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.

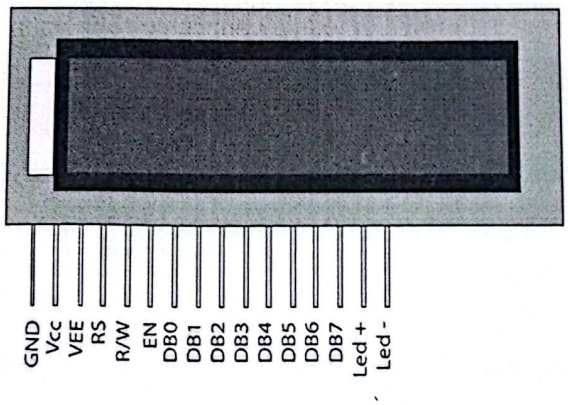


Figure: 16x2 Character LCD Module

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.

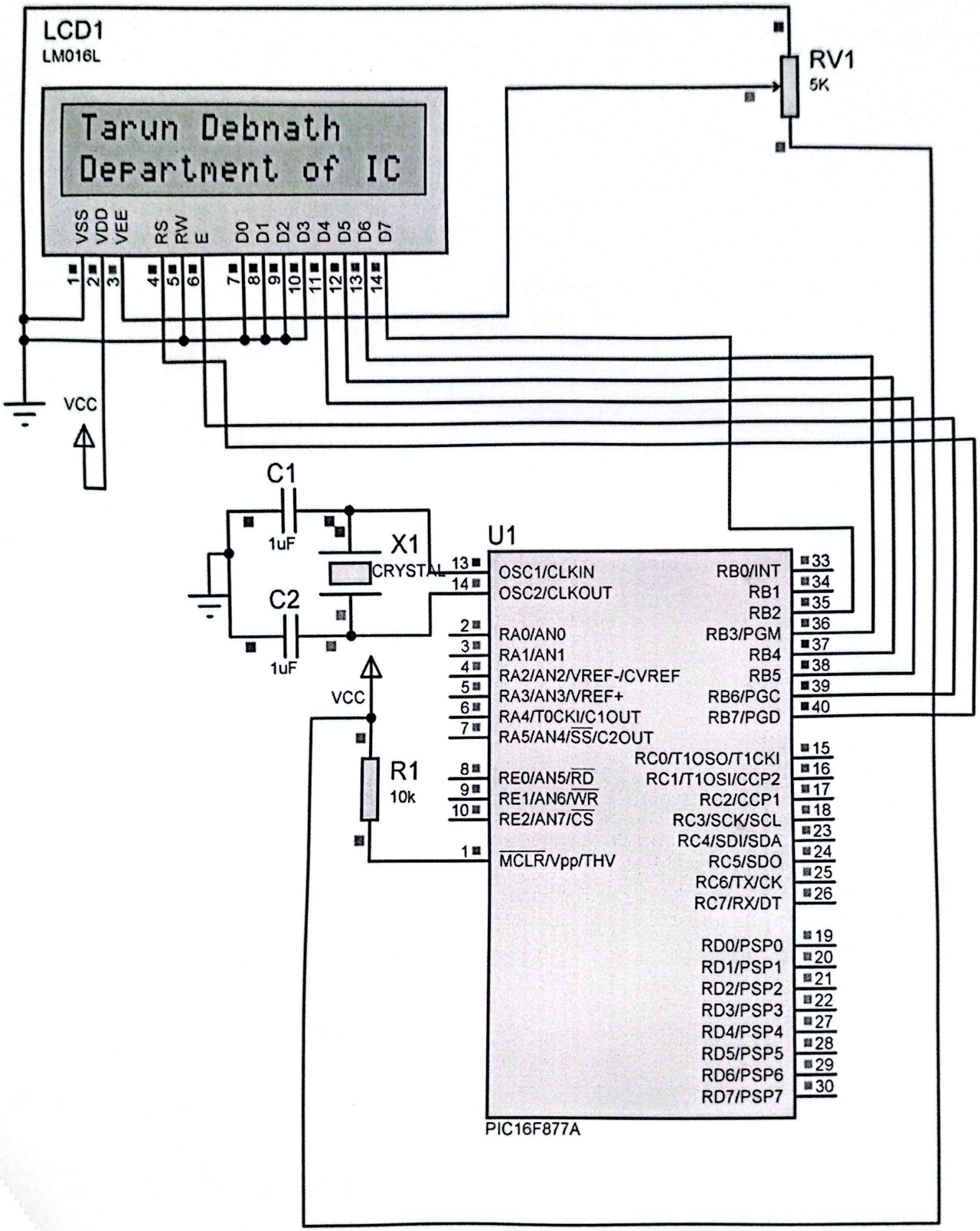
Micro C Code:

```
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-----