Peripheral Interference Controller (PIC)- Simulation using Proteus

PIC is a microcontroller with a RISC architecture operating in a frequency DC to 40 at operating voltage 12v Volt with 40-100 pins .
We are using a PIC18F4580 microcontroller which has a total of 40 ports .
Microchip Inc. has developed the PIC18 series of microcontrollers for use in high-pin-count, high-density, and complex applications. The PIC18F microcontrollers offer cost-efficient solutions for general purpose applications written in C that use a real-time operating system (RTOS) and require a complex communication protocol stack such as TCP/IP, CAN, USB, or ZigBee. PIC18F devices provide flash program memory in sizes from 8 to 128Kbytes and data memory from 256 to 4Kbytes, operating at a range of 2.0 to 5.0 volts, at speeds from DC to 40MHz.

The basic features of PIC18F-series microcontrollers are:

- 77 instructions
- PIC16 source code compatible
- Program memory addressing up to 2Mbytes
- Data memory addressing up to 4Kbytes
- DC to 40MHz operation
- 8 8 hardware multiplier
- Interrupt priority levels
- 16-bit-wide instructions, 8-bit-wide data path
- Up to two 8-bit timers/counters
- Up to three 16-bit timers/counters
- Up to four external interrupts

Hexadecimal Number System

The **hexadecimal number system** is a type of number system, that has a base value equal to 16. It is also pronounced sometimes as **'hex'**. Hexadecimal numbers are represented by only 16 symbols. These symbols or values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Each digit represents a decimal value. For example, D is equal to base-10 13.

Decimal Numbers	4-bit Binary Number	Hexadecimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

PROJECT 1 :- LED BLINKING

Objective:

The aim is to simulate blinking of LED in one port(eg:PORT C)

Softwares Required:

MPLAB and PROTEUS

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include<pic18.h>
void delay()
{
        int i,j;
         for(i=0;i<100;i++)
         for(j=0;j<100;j++);
}
void main()
{
        TRISC=0X00;
        while(1)
        {
            PORTC=0XFF;
               delay();
           PORTC=0X00;
            delay();
          }
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

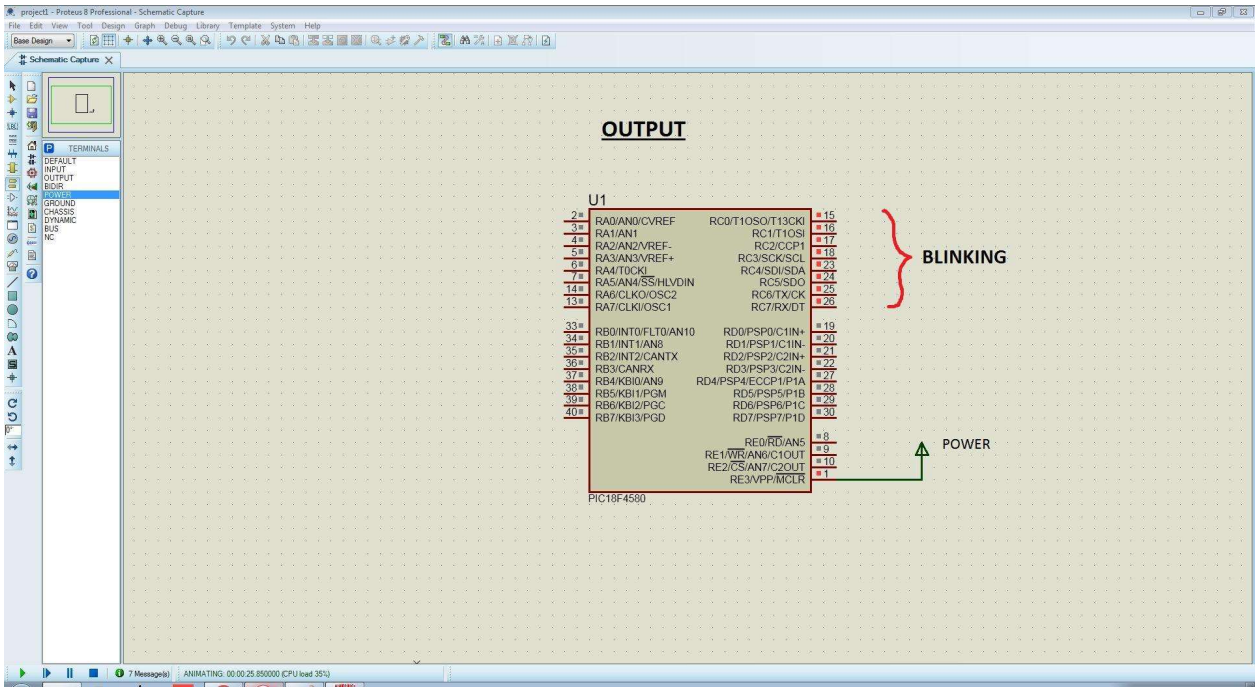
PIC18f4580 microcontroller, Power.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output of the simulation is given below 8 pins of PORT C will blink



<**********>

Program For blinking odd pins of PORTC

Program:

```
#include<pic18.h>
 void delay();
void main()
{

    TRISC=0X00;
    while(1)
    {
        PORTC=0X55;//odd pins in PORTD
        delay();
        PORTC=0X00;
        delay();

    }
}
void delay()
{
    for(int i=0;i<10000;i++);
}
```

Program For blinking even pins of PORTD

Program:

```
#include<pic18.h>
 void delay();
```

```
void main()
{

   TRISC=0X00;
   while(1)
   {
      PORTC=0XAA; //odd pins in PORTD
      delay();
      PORTC=0X00;
      delay();




   }
}
void delay()
{
   for(int i=0;i<10000;i++);
}
```

PROGRAM FOR TURNING ON ONE BY ONE LED RB0 TO RB7 THEN RC0 TO RC7

```
#include<pic18.h>
void delay();
void main()
{
   TRISB=0X00;
```

```
    TRISC=0X00;
    ADCON1=0X00;
    while(1)
    {
        for(int i=0;i<8;i++)
        {
            PORTB=0X01<<i;
            delay();
        }
        PORTB=0X00;
        delay();
        for(int i=0;i<8;i++)
        {
            PORTC=0X01<<i;
            delay();
        }
        PORTC=0X00;
        delay();
    }




}
void delay()
{
    for(int i=0;i<25000;i++);
}
```

PROGRAM FOR PERFORM LEFT SHIFTING AT PORTC

```
#include<pic18.h>
 void delay();
void main()
{

    TRISC=0X00;

    while(1)
    {
        for(int i=0;i<8;i++)
        {
            PORTC=0X01<<i;
            delay();
        }


    }
}
void delay()
{
    for(int i=0;i<10000;i++);
}
```

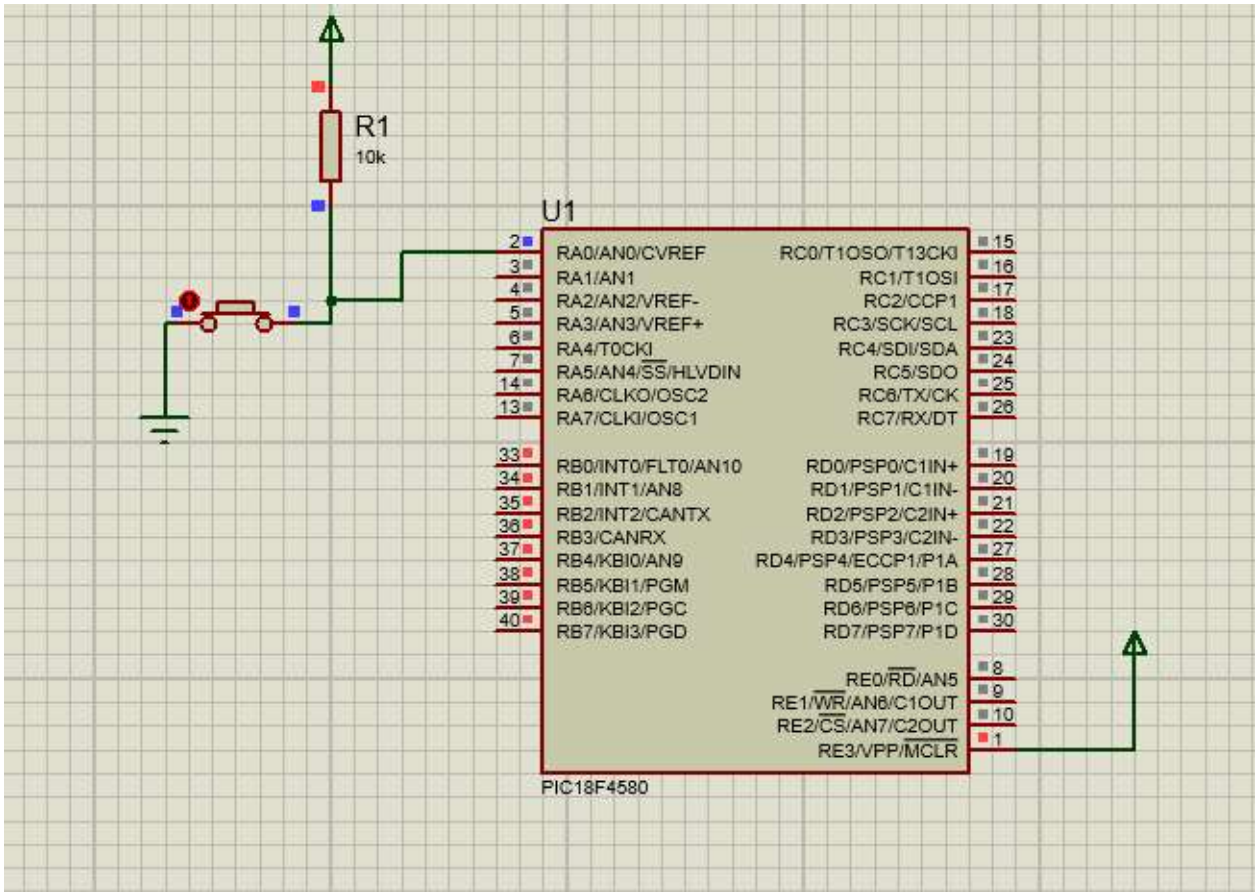
PROGRAM FOR PERFORM RIGHT SHIFTING AT PORTC

```
#include<pic18.h>
 void delay();
void main()
{
```

```
    TRISC=0X00;

    while(1)
    {
        for(int j=0;j<8;j++)
        {
            PORTC=0X80>>j;
            delay();
        }
    }
}
void delay()
{
    for(int i=0;i<10000;i++);
}
```

SWITCH CONCEPT (PULLUP & PULLDOWN)

PULLDOWN…

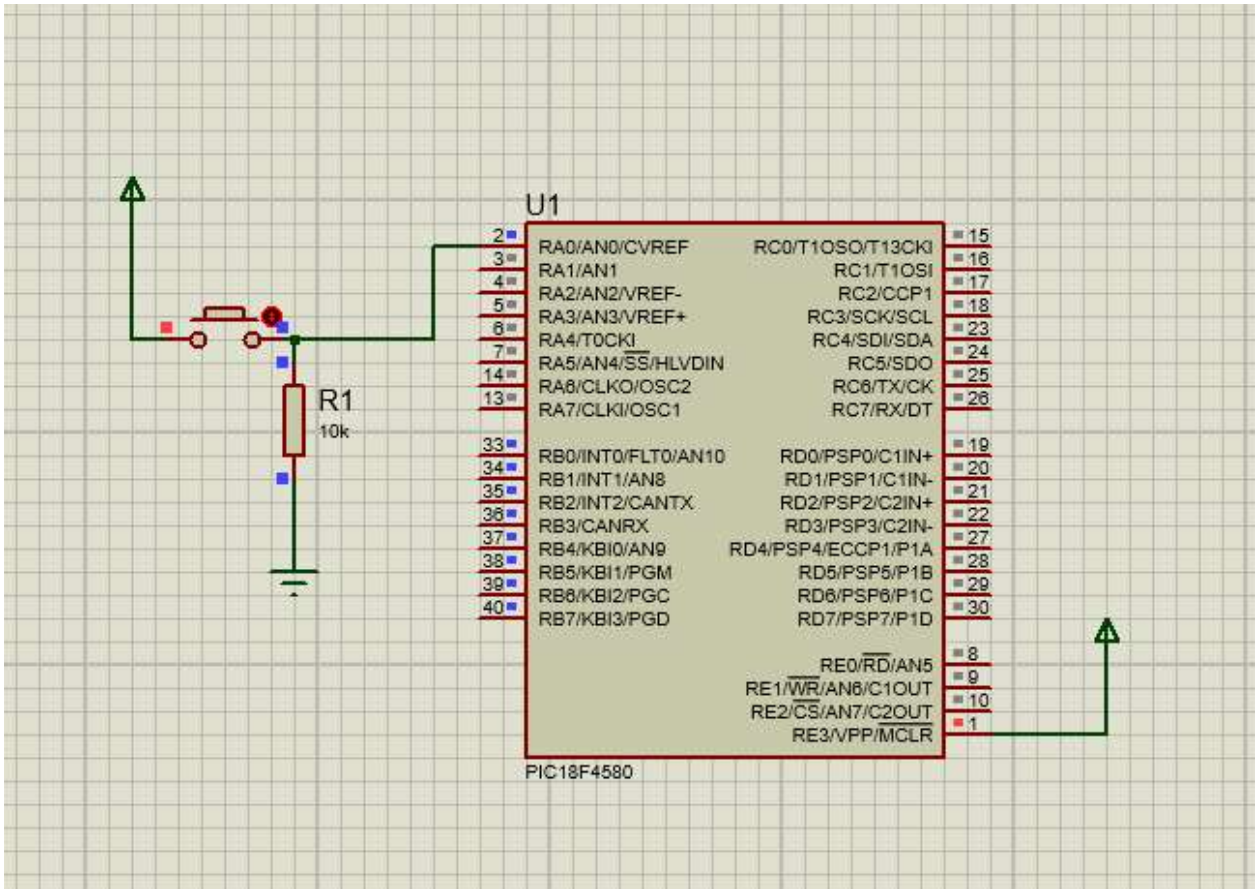


PROGRAM:

```
#include<pic18.h>
void main()
{
   ADCON1=0X0F;
   TRISA=0XFF;
   TRISB=0X00;
   while(1)
   {
      if(RA0==0)
      {
         PORTB=0XFF;
      }
      else
```

```
        {
            PORTB=0X00;
        }
    }
}
```

PULLUP:

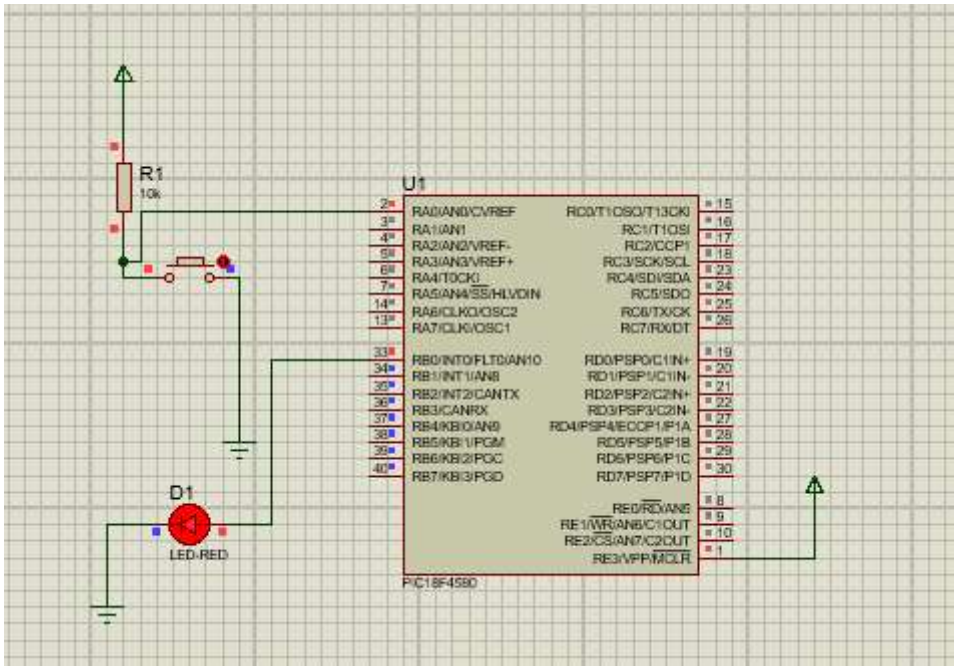


PROGRAM:

```
#include<pic18.h>
void main()
{
```

```
    ADCON1=0X0F;
    TRISA=0XFF;
    TRISB=0X00;
    while(1)
    {
        if(RA0==1)
        {
            PORTB=0XFF;
        }
        else
        {
            PORTB=0X00;
        }
    }
}
```

LED ON IN FIRST PUSH OFF IN SECOND PUSH



PROGRAM:

```
#include<pic18.h>
void main()
{
   ADCON1=0X0F;
   TRISA=0XFF;
   TRISB=0X00;
   int c=0;
   int a;
   while(1)
   {
      if(RA0==0)
      {
         c++;
         while(RA0==0);

    }
         if(c%2==0)
         {
            RB0=1;
         }
         else if(c%2==1)
        {
        RB0=0;
        }
   }

}
```

PROJECT 2:- Motor control

Objective:

The aim is to simulate two dc motors rotating in different direction controlled by 3 switches

Softwares Required:

MPLAB and PROTEUS

STEP 1:

Using **MPLAB** compile the following code and create hex file

MOTOR AND SWITCH

```
#include<pic18.h>
void main()
{
```

Open Proteus and place the components on the workspace

STEP 2:Components Required :-

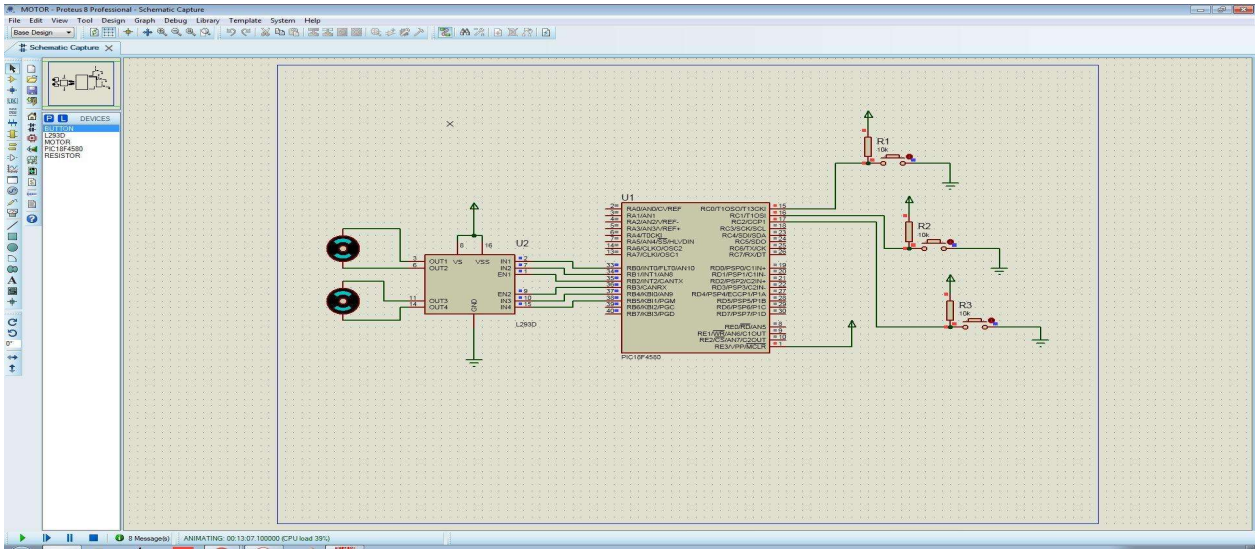
PIC18f4580 microcontroller, Power, Ground, l293d, Active motor, Resistor, Push button.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output of the simulation is given below 3 motors now controlled by 3 switches to rotate different directions and different combinations



FIRST MOTOR CLOCKWISE

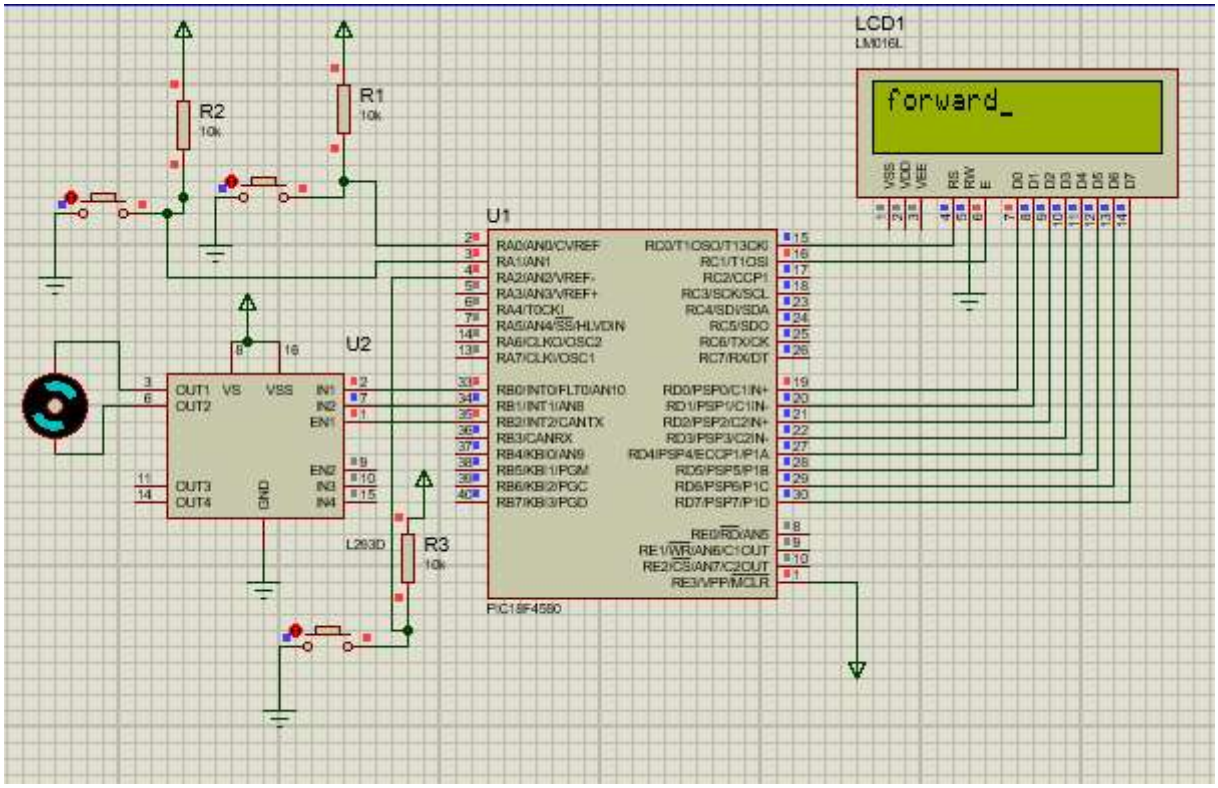
```
#include<pic18.h>
void main()
{
        TRISB=0X00;
         PORTB=0X05;

}
```

FIRST MOTOR ANTI-CLOCKWISE

```
#include<pic18.h>
void main()
{
        TRISB=0X00;
```

```
        PORTB=0X05;

}
```

Control motor using push button and print status on lcd



Program:

```
#include<pic18.h>
void cmd(char a);
void data(char b);
void delay();
void display(char *p);
```

```
void main()
{
ADCON1=0X0F;
TRISC=0x00;
TRISD=0x00;
TRISB=0X00;
TRISA=0XFF;
cmd(0x38);
cmd(0x01);
cmd(0x06);
cmd(0x0E);
cmd(0x80);
while(1)
{

        if(RA0==0)
        {
                PORTB=0X05;
        display("forward");
        }
        if(RA1==0)
        {
                PORTB=0X06;
       display("reverse");
        }

        cmd(0x01);
}

}
```

```
void cmd(char a)
{
PORTD=a;
RC0=0;
RC1=1;
delay();
RC1=0;
}
void data(char b)
{
PORTD=b;
RC0=1;
RC1=1;
delay();
RC1=0;
}
void display(char *p)
{
while(*p!=0)
{
data(*p);
p++;
}
}
void delay()
{
for(int i=0;i<300;i++)
for(int j=0;j<200;j++);
}
```

PROJECT 3:-LCD

Objective:

The aim is to simulate "WELCOME" in lcd display

Softwares Require MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include<pic18.h>
void cmd(char a);
void data(char b);
void delay();
void display(char *p);
void main()
{
TRISC=0x00;
TRISD=0x00;
cmd(0x38);
cmd(0x01);
cmd(0x06);
cmd(0x0E);
cmd(0x80);
display("WELCOME");
}
void cmd(char a)
{
```

```
PORTD=a;
RC0=0;
RC1=1;
delay();
RC1=0;
}
void data(char b)
{
PORTD=b;
RC0=1;
RC1=1;
delay();
RC1=0;
}
void display(char *p)
{
while(*p!=0)
{
data(*p);
p++;
}
}
void delay()
{
for(int i=0;i<300;i++)
for(int j=0;j<200;j++);
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

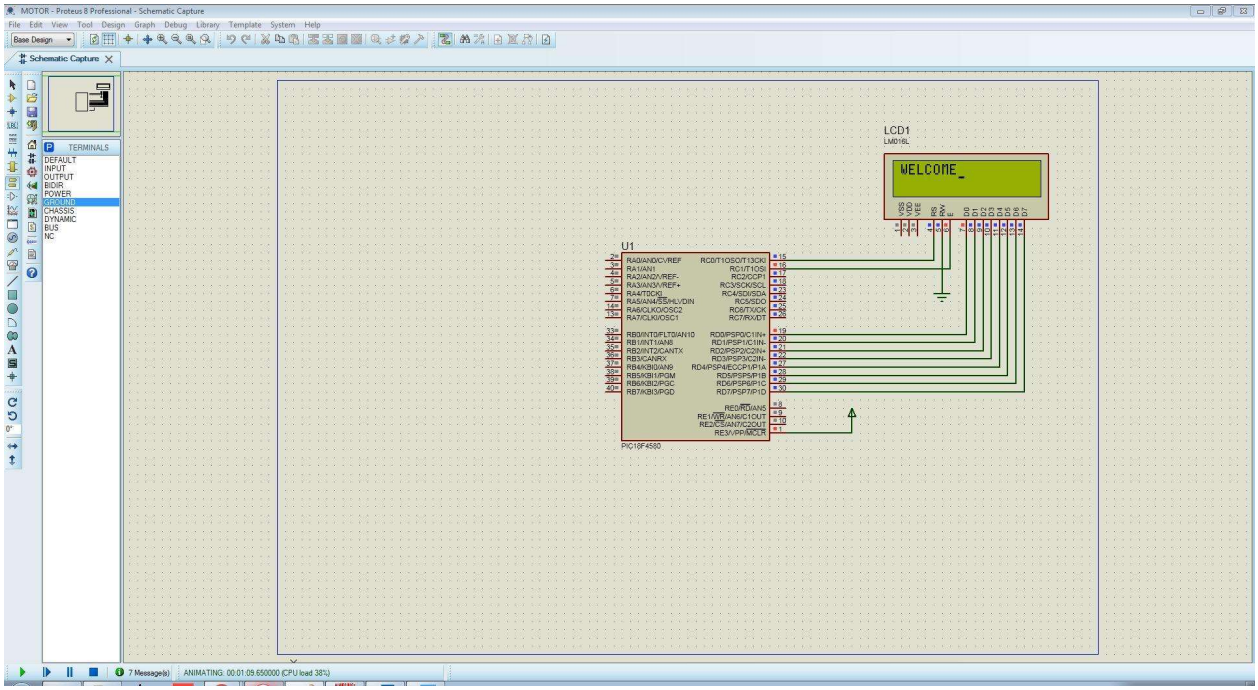
PIC18f4580 microcontroller,LM016L, Power, Ground.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output of the simulation is given below LCD displays “WELCOME” as output



PROGRAM FOR DISPLAYING STRING ON LCD WITHOUT POINTER FUNCTION

```
#include<pic18.h>
void delay();
void data(char b);
void cmd(char a);
void main()
{
```

```
TRISC=0x00;
TRISD=0x00;
cmd(0x38);
cmd(0x01);
cmd(0x06);
cmd(0x0E);
cmd(0x80);
char a[]="hello";
for(int i=0;i<5;i++)
{
    data(a[i]);
}
}
void cmd(char a)
{
PORTD=a;
RC0=0;
RC1=1;
delay();
RC1=0;
}
void data(char b)
{
PORTD=b;
RC0=1;
RC1=1;
delay();
RC1=0;
}

void delay()
{
```

```
for(int i=0;i<300;i++)
for(int j=0;j<200;j++);
}
```

PROGRAM FOR DISPLAYING STRING ON LCD WITHOUT FUNCTION

```
#include<pic18.h>
void main()
{
TRISC=0x00;
TRISD=0x00;
PORTD=0x00;
int i;

char a='h',b='a',c='i';
PORTD=0X0F;
RC0=0;
RC1=1;
for(i=0;i<10000;i++);
RC1=0;
PORTD=0x01;
RC0=0;
RC1=1;
for(i=0;i<10000;i++);
RC1=0;
PORTD=a;
RC0=1;
RC1=1;
for(i=0;i<10000;i++);
RC1=0;
```

```
PORTD=b;
RC0=1;
RC1=1;
for(i=0;i<10000;i++);
RC1=0;

PORTD=c;
RC0=1;
RC1=1;
for(i=0;i<10000;i++);
RC1=0;
  while(1);
}
```

TIMER

A timer (sometimes referred to as a counter) is a special piece of hardware inside many microcontrollers. Their function is simple: they count (up or down, depending on the configuration--we'll assume up for now). For example, an 8-bit timer will count from 0 to 255

PROJECT 4:- TIMER'S

Objective:

The aim is to create a simulation of timer 0,timer1 and timer 2 with a delay of 10ms

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

TIMER 0

```
#include<pic18.h>
void timer();
void main()
{
TRISC=0X00;
T0CON=0XC7;
while(1)
{
PORTC=0X0F;
timer();
PORTC=0X00;
timer();
}
}
void timer()
{
while(TMR0IF==0);
TMR0IF=0;
TMR0L=61;
}
```

TIMER 1

```
#include<pic18.h>
void timer();
void main()
{
TRISC=0x00;
   TRISD=0x00;
T1CON=0x41;
while(1)
{
PORTC=0x0F;
timer();
PORTC=0x00;
timer();
}
}
void timer()
{
while(TMR1IF==0);
TMR1IF=0;
TMR1L=0xB0;
TMR1H=0x3C;
}
```

TIMER 2:

```
#include<pic18.h>
void timer();
void main()
{
TRISC=0x00;
T2CON=0x7F;
```

```
while(1)
{
PORTC=0x0F;
timer();
PORTC=0x00;
timer();
}
}
void timer()
{

while(TMR2IF==0);
TMR2IF=0;
PR2=195;

}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

PIC18f4580 microcontroller, Power, Ground, Oscilloscope.

/**********

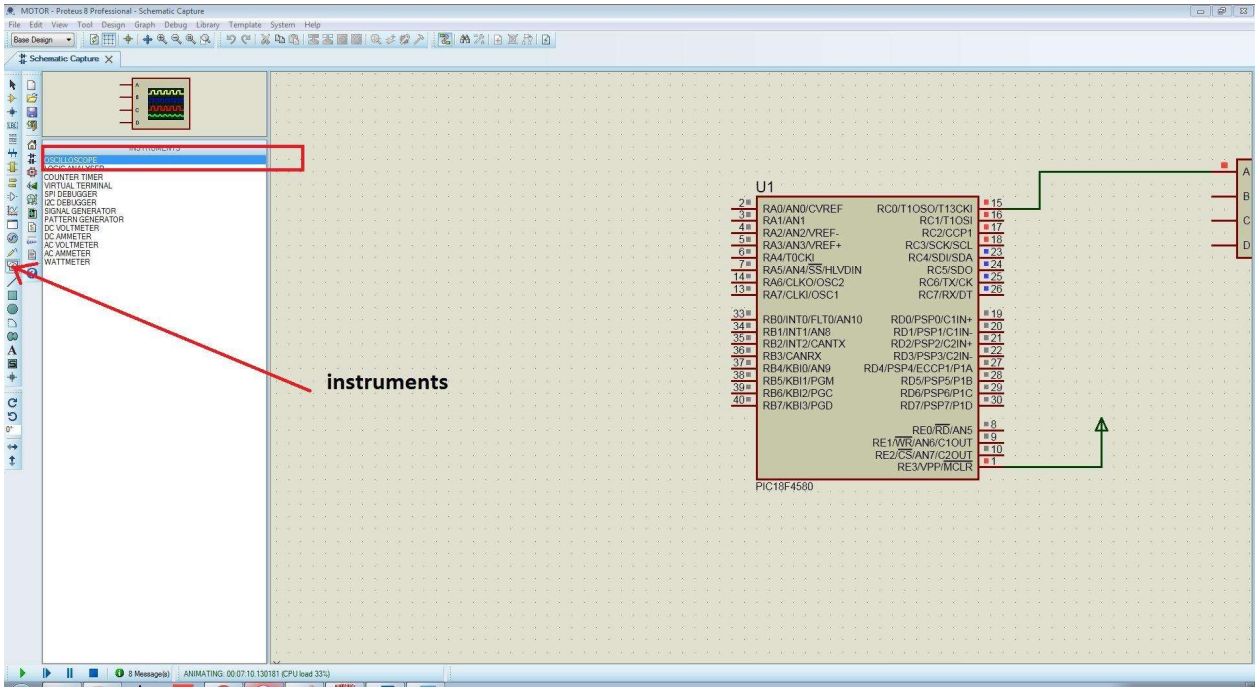
HOW SETUP OSCILLOSCOPE IN PROTEUS

Step(i):

Choose the instruments option from the tool menu present on left side of the proteus

Step(ii):

Now Choose the oscilloscope from the list and place it in proteus workspace then connect any of the channel terminal to the outputted port



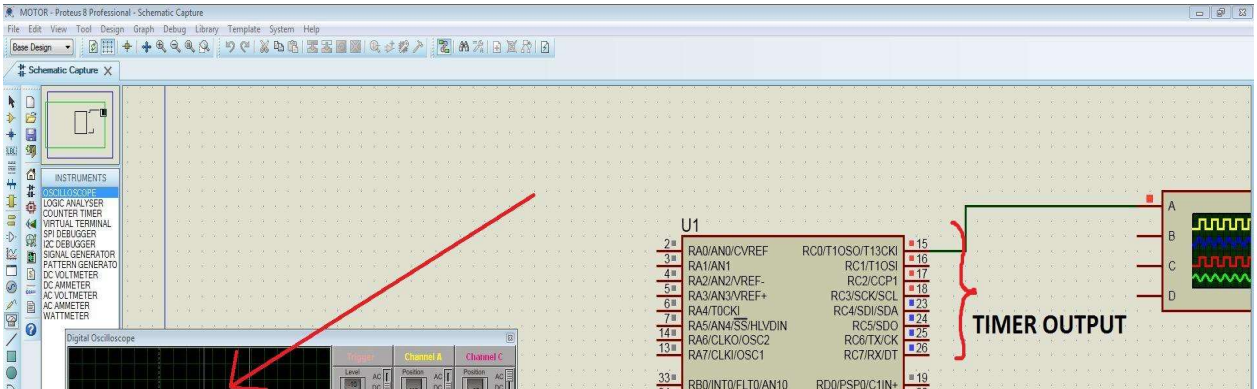
**********/

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output simulation OF TIMER'S (0,1,2) are given below



PROJECT 5:- USART

Objective:

The aim is to create a simulation of USART communication

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include<pic18.h>
void trans(char a);
void display(const char *p);
char rec();
void main()
{
	char h='a';
	TRISC=0X80;
```

```
        TXSTA=0x24;
        RCSTA=0X90;
        SPBRG=129;
        while(1)
        {
                h=rec();
                trans(h);
        }
}
void trans(char a)
{
        TXREG=a;
        while(TXIF==0);
        TXIF=0;
}
char rec()
{
        while(RCIF==0);
        RCIF=0;
        return RCREG;
}
void display(const char*p)
{
        while(*p!='\0')
        {
                trans(*p);
                p++;
        }
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

PIC18f4580 microcontroller, Power, Ground, Virtual Terminal.

/***

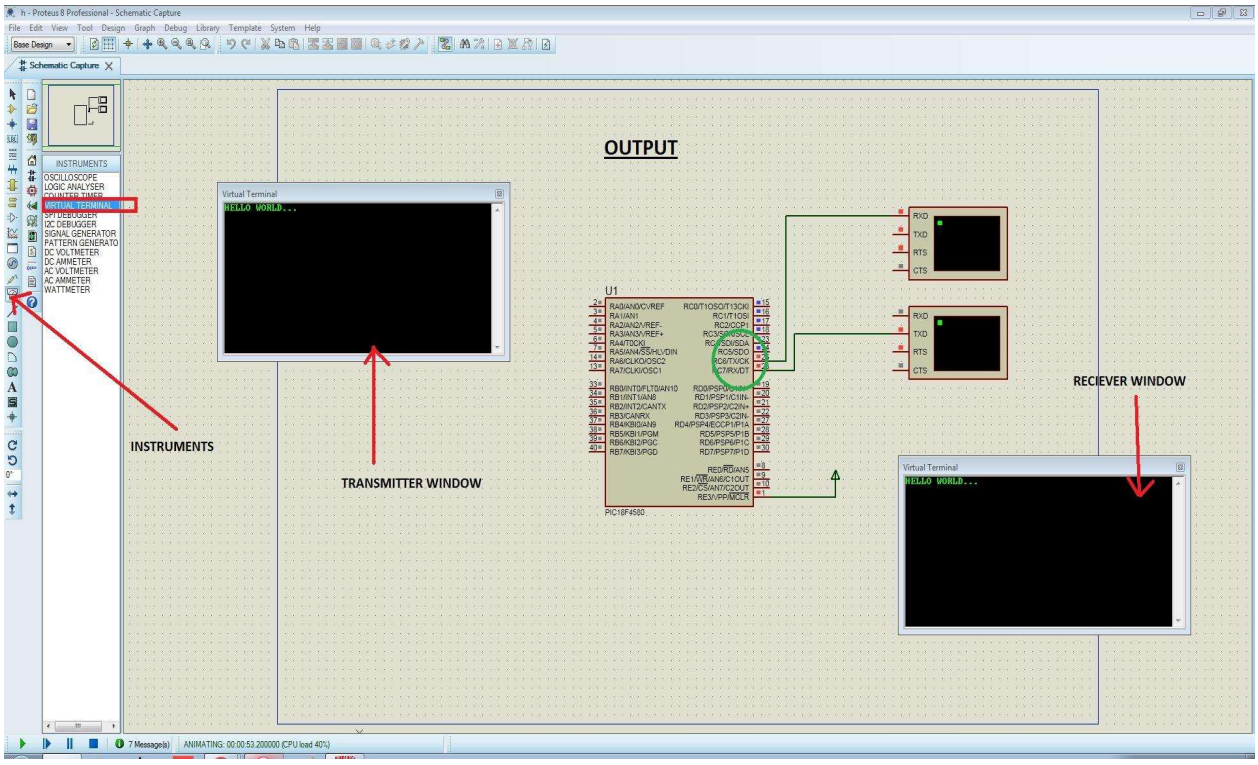
HOW SETUP VIRTUAL TERMINAL IN PROTEUS

Step(i):

Choose the instruments option from the tool menu present on left side of the proteus

Step(ii):

Now Choose the virtual terminal from the list and place it in proteus workspace then connect any of the channel terminal to the outputted port



Program for passing characters between * and #.

#include<pic18.h>

```
void trans(char a);
void display(const char *p);
char rec();
void main()
{
        char h='a';
        TRISC=0X80;
        TXSTA=0x24;
        RCSTA=0X90;
        SPBRG=129;
         Int c;
        while(1)
        {
                h=rec();
                if(h==’#’)
                {
                        c=0;
                }
                if(c==1)
                {
                trans(h);
                }
                if(h==’*’)
                {
                        c=1;
                }
        }
}
void trans(char a)
{
        TXREG=a;
        while(TXIF==0);
```

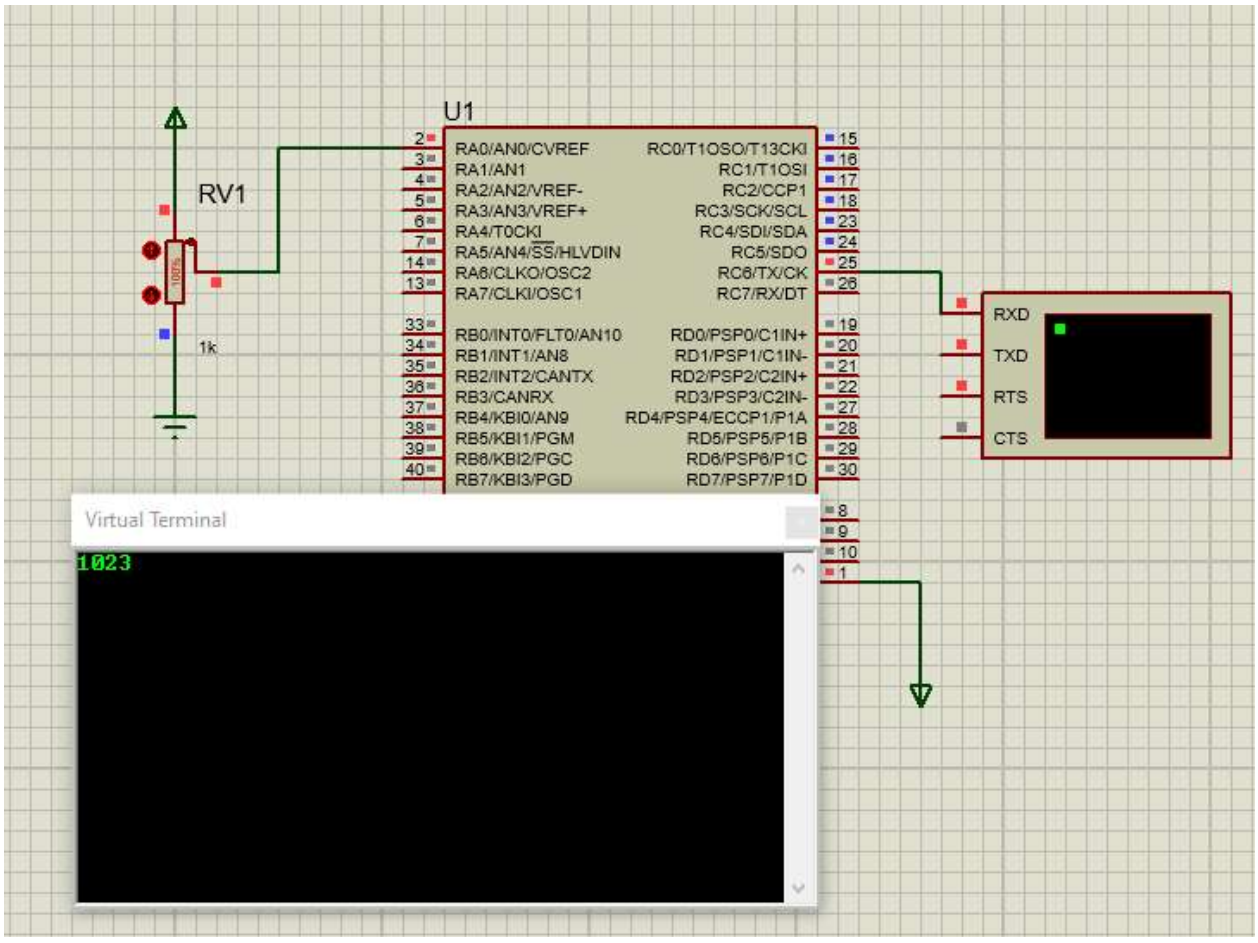
```
        TXIF=0;
}
char rec()
{
        while(RCIF==0);
        RCIF=0;
        return RCREG;
}
void display(const char*p)
{
        while(*p!='\0')
        {
                trans(*p);
                p++;
        }
}
```

USART Transmission only

```
#include<pic18.h>
void trans(char a);

void main()
{
        char h;
        TRISC=0X80;
        TXSTA=0x24;
        RCSTA=0X90;
        SPBRG=129;
    trans('h');
        while(1);

}
```

```
void trans(char a)
{
        TXREG=a;
        while(TXIF==0);
        TXIF=0;
}
```

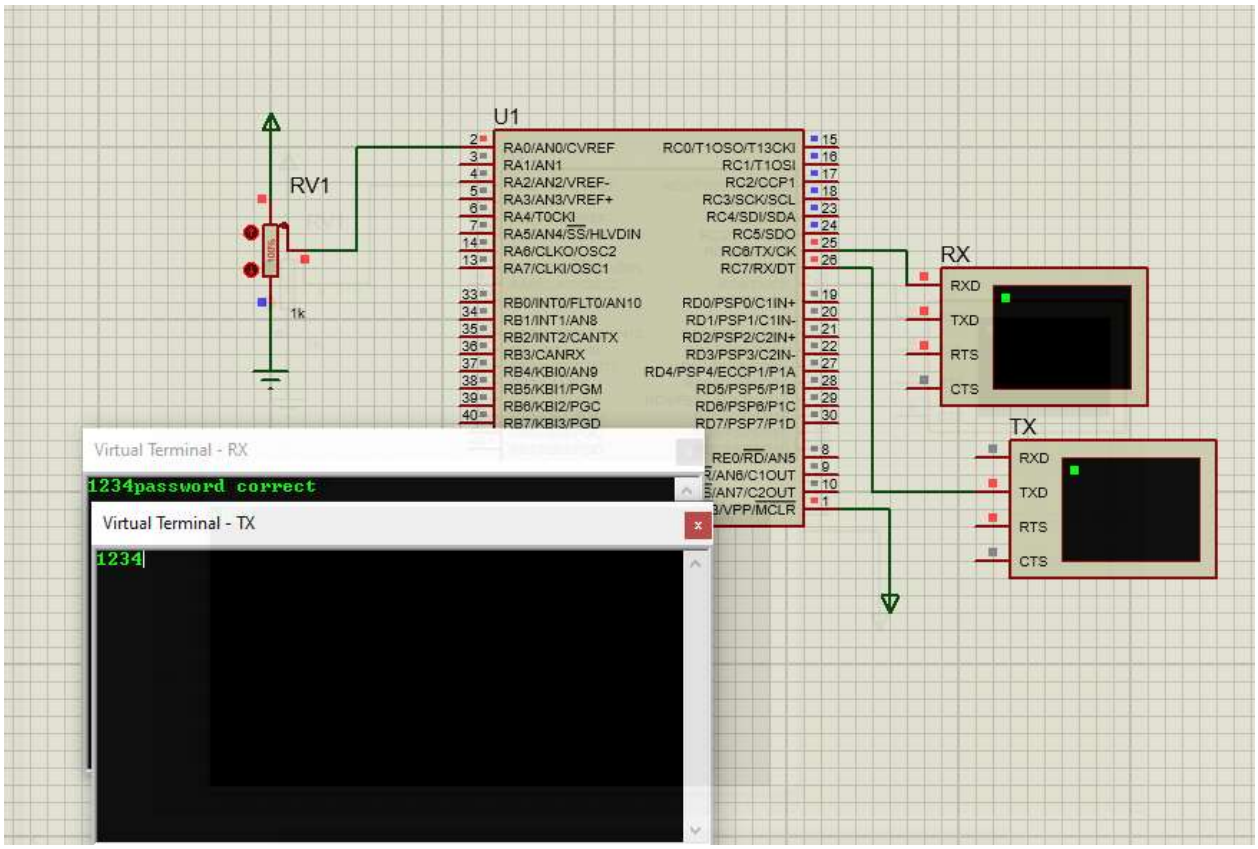
ADC OUTPUT IN UART



```
#include<pic18.h>
void trans(char a)
{
        TXREG=a;
        while(TXIF==0);
        TXIF=0;
}
void main()
{
    ADCON0=0X01;
    ADCON1=0X00;
    ADCON2=0X86;
    TRISA=0XFF;
        TRISC=0X80;
        TXSTA=0x24;
        RCSTA=0X90;
        SPBRG=129;

    int a,b[5],i;
    while(1)
    {
    GODONE=1;
    while(GODONE==1);

        a=ADRESL;
        a+=(ADRESH<<8);
        for(i=0;i<4;i++)
        {
            b[3-i]=(a%10)+48;
            a=a/10;
        }
        for(i=0;i<4;i++)
```

```
    {
        trans(b[i]);

    }
    trans(0x08);
    trans(0x08);
    trans(0x08);
    trans(0x08);

}
}
```

UART PASSWORD



PROGRAM:

```
#include<pic18.h>
int i=0,c=0;
char b[5],k[5]="1234";
char rec()
{
    while(RCIF==0);
    RCIF=0;
    return RCREG;
}
void trans(char a)
{
    while(TXIF==0);
    TXIF=0;
    TXREG=a;
     b[i]=a;
     i++;
}
void display(const char *p)
{
    while(*p)
    {
        trans(*p);
        p++;
    }
}


void main()
{
```

```
    TXSTA=0X24;
  RCSTA=0X90;
  SPBRG=129;
    TRISC=0x80;
    while(1)
    {

    char h=rec();
    trans(h);
          if(i==4)
    {
       for(int j=0;j<4;j++)
       {
          if(k[j]==b[j])
          {
             c++;
          }
       }
       if(c==4)
       {
          display("password correct");
       }
       else
       {
          display("password incorrect");
       }
    }
     }

}
```

PROJECT 6:- PWM

Objective:

The aim is to create a simulation of Pulse Width Modulation with 50% duty cycle.

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include <pic18.h>
void main()

{
    TRISC=0X00;
    TRISB=0X00;
    PORTB=0X1D;
    T2CON=0X7E;
    CCP1CON=0X0F;
    CCPR1L=0X7E;
    while(1);
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

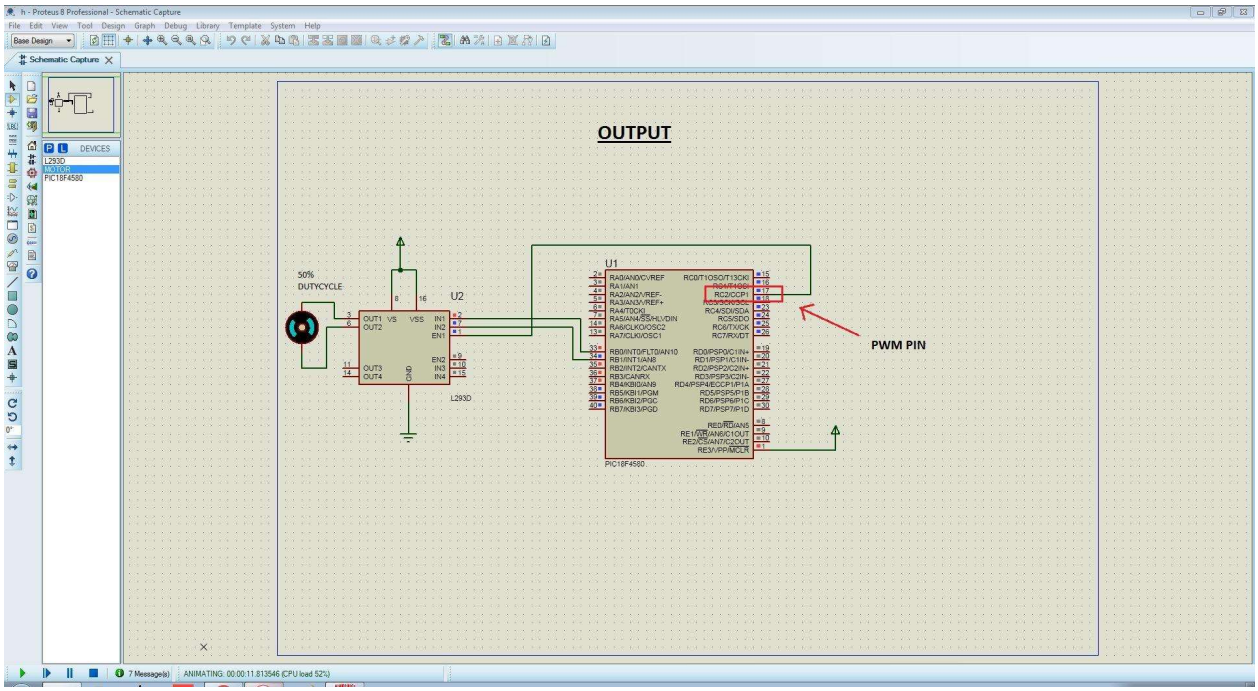
PIC18f4580 microcontroller, Power, Ground, Oscilloscope, Active Motor, l293d.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output simulation of PWM with 50% duty cycle is given below



PROJECT 5:- INTERRUPT

Objective:

The aim is to create a simulation of interrupt

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include<pic18.h>
void delay();
void main()
{
TRISB=0xFF;
TRISD=0x00;
ADCON1=0x0F;
PORTD=0xFF;
GIE=1;
PEIE=1;
INT0IE=1;
   while(1)
  {

  }
}
void interrupt isr()
{
    if(INT0IF==1)
    {
    PORTD=~PORTD;

    INT0IF=0;
    }

}
```

```
void delay()
{
for(int i=0;i<500;i++);
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

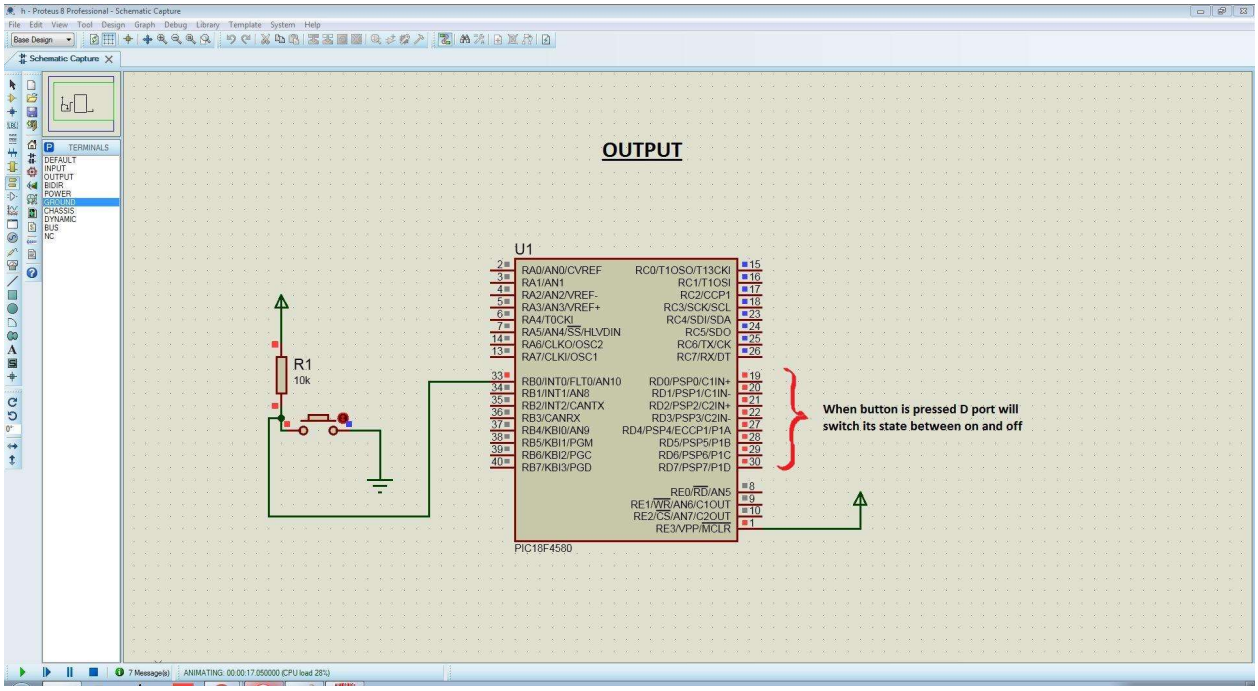
PIC18f4580 microcontroller, Power, Ground.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output simulation of interrupt given below



PROJECT :6

ANALOG TO DIGITAL CONVERTER (8 BIT)

Objective:

The aim is to create a simulation of adc.

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

```
#include<pic18.h>
void delay()
{
   for(int i=0;i<200;i++)
```

```
    {
        for(int j=0;j<200;j++);
    }
}
void data(char a)
{
    PORTD=a;
    RC0=1;
    RC1=1;
    delay();
    RC1=0;

}
void cmd(char a)
{
    PORTD=a;
    RC0=0;
    RC1=1;
    delay();
    RC1=0;
}
void main()
{
    ADCON0=0X01;
    ADCON1=0X00;
    ADCON2=0X06;
    TRISA=0XFF;
    TRISC=0X00;
    TRISD=0X00;
    cmd(0x0F);
    cmd(0x01);
    cmd(0x06);
```

```
    cmd(0x38);
    cmd(0x80);
    int a,b[5],i;
    while(1)
    {
    GODONE=1;
    while(GODONE==1);

        a=ADRESH;
        for(i=0;i<3;i++)
        {
            b[2-i]=(a%10)+48;
            a=a/10;
        }
        for(i=0;i<3;i++)
        {
            data(b[i]);

        }
        cmd(0x01);

}
}
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

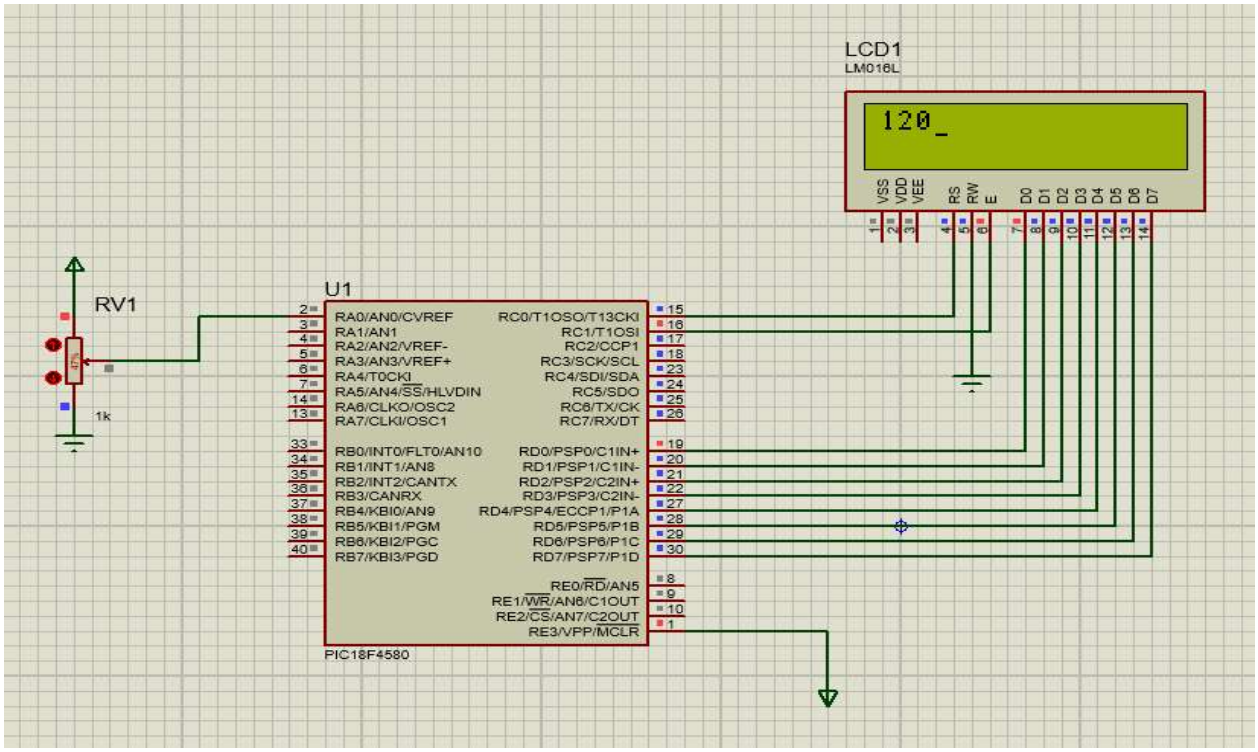
PIC18f4580 microcontroller, Power, Ground, LM016L (lcd), Potentiometer (POT/POT-HG).

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:-

The output simulation of adc(8 bit) is given below



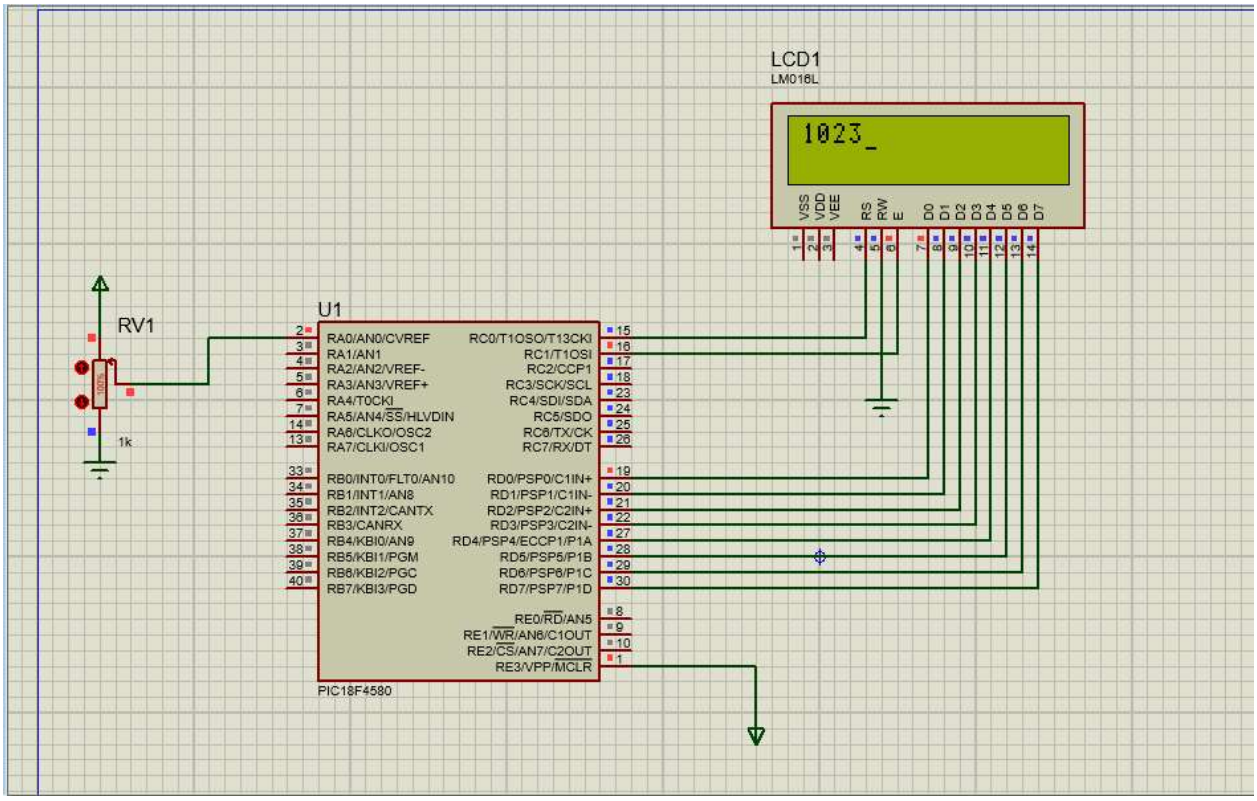
ANALOG TO DIGITAL CONVERTER (10 BIT):

```
#include<pic18.h>
void delay()
{
   for(int i=0;i<200;i++)
   {
      for(int j=0;j<200;j++);
   }
}
void data(char a)
```

```
{
    PORTD=a;
    RC0=1;
    RC1=1;
    delay();
    RC1=0;

}
void cmd(char a)
{
    PORTD=a;
    RC0=0;
    RC1=1;
    delay();
    RC1=0;
}
void main()
{
    ADCON0=0X01;
    ADCON1=0X00;
    ADCON2=0X86;
    TRISA=0XFF;
    TRISC=0X00;
    TRISD=0X00;
    cmd(0x0F);
    cmd(0x01);
    cmd(0x06);
    cmd(0x38);
    cmd(0x80);
    int a,b[5],i;
    while(1)
    {
```

```
    GODONE=1;
    while(GODONE==1);

        a=ADRESL;
        a+=(ADRESH<<8);
        for(i=0;i<4;i++)
        {
            b[3-i]=(a%10)+48;
            a=a/10;
        }
        for(i=0;i<4;i++)
        {
            data(b[i]);

        }
        cmd(0x01);

}
}
```

OUTPUT



PROJECT :7

MATRIX KEYPAD

Objective:

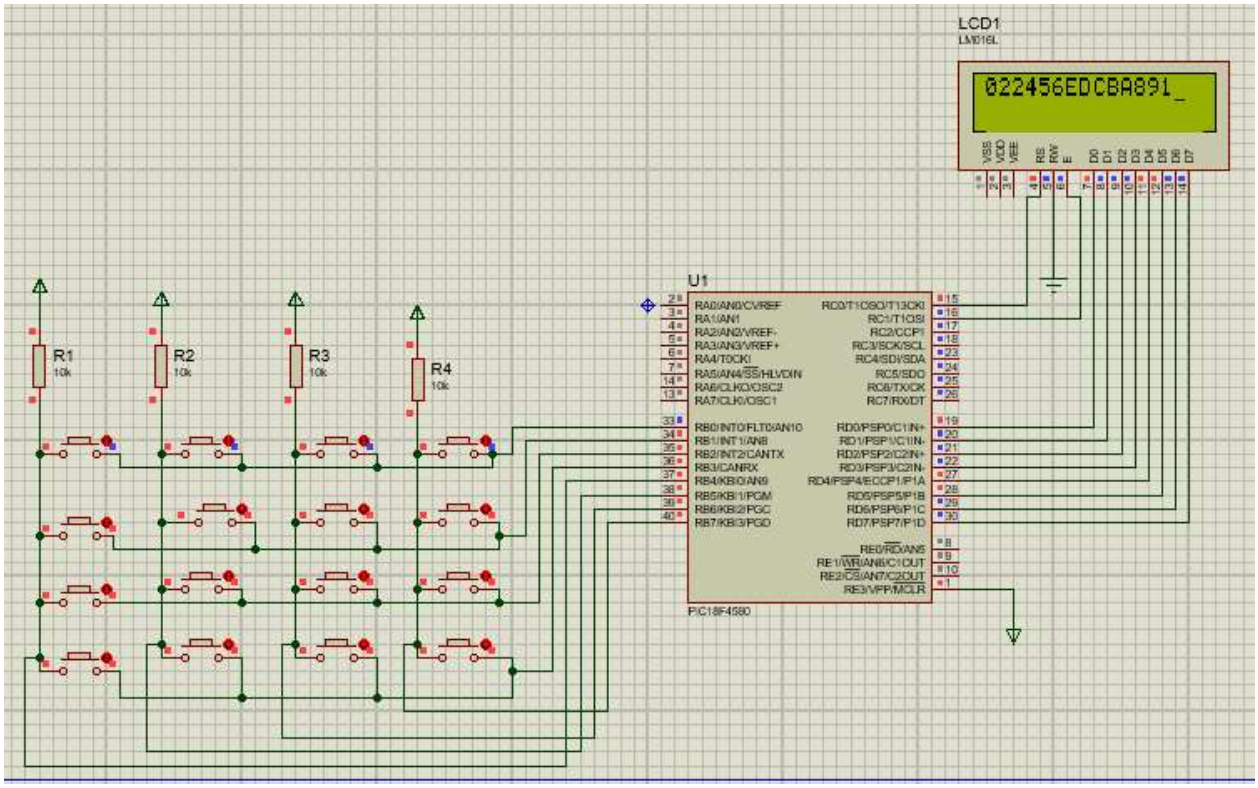
The aim is to create a simulation of matrix keypad

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file




```
#include<pic18.h>
void delay()
{
    for(int i=0;i<10000;i++);
}
void data(char a)
{
    int i;
    RC0=1;
    PORTD=a;
    RC1=1;
    delay();
    RC1=0;
```

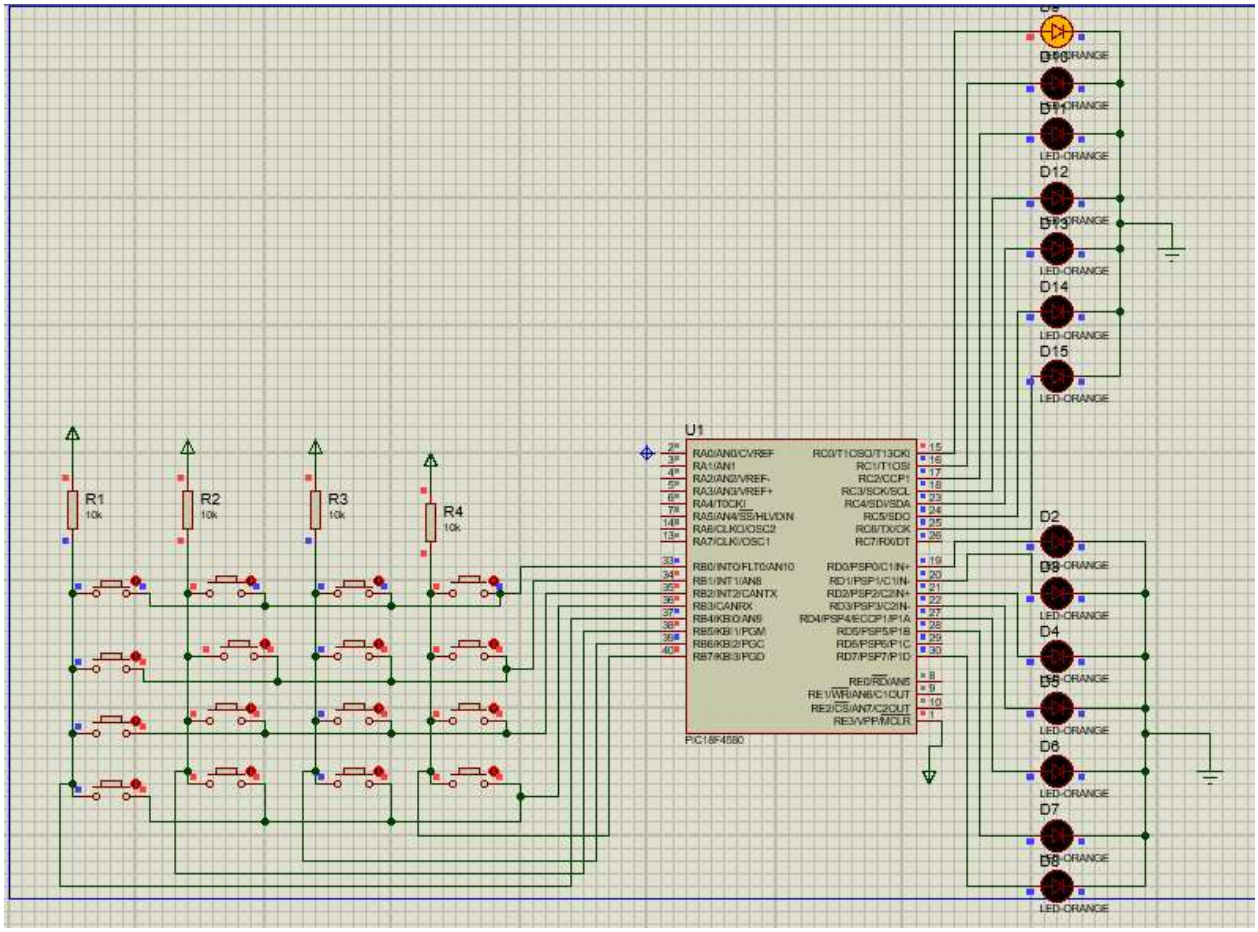
```
      while(RB4==0);
      while(RB5==0);
      while(RB6==0);
      while(RB7==0);
}
void cmd(char a)
{
    int i;
    RC0=0;
    PORTD=a;
    RC1=1;
    delay();
    RC1=0;
}

void main()
{
    int i=0;
    TRISB=0xf0;
    TRISC=0x00;
    TRISD=0x00;
    ADCON1=0X0F;
    cmd(0x01);
    cmd(0x0f);
    cmd(0x06);
    cmd(0x38);
    cmd(0x80);
    while(1)
    {

          PORTB=0x0E;
        if(RB4==0)
        {
```

```
    data('0');

}
if(RB5==0)
{

    data('1');
}
if(RB6==0)
{

    data('2');
}
if(RB7==0)
{

    data('3');
}
PORTB=0x0D;
if(RB4==0)
{

    data('4');
}
if(RB5==0)
{

    data('5');
}
if(RB6==0)
{
```

```
    data('6');
}
if(RB7==0)
{

    data('7');
}
PORTB=0x0B;
if(RB4==0)
{

    data('8');
}
if(RB5==0)
{

    data('9');
}
if(RB6==0)
{

    data('A');
}
if(RB7==0)
{

    data('B');
}
PORTB=0x07;
if(RB4==0)
{

    data('C');
```

```
        }
        if(RB5==0)
        {

            data('D');
        }
        if(RB6==0)
        {

            data('E');
        }
         if(RB7==0)
        {

            data('F');
        }
}
}
```

CONTROLLING LED USING MATRIX KEYPAD




```
#include<pic18.h>
void main()
{
     ADCON1=0X0F;
    int i=0;
    TRISB=0xf0;
    TRISC=0x00;
    TRISD=0X00;


    while(1)
    {
        PORTB=0x0E;
```

```
if(RB4==0)
{

    PORTC=0X01;
    while(RB4==0);
}
if(RB5==0)
{

    PORTC=0X02;
    while(RB5==0);
}
if(RB6==0)
{

    PORTC=0X04;
    while(RB6==0);
}
if(RB7==0)
{

    PORTC=0X08;
    while(RB7==0);
}
PORTB=0x0D;
if(RB4==0)
{

    PORTC=0X10;
    while(RB4==0);
}
if(RB5==0)
{
```

```
    PORTC=0X20;
    while(RB5==0);
}
if(RB6==0)
{

    PORTC=0X40;
    while(RB6==0);
}
if(RB7==0)
{

    PORTB=0X80;
    while(RB7==0);
}
PORTB=0x0B;
if(RB4==0)
{

    PORTD=0X01;
    while(RB4==0);
}
if(RB5==0)
{

    PORTD=0X02;
    while(RB5==0);
}
if(RB6==0)
{

    PORTD=0X04;
```

```
    while(RB6==0);
}
if(RB7==0)
{

    PORTD=0X08;
    while(RB7==0);
}
PORTB=0x07;
if(RB4==0)
{

    PORTD=0X10;
    while(RB4==0);
}
if(RB5==0)
{

    PORTD=0X20;
    while(RB5==0);
}
if(RB6==0)
{

    PORTD=0X40;
    while(RB6==0);
}
 if(RB7==0)
{

    PORTD=0X80;
    while(RB7==0);
}
```

```
        PORTC=0X00;
        PORTD=0X00;
    }
}
```

PASSWORD USING MATRIX KEYPAD

```
#include<pic18.h>
int i=0,c=0;
char b[5],k[5]="1234";
void delay()
{
    for(int i=0;i<10000;i++);
}
void data(char a)
{

    RC0=1;
    PORTD=a;
    RC1=1;
    delay();
    RC1=0;
     while(RB4==0);
     while(RB5==0);
     while(RB6==0);
     while(RB7==0);
     b[i]=a;
     i++;
}
```

```
void display(const char *p)
{
    while(*p)
    {
        data(*p);
        p++;
    }
}
void cmd(char a)
{

    RC0=0;
    PORTD=a;
    RC1=1;
    delay();
    RC1=0;
}

void main()
{

    TRISB=0xf0;
    TRISC=0x00;
    TRISD=0x00;
    ADCON1=0X0F;
    cmd(0x01);
    cmd(0x0f);
    cmd(0x06);
    cmd(0x38);
    cmd(0x80);
    while(1)
    {
```

```
PORTB=0x0E;
if(RB4==0)
{

    data('0');

}
if(RB5==0)
{

    data('1');
}
if(RB6==0)
{

    data('2');
}
if(RB7==0)
{

    data('3');
}
PORTB=0x0D;
if(RB4==0)
{

    data('4');
}
if(RB5==0)
{

    data('5');
}
```

```
if(RB6==0)
{

    data('6');
}
if(RB7==0)
{

    data('7');
}
PORTB=0x0B;
if(RB4==0)
{

    data('8');
}
if(RB5==0)
{

    data('9');
}
if(RB6==0)
{

    data('A');
}
if(RB7==0)
{

    data('B');
}
PORTB=0x07;
if(RB4==0)
```

```
{

    data('C');
}
if(RB5==0)
{

    data('D');
}
if(RB6==0)
{

    data('E');
}
 if(RB7==0)
{

    data('F');
}
if(i==4)
{
    for(int j=0;j<4;j++)
    {
        if(k[j]==b[j])
        {
            c++;
        }
    }
    if(c==4)
    {
        display("password correct");
    }
    else
```

```
                {
                    display("password incorrect");
                }
            }
        }

}
```

PROJECT :8

CAPTURE COMPARE

Objective:

The aim is to create a simulation of capture and compare using CCP module

Softwares Require

MPLAB and PROTEUS.

STEP 1:

Using **MPLAB** compile the following code and create hex file

i) CAPTURE

```
#include<pic18.h>
void main()
{
   ADCON1=0X0F;
   TRISC=0X00;
```

```
    T1CON=0X71;
    CCP1CON=0X02;
    CCPR1L=0X00;
    CCPR1H=0X01;
    TMR1H=0X00;
    while(1)
    {

    }
}
```

STEP 2:

Open Proteus and place the components on the workspace

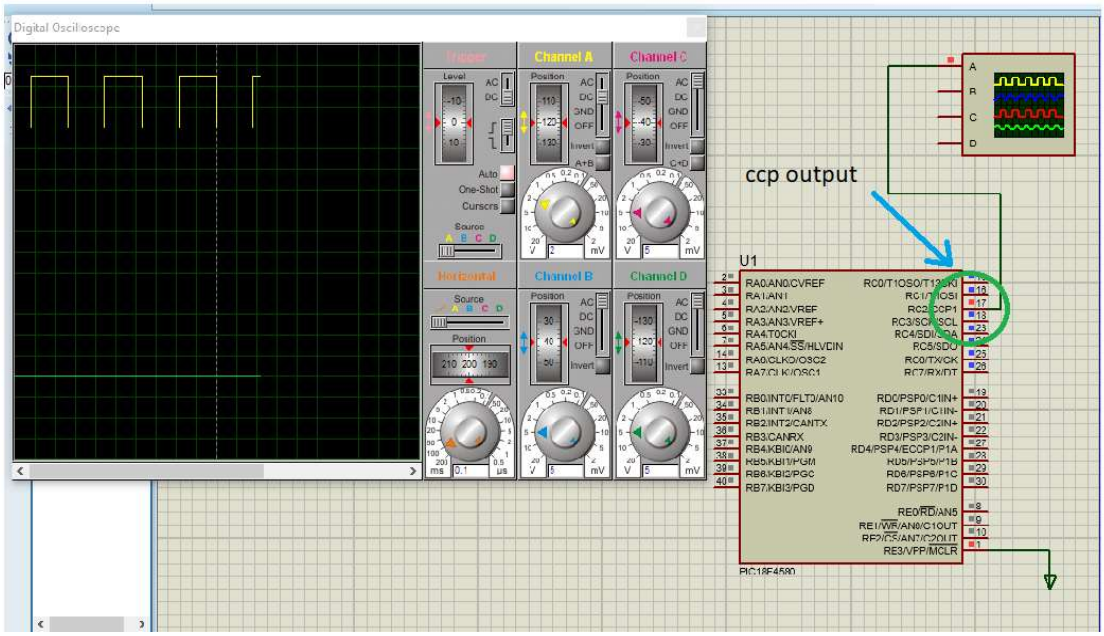
Components Required :-

PIC18f4580 microcontroller, Power, Ground, Oscilloscope.

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:- The output simulation of the compare is given below.



ii) CAPTURE .

```
#include<pic18.h>
void main()
{
        CCP1CON=0X04;
        T1CON=0X71;
        TRISC=0XFF;
        TMR1L=0X00;
        TMR1H=0X00;
        while(1);
        }
```

STEP 2:

Open Proteus and place the components on the workspace

Components Required :-

PIC18f4580 microcontroller, Power, Ground, pull down/up switch

STEP 3:

After placing components load the hex file created into the microcontroller at required clock frequency then simulate

OUTPUT:- The output simulation of the compare is given below.