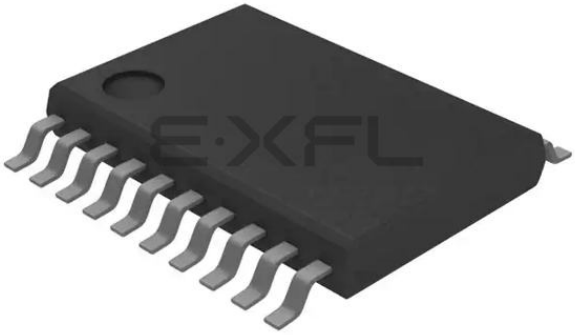


Welcome to E-XFL.COM

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	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny861-15xd

The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in     r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi    EECR, EEMPE   ; start EEPROM write
sbi    EECR, EEPE
out    SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;       /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

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

Assembly Code Example

```
sei    ; set Global Interrupt Enable
sleep  ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set Global Interrupt Enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

5.7.1 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 four 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 (two bytes) is popped back from the stack, the stack pointer is incremented by two, and the I-bit in SREG is set.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM program enable signal EEPE is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two 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 written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

6.5.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	**MSB**							**LSB**	**GPIOR2**
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	

6.5.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	**MSB**							**LSB**	**GPIOR1**
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	

6.5.6 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	**MSB**							**LSB**	**GPIOR0**
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	

7.1.3 Flash Clock – clk_{FLASH}

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

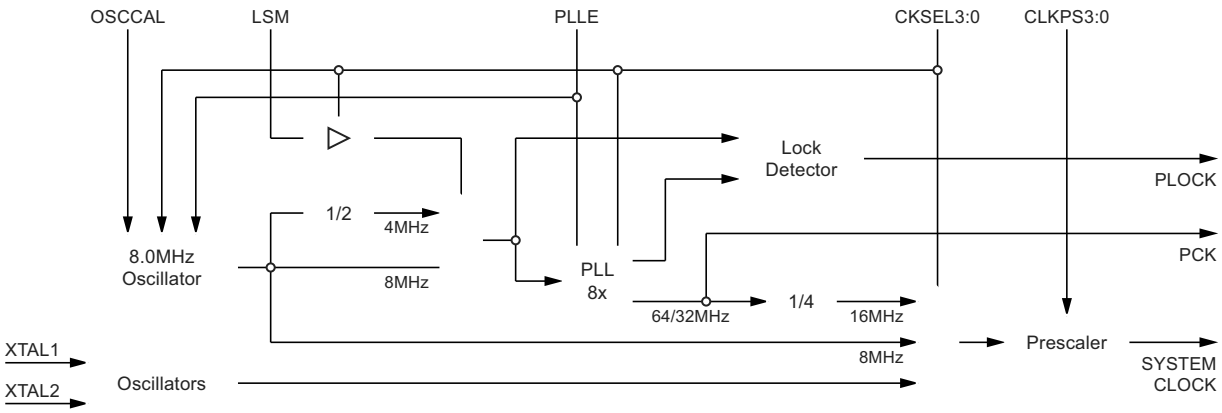
7.1.4 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

7.1.5 Internal PLL for Fast Peripheral Clock Generation - clk_{PCK}

The internal PLL in Atmel® ATtiny261/461/861 generates a clock frequency that is 8x multiplied from a source input. By default, the PLL uses the output of the internal 8.0MHz RC oscillator as source. Alternatively, if the LSM bit of the PLLCSR is set the PLL will use the output of the RC oscillator divided by two. Thus the output of the PLL, the fast peripheral clock is 64MHz. The fast peripheral clock, or a clock prescaled from that, can be selected as the clock source for Timer/Counter1or as a system clock. See Figure 7-2. The frequency of the fast peripheral clock is divided by two when LSM of PLLCSR is set, resulting in a clock frequency of 32MHz. Note, that LSM can not be set if PLL_{CLK} is used as a system clock.

Figure 7-2. PCK Clocking System



The PLL is locked on the RC oscillator and adjusting the RC oscillator via OSCCAL register will adjust the fast peripheral clock at the same time. However, even if the RC oscillator is taken to a higher frequency than 8MHz, the fast peripheral clock frequency saturates at 85MHz (worst case) and remains oscillating at the maximum frequency. It should be noted that the PLL in this case is not locked any longer with the RC oscillator clock. Therefore, it is recommended not to take the OSCCAL adjustments to a higher frequency than 8MHz in order to keep the PLL in the correct operating range.

The internal PLL is enabled when:

- The PLLE bit of the PLLCSR register is set.
- The CKSEL fuse are programmed to ‘0001’.

The PLLCSR bit PLOCK is set when PLL is locked.

Both internal RC oscillator and PLL are switched off in power down and stand-by sleep modes.

- **Bit 5 – PCIF: Pin Change Interrupt Flag**

When a logic change on any PCINT15 pin triggers an interrupt request, PCIF becomes set (one). If the I-bit in SREG and the PCIE bit in GIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bits 4:0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny261/461/861 and will always read as zero.

11.1.4 PCMSK0 – Pin Change Mask Register A

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**PCINT7**	**PCINT6**	**PCINT5**	**PCINT4**	**PCINT3**	**PCINT2**	**PCINT1**	**PCINT0**	**PCMSK0**
Read/Write	R/W	R/W	R/W	R/w	R/W	R/W	R/W	R/W	
Initial Value	1	1	0	0	1	0	0	0	

- **Bits 7:0 – PCINT7:0: Pin Change Enable Mask 7..0**

Each PCINT7:0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.1.5 PCMSK1 – Pin Change Mask Register B

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	**PCINT15**	**PCINT14**	**PCINT13**	**PCINT12**	**PCINT11**	**PCINT10**	**PCINT9**	**PCINT8**	**PCMSK1**
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	1	

- **Bits 7:0 – PCINT15:8: Pin Change Enable Mask 15..8**

Each PCINT15:8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT11:8 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin, and if PCINT15:12 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT15:8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

Table 12-4 and Table 12-5 relate the alternate functions of Port B to the overriding signals shown in Figure 12-5 on page 57.

Table 12-4. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/RESET/dW/ ADC10/PCINT15	PB6/ADC9/T0/INT0/ PCINT14	PB5/XTAL2/CLKO/ OC1D/ADC8/PCINT13(1)	PB4/XTAL1/OC1D/ADC7/PCINT12(1)
PUOE	$\overline{\text{RSTDISBL}}$(1) × DWEN(1)	0	$\overline{\text{INTRC}}$ × $\overline{\text{EXTCLK}}$	INTRC
PUOV	1	0	0	0
DDOE	RSTDISBL(1) × DWEN(1)	0	$\overline{\text{INTRC}}$ × $\overline{\text{EXTCLK}}$	INTRC
DDOV	debugWire transmit	0	0	0
PVOE	0	0	OC1D Enable	OC1D enable
PVOV	0	0	OC1D	OC1D
PTOE	0	0	0	0
DIEOE	0	$\overline{\text{RSTDISBL}}$ + (PCINT5 × PCIE + ADC9D)	INTRC × EXTCLK + PCINT4 × PCIE + ADC8D	$\overline{\text{INTRC}}$ + PCINT12 × PCIE + ADC7D
DIEOV	ADC10D	ADC9D	(INTRC × EXTCLK) + ADC8D	INTRC × ADC7D
DI	PCINT15	T0/INT0/PCINT14	PCINT13	PCINT12
AIO	RESET / ADC10	ADC9	XTAL2, ADC8	XTAL1, ADC7

Note: 1. 1 when the fuse is “0” (programmed).

Table 12-5. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC1B/ PCINT11	PB2/SCK/USCK/SCL/ OC1B/PCINT10	PB1/MISO/DO/OC1A/ PCINT9	PB0/MOSI/DI/SDA/ OC1A/PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	USI_TWO_WIRE × $\overline{\text{USIPOS}}$	0	USI_TWO_WIRE × $\overline{\text{USIPOS}}$
DDOV	0	(USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2 × $\overline{\text{USIPOS}}$	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB0}}$) × DDB0 × $\overline{\text{USIPOS}}$
PVOE	OC1B enable	$\overline{\text{OC1B Enable}}$ + $\overline{\text{USIPOS}}$ × USI_TWO_WIRE × DDB2	OC1A Enable + $\overline{\text{USIPOS}}$ × USI_THREE_WIRE	$\overline{\text{OC1A Enable}}$ + (USI_TWO_WIRE × DDB0 × $\overline{\text{USIPOS}}$)
PVOV	OC1B	OC1B	OC1A + (DO × $\overline{\text{USIPOS}}$)	OC1A
PTOE	0	USITC × $\overline{\text{USIPOS}}$	0	0
DIEOE	PCINT11 × PCIE	PCINT10 × PCIE + USISIE × $\overline{\text{USIPOS}}$	PCINT9 × PCIE	PCINT8 × PCIE + (USISIE × $\overline{\text{USIPOS}}$)
DIEOV	0	0	0	0
DI	PCINT11	USCK/SCL/PCINT10	PCINT9	DI/SDA/PCINT8
AIO				

Note: INTRC means that one of the internal RC oscillators are selected (by the CKSEL fuses), EXTCK means that external clock is selected (by the CKSEL fuses).

13. Timer/Counter0 Prescaler

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$. See Table 13-1 on page 68 for details.

13.1 Prescaler Reset

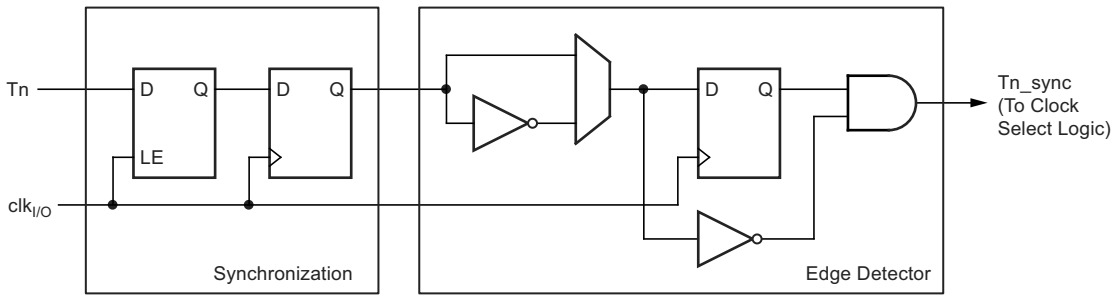
The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024). It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution.

13.2 External Clock Source

An external clock source applied to the T0 pin can be used as Timer/Counter clock (clk_{T0}). The T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 13-1 shows a functional equivalent block diagram of the T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects. See Table 13-1 on page 68 for details.

Figure 13-1. T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem).

However, due to variation of the system clock frequency and duty cycle caused by oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

14.2.1 Registers

The Timer/Counter0 low byte register (TCNT0L) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in Figure 14-1) signals are all visible in the timer interrupt flag register (TIFR). All interrupts are individually masked with the timer interrupt mask register (TIMSK). TIFR and TIMSK are not shown in the figure.

In 16-bit mode the Timer/Counter consists one more 8-bit register, the Timer/Counter0 high byte register (TCNT0H). Furthermore, there is only one output compare unit in 16-bit mode as the two output compare registers, OCR0A and OCR0B, are combined to one 16-bit output compare register.

OCR0A contains the low byte of the word and OCR0B contains the high byte of the word. When accessing 16-bit registers, special procedures described in Section 14.9 “Accessing Registers in 16-bit Mode” on page 76 must be followed.

14.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0L for accessing Timer/Counter0 counter value and so on.

The definitions in Table 14-1 are also used extensively throughout the document.

Table 14-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255) in 8-bit mode or 0xFFFF (decimal 65535) in 16-bit mode.
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF/0xFFFF (MAX) or the value stored in the OCR0A register.

14.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The clock select logic is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter control register 0 B (TCCR0B), and controls which clock source and edge the Timer/Counter uses to increment its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T0}). For details on clock sources and prescaler, see Section 13. “Timer/Counter0 Prescaler” on page 66.

The following code examples show how to do an atomic write of the TCNT0H/L register contents. Writing any of the OCR0A/B registers can be done by using the same principle.

```
Assembly Code Example
TIM0_WriteTCNT0:
        ; Save global interrupt flag
        in     r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT0 to r17:r16
        out    TCNT0H,r17
        out    TCNT0L,r16
        ; Restore global interrupt flag
        out    SREG,r18
        ret
```

```
C Code Example
void TIM0_WriteTCNT0(unsigned int i)
{
unsigned char sreg;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Set TCNT0 to i */
TCNT0H = (i >> 8);
TCNT0L = (unsigned char)i;
/* Restore global interrupt flag */
SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT0H/L.

14.9.1 Reusing the temporary high byte register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

16.2.3 Registers

The Timer/Counter (TCNT1) and output compare registers (OCR1A, OCR1B, OCR1C and OCR1D) are 8-bit registers that are used as a data source to be compared with the TCNT1 contents. The OCR1A, OCR1B and OCR1D registers determine the action on the OC1A, OC1B and OC1D pins and they can also generate the compare match interrupts. The OCR1C holds the Timer/Counter TOP value, i.e. the clear on compare match value. The Timer/Counter1 high byte register (TC1H) is a 2-bit register that is used as a common temporary buffer to access the MSB bits of the Timer/Counter1 registers, if the 10-bit accuracy is used.

Interrupt request (overflow TOV1, and compare matches OCF1A, OCF1B, OCF1D and fault protection FPF1) signals are visible in the timer interrupt flag register (TIFR) and Timer/Counter1 control register D (TCCR1D). The interrupts are individually masked with the timer interrupt mask register (TIMSK) and the FPIE1 bit in the Timer/Counter1 control register D (TCCR1D).

Control signals are found in the Timer/Counter control registers TCCR1A, TCCR1B, TCCR1C, TCCR1D and TCCR1E.

16.2.4 Synchronization

In asynchronous clocking mode the Timer/Counter1 and the prescaler allow running the CPU from any clock source while the prescaler is operating on the fast peripheral clock (PCK) having frequency of 64MHz (or 32MHz in low speed mode). This is possible because there is a synchronization boundary between the CPU clock domain and the fast peripheral clock domain. Figure 16-2 shows Timer/Counter 1 synchronization register block diagram and describes synchronization delays in between registers. Note that all clock gating details are not shown in the figure.

The Timer/Counter1 register values go through the internal synchronization registers, which cause the input synchronization delay, before affecting the counter operation. The registers TCCR1A, TCCR1B, TCCR1C, TCCR1D, OCR1A, OCR1B, OCR1C and OCR1D can be read back right after writing the register. The read back values are delayed for the Timer/Counter1 (TCNT1) register, Timer/Counter1 High Byte Register (TC1H) and flags (OCF1A, OCF1B, OCF1D and TOV1), because of the input and output synchronization.

The system clock frequency must be lower than half of the PCK frequency, because the synchronization mechanism of the asynchronous Timer/Counter1 needs at least two edges of the PCK when the system clock is high. If the frequency of the system clock is too high, it is a risk that data or control values are lost.

16.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM1x1:0 bits differently in normal mode and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the waveform generator that no action on the OCW1x output is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 16-6 on page 108. For fast PWM mode, refer to Table 16-7 on page 108, and for the phase and frequency correct PWM refer to Table 16-8 on page 109. A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

16.7 Modes of Operation

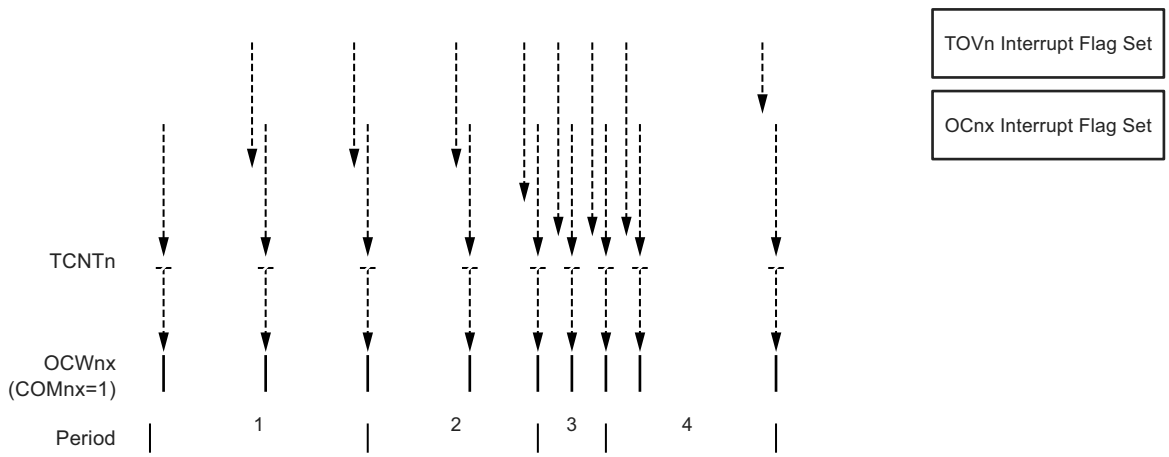
The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (bits PWM1x and WGM10) and compare output mode (COM1x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted, non-inverted or complementary. For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared, or toggled at a compare match.

16.7.1 Normal Mode

The simplest mode of operation is the Normal mode (PWM1x = 0), the counter counts from BOTTOM to TOP (defined as OCR1C) then restarts from BOTTOM. The OCR1C defines the TOP value for the counter, hence also its resolution, and allows control of the compare match output frequency. In toggle compare output mode the waveform output (OCW1x) is toggled at compare match between TCNT1 and OCR1x. In non-inverting compare output mode the waveform output is cleared on the compare match. In inverting compare output mode the waveform output is set on compare match.

The timing diagram for the normal mode is shown in Figure 16-10. The counter value (TCNT1) that is shown as a histogram in the timing diagram is incremented until the counter value matches the TOP value. The counter is then cleared at the following clock cycle The diagram includes the waveform output (OCW1x) in toggle compare mode. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1.

Figure 16-10. Normal Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set in the same clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 11th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt, that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time. For generating a waveform, the OCW1x output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1x1:0 = 1). The OC1x value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC1x} = f_{clkT1}/4$ when OCR1C is set to zero. The waveform frequency is defined by the following equation:

$$f_{OC1x} = \frac{f_{\mathrm{clkT1}}}{2 \cdot (1 + OCR1C)}$$

The following code examples show how to do an atomic write of the TCNT1 register contents. Writing any of the OCR1A/B/C/D registers can be done by using the same principle.

```
Assembly Code Example
TIM1_WriteTCNT1:
        ; Save global interrupt flag
        in     r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT1 to r17:r16
        out    TC1H,r17
        out    TCNT1,r16
        ; Restore global interrupt flag
        out    SREG,r18
        ret

C Code Example
void TIM1_WriteTCNT1(unsigned int i)
{
unsigned char sreg;
unsigned int i;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Set TCNT1 to i */
TC1H = (i >> 8);
TCNT1 = (unsigned char)i;
/* Restore global interrupt flag */
SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

16.10.1 Reusing the temporary high byte register

If writing to more than one 10-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

16.11 Register Description

16.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM1A	PWM1B	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7,6 - COM1A1, COM1A0: Comparator A Output Mode, Bits 1 and 0**

These bits control the behavior of the waveform output (OCW1A) and the connection of the output compare pin (OC1A). If one or both of the COM1A1:0 bits are set, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{OC1B}$ output is connected only in PWM modes when the COM1A1:0 bits are set to “01”. Note that the data direction register (DDR) bit corresponding to the OC1A and $\overline{OC1A}$ pins must be set in order to enable the output driver.

The function of the COM1A1:0 bits depends on the PWM1A, WGM10 and WGM11 bit settings. Table 16-6 shows the COM1A1:0 bit functionality when the PWM1A bit is set to normal mode (non-PWM).

Table 16-6. Compare Output Mode, Normal Mode (non-PWM)

COM1A1..0	OCW1A Behavior	OC1A Pin	$\overline{OC1A}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Toggle on compare match	Connected	Disconnected
10	Clear on compare match	Connected	Disconnected
11	Set on compare match	Connected	Disconnected

Table 16-7 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to fast PWM mode.

Table 16-7. Compare Output Mode, Fast PWM Mode

COM1A1..0	OCW1A Behavior	OC1A	OC1A
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	Connected	Connected
10	Cleared on compare match Set when TCNT1 = 0x000	Connected	Disconnected
11	Set on compare match Cleared when TCNT1 = 0x000	Connected	Disconnected

- **Bit 2 - FOC1B: Force Output Compare Match 1B**

The FOC1B bit is only active when the PWM1B bit specify a non-PWM mode.

Writing a logical one to this bit forces a change in the waveform output (OCW1B) and the output compare pin (OC1B) according to the values already set in COM1B1 and COM1B0. If COM1B1 and COM1B0 written in the same cycle as FOC1B, the new settings will be used. The force output compare bit can be used to change the output pin value regardless of the timer value. The automatic action programmed in COM1B1 and COM1B0 takes place as if a compare match had occurred, but no interrupt is generated.

The FOC1B bit is always read as zero.

- **Bit 1 - PWM1A: Pulse Width Modulator A Enable**

When set (one) this bit enables PWM mode based on comparator OCR1A

- **Bit 0 - PWM1B: Pulse Width Modulator B Enable**

When set (one) this bit enables PWM mode based on comparator OCR1B.

16.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	**PWM1X**	**PSR1**	**DTPS11**	**DTPS10**	**CS13**	**CS12**	**CS11**	**CS10**	**TCCR1B**
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 - PWM1X: PWM Inversion Mode**

When this bit is set (one), the PWM Inversion Mode is selected and the dead time generator outputs, OC1x and $\overline{OC1x}$ are inverted.

- **Bit 6 - PSR1: Prescaler Reset Timer/Counter1**

When this bit is set (one), the Timer/Counter1 prescaler (TCNT1 is unaffected) will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always read as zero.

- **Bits 5,4 - DTPS11, DTPS10: Dead Time Prescaler Bits**

The Timer/Counter1 control register B is a 8-bit read/write register.

The dedicated dead time prescaler in front of the dead time generator can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8 providing a large range of dead times that can be generated. The dead time prescaler is controlled by two bits DTPS11 and DTPS10 from the dead time Prescaler register. These bits define the division factor of the dead time prescaler. The division factors are given in Table 16-14.

Table 16-14. Division factors of the Dead Time prescaler

DTPS11	DTPS10	Prescaler divides the T/C1 clock by
0	0	1x (no division)
0	1	2x
1	0	4x
1	1	8x

16.11.8 OCR1A – Timer/Counter1 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	MSB							LSB	OCR1A
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 output compare register A is an 8-bit read/write register.

The Timer/Counter output compare register A contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1A. A compare match does only occur if Timer/Counter1 counts to the OCR1A value. A software write that sets TCNT1 and OCR1A to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1A after a synchronization delay following the compare event.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.9 OCR1B – Timer/Counter1 Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	MSB							LSB	OCR1B
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 output compare register B is an 8-bit read/write register.

The Timer/Counter output compare register B contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1. A compare match does only occur if Timer/Counter1 counts to the OCR1B value. A software write that sets TCNT1 and OCR1B to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1B after a synchronization delay following the compare event.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.10 OCR1C – Timer/Counter1 Output Compare Register C

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	MSB							LSB	OCR1C
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	1	

The output compare register C is an 8-bit read/write register.

The Timer/Counter output compare register C contains data to be continuously compared with Timer/Counter1, and a compare match will clear TCNT1. This register has the same function in normal mode and PWM modes.

Note that, if a smaller value than three is written to the output compare register C, the value is automatically replaced by three as it is a minimum value allowed to be written to this register.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

17.5.2 USIBR – USI Buffer Register

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	MSB							LSB	USIBR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The content of the serial register is loaded to the USI buffer register when the transfer is completed, and instead of accessing the USI data register (the serial register) the USI data buffer can be accessed when the CPU reads the received data. This gives the CPU time to handle other program tasks too as the controlling of the USI is not so timing critical. The USI flags as set same as when reading the USIDR register.

17.5.3 USISR – USI Status Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The status register contains interrupt flags, line status flags and the counter value.

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When two-wire mode is selected, the USISIF flag is set (to one) when a start condition is detected. When output disable mode or three-wire mode is selected and (USICSx = 0b11 and USICLK = 0) or (USICS = 0b10 and USICLK = 0), any edge on the SCK pin sets the flag.

An interrupt will be generated when the flag is set while the USISIE bit in USICR and the global interrupt enable flag are set. The flag will only be cleared by writing a logical one to the USISIF bit. Clearing this bit will release the start detection hold of USCL in two-wire mode.

A start condition interrupt will wakeup the processor from all sleep modes.

- **Bit 6 – USIOIF: Counter Overflow Interrupt Flag**

This flag is set (one) when the 4-bit counter overflows (i.e., at the transition from 15 to 0). An interrupt will be generated when the flag is set while the USIOIE bit in USICR and the global interrupt enable flag are set. The flag will only be cleared if a one is written to the USIOIF bit. Clearing this bit will release the counter overflow hold of SCL in two-wire mode.

A counter overflow interrupt will wake up the processor from Idle sleep mode.

- **Bit 5 – USIPF: Stop Condition Flag**

When Two-wire mode is selected, the USIPF Flag is set (one) when a stop condition is detected. The flag is cleared by writing a one to this bit. Note that this is not an interrupt flag. This signal is useful when implementing two-wire bus master arbitration.

- **Bit 4 – USIDC: Data Output Collision**

This bit is logical one when bit 7 in the USI data register differs from the physical pin value. The flag is only valid when two-wire mode is used. This signal is useful when implementing two-wire bus master arbitration.

- **Bits 3:0 – USICNT3..0: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can directly be read or written by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter0 compare match, or by software using USICLK or USITC strobe bits. The clock source depends of the setting of the USICS1..0 bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) are can still be used by the counter.

- **Bit 3:2 – USICS1:0: Clock Source Select**

These bits set the clock source for the USI data register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or Timer/Counter0 compare match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1:0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the USI data register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 17-2 on page 128 shows the relationship between the USICS1..0 and USICLK setting and clock source used for the USI data register and the 4-bit counter.

Table 17-2. Relations between the USICS1..0 and USICLK Setting

USICS1	USICS0	USICLK	USI Data Register Clock Source	4-bit Counter Clock Source
0	0	0	No Clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 compare match	Timer/Counter0 compare match
1	0	0	External, positive edge	External, both edges
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a one to this bit location strobes the USI data register to shift one step and the counter to increment by one, provided that the USICS1..0 bits are set to zero and by doing so the software clock strobe option is selected. The output will change immediately when the clock strobe is executed, i.e., in the same instruction cycle. The value shifted into the USI data register is sampled the previous instruction cycle. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a clock select register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 17-2).

- **Bit 0 – USITC: Toggle Clock Port Pin**

Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the data direction register, but if the PORT value is to be shown on the pin the DDB2 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

17.5.5 USIPP – USI Pin Position

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	-	-	-	-	-	-	-	**USIPOS**	**USIPP**
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and always reads as zero.

- **Bit 0 – USIPOS: USI Pin Position**

Setting this bit to one changes the USI pin position. As default pins PB2..PB0 are used for the USI pin functions, but when writing this bit to one the USIPOS bit is set the USI pin functions are on pins PA2..PA0.

20. debugWIRE On-chip Debug System

20.1 Features

- Complete program flow control
- Emulates all on-chip functions, both digital and analog, except RESET pin
- Real-time operation
- Symbolic debugging support (both at C and assembler source level, or for other HLLs)
- Unlimited number of program break points (using software break points)
- Non-intrusive operation
- Electrical characteristics identical to real device
- Automatic configuration system
- High-speed operation
- Programming of non-volatile memories

20.2 Overview

The debugWIRE on-chip debug system uses a one-wire, bi-directional interface to control the program flow, execute AVR® instructions in the CPU and to program the different non-volatile memories.

20.3 Physical Interface

When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 20-1. The debugWIRE Setup

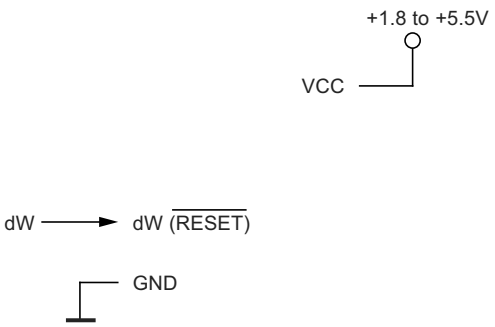


Figure 20-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistor on the dW/(RESET) line must be in the range of 10k to 20 kΩ. However, the pull-up resistor is optional.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors inserted on the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

Table 22-12. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip erase
0100 0000	Write fuse bits
0010 0000	Write lock bits
0001 0000	Write flash
0001 0001	Write EEPROM
0000 1000	Read signature bytes and calibration byte
0000 0100	Read fuse and lock bits
0000 0010	Read flash
0000 0011	Read EEPROM

22.8 Parallel Programming

22.8.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set $\overline{\text{RESET}}$ to “0” and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 22-10 on page 160 to “0000” and wait at least 100ns.
4. Apply 11.5 - 12.5V to $\overline{\text{RESET}}$. Any activity on Prog_enable pins within 100ns after +12V has been applied to $\overline{\text{RESET}}$, will cause the device to fail entering programming mode.
5. Wait at least 50µs before sending a new command.

22.8.2 Considerations for Efficient Programming

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.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE fuse is programmed) and flash after a chip erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in flash or 256 byte EEPROM. This consideration also applies to signature bytes reading.

22.8.3 Chip Erase

The chip erase will erase the flash and EEPROM[1] memories plus lock bits. The lock bits are not reset until the program memory has been completely erased. The fuse bits are not changed. A chip erase must be performed before the flash and/or EEPROM are reprogrammed.

Note: 1. The EEPRPOM memory is preserved during chip erase if the EESAVE fuse is programmed.

Load command “Chip Erase”

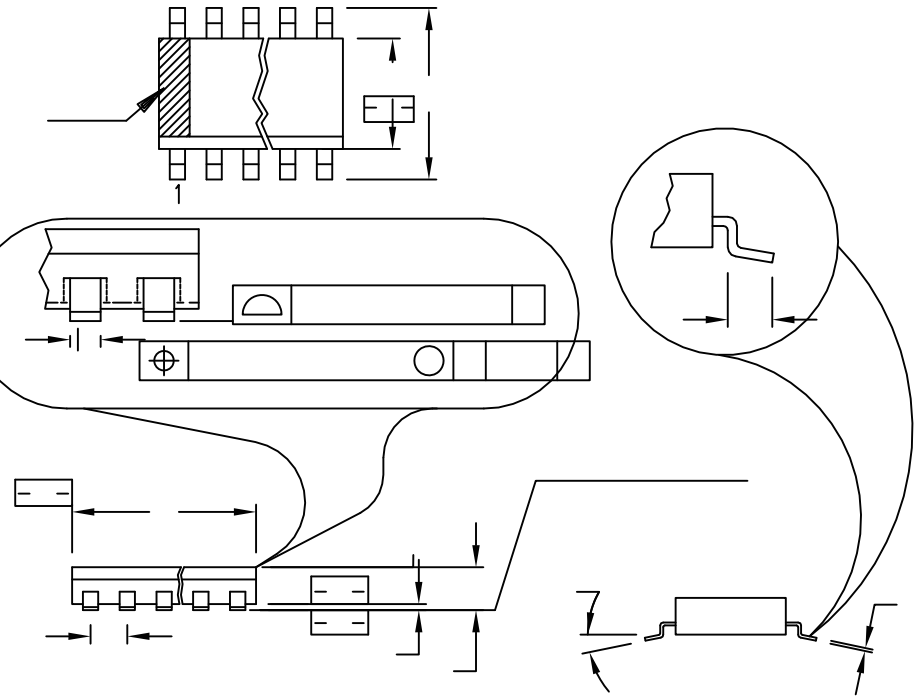
1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for chip erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{\text{WR}}$ a negative pulse. This starts the chip erase. RDY/$\overline{\text{BSY}}$ goes low.
6. Wait until RDY/$\overline{\text{BSY}}$ goes high before loading a new command.

25. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x15 (0x35)	TCCR0A	TCW0	ICEN0	ICNC0	ICES0	ACIC0			CTC0	80
0x14 (0x34)	TCNT0H	Timer/Counter0 counter register high byte								81
0x13 (0x33)	OCR0A	Timer/Counter0 output compare register A								81
0x12 (0x32)	OCR0B	Timer/Counter0 output compare register B								81
0x11 (0x31)	USIPP								USIPOS	128
0x10 (0x30)	USIBR	USI buffer register								126
0x0F (0x2F)	USIDR	USI data register								125
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	126
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	127
0x0C (0x2C)	GPIOR2	General purpose I/O register 2								23
0x0B (0x2B)	GPIOR1	General purpose I/O register 1								23
0x0A (0x2A)	GPIOR0	General purpose I/O register 0								23
0x09 (0x29)	ACSRB	HSEL	HLEV				ACM2	ACM1	ACM0	131
0x08 (0x28)	ACSRA	ACD	ACBG	ACO	ACI	ACIE	ACME	ACIS1	ACIS0	129
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	144
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	146
0x05 (0x25)	ADCH	ADC data register high byte								147
0x04 (0x24)	ADCL	ADC data register low byte								147
0x03 (0x23)	ADCSRB	BIN	GSEL		REFS2	MUX5	ADTS2	ADTS1	ADTS0	148
0x02 (0x22)	DIDR1	ADC10D	ADC9D	ADC8D	ADC7D					149
0x01 (0x21)	DIDR0	ADC6D	ADC5D	ADC4D	ADC3D	AREFD	ADC2D	ADC1D	ADC0D	149
0x00 (0x20)	TCCR1E	–	-	OC1OE5	OC1OE4	OC1OE3	OC1OE2	OC1OE1	OC1OE0	115

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operation the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

Figure 28-3. 6G



	----		----	

20/12/07

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	6G, 20 Leads - 4.4x6.5mm Body - 0.65mm Pitch - Lead length: 0.6mm THIN SHRINK SMALL OUTLINE		6G	A