Experiment Name: DC Motor Speed Control Using PWM and Microcontroller.

PWM: PWM stands for Pulse Width Modulation. A modulation technique that generates variable-width pulses to represent the amplitude of an analog input signal.

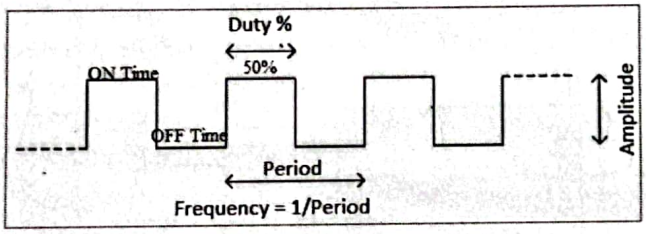
To understand PWM as a type of signal which can be produced from a digital IC such as microcontroller or 555 timer. The signal thus produced will have a train of pulses and these pulses will be in form of a square wave. That is, at any given instance of time the wave will either be high or will be low. For the ease of understanding let us consider a 5V PWM signal, in this case the PWM signal will either be 5V (high) or at ground level 0V (low). The duration at which the signals stays high is called the "*on time*" and the duration at which the signal stays low is called as the "*off time*".

Duty cycle of the PWM:

As told earlier, a PWM signal stays on for a particular time and then stays off for the rest of the period. The percentage of time in which the PWM signal remains HIGH (on time) is called as duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle. The formulae to calculate the duty cycle is shown below.

Duty Cycle =Turn ON time/ (Turn ON time + Turn OFF time)

The following image represents a PWM signal with 50% duty cycle. As you can see, considering an entire **time period** (on time + off time) the PWM signal stays on only for 50% of the time period.



By controlling the Duty cycle from 0% to 100% we can control the "*on time*" of PWM signal and thus the width of signal. Since we can modulate the width of the pulse, it got its iconic name "*Pulse width Modulation*"

Frequency of a PWM:

The frequency of a PWM signal determines how fast a PWM completes one period. One Period is the complete ON and OFF time of a PWM signal as shown in the above figure. The formulae to calculate the Frequency is given below

Frequency = 1/Time Period
Time Period = On time + Off time

Normally the PWM signals generated by microcontroller will be around 500 Hz, such high frequencies will be used in high speed switching devices like inverters or converters. But not all applications require high frequency. For example to control a servo motor we need to produce PWM signals with 50Hz frequency, so the frequency of a PWM signal is also controllable by program for all microcontrollers.

Features of L293D:

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)

- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

The Pin diagram of the L293D is given below:

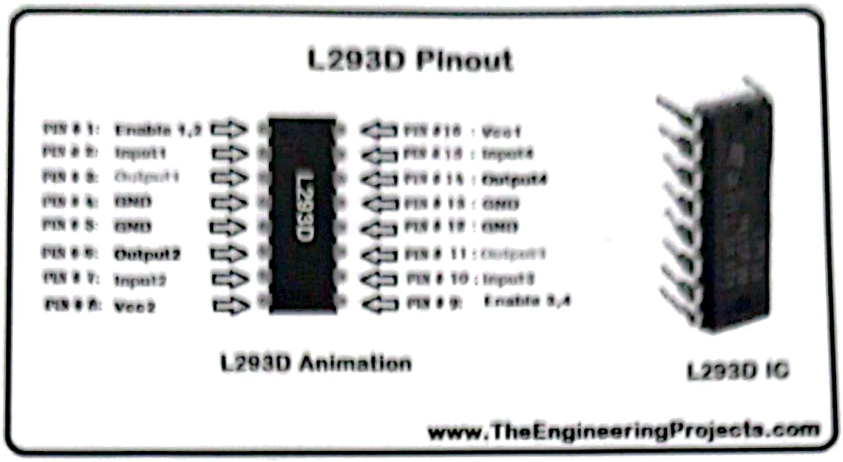


Figure: Pin diagram of L293D

Circuit Diagram:

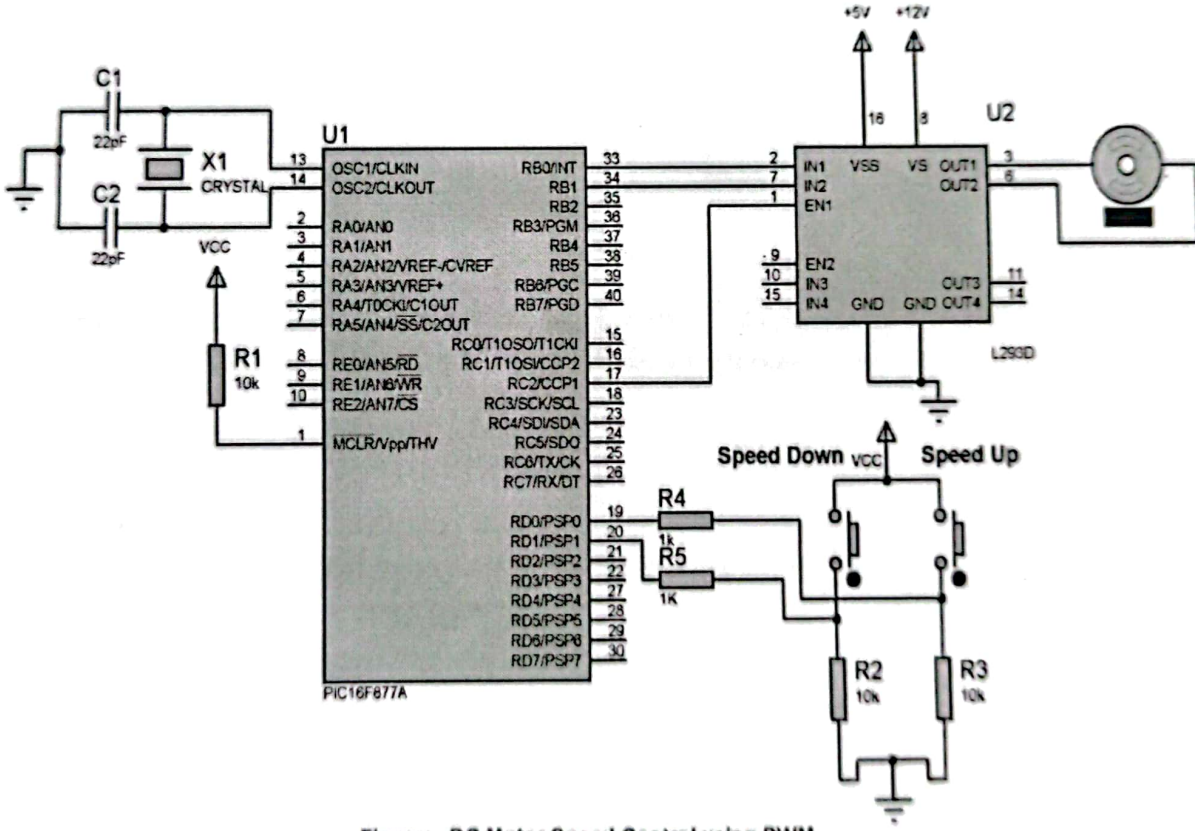


Figure: DC Motor Speed Control using PWM

MikroC Code:

```
void main()
{
 short duty  = 0; //initial value for duty

 TRISD = 0xFF; //PORTD as input
 TRISB = 0x00; //PORTB as output
 //Run motor in anticlock wise
 PORTB.F0=0xff;
 PORTB.F1=0x00;

 PWM1_Init(1000);  //Initialize PWM1
 PWM1_Start();  //start PWM1
 PWM1_Set_Duty(duty); //Set current duty for PWM1

 while (1)       // endless loop
 {
   if (RD0_bit && duty<250) //if button on RD0 pressed
   {
     Delay_ms(100);
     if (RD0_bit && duty<250)
     {
        duty = duty + 10;  //increment current_duty
        PWM1_Set_Duty(duty);  //Change the duty cycle
     }
   }
   if (RD1_bit && duty >0) //button on RD1 pressed
   {
     Delay_ms(100);
     if (RD1_bit && duty >0)
     {
        duty = duty - 10;  //decrement duty
        PWM1_Set_Duty(duty);
     }
   }
   Delay_ms(10);      // slow down change pace a little
  }
 }
```

TL5

Liquid Crystal Display (LCD)

Project Name: Moving Display with PIC Microcontroller and LCD.

Objectives:

I) To understand the Pin configuration & theory of LCD.
II) To interface with an LCD with a PIC microcontroller.
III) Successful Simulation in Proteus & Mikro C Compiler.

Apparatus List:

I) 16×2 Character LCD
II) PIC16F877A Microcontroller
III) Crystal Oscillator
IV) 22 pf capacitor
V) Trimpot 5K/10K
VI) Proteus ISIS, Mikro C, PICKIT2 Burner.

Theory on LCD: 16×2 Character LCD is a very basic LCD module which is commonly used in electronics projects and products. It contains 2 rows that can display 16 characters. Each character is displayed using 5×8 or 5×10 dot matrix. It can be easily interfaced with a microcontroller.

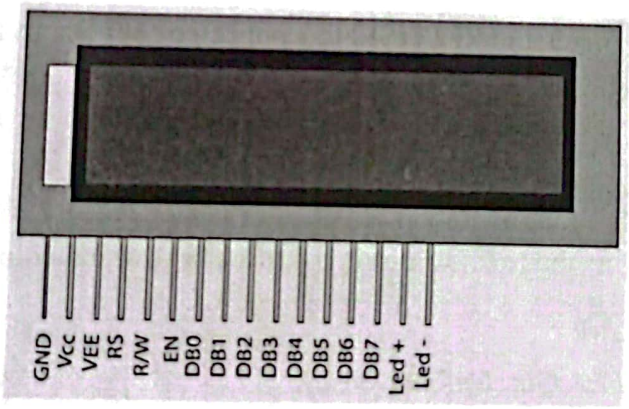


Figure: 16x2 Character LCD Module

This is the pin diagram of a 16×2 Character LCD display. As in all devices it also has two inputs to give power Vcc and GND. Voltage at VEE determines the Contrast of the display. A 10K potentiometer whose fixed ends are connected to Vcc, GND and variable end is connected to VEE can be used to adjust contrast. A microcontroller needs to send two information's to operate this LCD module, Data and Commands. Data represents the ASCII value (8 bits) of the character to be displayed and Command determines the other operations of LCD such as position to be displayed. Data and Commands are send through the same data lines, which are multiplexed using the RS (Register Select) input of LCD. When it is HIGH, LCD takes it as data to be displayed and when it is LOW, LCD

takes it as a command. Data Strobe is given using E (Enable) input of the LCD. When the E (Enable) is HIGH, LCD takes it as valid data or command. The input signal R/W (Read or Write) determines whether data is written to or read from the LCD. In normal cases we need only writing hence it is tied to GROUND in circuits shown below.

Data Transmission Process Theory: A PIC Microcontroller can be easily made to communicate with LCD by using the built in Libraries of MikroC. Interfacing between PIC and LCD can be 4-bit or 8-bit. The difference between 4-bit and 8-bit is how data are send to the LCD. In the 8-bit mode to write an 8-bit character to the LCD module, ASCII data is send through the data lines DB0- DB7 and data strobe is given through the E line.

But 4-bit mode uses only 4 data lines. In this mode the 8-bit ASCII data is divided into 2 parts which are send sequentially through data lines DB4 – DB7 with its own data strobe through the E line. The idea of 4-bit communication is to save as much pins that used to interface with LCD. The 4-bit communication is a bit slower when compared to 8-bit. The speed difference is only minimal, as LCDs are slow speed devices the tiny speed difference between these two modes is not significant. Thus the 4-bit mode data transmission is most commonly used.

Library Functions:

Lcd_Init

Prototype : *void Lcd_Init();*

This function initializes the LCD module connected to the above defined pins of the PIC Microcontroller.

Lcd_Out

Prototype : *void Lcd_Out(char row, char column, char *text);*

This functions prints the text (string) in a particular row and column.

Lcd_Out_Cp

Prototype : *void Lcd_Out_Cp(char *text);*

This function prints the text (string) in the current cursor position. When we write data to LCD Screen, it automatically increments the cursor position.

Lcd_Chr

Prototype : *void Lcd_Chr(char row, char column, char out_char);*

It prints the character (out_char) in the specified row and column of the LCD Screen.

Lcd_Chr_Cp

Prototype : *void Lcd_Chr_Cp(char out_char);*

It prints the character (out_char) in the current cursor position.

Lcd_Cmd

Prototype : *void Lcd_Cmd(char out_char);*

This function is used to send commands to LCD. You can use any one of the following constants as command.

- _LCD_TURN_ON – Turns ON the LCD Display.
- _LCD_TURN_OFF – Turns OFF the LCD Display.
- _LCD_FIRST_ROW – Moves the cursor to the first row.
- _LCD_SECOND_ROW – Moves the cursor to the the second row.
- _LCD_THIRD_ROW – Moves the cursor to the third row.
- _LCD_FOURTH_ROW – Moves the cursor to the fourth row.
- _LCD_CLEAR – Clears the LCD Display.
- _LCD_CURSOR_OFF – Turns ON the cursor.
- _LCD_UNDERLINE_ON – Turns ON the cursor underline.
- _LCD_BLINK_CURSOR_ON – Turns ON the cursor blink.
- _LCD_MOVE_CURSOR_LEFT – Moves cursor LEFT without changing the data.
- _LCD_MOVE_CURSOR_RIGHT – Moves cursor RIGHT without changing the data.
- _LCD_SHIFT_LEFT – Shifts the display left without changing the data in the display RAM.
- _LCD_SHIFT_RIGHT – Shifts the display right without changing the data in the display RAM.
- _LCD_RETURN_HOME – Returns the cursor and shifted display to Home position.

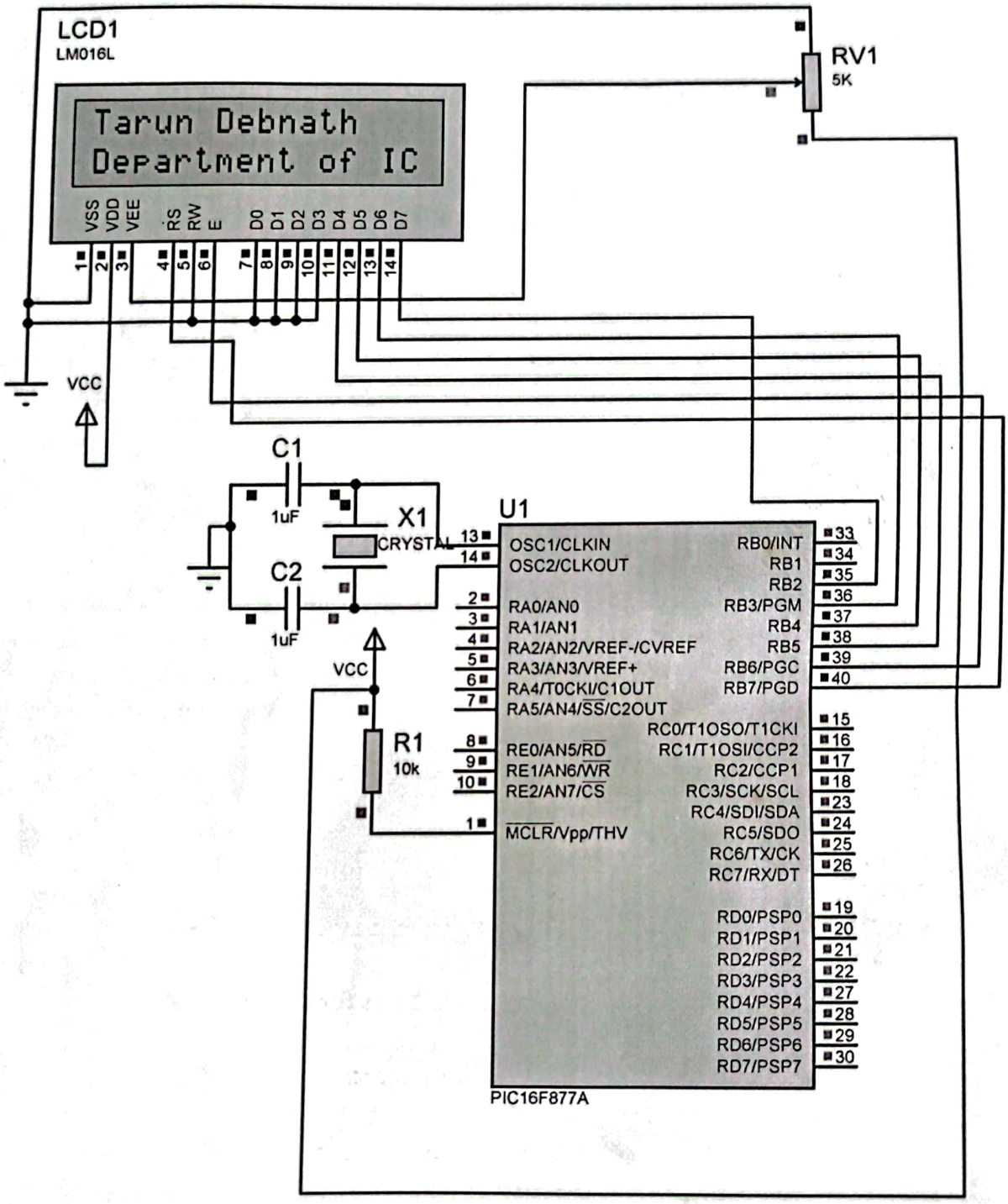
Micro C Code:

```
sbit LCD_RS at RB7_bit;
sbit LCD_EN at RB6_bit;
sbit LCD_D4 at RB5_bit;
sbit LCD_D5 at RB4_bit;
sbit LCD_D6 at RB3_bit;
sbit LCD_D7 at RB2_bit;
sbit LCD_RS_Direction at TRISB7_bit;
sbit LCD_EN_Direction at TRISB6_bit;
sbit LCD_D4_Direction at TRISB5_bit;
sbit LCD_D5_Direction at TRISB4_bit;
sbit LCD_D6_Direction at TRISB3_bit;
sbit LCD_D7_Direction at TRISB2_bit;
// End LCD module connections

void main()
{
  int i;
  char a[]="Tarun Debnath";
  char b[]="Department of ICE";
  Lcd_Init(); // Initialize LCD
 Lcd_Cmd(_LCD_CURSOR_OFF);
 Lcd_Out(1,1,a);               // Write text in first row
 Lcd_Out(2,1,b);               // Write text in second row

 Delay_ms(80);
 while(1) {                    // Endless loop
   for(i=0; i<17; i++) {       // Move text character to the left 7 times
     Lcd_Cmd(_LCD_SHIFT_LEFT);
     Delay_ms(1000);           // use 20ms to 50ms for proteus simulation
   }
   for(i=0; i<17; i++) {       // Move text character to the right 7 times
     Lcd_Cmd(_LCD_SHIFT_RIGHT);
     Delay_ms(1000);           // use 20ms to 50ms for proteus simulation
   }
 }
}
```

Circuit diagram & Simulation in Proteus ISIS:



------X-----