**CHAPTER IV**

**LCD INTERFACING**

**4.1 Introduction**  
The most commonly used Character based LCDs are based on Hitachi's HD44780 controller or other which are compatible with HD44580. In this tutorial, we will discuss about character based LCDs, their interfacing with various microcontrollers, various interfaces (8-bit/4-bit), programming, special stuff and tricks you can do with these simple looking LCDs which can give a new look to your application.

**Pin Description**

The most commonly used LCD’s found in the market today are 1 Line, 2 Line or 4 Line LCDs which have only 1 controller and support at most of 80 characters, whereas LCDs supporting more than 80 characters make use of 2 HD44780 controllers.  
  
Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections). Pin description is shown in the table below.

|  |  |  |
| --- | --- | --- |
| **Pin No.** | **Name** | **Description** |
| Pin no. 1 | **VSS** | Power supply (GND) |
| Pin no. 2 | **VCC** | Power supply (+5V) |
| Pin no. 3 | **VEE** | Contrast adjust |
| Pin no. 4 | **RS** | 0 = Instruction input 1 = Data input |
| Pin no. 5 | **R/W** | 0 = Write to LCD module 1 = Read from LCD module |
| Pin no. 6 | **EN** | Enable signal |
| Pin no. 7 | **D0** | Data bus line 0 (LSB) |
| Pin no. 8 | **D1** | Data bus line 1 |
| Pin no. 9 | **D2** | Data bus line 2 |
| Pin no. 10 | **D3** | Data bus line 3 |
| Pin no. 11 | **D4** | Data bus line 4 |
| Pin no. 12 | **D5** | Data bus line 5 |
| Pin no. 13 | **D6** | Data bus line 6 |
| Pin no. 14 | **D7** | Data bus line 7 (MSB) |

**DDRAM - Display Data RAM**

[Display data](http://www.8051projects.net/lcd-interfacing/basics.php) RAM (DDRAM) stores display data represented in 8-bit character codes. Its extended capacity is 80 X 8 bits, or 80 characters. The area in display data RAM (DDRAM) that is not used for display can be used as general data RAM. So whatever you send on the DDRAM is actually displayed on the [LCD](http://www.8051projects.net/lcd-interfacing/basics.php). For LCDs like 1x16, only 16 characters are visible, so whatever you write after 16 chars is written in DDRAM but is not visible to the user.

**4-bit programming of LCD**

In 4-bit mode the data is sent in nibbles, first we send the higher nibble and then the lower nibble. To enable the 4-bit mode of LCD, we need to follow special sequence of initialization that tells the LCD controller that user has selected 4-bit mode of operation. We call this special sequence as resetting the LCD. Following is the reset sequence of LCD.

* Wait for about 20mS
* Send the first init value (0x30)
* Wait for about 10mS
* Send second init value (0x30)
* Wait for about 1mS
* Send third init value (0x30)
* Wait for 1mS
* Select bus width (0x30 - for 8-bit and 0x20 for 4-bit)
* Wait for 1mS

The busy flag will only be valid after the above reset sequence. Usually we do not use busy flag in 4-bit mode as we have to write code for reading two nibbles from the LCD. Instead we simply put a certain amount of delay usually 300 to 600uS. This delay might vary depending on the LCD you are using, as you might have a different crystal frequency on which LCD controller is running. So it actually depends on the LCD module you are using.

In 4-bit mode, we only need 6 pins to interface an LCD. D4-D7 are the data pins connection and Enable and Register select are for LCD control pins. We are not using Read/Write (RW) Pin of the LCD, as we are only writing on the [LCD](http://www.8051projects.net/lcd-interfacing/lcd-4-bit.php) so we have made it grounded permanently. If you want to use it, then you may connect it on your controller but that will only increase another pin and does not make any big difference. Potentiometer RV1 is used to control the LCD contrast. The unwanted data pins of LCD i.e. D0-D3 are connected to ground.

**Sending data/command in 4-bit Mode**

We will now look into the common steps to send data/command to LCD when working in 4-bit mode. In 4-bit mode data is sent nibble by nibble, first we send higher nibble and then lower nibble. This means in both command and data sending function we need to separate the higher 4-bits and lower 4-bits.

The common steps are:

* Mask lower 4-bits
* Send to the LCD port
* Send enable signal
* Mask higher 4-bits
* Send to LCD port
* *Send enable signal*