Character Based LCD Module Interface.

xBoard MINI™ Documentation

Using LCD Modules.

When you start working with LCD modules you will start feeling the real power of MCUs and your imaginations will be touching sky you will wonder how many exciting a powerful gadgets you can create and that's so very easily.

LCD Modules can present textual information to user. It's like a cheap "monitor" that you can hook in all of your gadgets. They come in various types. The most popular one can display 2 lines of 16 characters.

A 16x2 LCD Module

These can be easily interfaced to MCUs, thanks to the API(Functions used to easily access the modules) we provide. LCD interfacing is just fun !

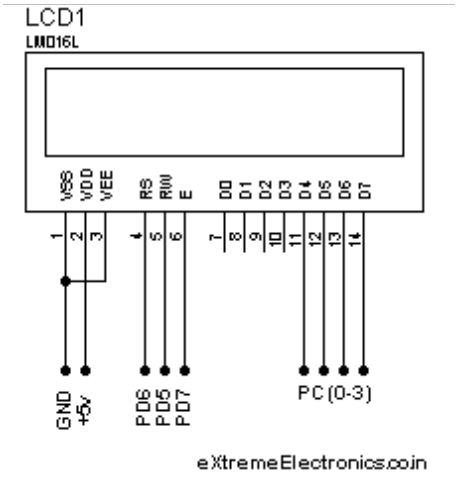
PIN Configurations.

The lcd modules has 16 PINs for interfacing. The details are given below.

No	Name
1	VSS (GND supply)
2	VCC (+5V supply)
3	VEE (contrast adjst)
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	LED(+)
16	LED(-)

Connection with xBoard MINI.

The lcd module can be easily connected to the xBoard™ MINI the connection is as follows.



Connection with MCU

Connect the required pins of PORTC and PORTD as shown in the diagram. The PORTs are clearly marked in the board. Connect to PORTD using a 8 PIN connecter and to PORTC using a 6 PIN connecter. Then supply the LCD using the onboard 5V supply output using a 2 PIN connecter. Leave D0-D4 of LCD unconnected. ***Note all the required connecters are provided with the kit.***

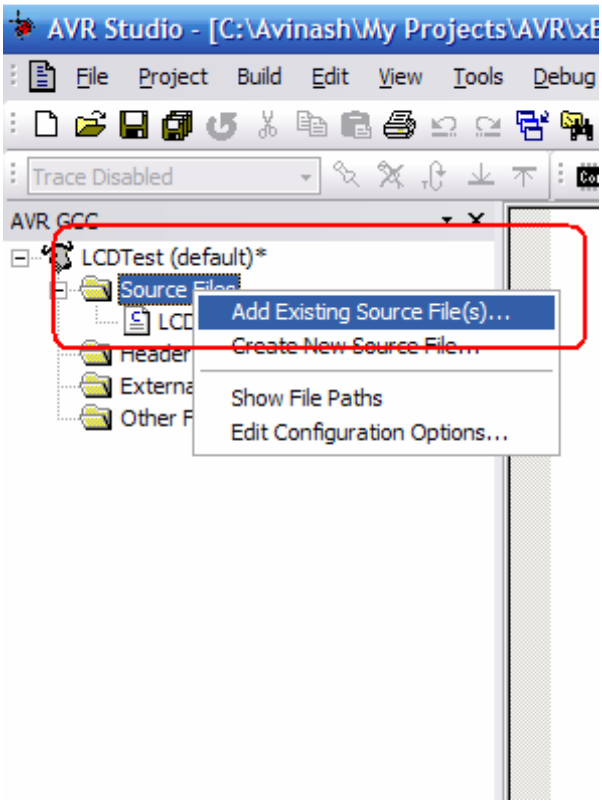
Adding LCD support to your project

To add LCD support to your C projects we have made a easy to use library. To use this library first create a new project in AVRStudio then copy the following files to your project folder.

lcd.c
lcd.h
myutils.h

from
x:\xAPI\lcd-v20\ (where x:\ is your CD/DVD drive)

then add them to your project by right clicking project view and selecting "Add Existing Source File(s)…" and then select the "lcd.c".



Adding files to project.

similarly add "lcd.h" and "myutils.h" in Header Files section. Now you are ready to start coding LCD applications !!!

Programming.

In your main C file include the file lcd.h as

#include "lcd.h"

then initialize the LCD subsystem using a call to

LCDInit(LS_BLINK|LS_ULINE);

the argument specify the type of cursor required the LS_BLINK gives a blinking cursor. LS_ULINE gives a underlined cursor.

To write any text call

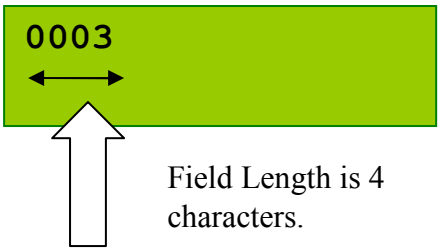
LCDWriteString("Welcome");

To write any number call

void LCDWriteInt(int val,unsigned int field_length);

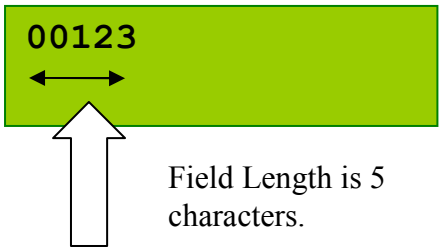
This will print a integer contained in "val" . The field length is the length of field in which the number is printed. For example

LCDWriteInt(3,4);
will print as follows



Field Length is 4 characters.

While
LCDWriteInt(123,5) will print as follows.

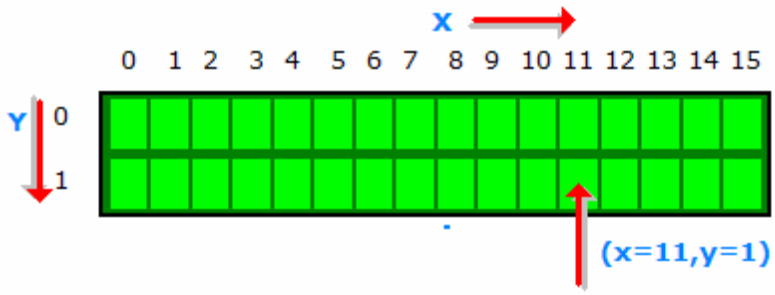


To goto any particular position on screen call.

```
void LCDGotoXY(uint8_t x,uint8_t y);
```

For example the following will take the cursor to (11,1) i.e. 12th column of second line.

```
LCDGotoXY(11,1);
```



Cursor Positioning.

Now anything you write to LCD will be printed at (11,1).

Clearing the display

```
LCDClear()
```

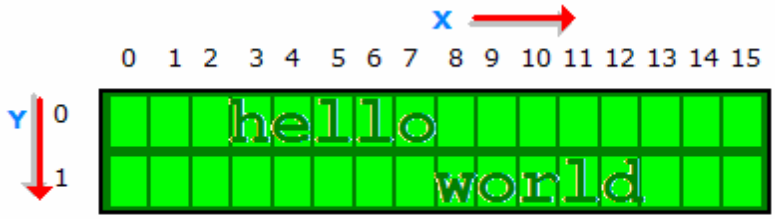
This will clear the display and bring the cursor back to (0,0).

There a two more functions that will go to specific position and print in one call.

```
LCDWriteStringXY(x,y,msg);
x,y :the location where to print "msg"
msg :the message to print

Ex:
LCDWriteStringXY(3,0,"hello");
LCDWriteStringXY(8,1,"world");

Output:
```



Cursor Positioning.

Similarly there is a function for integers.

```
LCDWriteIntXY(x,y,num,field_length);
x,y :the location where to print "msg"
num:the integer number to print
field_length: the length of field (see LCDWriteInt() function
above).
```

Now you know the basics of LCD interfacing lets jump to a sample program that will demonstrate the functions you learned.

Sample Program.

```
#include <avr/io.h>
#include <util/delay.h>

#include "lcd.h"

void main()
{
	unsigned char i;

	//Initialize LCD module
	InitLCD(LS_BLINK|LS_ULINE);

	//Clear the screen
	LCDClear();

	//Simple string printing
	LCDWriteString("Congrats ");

	//A string on line 2
	LCDWriteStringXY(0,1,"Loading ");

	//Print some numbers
	for (i=0;i<99;i+=1)
	{
		LCDWriteIntXY(9,1,i,3);
		LCDWriteStringXY(12,1,"%");
		_delay_loop_2(0);
		_delay_loop_2(0);
		_delay_loop_2(0);
		_delay_loop_2(0);


	}

	//Clear the screen
	LCDClear();

	//Some more text

	LCDWriteString("Hello world");
	LCDWriteStringXY(0,1,"By YourName Here"); // <--- Write ur NAME HERE

	//Wait
	for(i=0;i<100;i++) _delay_loop_2(0);

	//Some More ......
	LCDClear();
	LCDWriteString("    eXtreme");
	LCDWriteStringXY(0,1,"  Electronics");

}
```

Note: The sample programs are available under "samples" folder in support CD. The hex files ready to burn are available under "hex" folder under the "samples" folder.

Advance Use – Configuring Connections.

The library is designed to be fully customizable. If you want to connect the LCD module to some different i/o ports of the MCU then you can do so easily. You just have to modify the lcd.h file. Let's see how.

Open lcd.h and find a section "LCD Connections" it looks like

```
/************************************************
    LCD CONNECTIONS
************************************************/

#define LCD_DATA C  //Port PC0-PC3 are connected to D4-D7

#define LCD_E D         //Enable OR strobe signal
#define LCD_E_POS   PD7 //Position of enable in above port

#define LCD_RS D
#define LCD_RS_POS  PD6

#define LCD_RW D
#define LCD_RW_POS  PD5


//************************************************
```

Set LCD_DATA to the port where you have connected the LCD data lines. Data Lines must be connected to any port say PORTB starting from pin-0 to pin-3. i.e. If you set

#define LCD_DATA B

you should connect
PORTB.0->DATA4
PORTB.1->DATA5
PORTB.2->DATA6
PORTB.3->DATA7

The library uses advance 4-bit mode so DATA0-DATA-3 of LCD are not used, saving 4 MCU pins!

Configuring LCD Connection.

Now set the port where you have connected LCD's 'E' signal. In example it is PORTD so

#define LCD_E D

Then specify to which PIN of PORTD it is connected, this is done by

#define LCD_E_POS PD6

So 'E' pin of LCD is connected to PORTD-6
In same way set RS and RW signals. And that's all! So you saw how easy is to customize the libray.

Whats next

We will see what different Internal peripherals available on the AVR mcu and the methods of interface/communication from them. This will help you use internal peripheral like ADC,Timers,USARTs etc which will be covered in later tutorials.