The codes below which help in better understanding of timers and counters. I have tested this code for atmega32. I have taken reference from www.avrfreaks.net . Hope you all will find this useful.

– Darsh Shah

```
// blinking led on PORTC using interrupt and Timer0.

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>


char togg=00000000;


SIGNAL (SIG_OVERFLOW0)

{

togg=~togg;    // togg^=0xFF;

PORTC=togg;

}


void main( void )

{

DDRC=0xFF; /* use all pins on PORTB for output */


TIMSK=0b00000001; /* enables the T/C0 overflow interrupt in the T/C interrupt mask register for */

TCNT0=0; /* start value of T/C0 */

TCCR0=0b00000101; /* prescale ck/1024 */


sei(); /* set global interrupt enable */
for (;;){}

}
```

```
// blinking of led at .5 second rate using timer1 which is prescaled 8.overflow  interrupt is used.

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>


SIGNAL(SIG_OVERFLOW1)

{

  PORTC ^= (1 << 0); // Toggle the LED

}


int main (void)

{

   DDRC =0xFF; // Set LED as output

   TIMSK |= (1 << TOIE1); // Enable overflow interrupt

   sei(); // Enable global interrupts

   TCCR1B |= (1 << CS11); // Start timer at Fcpu/8


   for (;;)

   { }

}

// the below function can also be used to execute the interrupt. depends upon the user which to
use.
```

```
/*ISR(TIMER1_OVF_vect)

{

  PORTC ^= (1 << 0); // Toggle the LED

}*/
```

```
// blinking of led at 1 second rate using timer1 which is prescaled  64. CTC (Clear on Timer Compare)
is used

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>




int main (void)

{

  DDRC =0xFF; // Set LED as output


  // Configure timer 1 for CTC mode

  TCCR1B |= ((1 << WGM12)|(1 << CS10) | (1 << CS11)); // Start timer at Fcpu/64

  OCR1A   = 15625; // Set CTC compare value to 1Hz at 1MHz AVR clock, with a prescaler of 64


  for (;;)

  {

       if (TIFR & (1 << OCF1A))  // if flag is set i.e. comparison is true

        {

        PORTC ^= (1 << 0); // Toggle the LED
        TIFR = (1 << OCF1A); // clear the CTC flag (writing a logic one to the set flag clears it)

        }

  }

}
```

// blinking of led at 1 second rate using timer1 which is prescaled 64. CTC (Clear on Timer Compare) interrupt is //used.

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

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>


SIGNAL(SIG_OUTPUT_COMPARE1A)

{

 PORTC ^= (1 << 0); // Toggle the LED

}


int main (void)

{

  DDRC =0xFF; // Set LED as output


  // Configure timer 1 for CTC mode

  TCCR1B |= ((1 << WGM12)|(1 << CS10) | (1 << CS11)); // Start timer at Fcpu/64

  OCR1A   = 15625; // Set CTC compare value to 1Hz at 1MHz AVR clock, with a prescaler of 64

  TIMSK |= (1 << OCIE1A); // Enable CTC interrupt

   sei(); //  Enable global interrupts
  for (;;)

  { }

}


// the below function can also be used to execute the interrupt. depends upon the user which to use.

/*ISR(TIMER1_COMPA_vect)

{

  PORTC ^= (1 << 0); // Toggle the LED

}*/
```

```
// blinking of led at 1 second rate using timer1 which is prescaled 64.overflow  interrupt is used and bottom is changed

// Reload=65535(TOP)-15625(period required for 1Hz)=49910(start value)

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>


SIGNAL(SIG_OVERFLOW1)

{

 PORTC ^= (1 << 0); // Toggle the LED

 TCNT1  = 49910; // Reload timer with precalculated value

}


int main (void)
```

```
{
  DDRC =0xFF; // Set LED as output


  TIMSK |= (1 << TOIE1); // Enable overflow interrupt
  sei(); // Enable global interrupts


  TCNT1 = 49910; // Preload timer with precalculated value


  TCCR1B |= ((1 << CS10) | (1 << CS11)); // Set up timer at Fcpu/64


  for (;;)
  { }


 }

// the below function can also be used to execute the interrupt.depends upon the user which to use.
/*ISR(TIMER1_OVF_vect)
{
  PORTC ^= (1 << 0); // Toggle the LED
 TCNT1  = 49910; // Reload timer with precalculated value
}*/
```

```
// blinking of led at 10 second rate using timer1 which is prescaled 64.so long delay can be given
#include <avr/io.h>
#include <avr/interrupt.h>
```

```
#include <util/delay.h>

#include <compat/deprecated.h>


int main (void)

{ unsigned char SEC = 0; // Make a new counter variable and initialize to zero

  DDRC =0xFF; // Set LED as output


  TCCR1B |= ((1 << CS10) | (1 << CS11)); // Set up timer at Fcpu/64


  for (;;)

  {

    // Check timer value in if statement, true when count matches 1 second

    if (TCNT1 >= 15625)

    { TCNT1 = 0; // Reset timer value

      SEC++;


      if (SEC == 10) // Check if 10 sec has elapsed

      {

        SEC = 0; // Reset counter variable


      PORTC ^= (1 << 0); // Toggle the LED


      }

    }

  }

}
```

```
// blinking of led at 1 second rate using timer1 which is prescaled 64.

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>


int main (void)

{

  DDRC =0xFF; // Set LED as output


  TCCR1B |= ((1 << CS10) | (1 << CS11)); // Set up timer at Fcpu/64


  for (;;)

  {

    // Check timer value in if statement, true when count matches 1 second

    if (TCNT1 >= 15625)

    {

      PORTC ^= (1 << 0); // Toggle the LED


      TCNT1 = 0; // Reset timer value

    }

  }

}
```

```
//Timers running at Fcpu and we want the led to blink at 20hz

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <compat/deprecated.h>



int main (void)

{

  DDRC =0xFF; // Set LED as output


  TCCR1B |= (1 << CS10); // Set up timer


  for (;;)

  {

    // Check timer value in if statement, true when count matches 1/20 of a second

    if (TCNT1 >= 50000)

    {

      PORTC ^= (1 << 0); // Toggle the LED


      TCNT1 = 0; // Reset timer value

    }

  }

}
```