EMBEDDED HARDWARE DESIGN

Experiment3: Interfacing LCD with Microcontroller

LCD (Liquid Crystal Display) has become very popular option for displaying in Embedded Applications. Since they are very cheap and easy to interface with microcontrollers, they are widely found in devices like telephones, vending machines, washing machines, toys etc. LCD comes in several varieties i.e. 16*2, 20*2, 20*4 etc. These different LCD varieties can display different number of characters i.e. 16*2 can display 32 characters at a time.

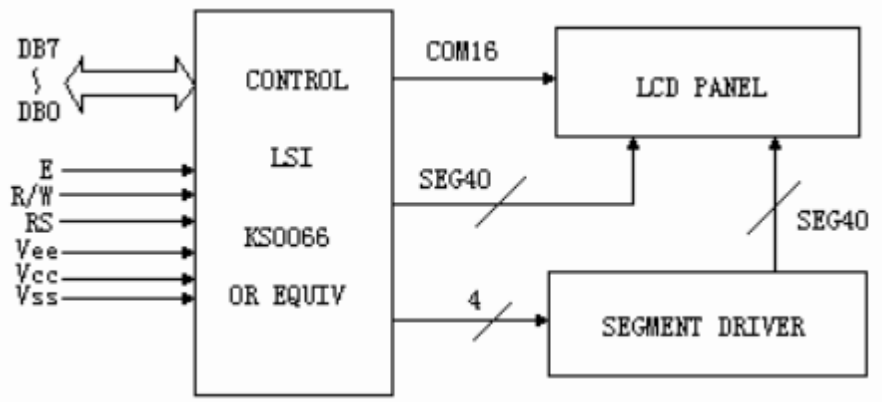
Introduction

The LCD module that we will be using is 16*2 JHD162A. This model has 2 lines and 16 rows of display blocks. Each block can be used to display 1 character. So there are total 32 such blocks. One block has 8*5 pixels. Depending on which pixel is ON and which is OFF we can display several Alpha-Numeric characters. This model also has a green backlight, which helps us to see the display even in dark. In reality this module consists of a controller chip, a segment driver chip, LCD display and some passive components. There are total 16 pins in the LCD module.

Pin Number	Pin Name	Pin Function	Connection to ATMega16
1	Vss	GND Supply	
2	Vcc	+5V LCD Power	
3	Vee	Contrast Adjustment	
4	RS	Register Signal Select(Command/Data)	PC7
5	R/W	Read/Write Select	PC6
6	E	Enable (read/write enable)	PC5
7	DB0	Data pin	-
8	DB1	Data pin	-
9	DB2	Data pin	-
10	DB3	Data pin	-
11	DB4	Data pin	PC0
12	DB5	Data pin	PC1
13	DB6	Data pin	PC2
14	DB7	Data pin	PC3
15	LED+	Backlight Power +5V	
16	LED-	Backlight Power GND	

These connections have already been made to the USB BootLoader Kit so when you insert the LCD module properly, the connections are made as mentioned in the Table.

While using LCD, we can think a simple analogy for its operation. Each of the 32 blocks is a memory, as soon as we write an ASCII number into one of these 32 memory locations the corresponding character is displayed on that block. The function of displaying the character after decoding the data is done by an onboard controller chip.



Block Diagram of LCD Module

In reality the LCD module consists of several memory locations apart from the 32 dedicated for the display. Since LCD module can be used in several ways we need to select one of these modes and configure the LCD for this mode. The various operations that we need to perform are:-

1. Write to LCD memory to display data
2. Configure LCD module by writing commands to memory location
3. Read data in Memory locations
4. Perform some special operations like Clear Screen , Bring Cursor to starting position etc

The various pins that are provided like RS, R/W etc acts as a way to do all these operations.The 3 control pins have important functions

1. **RS** – The LCD has 2 modes of operation Character and Instruction. Depending on its value the data on the data line (DB0-DB7) is treated as either command or character. If RS is LOW then the data on DB0-DB7 is an Instruction and if RS is HIGH then data on data line is character
2. **Enable** – The pin acts like a clock for the LCD module, when there is a falling clock edge then the data on the data lines is taken in by the LCD module and processed.
3. **R/W** - This pin identifies if the operation to be performed is a read or write.

If R/W = 0 write operation R/W = 1 read operation RS = 0 Command RS = 1 Character E -> falling edge , data processed by LCD Module
To be able to correctly perform an operation we have to assign the values of these 3 control pins and data lines in a proper sequence.

Let's look at one example i.e. displaying a character by writing to its memory location.
The steps are:-

1. Set the data on the data lines i.e. the 8 bits of data are assigned to each of the 8 pins (DB0-DB7) like LSB -> DB0 and MSB ->DB7.
2. Set RS pin i.e. make RS pin HIGH.
3. Clear RW pin i.e. make RW pin LOW.
4. Toggle E i.e. make E high, wait for some time, make E low.

Another example, if we want to write a command, the steps are:-

1. Set command on the data lines i.e. the 8 bits of data are assigned to each of the 8 pins (DB0-DB7) like LSB -> DB0 and MSB ->DB7.
2. Clear RS pin i.e. make RS pin LOW.
3. Clear RW pin i.e. make RW pin LOW.
4. Toggle E pin i.e. make E high, wait for some time, make E low.

A very important point to note is that the sequence of these steps can be done manually using toggle switches only and the microcontroller is only used because the repetitive steps need not be performed manually again and again.

In the above examples we used 8 data pins and 3 pins for control signal i.e. total 11 pins of the microcontrollers are used. An efficient way is to use 4 data pins instead of 8. In case we are using 4 data lines , the higher nibble is send on these 4 data lines first and then we send the lower nibble on the 4 data lines. In this case we are using 7 lines of the microcontroller. This mode is called the 4-bit transfer mode. By default when we are using 4-bit mode the data transfer happens only through DB4-DB7 (the LCD pins) whereas DB0-DB3 are unconnected.

In the datasheet of LCD module we find a table shown below. This is the Instruction set of the LCD Module JHD 162A.

Let's look at first row - SCREEN CLEAR. This Instruction tells the LCD module to return the cursor to the starting position i.e. (1, 1) and also to clear the screen of any characters. The code for this instruction is 0x01. If we want the LCD module to clear the screen, then we should **write** the **command 0x01** to the LCD Module.

Similarly we have DISPLAY SWITCH instruction. It selects options for the display like Turning the display on/off, turning the cursor on/off, turn blinking of cursor on/off. If we want to turn display on we set D (DB2) to 1, to turn cursor on we make C (DB1) 1 and to select blinking cursor B (DB0) is set to 1. Hence the final code of Display switch is 0x0F. If we **write** a **command 0x0F**, then the LCD module will perform the above mentioned 3 actions.

Finally we will look at FUNCTION SET- this Instruction selects the 4-bit/8-bit mode (DL =0 means 4-bit transfer mode selected whereas DL=1 means 8-bit mode selected), N selects the number of lines in LCD (N=1 means a 2-line display), F select font type (in our case we have 5*7 type font which means F=0). Hence to select a 4-bit mode, 2-line display and 5*7 font the value of FUNCTION SET becomes 0x28. If Instruction INPUT SET is not given then by default after every character write, the cursor moved to next block. [But this does not implies that after writing to (16, 1) the cursor will automatically come to (1, 2)!!! you will have to set the address of cursor to (1,2) yourself.]

COMMAND	COMMAND CODE										COMMAND CODE	E-CYCLE f_{osc}=250KHz
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
SCREEN CLEAR	0	0	0	0	0	0	0	0	0	1	Screen Clear, Set AC to 0 Cursor Reposition	1.64ms
CURSOR RETURN	0	0	0	0	0	0	0	0	1	*	DDRAM AD=0, Return, Content Changeless	1.64ms
INPUT SET	0	0	0	0	0	0	0	1	I/D	S	Set moving direction of cursor, Appoint if move	40us
DISPLAY SWITCH	0	0	0	0	0	0	1	D	C	B	Set display on/off,cursor on/off, blink on/off	40us
SHIFT	0	0	0	0	0	1	S/C	R/L	*	*	Remove cursor and whole display,DDRAM changeless	40us
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*	Set DL,display line,font	40us
CGRAM AD SET	0	0	0	1	ACG						Set CGRAM AD, send receive data	40us
DDRAM AD SET	0	0	1	ADD							Set DDRAM AD, send receive data	40us
BUSY/AD READ CT	0	1	BF	AC							Executing internal function, reading AD of CT	40us
CGRAM/ DDRAM DATA WRITE	1	0	DATA WRITE								Write data from CGRAM or DDRAM	40us
CGRAM/ DDRAM DATA READ	1	1	DATA READ								Read data from CGRAM or DDRAM	40us
	I/D=1: Increment Mode; I/D=0: Decrement Mode S=1: Shift S/C=1: Display Shift; S/C=0: Cursor Shift R/L=1: Right Shift; R/L=0: Left Shift DL=1: 8D DL=0: 4D N=1: 2R N=0: 1R F=1: 5x10 Style; F=0: 5x7 Style BF=1: Execute Internal Function; BF=0: Command Received										DDRAM: Display data RAM CGRAM: Character Generator RAM ACG: CGRAM AD ADD: DDRAM AD & Cursor AD AC: Address counter for DDRAM & CGRAM	E-cycle changing with main frequency. Example: If fcp or f_{osc}=270KHz 40us x 250/270 =37us

Instruction Set of LCD Module

0x80	0x81	0x82	0x83									0x8D	0x8E	0x8F
0xC0	0xC1	0xC2										0xCD	0xCE	0xCF

Memory Address of Display Block on the LCD Module

The Memory addresses of blocks on LCD display are as shown above. So if you **send** a **Command** say 0x80, then the cursor will reach to the first block i.e. (1,1) similarly if you **send** a **command** 0xC5, then cursor will reach location (5,2) and so on. In this way you can move cursor to any position.

To make the LCD work with microcontroller you will proceed in this way:-

1. All the functions of the LCD will be written in a separate file named lcd.h, so that whenever you need to use the LCD functionalities, you will just include this header file lcd.h, and call the functions written in lcd.h to display or configure LCD module.
2. In the lcd.h you should have the following functions:-

(a) LCD_init(void)

(b) LCD_Send(char c, unsigned char DC)

(c) LCD_Send_String(char *s)

(d) LCD_GotoXY(unsigned int x, unsigned int y)

(e) LCD_Print(int data, unsigned int x, unsigned int y)

LCD_init – This function initializes the LCD module by setting 4-bit data transfer mode, turning the display on, turning the cursor on and setting the cursor to blink.
LCD_Send – This function takes two arguments c, DC. Here c is the data which we want to write and DC identifies this data to be a command or character (data) i.e.
if DC=1 => Data.
if DC=0 => Command.
LCD_Send_String – This function takes a string as input and after parsing it ,send character by character to LCD_Send() function , to print the character on by one on the display.
LCD_GotoXY – This function takes the cursor to position (X, Y) in the display.
LCD_Print – This function takes 3 argument i.e. data (a numeric value), and x, y co-ordinate of the starting position of the printing of the data. First this function calls
LCD_GotoXY to move cursor to (x, y). Then the function calls itoa() function and send the resultant string to LCD_Send_String().

The Pseudo codes of the functions are written as follows:-

LCD_init(void)
{
Clear RS pin, clear RW pin.

Write a command to clear the screen.
Write a command to turn display on, turn cursor on, turn blinking of cursor on.
Write a command to select 4-bit mode, 2 line display, 5*7 font.
}
// when we say write a command it means that the function LCD_Send is called.

LCD_Send(char c, unsigned char DC)
{
Break the 8 bit value of char c into 2 parts, upper nibble and lower nibble.
If it is command
    Write upper nibble on data line
    Clear RS pin, clear RW pin
    Toggle E pin, put some delay (~1ms)
    Write lower nibble on data line
    Clear RS, clear RW pin
    Toggle E pin, put some delay (~1ms)
If it is Data
    Write upper nibble on data line
    Set RS pin, Clear RW pin
    Toggle E pin, put some delay (~1ms)
    Write lower nibble on data line
    Set RS pin, Clear RW pin
    Toggle E pin, put some delay (~1ms)
}
// you will need to put delay while toggling E i.e. Set E, put 1 ms delay, Clear E.

LCD_Send_String(char *s)
{
Break string s into characters, say c1, c2, c3…
LCD_Send(c1), LCD_Send(c2), …..
}

LCD_GotoXY(unsigned int x, unsigned int y)
{
If it is line 1, send the command corresponding to address of 1st line.
If it is line 2, send the command corresponding to address of 2nd line
}

LCD_Print(int data, unsigned int x, unsigned int y)
{
Call LCD_GotoXY(x,y)
Break integer data into character and call LCD_Send_String().
}