LCD Interfacing with AVR Microcontroller.

When you start working with LCD modules you will start feeling the real power of MCU 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. These can be easily interfaced to MCU's, thanks to the API(Functions used to easily access the modules) we provide. LCD interfacing is just fun !

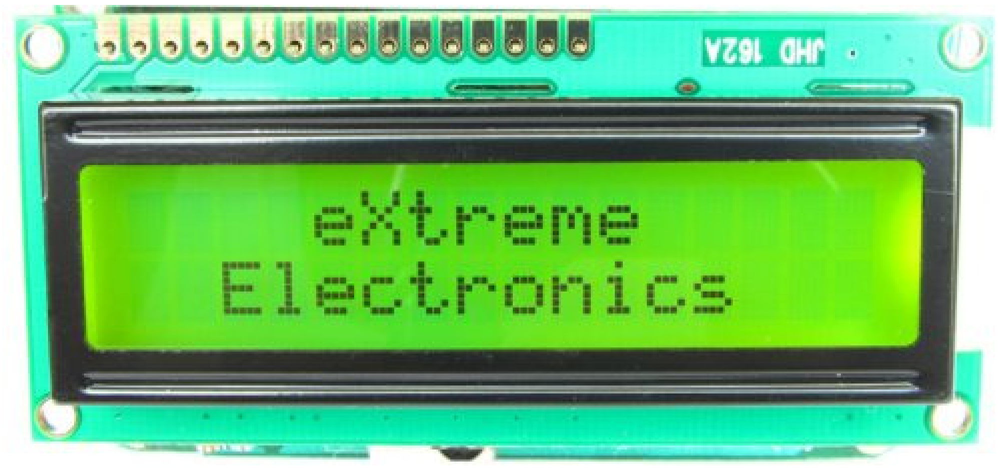


Fig: A 16x2 LCD Module

Buy LCD Modules Online In India

PIN Configurations.

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

LCD Module Pin Configuration

1 VSS (GND Supply)

2 VCC (+5V)

3 VEE (Contrast Adjust)

4 RS

5 R/W

6 E

7 DB0

8 DB1

9 DB2

10 DB3

11 DB4

12 DB5

13 DB6

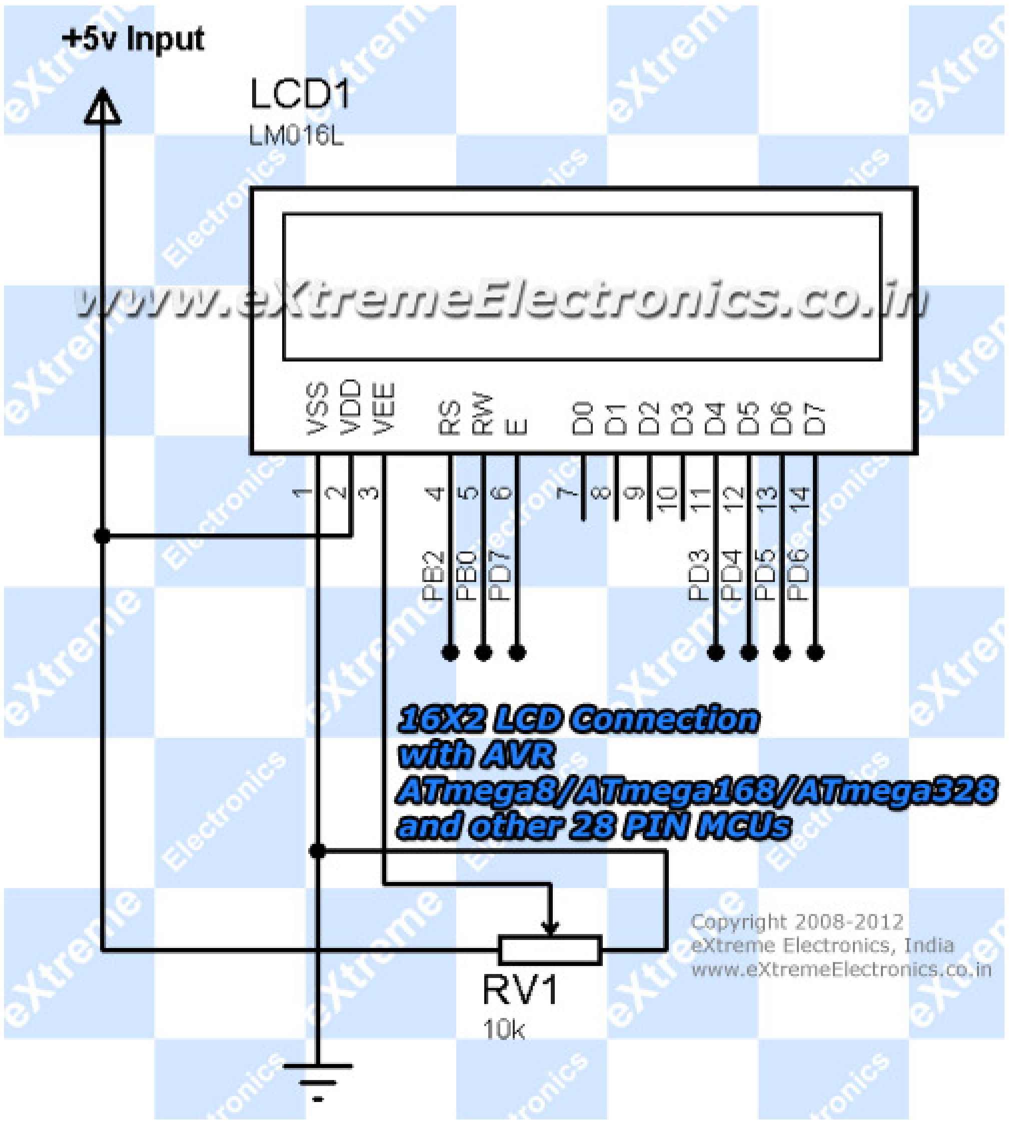
14 DB7

15 LED +

16 LED -

Connection with ATmega8/ATmega168 etc.

The lcd module can be easily connected to the any 28 pin AVR MCU like ATmega8/ATmega168 /ATmega328 etc. The diagram below shows the LCD connection with AVR MCUs port pins.



+5v Input
LCD1
LM016L
www.eXtremeElectronics.co.in
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
PB2
PB0
PD7
PD3
PD4
PD5
PD6
16X2 LCD Connection with AVR ATmega8/ATmega168/ATmega328 and other 28 PIN MCUs
RV1
10k
Copyright 2008-2012 eXtreme Electronics, India www.eXtremeElectronics.co.in

Fig: Connection with 28 PIN AVR MCUs

Connect the required pins of PORTB and PORTD as shown in the diagram. The PORTs are clearly marked in the board. For connection you will need single pin female to female wires. Supply the LCD using the onboard 5V supply output using a 2 PIN connecter. Leave D0-D4 of LCD unconnected.

Connection with 40 PIN MCUs like ATmega16/ATmega32

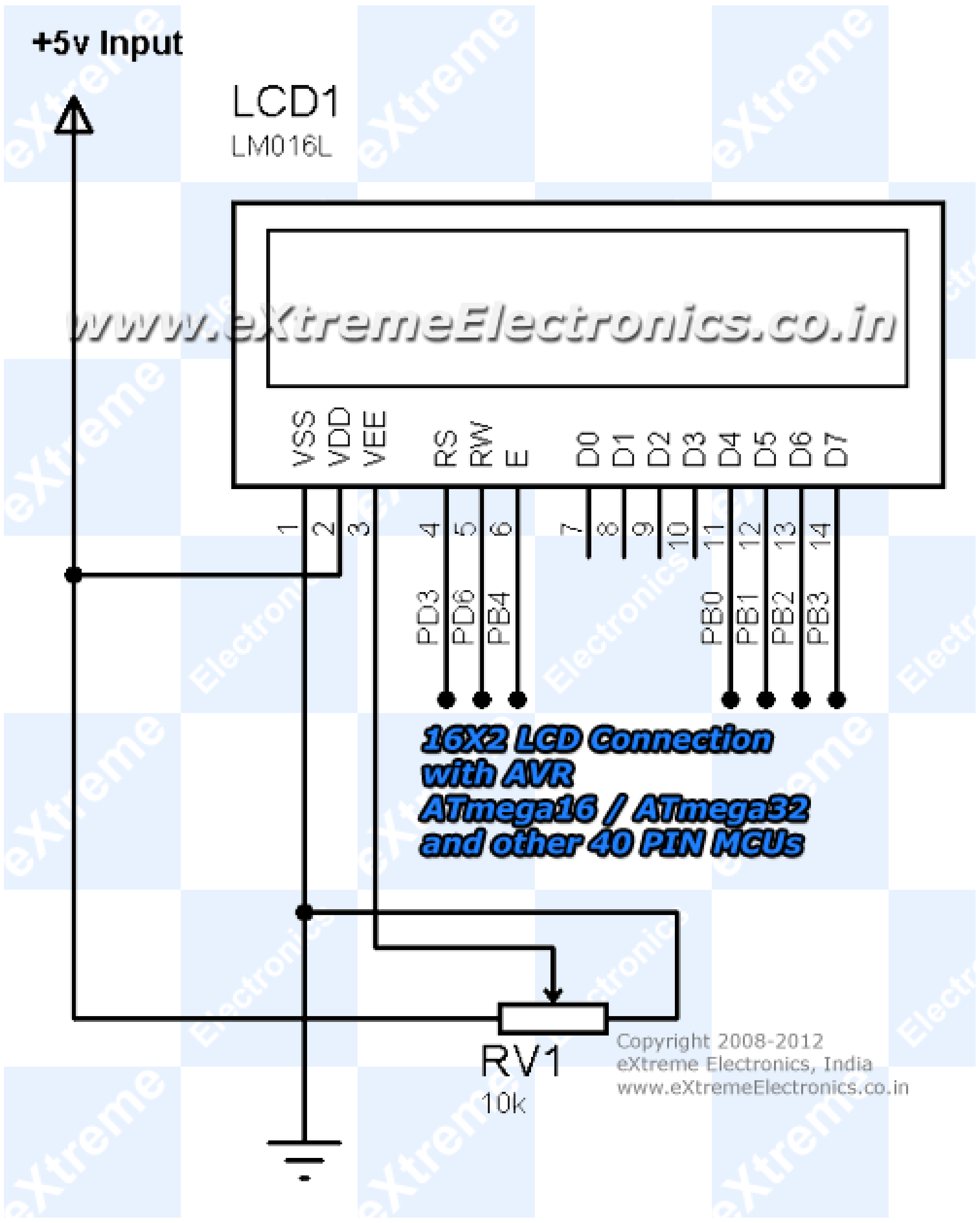

+5v Input
LCD1
LM016L
www.eXtremeElectronics.co.in
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
1 2 3 4 5 6 7 8 9 10 11 12 13 14
PD3 PD6 PB4 PB0 PB1 PB2 PB3
16X2 LCD Connection with AVR ATmega16 / ATmega32 and other 40 PIN MCUs
RV1
10k
Copyright 2008-2012 eXtreme Electronics, India www.eXtremeElectronics.co.in

Fig: Connection with 40 PIN AVR MCUs

NOTE: The 10K Pot (RV1) is very important, so please don't omit that ! When powered on for the first time you need to adjust this pot to get a clear display. Without proper adjustment of this pot you can even get a completely blank display !

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 AVR Studio then copy the following files to your **project folder**. lcd.c lcd.h myutils.h from lcdlibv20.zip then add them to your project by right clicking project view and selecting "***Add Existing Source File(s)…***" and then select the ***"lcd.c"***. Similarly add ***"lcd.h"*** and ***"myutils.h"*** in Header Files section. Now you are ready to start coding LCD applications !!!

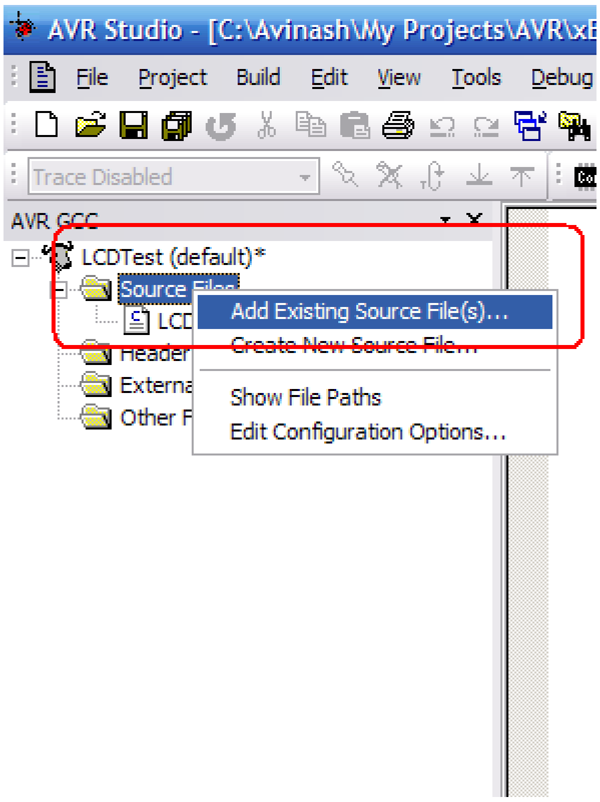
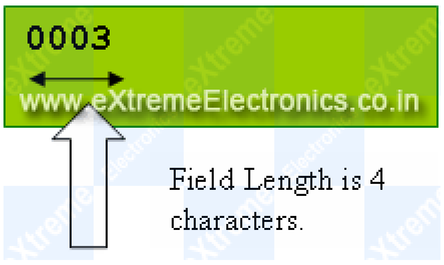

AVR Studio - [C:\Avinash\My Projects\AVR\xE
File Project Build Edit View Tools Debug
Trace Disabled
AVR GCC
LCDTest (default)*
Source Files
LCD
Header
Externa
Other F
Add Existing Source File(s)...
Create New Source File...
Show File Paths
Edit Configuration Options...

Fig: Adding files to projects.

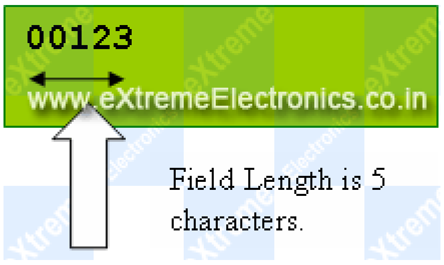
Programming.

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



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);
```

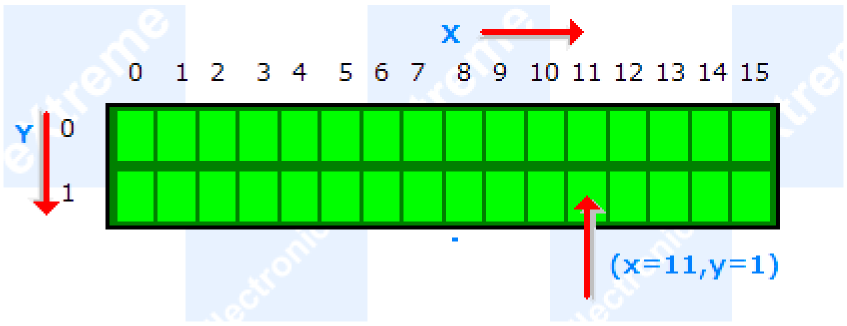


Fig: Cursor Positioning.

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

Writing Text at a specific position.

```
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:
```

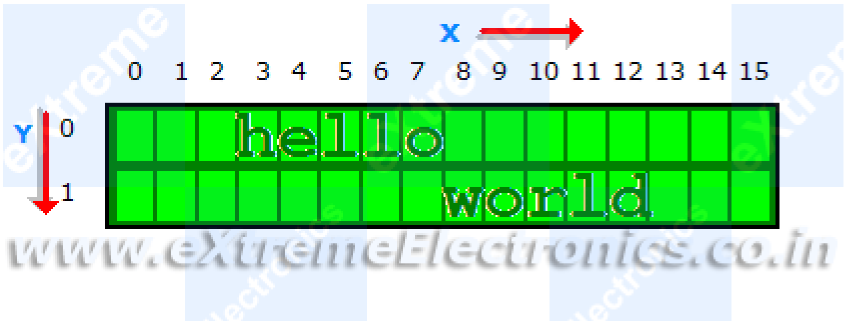


Fig: Cursor Positioning.

Writing Number at a specific position.

Similarly there is a function for integers.

```
LCDWriteIntXY(x,y,num,field_length);
____________________________________
x,y : the location where to print "num"
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.

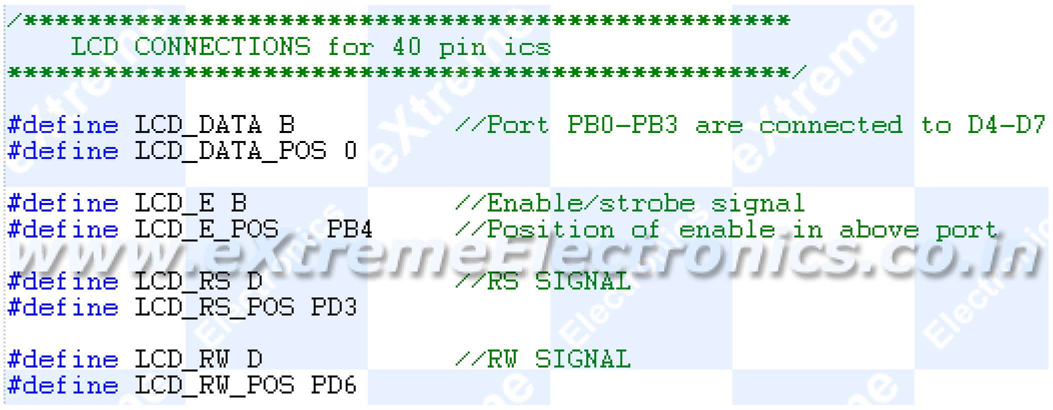
Sample Program

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

#include "lcd.h"

void main()
{
   unsigned char i;

   //Initialize LCD module
   LCDInit(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");

}
```

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 for 40 pin ics
************************************************/

#define LCD_DATA B          //Port PB0-PB3 are connected to D4-D7
#define LCD_DATA_POS 0

#define LCD_E B             //Enable/strobe signal
#define LCD_E_POS   PB4     //Position of enable in above port

#define LCD_RS D            //RS SIGNAL
#define LCD_RS_POS PD3

#define LCD_RW D            //RW SIGNAL
#define LCD_RW_POS PD6
```

Fig: Configuring LCD Connection.

Set LCD_DATA to the port where you have connected the LCD data lines. Then set LCD_DATA_POS to the starting pin of data lines. The LCD data lines D4 to D7 must be connected to consecutive pins starting from LCD_DATA_POS. For example if you wise to connect like this.

PORTD (MCU)	0	1	2	3	4	5	6	7
DATA LINEs (LCD)				D4	D5	D6	D7	

You must configure like this

#define LCD_DATA D //Because we are using PORTD

#define LCD_DATA_POS 3 // Because LCD DATA4 is connected to PD3

Please note that LCD Data lines DATA0 to DATA3 are always unused only DATA4 to DATA7 are required.

The library uses advance 4-bit mode so DATA0-DATA-3 of LCD are not used, saving 4 MCU pins! Now set the port where you have connected LCD's 'E' signal. In example it is PORTB so #define LCD_E B Then specify to which PIN of PORTB it is connected, this is done by #define LCD_E_POS PB4 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 library.

Downloads

- Core library files.
- AVR Studio Project for 28 PIN AVR MCUs (Tested on ATmega8).
- AVR Studio Project for 40 PIN AVR MCUs (Tested on ATmega32/ATmega16).
- HEX File ready to burn on ATmega8.
- HEX File ready to burn on ATmega16.
- HEX File ready to burn on ATmega32.

Custom Character ?

Some times we need to show characters that are not the part of standard character set, like some special symbols (say heart or symbols of other language like Hindi). In that case we can use custom characters. They are also easy to make and use. **Full article on using custom characters on alphanumeric lcds is available here**.

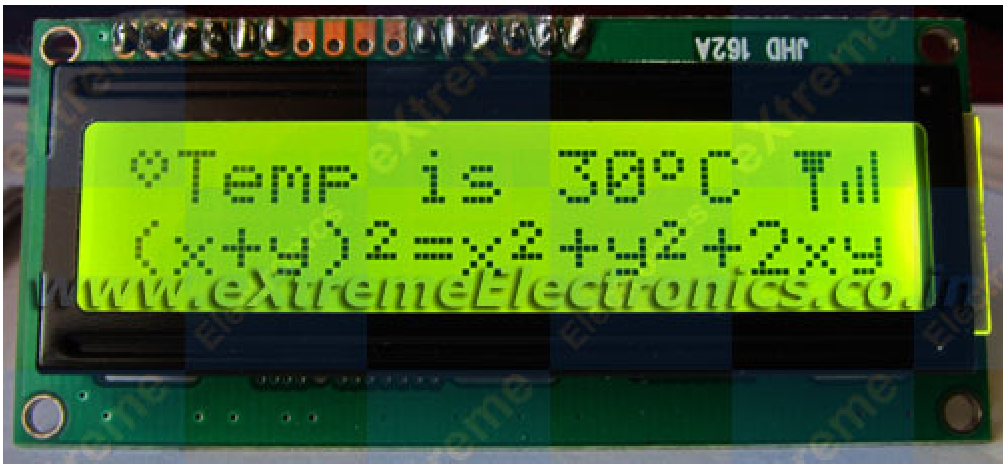


Fig: Custom Char Demo ! See the heart and other symbols?

Going Graphical ?

If you have more sophisticated display requirement then you may go for **Graphical LCDs (GLCD)** they can render graphics like Icons and bitmaps (images), support graphic primitives like line,circle, rectangle etc. Also they can draw text in different fonts and sizes. That means 100% fun! And they are cheap and very easy to get started (thanks to **ProGFX.org**). We have some beginners tutorials for it too. So why waiting? Go Graphical!

- **http://extremeelectronics.co.in/avr-tutorials/interfacing-ks0108-based-128x64-graphical-lcd-with-avr-mcu/**

Have fun and don't forget to bookmark this page. If you think this tutorial has helped you please post a **comment.**

Author

Avinash Gupta

My Facebook Profile.