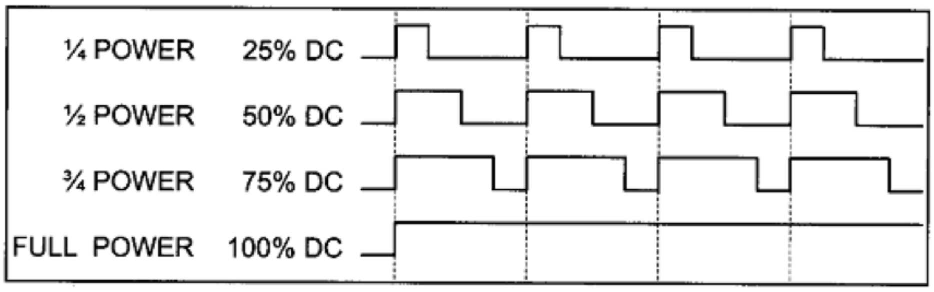
Introduction:

To learn the effective use of Timers in AVR microcontroller ATMEGA16 for the generation of PWM. By controlling the duty cycle of a signal, the power generated to your circuit load can be controlled.

Required Design Parameters:

Pulse width modulation (PWM) is a technique of controlling the amount of power delivered to an electronic load using an on-off digital signal. The fraction of the period for which the signal is on is known as the duty cycle. The average DC value of the signal can be varied by varying the duty cycle. The duty cycle can be anywhere between 0 (signal is always off) to 1 (signal is constantly on).

Suppose, if the signal has +5 V while it is on and 0 V during off condition, then by changing the duty cycle of the signal, any voltage between 0-5 V can be simulated. This method is commonly used for controlling speeds of DC motors and brightness of lamps.



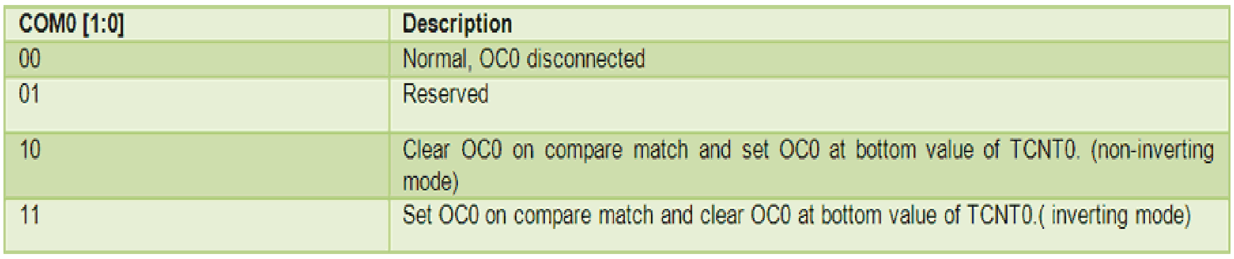
In AVR microcontrollers PWM signals are generated by the TIMER units. There are two methods by which you can generate PWM from AVR TIMER0 (for ATmega16 and ATmega32 MCUs).

1. Fast PWM
2. Phase Correct PWM

We will be using Fast PWM mode for this lab:

The Fast PWM mode is based on single-slope operation. In single slope operation, the register TCNTn counts from bottom value to maximum value and its value resets to zero. The counting starts again from bottom. The register OCRn compares the value with the TCNTn register constantly.

If the timer is configured in non-inverting mode, PWM output pin (OCn) goes low when the value of the above two registers matches. The OCn pin becomes high when the TCNTn register reaches at bottom value. In inverting mode OCn pin behaves opposite to non-inverting mode. For timer 0 fast PWM mode, following table shows the functionality of COM 0[1:0] bits.



COM0 [1:0]	Description
00	Normal, OC0 disconnected
01	Reserved
10	Clear OC0 on compare match and set OC0 at bottom value of TCNT0. (non-inverting mode)
11	Set OC0 on compare match and clear OC0 at bottom value of TCNT0.(inverting mode)

Frequency of fast PWM mode signal is twice than Phase Correct PWM mode signal because of its single slope operation. Output frequency of fast

$$PWM\ signal\ =\ Crystal\ frequency \div\ (Prescaler\ \times 256).$$

Design Strategy:

We select AVR studio 4 to develop and compile desired code for Atmega 16 micro-controller and code is simulated to check its integrity and working in Proteus.

We used timer 2 for generating PWM of Atmega 16 with given configuration.

8 bit TCCR2 register is initialized by:

WGM20 →Set, WGM21→Set //for fast PWM
COM20 →Not set, COM21→Set //for non-inverted output
CS20 →Not set, CS21→set, CS21→ Not set //for pre-scaler=8

And,

TCNT2 is timer register and OCR2 register is use to control duty cycle of PWM.

Now,

Our PWM frequency is:

$$PWM\ freq = \frac{clock\ freq}{256 * prescaler}$$

$$PWM\ freq = \frac{16MHz}{256 * 8}$$

$$PWM\ freq = 7812.5\ Hz$$

And, our duty cycle is periodically increasing and decreasing to ensure that Led light gradually gets bright and then gradually gets dim. This process remain continue forever.

Now, OCR2 register value:

0 ….→ 255…. → 0 // whole cycle completes in 2 seconds

Then, time required for one increment is:

$$Unit\ delay = 2/512$$

$$Unit\ delay \cong 4\ msec$$

So, delay is set to 4 msec.

Embedded C Code of Program:

```
#include <avr/io.h> // header files
#include <util/delay.h>

void Initiate_PWM()
{
  /*
  TCCR2 - Timer Counter Control Register (TIMER2)
  -----------------------------------------------
  BITS DESCRIPTION


  NO:   NAME   DESCRIPTION
  --------------------------
  BIT 7 : FOC2   Force Output Compare [Not used in this example]
  BIT 6 : WGM20  Wave form generation mode [SET to 1]
  BIT 5 : COM21  Compare Output Mode       [SET to 1]
  BIT 4 : COM20  Compare Output Mode       [SET to 0]

  BIT 3 : WGM21  Wave form generation mode [SET to 1]
  BIT 2 : CS22   Clock Select             [SET to 0]
  BIT 1 : CS21   Clock Select             [SET to 1]
  BIT 0 : CS20   Clock Select             [SET to 0]

  The above settings are for
  --------------------------

  Timer Clock = CPU Clock/8 (Pre-scaling by 8)
  Mode        = Fast PWM
  PWM Output  = Non Inverted

  */


  TCCR2|=(1<<WGM20)|(1<<WGM21)|(1<<COM21)|(1<<CS21);

  //Set OC2 PIN as output. It is  PD7 on ATmega16

  DDRD|=(1<<PD7);
}


//Function to set PWM duty cycle
void SetPWM_DutyCycle(uint8_t duty_cycle)
{
  OCR2=duty_cycle;
}

//Delay in Milli-second
void Wait()
{
 _delay_ms(4);
}

void main()
{
  uint8_t bright=0;

  //Initialize PWM Channel 2
  Initiate_PWM();

  //Do this forever
  while(1)
  {
    //Now Loop with increasing brightness

    for(bright=0;bright<255;bright++)
    {
      //Now Set The Brighness using PWM

      SetPWM_DutyCycle(bright);

      //Now Wait For Some Time
      Wait();
    }

    //Now Loop with decreasing brightness

    for(bright=255;bright>0;bright--)
    {
      //Now Set The Brighness using PWM

      SetPWM_DutyCycle(bright);

      //Now Wait For Some Time
      Wait();
    }
  }
  }
```

Proteus Simulation:

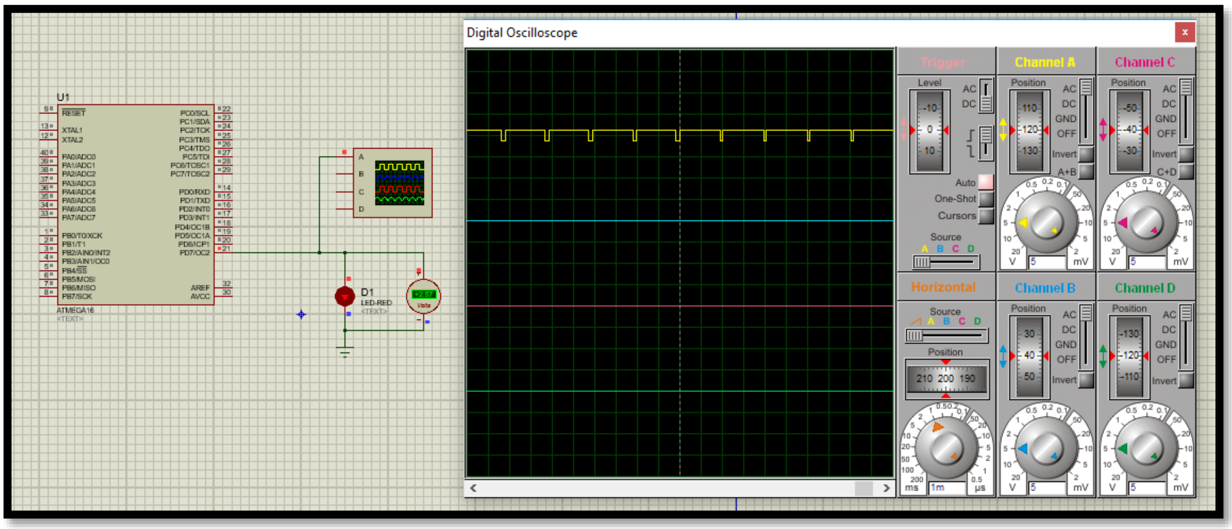


Figure No. 1 Proteus Simulation
