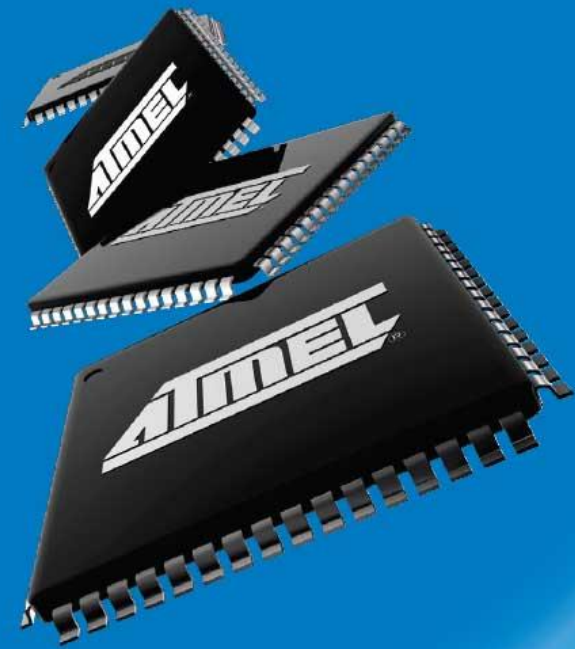


# AVR<sup>®</sup>

8-bit Microcontrollers

# AVR32<sup>®</sup>

32-bit Microcontrollers and Application Processors



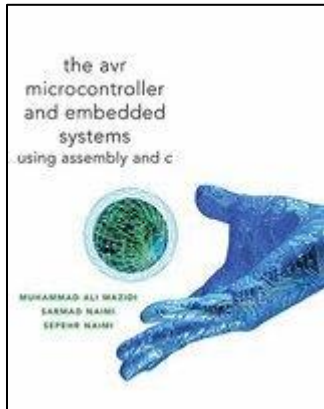
➔ *Interrupts and 16-bit Timer/Counter 1 (Normal Mode)*  
February 2009



Everywhere You Are<sup>®</sup>

## *Atmel ATmega328P Timing Subsystems*

### Reading



[The AVR Microcontroller and Embedded Systems using Assembly and C\)](#)  
by Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi

Chapter 9: Programming Timers 0, 1, and 2

9.1 Programming Timers 0, 1, and 2

9.2 Counter Programming

9.3 Programming Timers in C

## CONTENTS

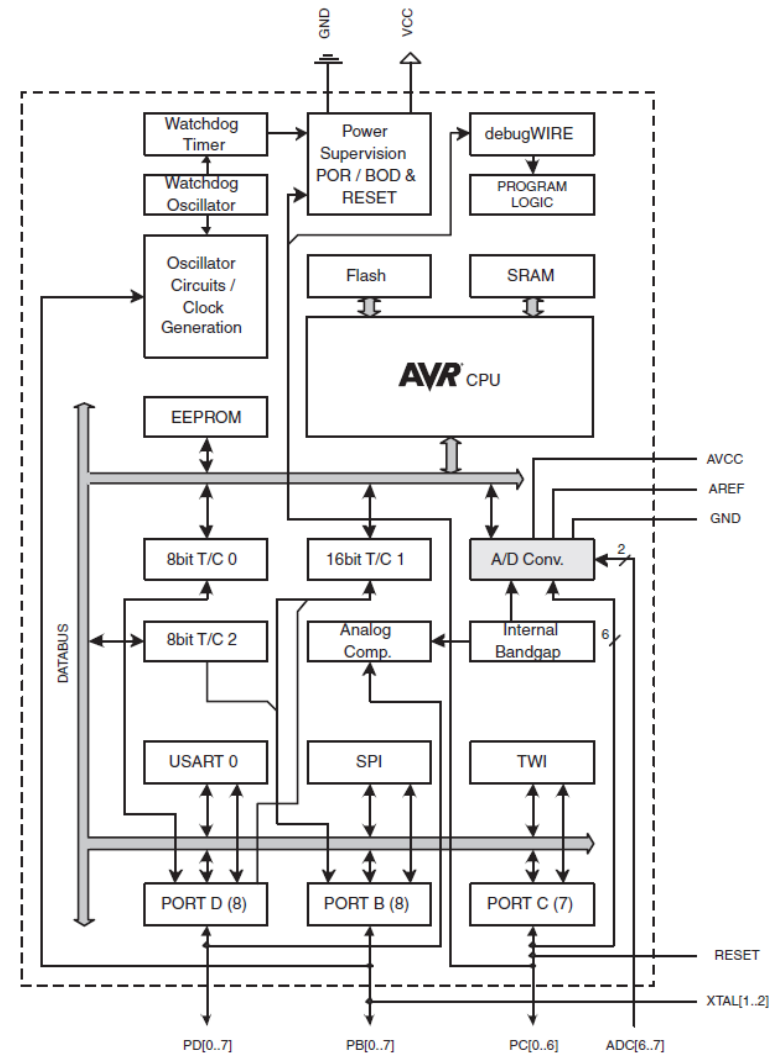
ATmega328P Timing Subsystem.....	4
What is a Flip-Flop and a Counter .....	5
Timing Terminology .....	6
Timer 1 Modes of Operation .....	6
Normal Mode .....	8
Timer/Counter 1 Prescaler .....	9
Timer/Counter 1 Normal Mode – Design Example .....	11
How to Calculate Timer Load Value (Whiteboard Illustration).....	12
Steps to Calculate Timer Load Value (Normal Mode) .....	13
Steps to Calculate Clock Divisor (Normal Mode).....	14
Polling Example – Assembly Version .....	15
Polling Example – C Version .....	16
More Looping Examples .....	17

## ATMEGA328P TIMING SUBSYSTEM<sup>1</sup>

The ATmega328P is equipped with two 8-bit timer/counters and one 16-bit counter. These Timer/Counters let you...

1. Turn on or turn off an external device at a programmed time.
2. Generate a precision output signal (period, duty cycle, frequency). For example, generate a complex digital waveform with varying pulse width to control the speed of a DC motor
3. Measure the characteristics (period, duty cycle, frequency) of an incoming digital signal
4. Count external events

Figure 2-1. Block Diagram

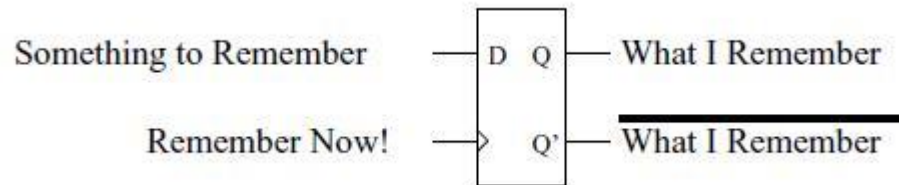


<sup>1</sup> Source: ATmega328P Data Sheet [http://www.atmel.com/dyn/resources/prod\\_documents/8161S.pdf](http://www.atmel.com/dyn/resources/prod_documents/8161S.pdf) page 5

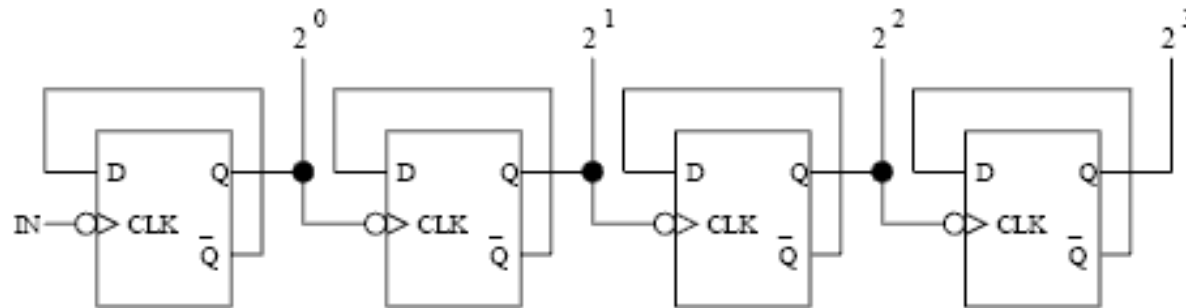
## WHAT IS A FLIP-FLOP AND A COUNTER

- You can think of a D flip-flop as a one-bit memory. The *something to remember* on the D input of flip-flop is remembered on the positive edge of the clock input<sup>2</sup>.

$D_t$	$Q_{t+1}$
0	0
1	1
X	$Q_t$



- The counter part of an ATmega328P Timer/Counter peripheral subsystem is an example of an asynchronous (ripple) counter, which is a collection of flip-flops with the clock input of stage n connected to the output of stage n - 1



- When compared with a synchronous counter, an asynchronous “ripple” counter: generates less noise and is less expensive. On the negative side, an asynchronous “ripple” counter is slower than a synchronous counter.

<sup>2</sup> Source: <http://sandbox.mc.edu/~bennet/cs314/slides/ch5me-4.pdf>

## TIMING TERMINOLOGY

### Frequency

The number of times a particular event repeats within a 1-s period. The unit of frequency is Hertz, or cycles per second. For example, a sinusoidal signal with a 60-Hz frequency means that a full cycle of a sinusoid signal repeats itself 60 times each second, or every 16.67 ms. For the digital waveform shown, the frequency is 2 Hz.

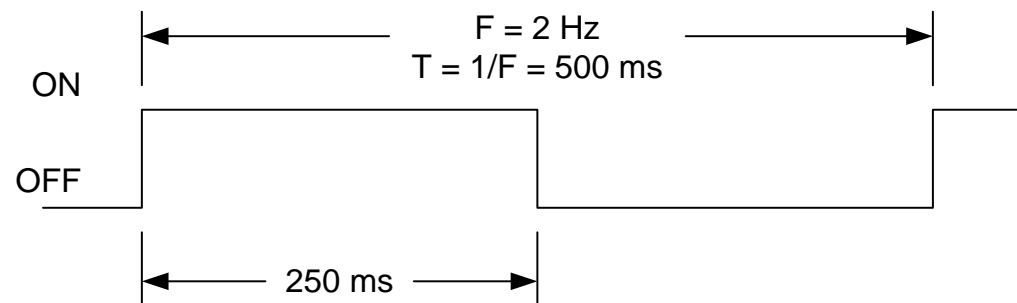
### Period

The flip side of a frequency is a period. If an event occurs with a rate of 2 Hz, the period of that event is 500 ms. To find a period, given a frequency, or vice versa, we simply need to remember their inverse relationship,  $F = 1/T$  where F and T represent a frequency and the corresponding period, respectively.

### Duty Cycle

In many applications, periodic pulses are used as control signals. A good example is the use of a periodic pulse to control a servo motor. To control the direction and sometimes the speed of a motor, a periodic pulse signal with a changing duty cycle over time is used.

Duty cycle is defined as the percentage of one period a signal is ON. The periodic pulse signal shown in the Figure is ON for 50% of the signal period and off for the rest of the period. Therefore, we call the signal in a periodic pulse signal with a 50% duty cycle. This special case is also called a square wave.



## TIMER 1 MODES OF OPERATION

**Table 15-4. Waveform Generation Mode Bit Description<sup>(1)</sup>**

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

## NORMAL MODE<sup>3</sup>

- The simplest AVR Timer mode of operation is the *Normal mode*. Waveform Generation Mode for Timer/Counter 1 (WGM1) bits 3:0 = 0. These bits are located in Timer/Counter Control Registers A/B (**TCCR1A** and **TCCR1B**).

Bit	7	6	5	4	3	2	1	0	
(0x80)	<b>COM1A1</b>	<b>COM1A0</b>	<b>COM1B1</b>	<b>COM1B0</b>	–	–	<b>WGM11</b>	<b>WGM10</b>	<b>TCCR1A</b>
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x81)	<b>ICNC1</b>	<b>ICES1</b>	–	<b>WGM13</b>	<b>WGM12</b>	<b>CS12</b>	<b>CS11</b>	<b>CS10</b>	<b>TCCR1B</b>
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- In this mode the Timer/Counter 1 Register (TCNT1H:TCNT1L) counts up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value 0xFFFF and then restarts 0x0000.
- There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

Bit	7	6	5	4	3	2	1	0	
(0x85)	<b>TCNT1[15:8]</b>								<b>TCNT1H</b>
(0x84)	<b>TCNT1[7:0]</b>								<b>TCNT1L</b>
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	

- In normal operation the Timer/Counter Overflow Flag (TOV1) bit located in the Timer/Counter1 Interrupt Flag Register (TIFR1) will be set in the same timer clock cycle as the Timer/Counter 1 Register (TCNT1H:TCNT1L) becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared.

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	<b>ICF1</b>	–	–	<b>OCF1B</b>	<b>OCF1A</b>	<b>TOV1</b>	<b>TIFR1</b>
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

<sup>3</sup> ATmega328P\_doc8161.pdf Section 15.9 Modes of Operation



## TIMER/COUNTER 1 PRESCALAR

- The clock input to Timer/Counter 1 (TCNT1) can be pre-scaled (divided down) by 5 preset values (1, 8, 64, 256, and 1024).

**Table 13-5.** Clock Select Bit Description

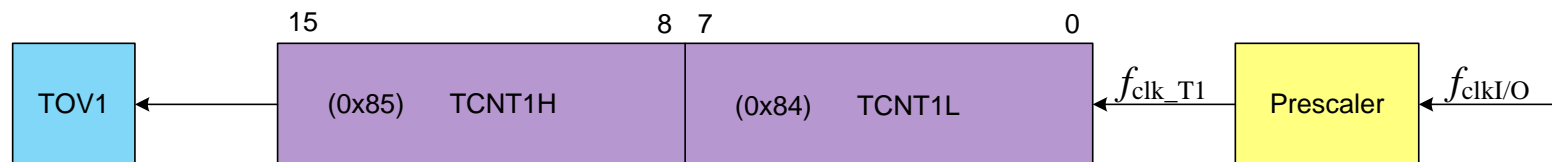
CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk <sub>IO</sub> /1 (No prescaling)
0	1	0	clk <sub>IO</sub> /8 (From prescaler)
0	1	1	clk <sub>IO</sub> /64 (From prescaler)
1	0	0	clk <sub>IO</sub> /256 (From prescaler)
1	0	1	clk <sub>IO</sub> /1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

- Clock Select Counter/Timer 1 (CS1) bits 2:0 are located in Timer/Counter Control Registers B.

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### Normal Mode (WGM 1 bits 3:0 = 0000<sub>2</sub>)

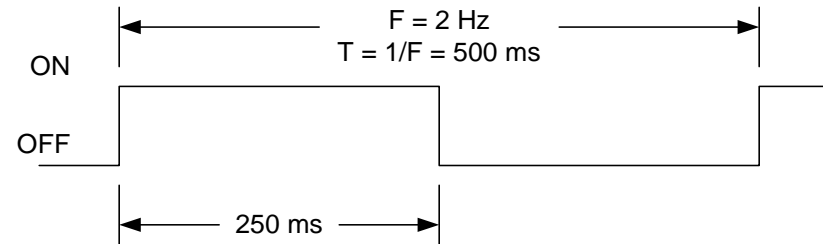
Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



**Table 13-5.** Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}/1$ (No prescaling)
0	1	0	$clk_{I/O}/8$ (From prescaler)
0	1	1	$clk_{I/O}/64$ (From prescaler)
1	0	0	$clk_{I/O}/256$ (From prescaler)
1	0	1	$clk_{I/O}/1024$ (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

## TIMER/COUNTER 1 NORMAL MODE – DESIGN EXAMPLE



- In this design example, we want to write a 250 msec delay routine assuming a system clock frequency of 16.000 MHz and a prescale divisor of 64.
- The first step is to discover if our 16-bit Timer/Counter 1 can generate a 250 ms delay.

### Variable Definitions

$t_{\text{clk\_T1}}$  : period of clock input to Timer/Counter1

$f_{\text{clk}}$  : AVR system clock frequency

$f_{\text{Tclk\_I/O}}$  : AVR Timer clock input frequency to Timer/Counter Waveform Generator

### How to Calculate Maximum Delay (Normal Mode)

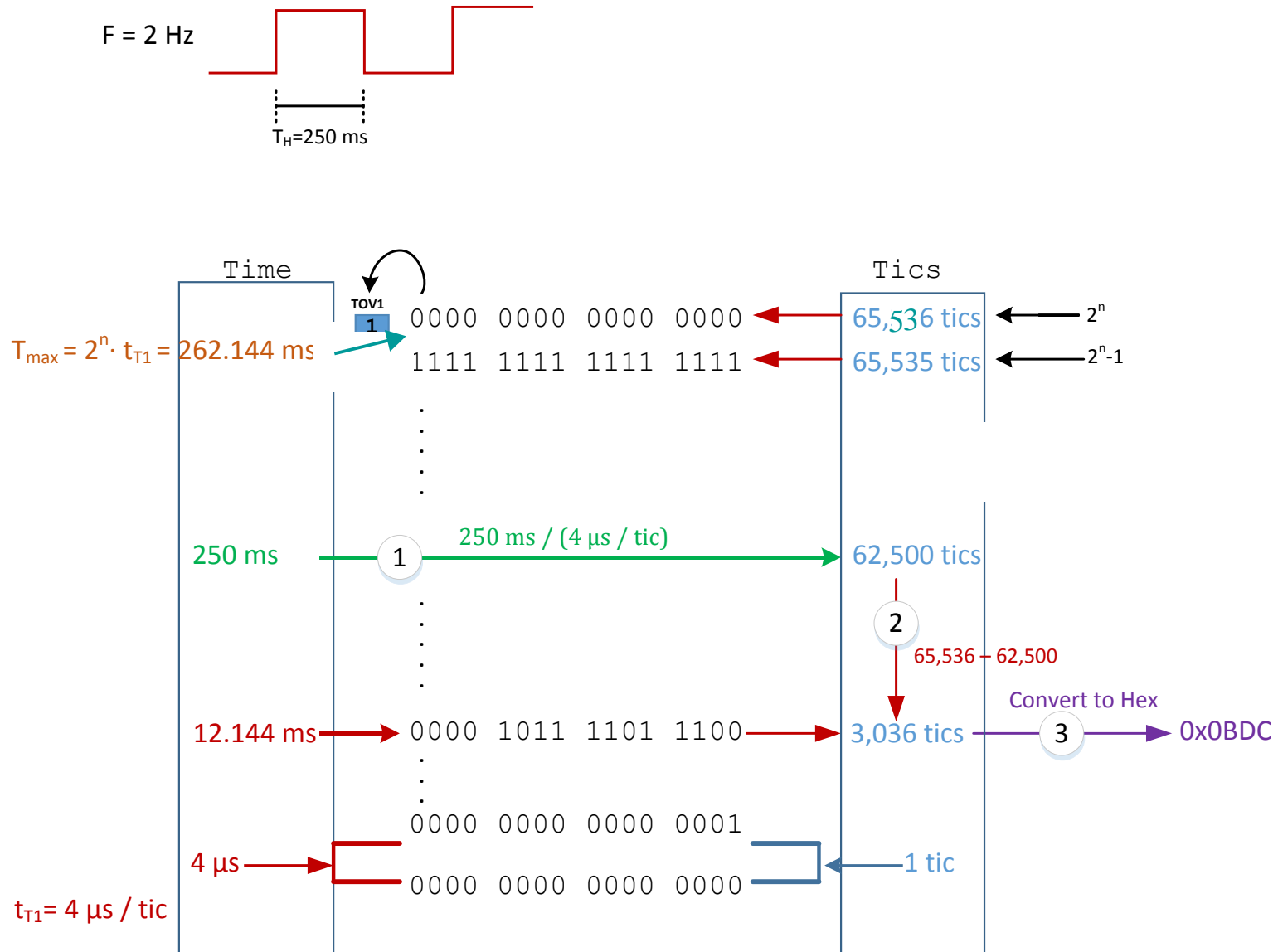
- The largest time delay possible is achieved by setting both TCNT1H and TCNT1L to zero, which results in the overflow flag TOV1 flag being set after  $2^{16} = 65,536$  tics of the Timer/Counter1 clock.

$$f_{\text{T1}} = f_{\text{Tclk\_I/O}}/64, \text{ given } f_{\text{Tclk\_I/O}} = f_{\text{clk}} \text{ then } f_{\text{T1}} = 16.000 \text{ MHz} / 64 = 250 \text{ KHz}$$

$$\text{and therefore } T1_{\text{max}} = 65,536 \text{ tics} / 250 \text{ KHz} = 262.14 \text{ msec}$$

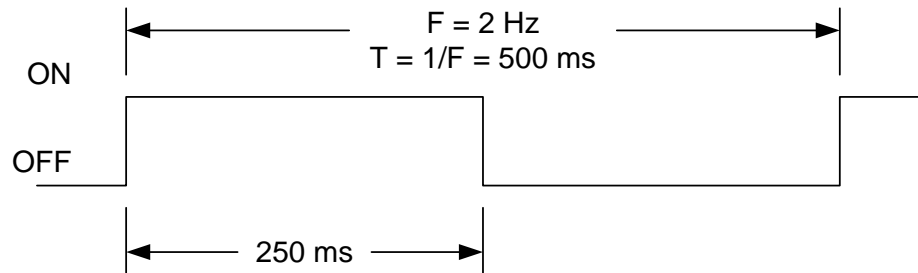
- Clearly, Timer 1 can generate a delay of 250 msec
- Our next step is to calculate the TCNT1 load value needed to generate a 250 ms delay.

## HOW TO CALCULATE TIMER LOAD VALUE (WHITEBOARD ILLUSTRATION)



## STEPS TO CALCULATE TIMER LOAD VALUE (NORMAL MODE)

### Problem



Generate a 250 msec delay assuming a clock frequency of 16 MHz and a prescale divisor of 64.

### Solution

1. Divide desired time delay by  $t_{clkT1}$  where  $t_{clkT1} = 64/f_{clk}/O = 64 / 16.000 \text{ MHz} = 4 \mu\text{sec/tic}$   
 $250\text{msec} / 4 \mu\text{s/tic} = 62,500 \text{ tics}$

**short-cut:**  $\text{TCNT1H} = \text{high}(-62,500)$  and  $\text{TCNT1L} = \text{low}(-62,500)$

2. Subtract 65,536 – step 1  
 $65,536 - 62,500 = 3,036$

3. Convert step 2 to hexadecimal.

$$3,036 = 0x0BDC$$

For our example  $\text{TCNT1H} = 0x0B$  and  $\text{TCNT1L} = 0xDC$

4. Check Answer

$$3,036 \text{ tics} \times 4 \mu\text{s/tic} = 12.14 \text{ msec}$$

$$262.14 \text{ msec} - 250 \text{ msec} = 12.14 \text{ msec} \quad \checkmark$$

## STEPS TO CALCULATE CLOCK DIVISOR (NORMAL MODE)

- In the previous example we assumed a divisor of 64, and then by calculating the maximum delay  $T_{MAX}$  verified that this assumption was correct. After that we simply followed the steps defined in the previous slide to calculate the value to be loaded into 16-bit timer/counter TCNT1.

$$T_{MAX} = \frac{2^n \cdot N}{f_{clk}} \quad \text{eq1.}$$

Where:  $T_{MAX}$  = maximum delay

$N$  = divisor

$n$  = number of flip-flops making-up the timer

$f_{clk}$  = system clock frequency

- But what if we are not given  $N$  and need to find TCNT1 for a given delay  $T$ . In this case we know that  $T \leq T_{MAX}$  and applying a little algebra can find an equation for  $N$ .

$$N \geq \frac{T \cdot f_{clk}}{2^n} \quad \text{eq2.}$$

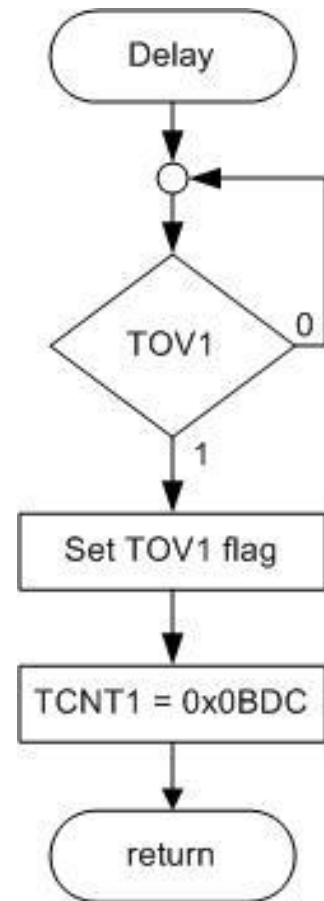
- Let's take a second look at our 250 msec delay problem. This time we will not assume a divisor of 64. Applying equation 2 we have:

$$N \geq (250 \text{ msec} \times 16 \text{ MHz}) / 2^{16} = 61.03$$

- From Table 13.5 "Clock Select Bit Description" on page 10, we see that the possible clock divisors are **1, 8, 64, 256, and 1024**. From this list we want to select the divisor that is the closest value, yet greater than or equal to  $N$ . For our example, not surprisingly the answer is again 64.

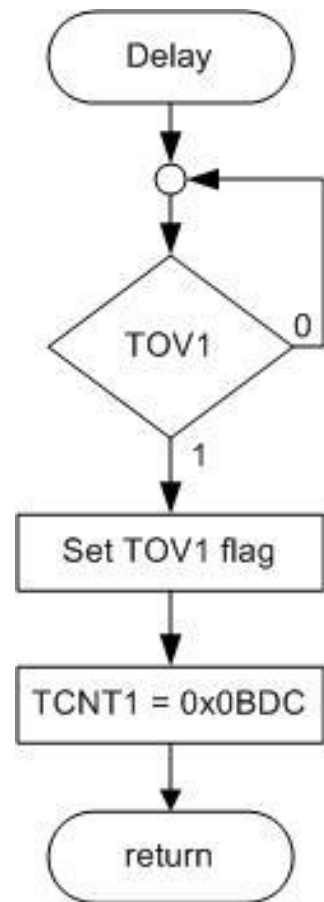
## POLLING EXAMPLE – ASSEMBLY VERSION

```
; -----  
; ----- Delay 250ms -----  
; Called from main program  
; Input: none      Output: none  
; no registers are modified by this subroutine  
Delay:  
    push    r15  
    in      r15, SREG  
    push    r16  
wait:  
    sbis    TIFR1, TOV1  
    rjmp    wait  
    sbi     TIFR1, TOV1    // clear flag bit by writing a one (1)  
    ldi     r16,0x0B        // load value high byte 0x0B  
    sts     TCNT1H,r16  
    ldi     r16,0xDC        // load value low byte 0xDC  
    sts     TCNT1L,r16  
    pop     r16  
    out     SREG, r15  
    pop     r15  
    ret
```



## POLLING EXAMPLE – C VERSION

```
; -----  
; ----- Delay 250ms -----  
; Called from main program  
; Input: none      Output: none  
void T1Delay()  
{  
    while (!(TIFR & (1<<TOV1))) // eq. to Ex: 9-42 expression  
        TIFR = 1<<TOV1;        // clear timer overflow flag  
    TCNT1H = 0x0B;  
    TCNT1L = 0xDC;  
}
```





## MORE LOOPING EXAMPLES

Here are six (6) other ways of implementing the looping part of the Polling Example written in assembly. See if you can come up with a few more.

<pre>wait:     sbis  TIFR1, TOV1  // targets a specific bit     rjmp  wait</pre>	<pre>wait:     in    r16, TIFR1     sbrs  r16, TOV1     rjmp  wait</pre>
<pre>wait:     in    r16, TIFR1     bst   r16, TOV1     brtc  wait</pre>	
<pre>wait:     in    r16, TIFR1     andi  r16, 0x01    // bitwise operation     breq  wait</pre>	<pre>wait:     in    r16, TIFR1     cbr   r16, 0xFE     breq  wait</pre>
<pre>wait:     in    r16, TIFR1     ror   r16     brcc  wait</pre>	<pre>wait:     in    r16, TIFR1     lsr   r16     brcc  wait</pre>