have alternate functions as shown in Table 49.

**Table 49.** Port D Pins Alternate Functions

| Port Pin | Alternate Function |
|---|---|
| PD0 | RXD (UART Input Pin) |
| PD1 | TXD (UART Output Pin) |
| PD2 | INT0 (External Interrupt 0 Input) |
| PD3 | INT1 (External Interrupt 1 Input) |
| PD4 | OC1B (Timer/Counter1 Output CompareB Match Output) |
| PD5 | OC1A (Timer/Counter1 Output CompareA Match Output) |
| PD6 | ICP (Timer/Counter1 Input Capture Pin) |
| PD7 | OC2 (Timer/Counter2 Output Compare Match Output) |

### The Port D Data Register – PORTD

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $12 ($32) | PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 | PORTD |
| 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 D Data Direction Register – DDRD

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $11 ($31) | DDD7 | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 | DDRD |
| 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 D Input Pins Address – PIND

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $10 ($30) | PIND7 | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 | PIND |
| 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 D Input Pins Address – PIND – is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

**Perform a Page Write**

To execute Page Write, set up the address in the Z-pointer, write “00101” to the five LSB in SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to Z13:Z7. During this operation, Z6:Z0 must be zero to ensure that the page is written correctly. It is recommended that the interrupts are disabled during the page write operation.

**Consideration while Updating the Boot Loader Section**

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit 11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock Bit 11 to protect the Boot Loader software from any internal software changes.

**Wait for SPM Instruction to Complete**

Though the CPU is halted during Page Write, Page Erase or Lock bit write, for future compatibility, the user software must poll for SPM complete by reading the SPMCR Register and loop until the SPMEN bit is cleared after a programming operation. See “Assembly code example for a Boot Loader” on page 141 for a code example.

**Instruction Word Read after Page Erase, Page Write, and Lock Bit Write**

To ensure proper instruction pipelining after programming action (Page Erase, Page Write, or Lock bit write), the SPM instruction must be followed with the sequence (.dw $FFFF - NOP) as shown below:

```
spm
.dw $FFFF
nop
```

If not, the instruction following SPM might fail. It is not necessary to add this sequence when the SPM instruction only loads the temporary buffer.

**Avoid Reading the Application Section During Self-Programming**

During Self-Programming (either Page Erase or Page Write), the user software should not read the application section. The user software itself must prevent addressing this section during the Self-Programming operations. This implies that interrupts must be disabled. Before addressing the application section after the programming is completed, for future compatibility, the user software must write “10001” to the five LSB in SPMCR and execute SPM within four clock cycles. Then the user software should verify that the ASB bit is cleared. See “Assembly code example for a Boot Loader” on page 141 for an example. Though the ASB and ASRE bits have no special function in this device, it is important for future code compatibility that they are treated as described above.

**Boot Loader Lock Bits**

ATmega163 has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU
- To only protect the Boot Loader Flash section from a software update by the MCU
- To only protect application Flash section from a software update by the MCU
- Allowing software update in the entire Flash

See Table  and Table  for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can only be cleared by a chip erase command.

```
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.

**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 60 for $t_{WD\_FLASH}$ value.

**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 60 for $t_{WD\_EEPROM}$ value.

**Programming Times for Non-volatile Memory**

The internal RC Oscillator is used to control programming time when programming or erasing Flash, EEPORM, Fuses, and Lock bits. During Parallel or Serial Programming, the device is in reset, and this Oscillator runs at its initial, uncalibrated frequency, which may vary from 0.5 MHz to 1.0 MHz. In software it is possible to calibrate this Oscillator to 1.0 MHz (see “Calibrated Internal RC Oscillator” on page 37). Consequently, programming times will be shorter and more accurate when Programming or erasing non-volatile memory from software, using SPM or the EEPROM interface. See Table 60 for a summary of programming times.

**Table 60.** Maximum Programming Times for Non-volatile Memory

| Operation | Symbol | Number of RC Oscillator Cycles | Parallel/Serial Programming | | Self-Programming[1] |
|---|---|---|---|---|---|
| | | | 2.7V | 5.0V | |
| Chip Erase | $t_{WD\_CE}$ | 16K | 32 ms | 30 ms | 17 ms |
| Flash Write[3] | $t_{WD\_FLASH}$ | 8K | 16 ms | 15 ms | 8.5 ms |
| EEPROM Write[2] | $t_{WD\_EEPROM}$ | 2K | 4 ms | 3.8 ms | 2.2 ms |
| Fuse/lock bit write | $t_{WD\_FUSE}$ | 1K | 2 ms | 1.9 ms | 1.1 ms |

Notes: 1. Includes variation over voltage and temperature after RC Oscillator has been calibrated to 1.0 MHz
2. Parallel EEPROM Programming takes 1K cycles
3. Per page

## DC Characteristics (Continued)

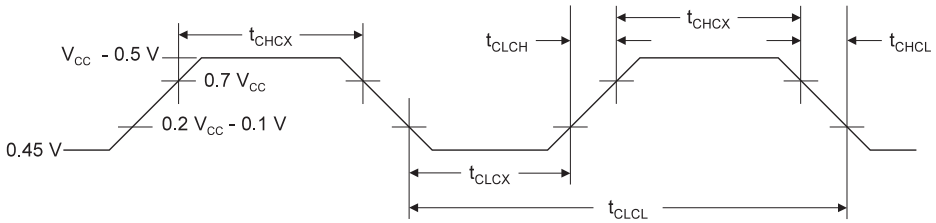
$T_A$ = -40°C to 85°C, $V_{CC}$ = 2.7V to 5.5V (unless otherwise noted)

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $I_{CC}$ | Power Supply Current | Active 4 MHz, $V_{CC}$ = 3V (ATmega163L) | | | 5.0 | mA |
| | | Active 8 MHz, $V_{CC}$ = 5V (ATmega163) | | | 15.0 | mA |
| | | Idle 4 MHz, $V_{CC}$ = 3V (ATmega163L) | | | 2.5 | mA |
| | | Idle 8 MHz, $V_{CC}$ = 5V (ATmega163) | | | 8 | mA |
| | Power-down mode(5) | WDT enabled, $V_{CC}$ = 3V | | 9 | 15.0 | µA |
| | | WDT disabled, $V_{CC}$ = 3V | | <1 | 4.0 | µA |
| $V_{ACIO}$ | Analog Comparator Input Offset Voltage | $V_{CC}$ = 5V<br>$V_{in}$ = $V_{CC}$/2 | | | 40 | mV |
| $I_{ACLK}$ | Analog Comparator Input Leakage Current | $V_{CC}$ = 5V<br>$V_{in}$ = $V_{CC}$/2 | -50 | | 50 | nA |
| $t_{ACID}$ | Analog Comparator Initialization Delay | $V_{CC}$ = 2.7V<br>$V_{CC}$ = 4.0V | | 750<br>500 | | ns |

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all $I_{OL}$, for all ports, should not exceed 200 mA.
   2] The sum of all $I_{OL}$, for ports B0 - B7, D0 - D7 and XTAL2, should not exceed 100 mA.
   3] The sum of all $I_{OL}$, for ports A0 - A7 and C0 - C7 should not exceed 100 mA.
   If IOL exceeds the test condition, $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (3 mA at Vcc = 5V, 1.5 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all $I_{OH}$, for all ports, should not exceed 200 mA.
   2] The sum of all $I_{OH}$, for ports B0 - B7, D0 - D7 and XTAL2, should not exceed 100 mA.
   3] The sum of all $I_{OH}$, for ports A0 - A7 and C0 - C7 should not exceed 100 mA.
   If IOH exceeds the test condition, $V_{OH}$ may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum $V_{CC}$ for Power-down is 2.5V.

## External Clock Drive Waveforms

**Figure 89.** External Clock Drive Waveforms



## Instruction Set Summary (Continued)

| | | | | | |
|---|---|---|---|---|---|
| BRIE | k | Branch if Interrupt Enabled | if ( I = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRID | k | Branch if Interrupt Disabled | if ( I = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| **DATA TRANSFER INSTRUCTIONS** | | | | | |
| MOV | Rd, Rr | Move Between Registers | Rd ← Rr | None | 1 |
| MOVW | Rd, Rr | Copy Register Word | Rd+1:Rd ← Rr+1:Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd ← K | None | 1 |
| LD | Rd, X | Load Indirect | Rd ← (X) | None | 2 |
| LD | Rd, X+ | Load Indirect and Post-Inc. | Rd ← (X), X ← X + 1 | None | 2 |
| LD | Rd, - X | Load Indirect and Pre-Dec. | X ← X - 1, Rd ← (X) | None | 2 |
| LD | Rd, Y | Load Indirect | Rd ← (Y) | None | 2 |
| LD | Rd, Y+ | Load Indirect and Post-Inc. | Rd ← (Y), Y ← Y + 1 | None | 2 |
| LD | Rd, - Y | Load Indirect and Pre-Dec. | Y ← Y - 1, Rd ← (Y) | None | 2 |
| LDD | Rd,Y+q | Load Indirect with Displacement | Rd ← (Y + q) | None | 2 |
| LD | Rd, Z | Load Indirect | Rd ← (Z) | None | 2 |
| LD | Rd, Z+ | Load Indirect and Post-Inc. | Rd ← (Z), Z ← Z+1 | None | 2 |
| LD | Rd, -Z | Load Indirect and Pre-Dec. | Z ← Z - 1, Rd ← (Z) | None | 2 |
| LDD | Rd, Z+q | Load Indirect with Displacement | Rd ← (Z + q) | None | 2 |
| LDS | Rd, k | Load Direct from SRAM | Rd ← (k) | None | 2 |
| ST | X, Rr | Store Indirect | (X) ← Rr | None | 2 |
| ST | X+, Rr | Store Indirect and Post-Inc. | (X) ← Rr, X ← X + 1 | None | 2 |
| ST | - X, Rr | Store Indirect and Pre-Dec. | X ← X - 1, (X) ← Rr | None | 2 |
| ST | Y, Rr | Store Indirect | (Y) ← Rr | None | 2 |
| ST | Y+, Rr | Store Indirect and Post-Inc. | (Y) ← Rr, Y ← Y + 1 | None | 2 |
| ST | - Y, Rr | Store Indirect and Pre-Dec. | Y ← Y - 1, (Y) ← Rr | None | 2 |
| STD | Y+q,Rr | Store Indirect with Displacement | (Y + q) ← Rr | None | 2 |
| ST | Z, Rr | Store Indirect | (Z) ← Rr | None | 2 |
| ST | Z+, Rr | Store Indirect and Post-Inc. | (Z) ← Rr, Z ← Z + 1 | None | 2 |
| ST | -Z, Rr | Store Indirect and Pre-Dec. | Z ← Z - 1, (Z) ← Rr | None | 2 |
| STD | Z+q,Rr | Store Indirect with Displacement | (Z + q) ← Rr | None | 2 |
| STS | k, Rr | Store Direct to SRAM | (k) ← Rr | None | 2 |
| LPM | | Load Program Memory | R0 ← (Z) | None | 3 |
| LPM | Rd, Z | Load Program Memory | Rd ← (Z) | None | 3 |
| LPM | Rd, Z+ | Load Program Memory and Post-Inc | Rd ← (Z), Z ← Z+1 | None | 3 |
| SPM | | Store Program Memory | (Z) ← R1:R0 | None | - |
| IN | Rd, P | In Port | Rd ← P | None | 1 |
| OUT | P, Rr | Out Port | P ← Rr | None | 1 |
| PUSH | Rr | Push Register on Stack | STACK ← Rr | None | 2 |
| POP | Rd | Pop Register from Stack | Rd ← STACK | None | 2 |
| **BIT AND BIT-TEST INSTRUCTIONS** | | | | | |
| SBI | P,b | Set Bit in I/O Register | I/O(P,b) ← 1 | None | 2 |
| CBI | P,b | Clear Bit in I/O Register | I/O(P,b) ← 0 | None | 2 |
| LSL | Rd | Logical Shift Left | Rd(n+1) ← Rd(n), Rd(0) ← 0 | Z,C,N,V | 1 |
| LSR | Rd | Logical Shift Right | Rd(n) ← Rd(n+1), Rd(7) ← 0 | Z,C,N,V | 1 |
| ROL | Rd | Rotate Left Through Carry | Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7) | Z,C,N,V | 1 |
| ROR | Rd | Rotate Right Through Carry | Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0) | Z,C,N,V | 1 |
| ASR | Rd | Arithmetic Shift Right | Rd(n) ← Rd(n+1), n=0..6 | Z,C,N,V | 1 |
| SWAP | Rd | Swap Nibbles | Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0) | None | 1 |
| BSET | s | Flag Set | SREG(s) ← 1 | SREG(s) | 1 |
| BCLR | s | Flag Clear | SREG(s) ← 0 | SREG(s) | 1 |
| BST | Rr, b | Bit Store from Register to T | T ← Rr(b) | T | 1 |
| BLD | Rd, b | Bit load from T to Register | Rd(b) ← T | None | 1 |
| SEC | | Set Carry | C ← 1 | C | 1 |
| CLC | | Clear Carry | C ← 0 | C | 1 |
| SEN | | Set Negative Flag | N ← 1 | N | 1 |
| CLN | | Clear Negative Flag | N ← 0 | N | 1 |
| SEZ | | Set Zero Flag | Z ← 1 | Z | 1 |
| CLZ | | Clear Zero Flag | Z ← 0 | Z | 1 |
| SEI | | Global Interrupt Enable | I ← 1 | I | 1 |
| CLI | | Global Interrupt Disable | I ← 0 | I | 1 |
| SES | | Set Signed Test Flag | S ← 1 | S | 1 |
| CLS | | Clear Signed Test Flag | S ← 0 | S | 1 |
| SEV | | Set Twos Complement Overflow. | V ← 1 | V | 1 |
| CLV | | Clear Twos Complement Overflow | V ← 0 | V | 1 |
| SET | | Set T in SREG | T ← 1 | T | 1 |
| CLT | | Clear T in SREG | T ← 0 | T | 1 |
| SEH | | Set Half Carry Flag in SREG | H ← 1 | H | 1 |