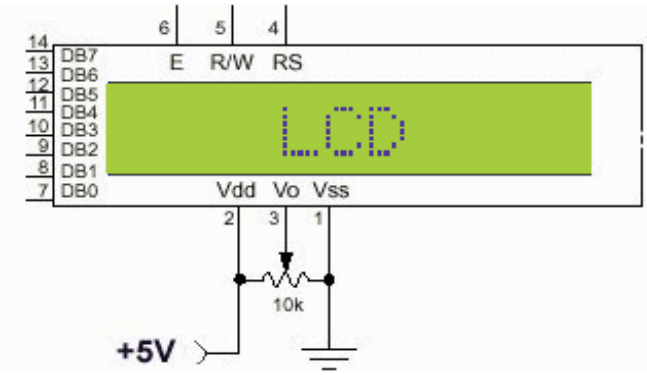
LCD - Liquid Crystal Display tutorial

In this tutorial we will study about the Liquid Crystal Display (LCD). The EDUPIC microcontroller board uses the LCD to display information such as text, messages, and data. We will look at:

- the HD44780U controller registers
- the steps to write character to the LCD display
- the LCD instructions set
- the LCD hardware configuration
- the FREE LCD program written in C language

The Schematics of the LCD:

The LCD display has two lines of characters, 16 characters per line. Each character is composed of matrix of pixels size 5x8. The matrix is controlled by Hitachi HD44780 controller, which performs all the operations that are required to run the matrix. Controller operation is done in accordance with the instructions it receives as described below:



- DB0 – DB7, the 8 data bus lines, which perform read/write of data
- Vss, Vdd – Voltage supply pins
- R/W – Pin writing/reading to/from – LCD
- RS – Pin selects registers between Instruction Register and Data Register
- E – "Enabling" pin; when this pin is set to logical low, the LCD does not care what is happening with R/W, RS, and the data bus lines; when this pin is set to logical high, the – LCD is processing the incoming data
- Vo – Pin for LCD contrast

LCD registers

The HD44780U controller has two 8-bit registers:

- *an instruction register (IR)* – the IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator RAM (CGRAM).
- *a data register (DR)* – the DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to be read from DDRAM or CGRAM. The DR is also used for data storage when reading data from DDRAM or CGRAM.

The choice between the two registers is made by the register selector (RS) signal as detailed the following table:

Register Selector		
RS	R/W	
0	0	Sends a command to LCD
0	1	Read busy flag (DB7) and address counter (DB0 to DB6)
1	0	Sends information to LCD
1	1	Reads information from LCD

Busy Flag (BF)
BF gives an indication whether the LCD is finished the previous instruction and ready with the next.

DDRAM Memory (Display Data RAM)
Display data RAM (DDRAM) stores the information we send to LCD in ASCII Code. For each letter there is a special code that represents it: for example, the letter A in ASCII code, "receives" a value of 65 in base 10 or 01000001 in binary base, or 41 in the base 16. The memory can contain up to 80 letters.

Some of the addresses represent the lines of LCD (0x00-0x0F- first line; 0x40-0x4F - second line). The rest of the addresses represent the "non-visible" memory of the DRAM, which can be also used as a general memory. The DDRAM address is the position of the cursor on the display LCD (the received information will be written at the place where the cursor is).

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

CGRAM Memory (Character Generator RAM)
Using CGRAM memory the user can "build" and store their own letters. For 5x8 dots, eight character patterns can be written, and for 5x10 dots, four character patterns can be written.The difference between the memories is that the DDRAM memory displays on the screen the "ready" characters in accordance with the ASCII code, while the CGRAM memory displays the special characters that the user has created.

Please CLICK here to access the LCD HD44780U data sheet for more information (279kb)

Address Counter (AC)
The address counter (AC) assigns addresses to both DDRAM and CGRAM. When an address of an instruction is written into the IR, the address information is sent from the IR to the AC. Selection of either DDRAM or CGRAM is also determined concurrently by the instruction. After writing into (reading from) DDRAM or CGRAM, the AC is automatically incremented by 1(decremented by 1). The AC contents are then output to DB0 to DB6 when RS = 0 and R/W = 1.

Writing a letter/character to the LCD display

To write a letter/character on the LCD display we have to do the following:

1. Perform an initialization.
2. Send the desired position to IR (DDRAM Address).
3. Send ASCII code of the letter to DR.

LCD display will show the letter that matches the code that was sent and the address counter AC will be updated (increment or decrement, depending on how it was initialized). You can write strings by sending characters in sequence.

LCD instruction set

The LCD instruction set consists of the commands you can send to LCD. Remember that the RS line needs to be set to zero to send instruction to the LCD. When the RS line is set to one, you are sending data to display memory or the character graphics (CG) memory. An "X" in any position means it does not matter what you enter there.

Clear Display:
This command clears the display and returns the cursor to the home position (address 0) and sets I/D to 1 in order to increment the cursor. Its line settings are as follows:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1

Home Cursor:
This returns the cursor to the home position, returns a shifted display to the correct position, and sets the display data (DD) RAM address to 0. Its line settings are as follows:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	X

Entry Mode Set:
This command sets the cursor move direction and specifies whether toshift the display or not. These operations are performed during the data write/read of the CG or DD RAM. Its line settings are as follows:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0

0	0	0	0	0	0	0	1	I/D	S

I/D=0 means the cursor position is decremented (moves right to left).
I/D=1 means the cursor position is incremented (moves left to right).
S=0 means normal operation, the display remains still, and the cursor moves.
S=1 means the display moves with the cursor.

Display On/Off Control:
This command sets the ON/OFF display as well as the cursor and blinking capabilities (0 equals OFF; 1 equals ON). D controls whether the display is ON or OFF, C controls whether the cursor is ON or OFF, B controls whether the blinking is ON or OFF. The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	C	B

Cursor or Display Shift:
This moves the cursor and shifts the display without changing DD RAM contents. The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	S/C	R/L	X	X

S/C=0 means move the cursor.
S/C=1 means shift display.
R/L= 0 means shift to the left.
R/L= 1 means shift to the right.

Function Set:
This sets the interface data length (DL), the number of display lines (N), and character font (F). The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	DL	N	F	X	X

DL=0 means 4 bits are being used (the standard)
DL=1 means a full 8 bits being utilized
N=0 means 1 line
N=1 means 2 lines or more
F=0 means that 5x7 dot characters are used (which is how 99% of all LCDs are set up)
F=1 means 5x10 dot characters are used

Set CG RAM Address:
This command sets the custom graphics (CG) RAM address. Setting RS to 1 sends data to CG RAM instead of the DD RAM. Eight CG characters are available, and they reside in the ASCII codes 0 through 7. The is sent in the 8-bit bytesfrom the top row to the bottom row and is left justified, meaning that only the bottom 5 bits matter (it is a 5x7 dot matrix). The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	MSB	CG	RAM	ADDRESS		LSB

Set DD RAM Address:
This sets the DD RAM address. Setting RS to 1 sends data to the display RAM, and the cursor advances in the direction where the I/D bit was set to. The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	MSB DD RAM ADDRESS LSB						

Read Busy Flag and Address:
This reads the busy flag (BF). If BF equals to 1, the LCD is busy and displays the location of the cursor. With the R/W line grounded, this command can not be used. The line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	0	0	0	0	0	0	1

Write Data to CG or DD RAM:
This command's line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	MSB ASCII code or CG bit pattern data							LSB

Read Data from CG or DD RAM:
This command's line settings are as follows:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	1	MSB ASCII code or CG bit pattern data							LSB

The LCD Initializing sequence:

The initializing sequence includes the steps that need to be executed in order for the LCD to work. In fact, these sequences of steps define in what form we want the LCD to work: the data length (8 bit or 4 bit); size of the letters; activation of the cursor; and more. When you send out an instruction (command or information) to the LCD it takes some time to execute it, so it's

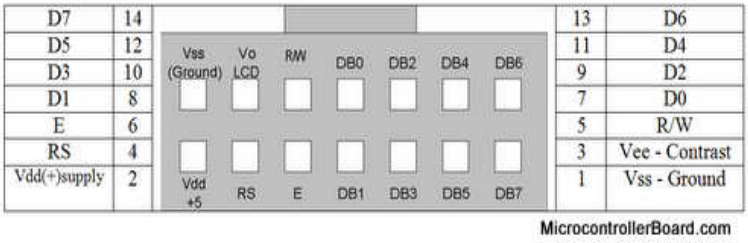
important to make sure that the LCD is "ready" for the next instruction/operation.

You can check if the LCD is ready in the following 2 ways:

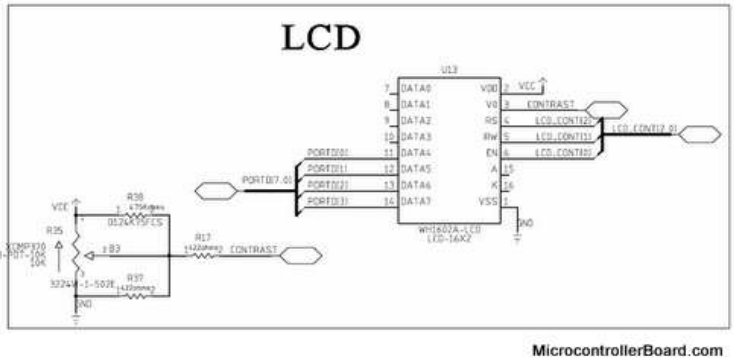
1. Create a delay subroutine to accommodate the minimum execution time.
2. Scanning BF (busy flag) bit - this bit gives an indication whether the LCD is finished working.

LCD hardware configuration

The traditional LCD connection is via a 14-pin dual in-line connector that works nicely with a 14-pin ribbon cable connector as show in the figure below:



Even though the cable pin out consists of 8 data lines (DB0-DB7), traditionally everyone uses the LCD in 4-bit mode to save on data lines and control signal lines. The following figure shows the LCD connection as it used with EduPIC development board.



We used 4 consecutive bits (PORTD0-PORTD3) in configurable nibble as the data lines. In addition, we used PORTE0-PORTE2 for the RS, EN and RW control signals lines.

Note: Typically, the LCD is used as an output-only device. If you want to keep track of the location of the cursor or what is in the special character buffer in your program, you need to tie the Read/NOT Write line to ground.

LCD interface program

This code will interface to a standard LCD controller like the Hitachi HD44780. It uses it in 4 bit mode. The LCD program is written is C language, and will display expression "MicrocontrollerBoard.com".
Here's a table with an explanation of functions:

Function	Explanation
Init_lcd()	Initializing the LCD to work in the 4-bit
Write_com_lcd()	Sends a command to LCD
Write_char_lcd()	Sends a letter to LCD
Write_string_lcd()	Sends a string to LCD
Clear_lcd()	Clears LCD
Goto_lcd()	Moves the cursor to the specified address

Download here the FREE LCD program written in C language

```
/ *
* This code will interface to a standard LCD controller
* like the Hitachi HD44780. It uses it in 4 bit mode, with
* the hardware connected as follows (the standard 14 pin
* LCD connector is used):
*
* PORTD bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
* PORTE bit 0 is connected to the LCD RS input (register select)
* PORTE bit 2 is connected to the LCD EN bit (enable)
* PORTE bit 1 is connected to the LCD RW bit (read or write)
* * To use these routines, set up the port I / O (TRISE, TRISD) then
* call Init_lcd (), then other routines as required.
*
```

```
* /
#include <pic.h>
static bit LCD_RS @ ((unsigned)&PORTE*8+0); // Register select
static bit LCD_EN @ ((unsigned)&PORTE*8+2); // Enable
static bit LCD_RW @ ((unsigned)&PORTE*8+1); // R/W
#define EN_TRIG ((LCD_EN = 1),(LCD_EN = 0));

//Functions

void DelayMs (int x);
void DelayUs (int x);
void Init_lcd (void);
void Write_com_lcd (unsigned char c);
void Write_char_lcd (unsigned char c);
void Write_string_lcd (const char * s);
void Clear_lcd (void);
void Goto_lcd (unsigned char pos);

void main ()
{
ADCON1 = 0x0e; // Set PORTA as digital port
TRISA = 0; // Set PORTA as output
TRISE = 0; // Set PORTE as output
TRISD = 0; // Set PORTD as output
Init_lcd(); // LCD Initialization
Clear_lcd(); // Reset LCD and move to start position
Write_string_lcd("MicrocontrollerBoard.com");
while(1);
}

/*-----------------------------------------------*/
void Init_lcd (void)
{
LCD_RW = 0; // write to LCD
LCD_RS = 0; // write control bytes
DelayMs(15); // power on delay
PORTD = 0x3; // attention!
EN_TRIG;
DelayMs(5);
EN_TRIG;
DelayUs(100);
EN_TRIG;
DelayMs(5);
PORTD = 0x2; // set 4 bit mode
EN_TRIG;
DelayUs(40);
Write_com_lcd(0x28); // 4 bit mode, 1/16 duty, 5x8 font
Write_com_lcd(0x08); // display off
Write_com_lcd(0x0F); // display on, blink curson on
Write_com_lcd(0x06); // entry mode

}

void DelayMs(int x)

{
int y=(x*1000)/15;
while(--y != 0)
continue;
}

void DelayUs(int x)

{
int y = x/15;
while(--y != 0)
continue;

}

void Write_com_lcd(unsigned char c)// send command to lcd

{
LCD_RW = 0;
LCD_RS = 0; // write the command
PORTD = (PORTD & 0xF0) | (c >> 4);
EN_TRIG;
PORTD = (PORTD & 0xF0) | (c & 0x0F);
EN_TRIG;
DelayUs(40);
}

void Write_char_lcd(unsigned char c)

{
LCD_RW = 0;
LCD_RS = 1; // write characters
PORTD = (PORTD & 0xF0) | (c >> 4);
EN_TRIG;
```

```
PORTD = (PORTD & 0xF0) | (c & 0x0F);
EN_TRIG;
DelayUs(40);
}

void Write_string_lcd(const char * s)

{
LCD_RS = 1; // write characters
while(*s)
Write_char_lcd(*s++);
}

void Clear_lcd(void)

{
LCD_RS = 0;
Write_com_lcd(0x1);
DelayMs(2);
}

void Goto_lcd(unsigned char pos)
{
LCD_RS = 0;
Write_com_lcd(0x80+pos);
}
```

Back to PIC microcontroller tutorial

Back to Development tools

Back to MicrocontrollerBoard.com Store
