Timers and Timer Interrupts in **AVR**

Author: *nobody*

Aughust 17, 2022

Abstract

sources of this article: electronicwings.com and extremeelectronics.co.in
Generally, we use a timer/counter to generate time delays, waveforms, or to count events.
Also, the timer is used for PWM generation, capturing events, etc
In AVR ATmega16 / ATmega32, there are three timers:

- Timer0: 8-bit timer
- Timer1: 16-bit timer
- Timer2: 8-bit timer

Timer x ($x = 0, 1, 2$)

First, we need to understand the basic registers of the Timer x

- TCNTx: Timer / Counter Register x
  It is an 8-bit or 16-bit register. It counts up with each clock. (TCNTxH is high byte and TCNTxL is low byte).
- TCCRx: Timer / Counter Control register 0

1. TCCR1A: This register is used to configure the TIMER x. It has the following bits:

7	6	5	4	3	2	1	0
COMxA1	COMxA0	COMxB1	COMxB0	FOCxA	FOCxB	WGMx1	WGMx0

COMxA1 and COMxA0
This are used to configure the action for the event when the timer has detected a "match".

WGMx1 and WGMx0
these combined with WGM12 and WGM13 found in TCCR1B are used for selecting proper mode of operation.
WGM= Waveform Generation Mode.

2. TCCRxB: This register is also used for configuration. The Bits are:

7	6	5	4	3	2	1	0
ICNCx	ICESx	-	WGMx3	WGMx2	CSx2	CSx1	CSx0

Note:
WGM13 – WGM12 – WGM11 – WGM10 are used to select the proper mode of operation. Please refer to the datasheet for complete combinations that can be used.
We need the CTC mode i.e. clear timer on match so we set WGM13=0 , WGM12=1 , WGM11=0 , WGM10=0.

The CS12,CS11,CS10:

TCCRxB			Clock Source
CSx2	CSx1	CSx0	
0	0	0	OFF
0	0	1	clk
0	1	0	clk/8
0	1	1	clk/64
1	0	0	clk/256
1	0	1	clk/1024
1	1	0	External clock source on T0 pin Clock on falling edge.
1	1	1	External clock source on T0 pin Clock on rising edge.

- Output Compare Register 1 A – OCR1A (OCR1AH,OCR1AL):
  if we set OC1A=250, then when timer value (TCNT) be equal to 250 we will get an interrupt, Therefore the frequency of occurrence is $f_{\text{counter}}/250$ and where f_{counter} means frequency of the counter's clock source.
- Timer Counter Interrupt Mask (TIMSK):
  This is the mask register used to selectively enable/disable interrupts. This register is related with the interrupts of timers.

7	6	5	4	3	2	1	0
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	-	TOIE0

Of all these bits 5,4,3,2 are for TIMER1 and we are interested in the OCIE1A which is Output Compare Interrupt Enable 1 A.

After enabling the interrupt we also need to enable interrupt globally by using the function `sei();`

Code Example:

blink a LED every 1 second
first note that (in this example and using atmega328p) F_CPU is 16 Mhz ($f_{\text{clk}} = 16000000$Hz).

First Example.
using `TIMSK1 |= (1<<OCIE1A);` as we saw before, we enable Output Compare Interrupt.
using `TCCR1B |= (1<<CS10) | (1<<CS12);` we set CS10 and CS12 bits to 1 so frequency of the clock source of the counter1 now equls to $fclk/1024 = 16000000/1024 = 15625$Hz.
then by seeting `OC1A=15625;` the frequency of interrupts will be 1Hz.
Note. in every interrupt we need to clear the value of the counter (`TCNT1=0;`).

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define F_CPU 16000000L

ISR(TIMER1_COMPA_vect){
    PORTB ^= _BV(PORTB5);
    TCNT1  = 0;
}
```

```
int main(void){
    DDRB   |= _BV(PORTB5);

    TCNT1   = 0;
    OC1A    = 15625;

    TCCR1A |= (1<<WGM12);
    TCCR1B |= (1<<CS10) | (1<<CS12);
    TIMSK1 |= (1<<OCIE1A);

    sei();

    while(1){}
}
```

Second Example. Like the first example, the frequency of the clock source of the counter1 equls to 15625Hz.
but this time set `OC1A=0;` so interrupts occur when TCNT1=0, but we set the initial value of timer1 to 49911 and by ressting it at each interrupt, the minimum of counter1 will be 49911.
as we know, maximum value of the timer1 is $2^{16} = 65536$, therefore we have:

$$f_{\text{interrupt}} = \frac{f_{\text{counter}}}{65536 - 49911} = 15625/15625 = 1\text{Hz}.$$

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define F_CPU 16000000L

ISR(TIMER1_COMPA_vect){
    PORTB ^= _BV(PORTB5);
    TCNT1  = 49911;
}

int main(void){
    DDRB   |= _BV(PORTB5);

    TCNT1   = 49911;
    OC1A    = 0;

    TCCR1A |= (1<<WGM12);
    TCCR1B |= (1<<CS10) | (1<<CS12);
    TIMSK1 |= (1<<OCIE1A);

    sei();

    while(1){}
}
```