

Welcome to [E-XFL.COM](http://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	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K 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 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-PDIP
Purchase URL	<a href="https://www.e-xfl.com/product-detail/atmel/atmega8l-8pi">https://www.e-xfl.com/product-detail/atmel/atmega8l-8pi</a>

## Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock  $clk_{CPU}$ , directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 5 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 5.** The Parallel Instruction Fetches and Instruction Executions

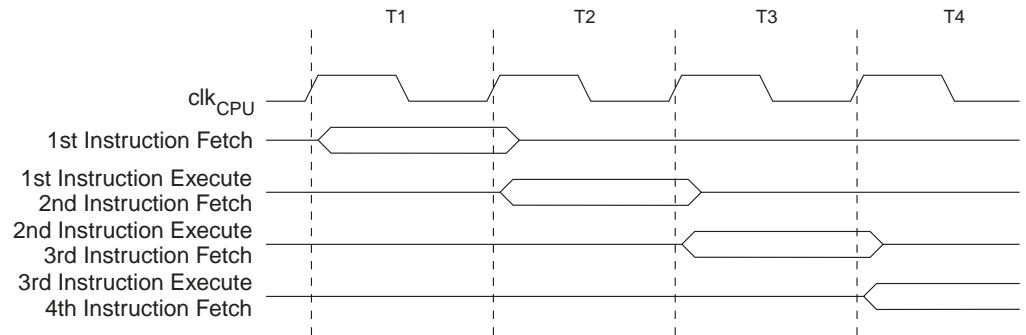
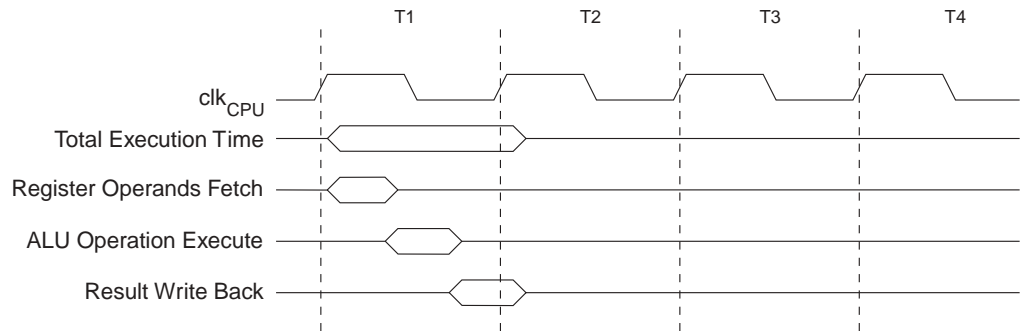


Figure 6 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

**Figure 6.** Single Cycle ALU Operation



## Reset and Interrupt Handling

The AVR provides several 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 written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock Bits BLB02 or BLB12 are programmed. This feature improves software security. See the section “Memory Programming” on page 219 for details.

The lowest addresses in the Program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of Vectors is shown in “Interrupts” on page 44. 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. The Interrupt Vectors can be moved to the start

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in the following example.

Assembly Code Example
<pre>sei ; set global interrupt enable sleep; enter sleep, waiting for interrupt ; note: will enter sleep before any pending ; interrupt(s)</pre>
C Code Example
<pre>_SEI(); /* set global interrupt enable */ _SLEEP(); /* enter sleep, waiting for interrupt */ /* note: will enter sleep before any pending interrupt(s) */</pre>

### Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles, the Program Vector address for the actual interrupt handling routine is executed. During this 4-clock cycle period, the Program Counter is pushed onto the Stack. The Vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (2 bytes) is popped back from the Stack, the Stack Pointer is incremented by 2, and the I-bit in SREG is set.

### The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

### The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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 – EEDR7..0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

### The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	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	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8 and will always read as zero.

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

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

- **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, setting EEWE within four clock cycles will write data to the EEPROM at the selected address. If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to 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**

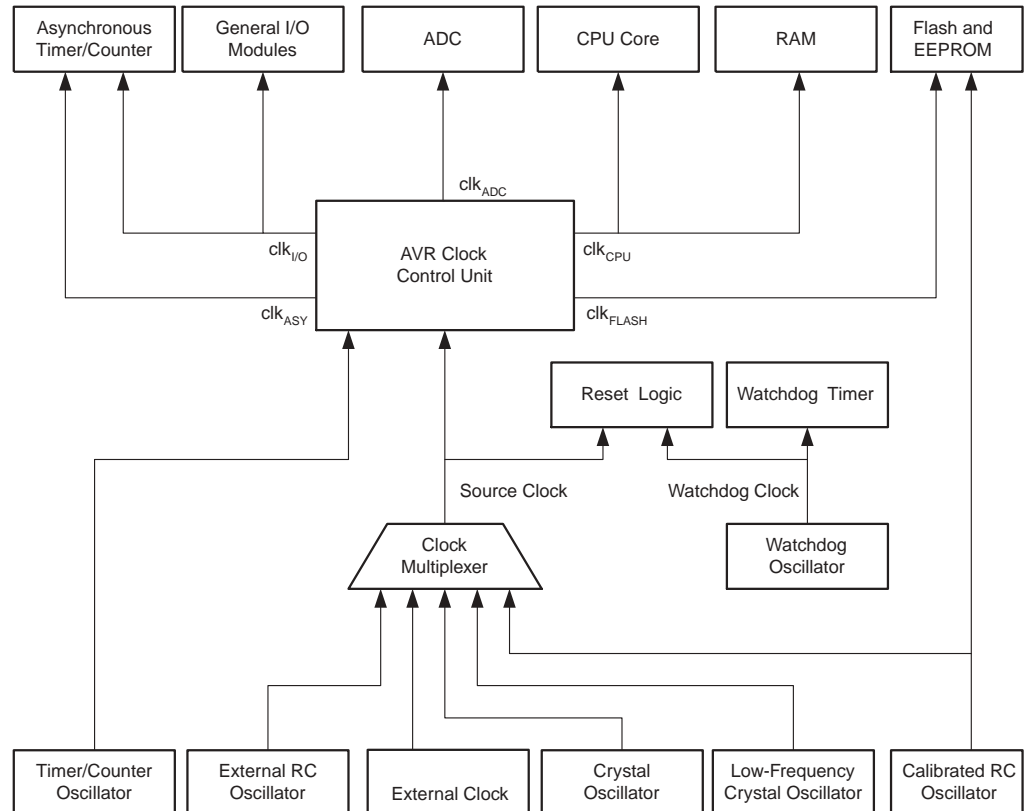
The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the

## System Clock and Clock Options

### Clock Systems and their Distribution

Figure 10 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in “Power Management and Sleep Modes” on page 31. The clock systems are detailed Figure 10.

**Figure 10.** Clock Distribution



#### CPU Clock – $clk_{CPU}$

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the Data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

#### I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when  $clk_{I/O}$  is halted, enabling TWI address reception in all sleep modes.

#### Flash Clock – $clk_{FLASH}$

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

**Table 5.** Start-up Times for the Crystal Oscillator Clock Selection

CKSELO	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset ( $V_{CC} = 5.0V$ )	Recommended Usage
0	00	258 CK <sup>(1)</sup>	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK <sup>(1)</sup>	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK <sup>(2)</sup>	–	Ceramic resonator, BOD enabled
0	11	1K CK <sup>(2)</sup>	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK <sup>(2)</sup>	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

- Notes:
1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
  2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

## Low-frequency Crystal Oscillator

To use a 32.768 kHz watch crystal as the clock source for the device, the Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to “1001”. The crystal should be connected as shown in Figure 11. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The internal capacitors have a nominal value of 36 pF.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 6.

**Table 6.** Start-up Times for the Low-frequency Crystal Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset ( $V_{CC} = 5.0V$ )	Recommended Usage
00	1K CK <sup>(1)</sup>	4.1 ms	Fast rising power or BOD enabled
01	1K CK <sup>(1)</sup>	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11	Reserved		

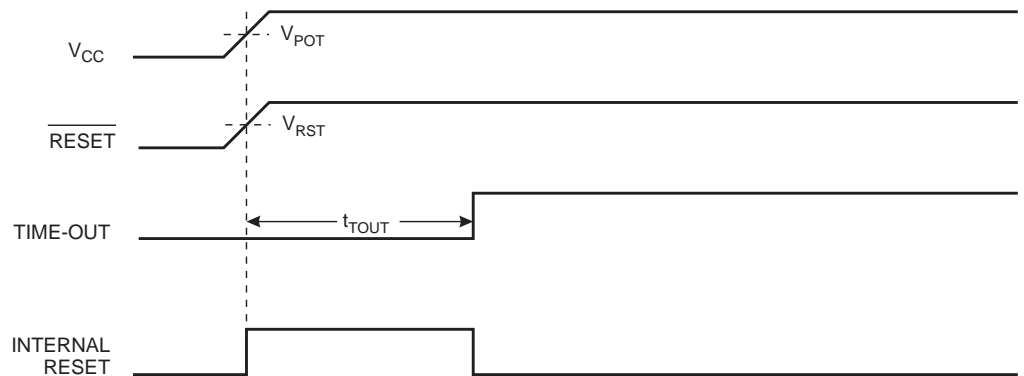
- Note:
1. These options should only be used if frequency stability at start-up is not important for the application.

## Power-on Reset

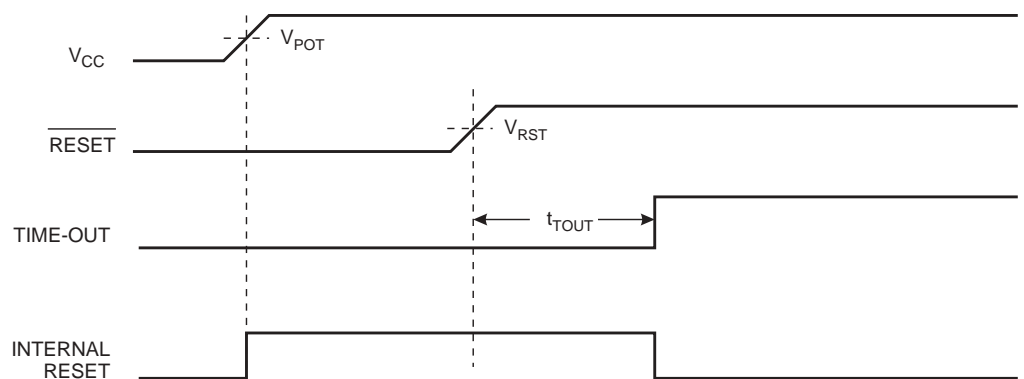
A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever  $V_{CC}$  is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after  $V_{CC}$  rise. The RESET signal is activated again, without any delay, when  $V_{CC}$  decreases below the detection level.

**Figure 15.** MCU Start-up,  $\overline{\text{RESET}}$  Tied to  $V_{CC}$



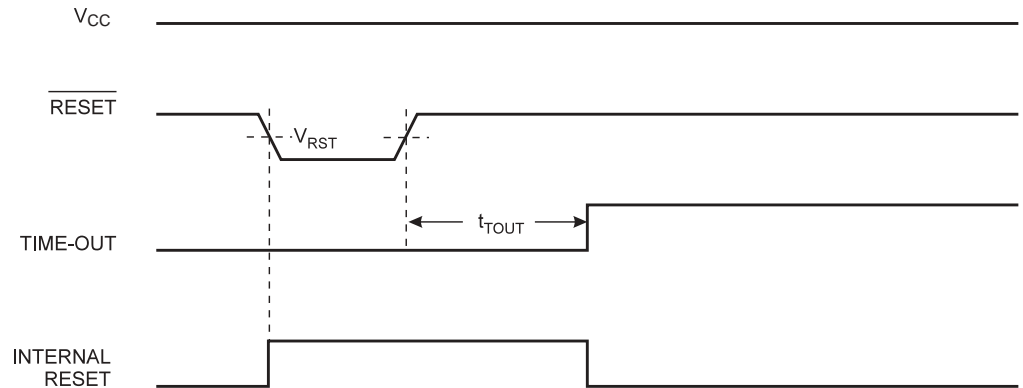
**Figure 16.** MCU Start-up,  $\overline{\text{RESET}}$  Extended Externally



## External Reset

An External Reset is generated by a low level on the  $\overline{\text{RESET}}$  pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage –  $V_{\text{RST}}$  on its positive edge, the delay counter starts the MCU after the time-out period  $t_{\text{TOUT}}$  has expired.

**Figure 17.** External Reset During Operation



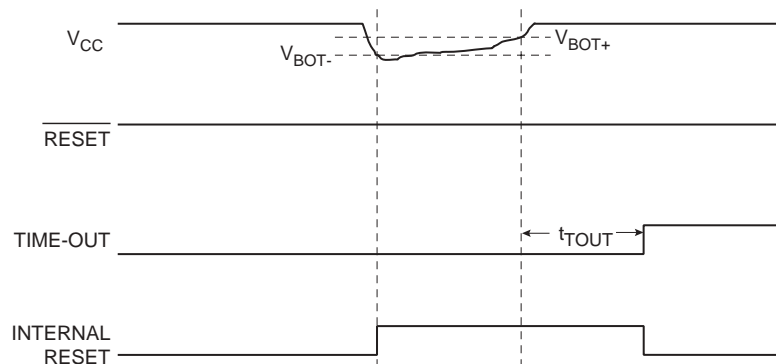
## Brown-out Detection

ATmega8 has an On-chip Brown-out Detection (BOD) circuit for monitoring the  $V_{\text{CC}}$  level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as  $V_{\text{BOT+}} = V_{\text{BOT}} + V_{\text{HYST}}/2$  and  $V_{\text{BOT-}} = V_{\text{BOT}} - V_{\text{HYST}}/2$ .

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and  $V_{\text{CC}}$  decreases to a value below the trigger level ( $V_{\text{BOT-}}$  in Figure 18), the Brown-out Reset is immediately activated. When  $V_{\text{CC}}$  increases above the trigger level ( $V_{\text{BOT+}}$  in Figure 18), the delay counter starts the MCU after the time-out period  $t_{\text{TOUT}}$  has expired.

The BOD circuit will only detect a drop in  $V_{\text{CC}}$  if the voltage stays below the trigger level for longer than  $t_{\text{BOD}}$  given in Table 15.

**Figure 18.** Brown-out Reset During Operation





- **Bit 0 – IVCE: Interrupt Vector Change Enable**

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example
-----------------------

<pre> Move_interrupts:     ; Enable change of Interrupt Vectors     ldi r16, (1&lt;&lt;IVCE)     out GICR, r16     ; Move interrupts to boot Flash section     ldi r16, (1&lt;&lt;IVSEL)     out GICR, r16     ret         </pre>
---

C Code Example
----------------

<pre> void Move_interrupts(void) {     /* Enable change of Interrupt Vectors */     GICR = (1&lt;&lt;IVCE);     /* Move interrupts to boot Flash section */     GICR = (1&lt;&lt;IVSEL); }         </pre>
---

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 32. The value on the INT0 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 32.** Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

## General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	<b>INT1</b>	<b>INT0</b>	–	–	–	–	<b>IVSEL</b>	<b>IVCE</b>	<b>GICR</b>
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

(ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 Flag is not required (if an interrupt handler is used).

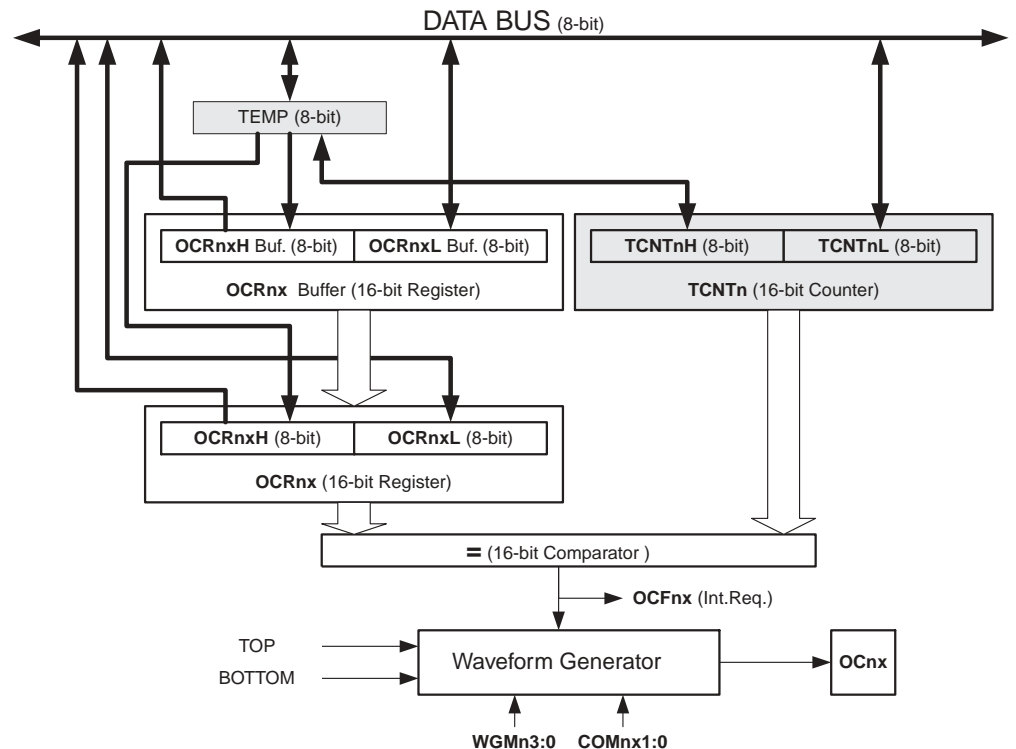
## Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the *Output Compare Flag* (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF1x Flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGM13:0) bits and *Compare Output mode* (COM1x1:0) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (See “Modes of Operation” on page 86.)

A special feature of Output Compare unit A allows it to define the Timer/Counter TOP value (i.e. counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the waveform generator.

Figure 35 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

**Figure 35.** Output Compare Unit, Block Diagram

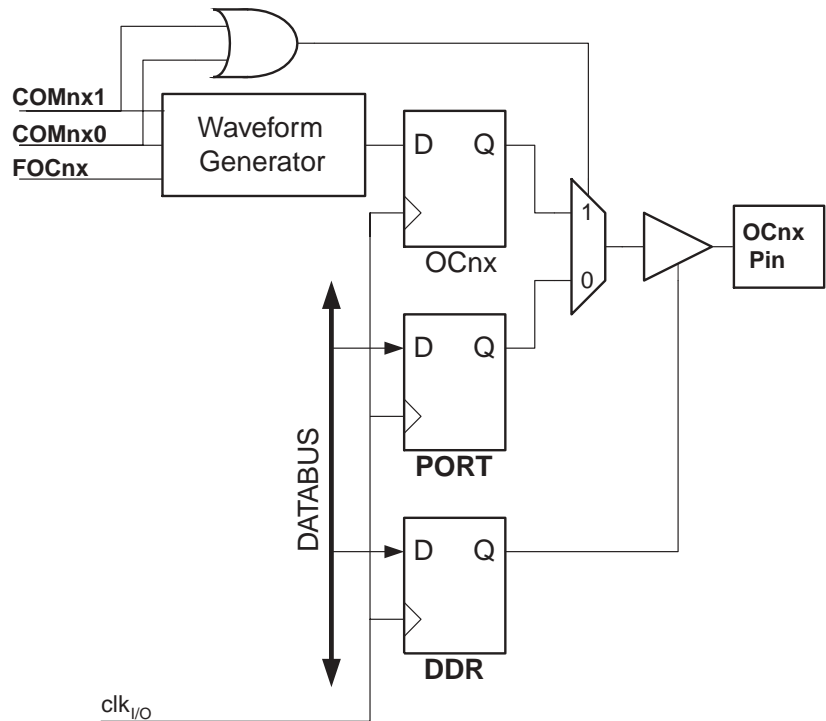


The OCR1x Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the

## Compare Match Output Unit

The *Compare Output mode* (COM1x1:0) bits have two functions. The waveform generator uses the COM1x1:0 bits for defining the Output Compare (OC1x) state at the next Compare Match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 36 shows a simplified schematic of the logic affected by the COM1x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a System Reset occur, the OC1x Register is reset to “0”.

**Figure 36.** Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC1x) from the waveform generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the *Data Direction Register* (DDR) for the port pin. The Data Direction Register bit for the OC1x pin (DDR\_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 36, Table 37 and Table 38 for details.

The design of the Output Compare Pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See “16-bit Timer/Counter Register Description” on page 95.

The COM1x1:0 bits have no effect on the Input Capture unit.

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

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

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

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

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

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

If a write is performed to any of the three Timer/Counter2 Registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2, and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

## Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- **Warning:** When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2, and TCCR2 might be corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2, and TCCR2.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
  5. Clear the Timer/Counter2 Interrupt Flags.
  6. Enable interrupts, if needed.
- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOSC1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the

## Accessing UBRRH/UCSRC Registers

The UBRRH Register shares the same I/O location as the UCSRC Register. Therefore some special consideration must be taken when accessing this I/O location.

### Write Access

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

Assembly Code Examples <sup>(1)</sup>
<pre> ... ; Set UBRRH to 2 ldi r16,0x02 out UBRRH,r16 ... ; Set the USBS and the UCSZ1 bit to one, and ; the remaining bits to zero. ldi r16,(1&lt;&lt;URSEL) (1&lt;&lt;USBS) (1&lt;&lt;UCSZ1) out UCSRC,r16 ... </pre>
C Code Examples <sup>(1)</sup>
<pre> ... /* Set UBRRH to 2 */ UBRRH = 0x02; ... /* Set the USBS and the UCSZ1 bit to one, and */ /* the remaining bits to zero. */ UCSRC = (1&lt;&lt;URSEL) (1&lt;&lt;USBS) (1&lt;&lt;UCSZ1); ... </pre>

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

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 148.

## USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	<b>RXCIE</b>	<b>TXCIE</b>	<b>UDRIE</b>	<b>RXEN</b>	<b>TXEN</b>	<b>UCSZ2</b>	<b>RXB8</b>	<b>TXB8</b>	<b>UCSRB</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR and PE Flags.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed (i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted). When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

This bit is used for Synchronous mode only. Write 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).

**Table 59.** UCPOL Bit Settings

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

## USART Baud Rate Registers – UBRRL and UBRRHs

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	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 149 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 written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.



	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	R/W
Initial Value	1	1	1	1	1	1	1	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. In multimaster systems, TWAR must be set in masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bits 7..1 – TWA: TWI (Slave) Address Register**

These seven bits constitute the slave address of the TWI unit.

- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the Two-wire Serial Bus.

## Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 77 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

**Table 86.** Lock Bit Protection Modes<sup>(2)</sup> (Continued)

Memory Lock Bits			Protection Type
1	1	1	No restrictions for SPM or 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 Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Notes: 1. Program the Fuse Bits before programming the Lock Bits.  
 2. "1" means unprogrammed, "0" means programmed

## Fuse Bits

The ATmega8 has two fuse bytes. Table 87 and Table 88 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

**Table 87.** Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL <sup>(4)</sup>	7	Select if PC6 is I/O pin or RESET pin	1 (unprogrammed, PC6 is RESET-pin)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN <sup>(1)</sup>	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT <sup>(2)</sup>	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 82 for details)	0 (programmed) <sup>(3)</sup>
BOOTSZ0	1	Select Boot Size (see Table 82 for details)	0 (programmed) <sup>(3)</sup>
BOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in Serial Programming mode.  
 2. The CKOPT Fuse functionality depends on the setting of the CKSEL bits, see "Clock Sources" on page 24 for details.  
 3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 82 on page 217.  
 4. When programming the RSTDISBL Fuse Parallel Programming has to be used to change fuses or perform further programming.

**Table 98.** Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	<u>Enable</u> Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program Memory	0010 H000	0000 aaaa	bbbb bbbb	oooo oooo	Read H (high or low) data o from Program memory at word address a:b.
Load Program Memory Page	0100 H000	0000 xxxx	xxx <b>b</b> bbbb	iiii iiii	Write H (high or low) data i to Program memory page at word address b. Data Low byte must be loaded before Data High byte is applied within the same address.
Write Program Memory Page	0100 1100	0000 aaaa	bbb <b>x</b> xxxx	xxxx xxxx	Write Program memory Page at address a:b.
Read EEPROM Memory	1010 0000	00xx xxxa	bbbb bbbb	oooo oooo	Read data o from EEPROM memory at address a:b.
Write EEPROM Memory	1100 0000	00xx xxxa	bbbb bbbb	iiii iiii	Write data i to EEPROM memory at address a:b.
Read Lock Bits	0101 1000	0000 0000	xxxx xxxx	xx <b>00</b> oooo	Read Lock Bits. "0" = programmed, "1" = unprogrammed. See Table 85 on page 219 for details.
Write Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11 <b>ii</b> iiii	Write Lock Bits. Set bits = "0" to program Lock Bits. See Table 85 on page 219 for details.
Read Signature Byte	0011 0000	00xx xxxx	xxxx xx <b>bb</b>	oooo oooo	Read Signature Byte o at address b.
Write Fuse Bits	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Set bits = "0" to program, "1" to unprogram. See Table 88 on page 221 for details.
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Set bits = "0" to program, "1" to unprogram. See Table 87 on page 220 for details.
Read Fuse Bits	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Read Fuse Bits. "0" = programmed, "1" = unprogrammed. See Table 88 on page 221 for details.
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Read Fuse high bits. "0" = programmed, "1" = unprogrammed. See Table 87 on page 220 for details.
Read Calibration Byte	0011 1000	00xx xxxx	0000 00 <b>bb</b>	oooo oooo	Read Calibration Byte

Note: **a** = address high bits  
**b** = address low bits  
**H** = 0 – Low byte, 1 – High byte  
**o** = data out  
**i** = data in  
**x** = don't care

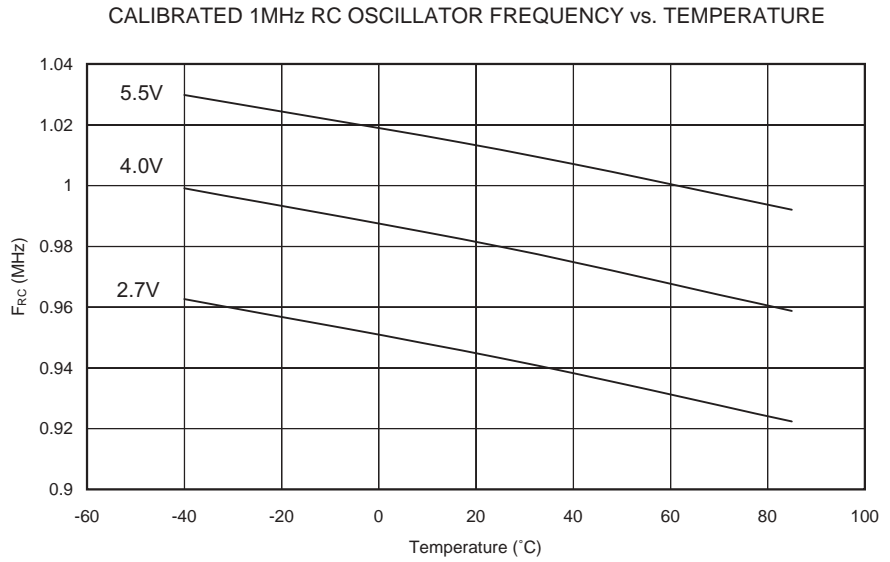


$T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ ,  $V_{CC} = 2.7\text{V}$  to  $5.5\text{V}$  (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units	
$I_{CC}$	Power Supply Current	Active 4 MHz, $V_{CC} = 3\text{V}$ (ATmega8L)			5	mA	
		Active 8 MHz, $V_{CC} = 5\text{V}$ (ATmega8)			15	mA	
		Idle 4 MHz, $V_{CC} = 3\text{V}$ (ATmega8L)			2	mA	
		Idle 8 MHz, $V_{CC} = 5\text{V}$ (ATmega8)			7	mA	
	Power-down mode <sup>(5)</sup>	WDT enabled, $V_{CC} = 3\text{V}$				25	$\mu\text{A}$
		WDT disabled, $V_{CC} = 3\text{V}$				2	$\mu\text{A}$
$V_{ACIO}$	Analog Comparator Input Offset Voltage	$V_{CC} = 5\text{V}$ $V_{in} = V_{CC}/2$			20	mV	
$I_{ACLK}$	Analog Comparator Input Leakage Current	$V_{CC} = 5\text{V}$ $V_{in} = V_{CC}/2$	-50		50	nA	
$t_{ACID}$	Analog Comparator Propagation Delay	$V_{CC} = 2.7\text{V}$ $V_{CC} = 4.0\text{V}$		750 500		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 (20mA at  $V_{CC} = 5\text{V}$ , 10mA at  $V_{CC} = 3\text{V}$ ) under steady state conditions (non-transient), the following must be observed:  
 PDIP Package:  
 1) The sum of all IOL, for all ports, should not exceed 400 mA.  
 2) The sum of all IOL, for ports C0 - C5 should not exceed 200 mA.  
 3) The sum of all IOL, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 100 mA.  
 TQFP and MLF Package:  
 1) The sum of all IOL, for all ports, should not exceed 400 mA.  
 2) The sum of all IOL, for ports C0 - C5, should not exceed 200 mA.  
 3) The sum of all IOL, for ports C6, D0 - D4, should not exceed 300 mA.  
 4) The sum of all IOL, for ports B0 - B7, D5 - D7, should not exceed 300 mA.  
 If IOL exceeds the test condition, VOL 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 (20mA at  $V_{CC} = 5\text{V}$ , 10mA at  $V_{CC} = 3\text{V}$ ) under steady state conditions (non-transient), the following must be observed:  
 PDIP Package:  
 1) The sum of all IOH, for all ports, should not exceed 400 mA.  
 2) The sum of all IOH, for port C0 - C5, should not exceed 100 mA.  
 3) The sum of all IOH, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 100 mA.  
 TQFP and MLF Package:  
 1) The sum of all IOH, for all ports, should not exceed 400 mA.  
 2) The sum of all IOH, for ports C0 - C5, should not exceed 200 mA.  
 3) The sum of all IOH, for ports C6, D0 - D4, should not exceed 300 mA.  
 4) The sum of all IOH, for ports B0 - B7, D5 - D7, should not exceed 300 mA.  
 If IOH exceeds the test condition, VOH 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.

**Figure 179.** Calibrated 1 MHz RC Oscillator Frequency vs. Temperature



**Figure 180.** Calibrated 1 MHz RC Oscillator Frequency vs.  $V_{CC}$

