Interfacing to Liquid Crystal Display(LCD) module

For microcontroller applications, it may be required to interact with a human being. Examples include displaying messages using a LCD module (known as output device) and entering information using the keypad (known as input device).

One of the most commonly used LCD module is the 16x2 displays that can display 2 lines of 16 characters each. Fortunately, a very popular standard from Hitachi exists that allows the users to communicate with the vast majority of LCDs regardless of the manufacturer.

The standard requires **3 control lines** as well as either **4 or 8 data lines**. The user may select whether the LCD module is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used, the LCD module will require a total of 7-lines (3 control lines plus the 4 data lines). If an 8-bit data bus is used, the LCD module will require a total of 11-lines (3 control lines plus the 8 data lines). The three control lines are referred to as **RS**, **E** and **R/~W**.

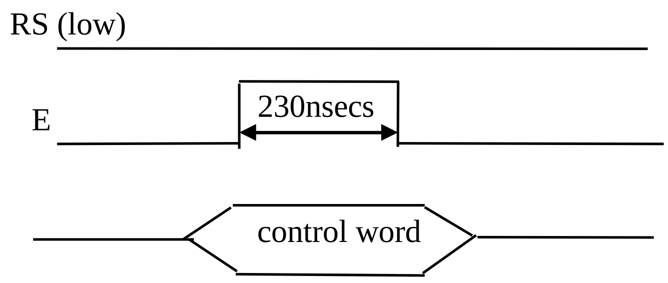
The **RS** line is the "**Register Select**" line.
When **RS** is low (Logic '0'), the data on the data bus is to be treated as a control word (such as clear display, position cursor, etc.). When **RS** is high (Logic '1'), the data on the data bus is text data that should be displayed on the LCD.

The **E** line is the "**Enable**" line.
To send data to the LCD module, your program should first set '**E'** high (Logic '1') and set the other two control lines and put the data on the data bus. When everything is completed or ready, bring '**E'** low (Logic '0'). The 1-0 transition will latch the data on the data bus into the LCD.

The **R/~W** line is the "**Read/Write**" line.
When **R/~W** is low (Logic '0'), the data on the data bus is being written to the LCD. When **R/~W** is high (Logic '1'), the program is effectively reading the LCD.

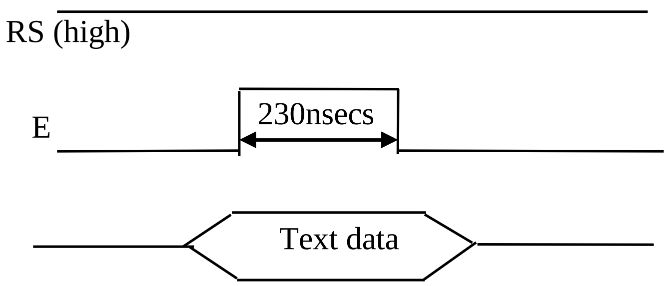
Writing a control word to LCD module

In order to write a control word to LCD we must follow the following timing diagram. **RS** signal must be low and the pulse width for the **E** signal must be at least 230nsecs.



Writing text data to LCD module for displaying

In order to write text data to LCD we must follow the following timing diagram. **RS** signal must be high and the pulse width for the **E** signal must be at least 230nSecs.



For the **target board** in the laboratory, a **16 characters by 2 lines** LCD module is used and each character is made up of a **5 by 7 dots matrix** (5 dots in horizontal and 7 dots in vertical). The LCD module interfaces to the PIC18F4520 microcontroller through a **8-bit data bus** and the **R/~W** line is tied to ground so as to enable LCD for "**Write Operation**" only (**R/~W** = logic '1'). **Appendix A** shows the schematic diagram of the target board.

To display messages on the LCD, you must follow the steps mentioned below:

1. Configure the ports for the microcontroller that interface to the LCD module.
2. Initialise the LCD module by writing the control word.
3. Output text data to the LCD module for display.

Configure the ports

Before you can output data to the LCD module you must first set the respective ports of the microcontroller that are used for the data lines and the control lines, as outputs.

Initialise the LCD

For the LCD used on the target board, we need to

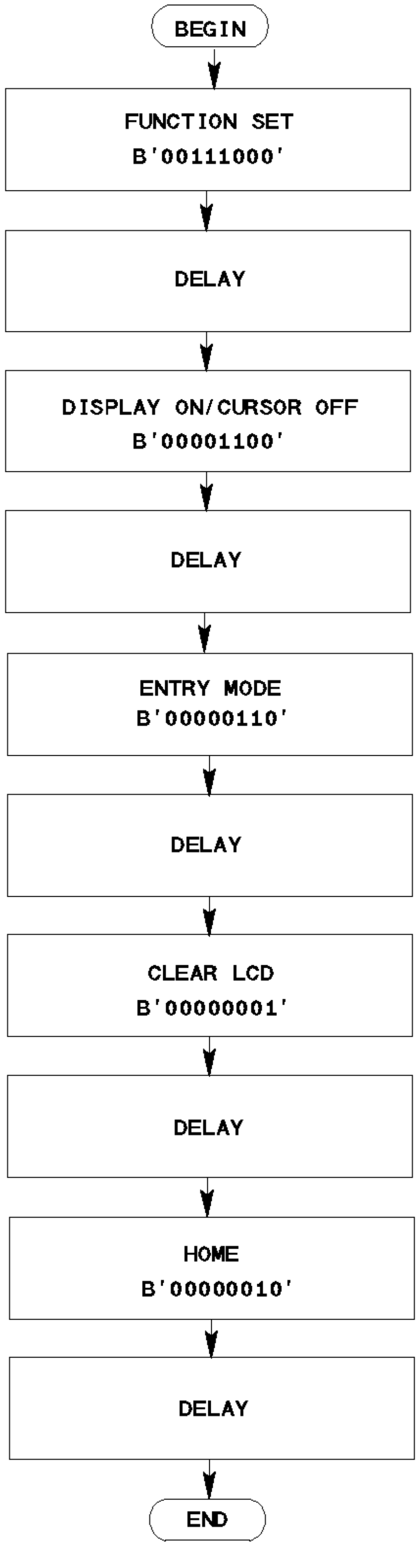
- first set the **"Function Set"** control word to **8-bit** interfacing, **2 lines** and **5 by 7** dots matrix;

- next set the **“Display On/Off”** control word to turn **Display “ON”** ;
- followed by the **“Entry Mode Set”** control word to **increment the position of the cursor** after each character is displayed.
- The last two control words are the **“Home”** and **“Clear”** which **return the cursor to the home position** and **clear the display** of the LCD. Refer to the Table and Flowchart below.

Control word	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**Description**
Function Set	0	0	1	D L	N	F	0	0	DL = 1(8-bit data length) DL = 0(4-bit data length) N = 1(2 lines) N = 0(1 line) F = 1(5 by 10 dots) F = 0(5 by 7 dots)
Display On/Off Control	0	0	0	0	1	D	C	B	D = 1(Display ON) D = 0(Display OFF) C = 1(Cursor Display ON) C = 0(Cursor Display OFF) B = 1(Blink) B = 0(Don’t Blink)
Entry Mode Set	0	0	0	0	0	1	I/D	S	I/D= 1(Increment position) I/D= 0(Decrement position) S = 1(Shift) S = 0(Don’t Shift)
Cursor at Home	0	0	0	0	0	0	1	0	Return Cursor to home position (Address 0)
Clear Display	0	0	0	0	0	0	0	1	Clear all display
Set DD RAM Address	1	A6	A5	A4	A3	A2	A1	A0	Setting the cursor position

Table 1 : Control words for the LCD module.

Refer to the program given in **Appendix B** of this lab. **Identify the program statement that corresponds to each of the steps below and write it beside the boxes.**



Flowchart 1 : For initialising the LCD module.

Setting the cursor position - Using Display Data RAM Address (DD RAM Address)

Refer to the table of control words (**Table 1**). Identify the name of the control word that can be used to set the cursor position ______________________________.

The LCD module contains a certain amount of memory that is assigned for display. All the text data written to the LCD module are stored in this memory before displaying on the LCD screen. This memory can be represented by the following "memory map".

Display	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Line 1	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
Line 2	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F

Memory map for 16x2 LCD module.

In the above memory map, the numbers in each box is the DD RAM Address that corresponds to the LCD screen positions (cursor position). Thus, the first character on the upper left-hand corner of the LCD screen is at address '0x00'. The next character position (character #2 on the first line) is at address '0x01', etc. This continues until we reach the 16^{th} character of the first line which is at address '0x0F'.

However, the first character of line 2, as shown in the memory map above, is at address '0x40'.
If we want to position the cursor on the second line, we need to send a "Set DD RAM Address"control word to the LCD module with the value B'11000000', where the MSB bit is always high (refer to the table for control word) and the remaining bits represent the binary for 0x40.

Exercise:
Modify the program in **LCD.c** to add the following message "MCT LAB" on the **second line** of the LCD. Centre your messages on the LCD screen.

Appendix A

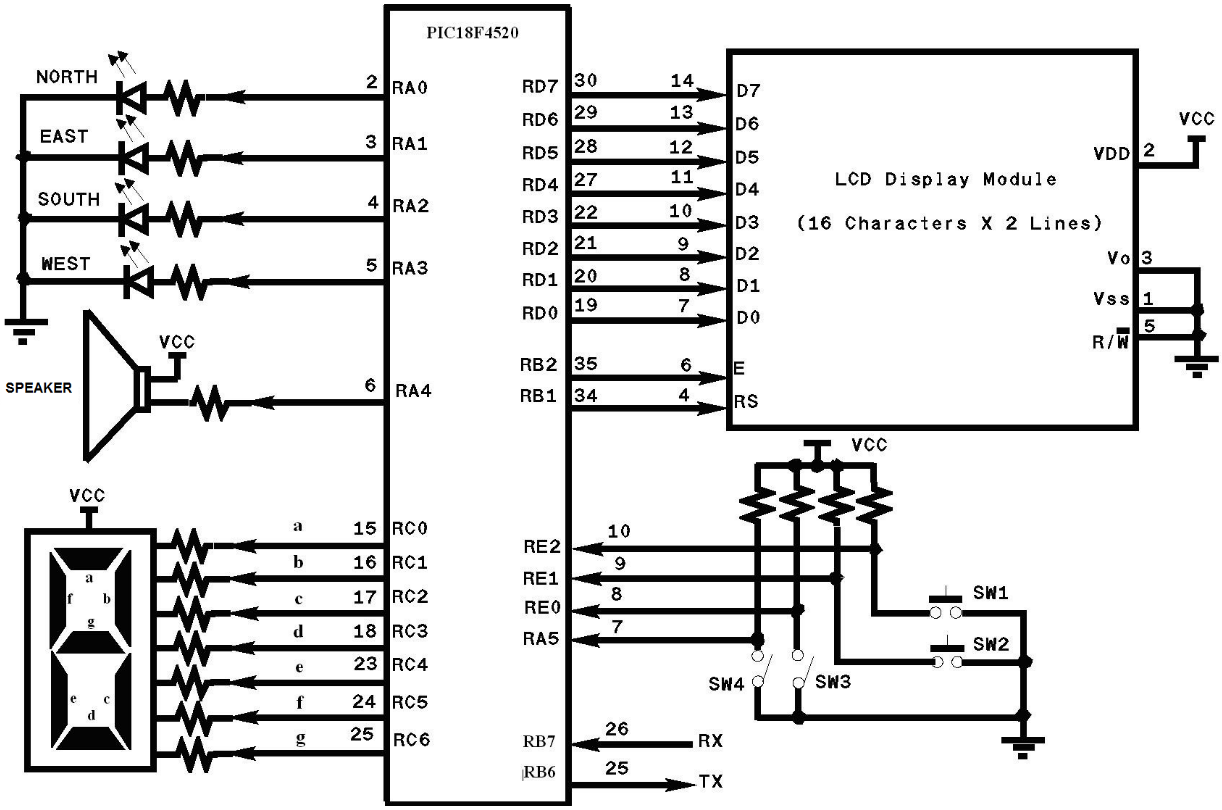


Diagram of the Target Board

Appendix B: Listing of "**LCD.c**" Program

```
//* display "WELCOME" on LCD when start up *
//* The target board uses 2MHz RC oscillator module *

#include <xc.h>
void Init_LCD(void);                    // Initialize the LCD
void LCD_sendCW(char);                  // 8-bit Control word for LCD
void LCD_sendData(char);                // 8-bit Text Data for LCD

#define LCD_DATA  PORTD
#define LCD_RS    PORTBbits.RB1         // RS signal for LCD
#define LCD_E     PORTBbits.RB2         // E signal for LCD

char MESS[] = "WELCOME";

void main()
{
   unsigned char i;
   ADCON1=0x0F;                   // Set ports A,B & E as digital I/O
   TRISB = 0b11111001;            // RB1 & 2 for LCD interface RS & E
   TRISD = 0;                     // Port D as LCD data output
   Init_LCD();                    // Init LCD 8-bit interface,multiple line

   while (1)
   {
      LCD_sendCW(0b00000010);           // Return cursor to home position
      for (i=0; MESS[i]!=0; i++)
         LCD_sendData(MESS[i]);
   }
}

/* LCD display initialization */
void Init_LCD(){
      LCD_sendCW(0b00111000);           // Function Set - 8-bit, 2 lines, 5X7
      LCD_sendCW(0b00001100);           // Display on, cursor on
      LCD_sendCW(0b00000110);           // Entry mode - inc addr, no shift
      LCD_sendCW(0b00000001);           // Clear display
      LCD_sendCW(0b00000010);           // Return cursor to home position
}

/* Write control word to LCD */
void LCD_sendCW(char x){
   LCD_RS   = 0;
   LCD_E    = 1;
   LCD_DATA = x;
   LCD_E    = 0;
   _delay(1000);                        // 2ms delay
}

/* Write text data to LCD */
void LCD_sendData(char x){
   LCD_RS    = 1;
   LCD_E     = 1;
   LCD_DATA  = x;
   LCD_E     = 0;
   _delay(500);                         // 1ms delay
}
```