Search Tutorials

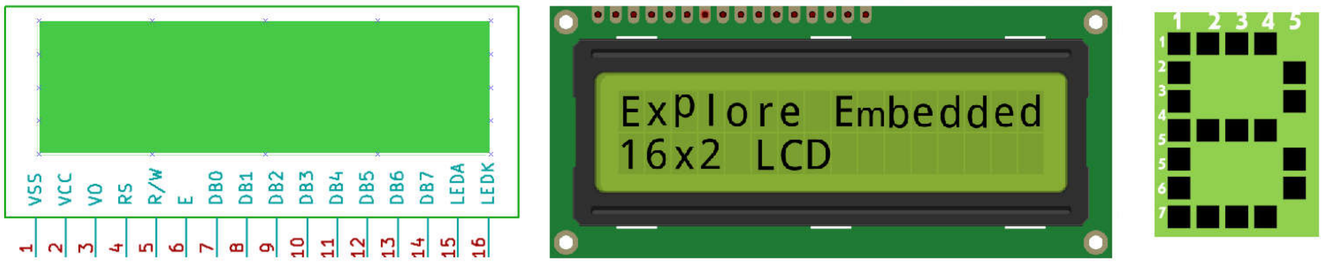
Contents ▾

# A1.8051 Interfacing:LCD 16x2

In this tutorial we are going to see how to interface a 2x16 LCD with 8051 in 8-bit mode. As per the name the 2x16 has 2 lines with 16 chars on each lines. It supports all the ascii chars and is basically used for displaying the alpha numeric characters. Here each character is displayed in a matrix of 5x7 pixels. Apart from alpha numeric chars it also provides the provision to display the custom characters by creating the pattern.

## LCD UNIT

Let us look at a pin diagram of a commercially available LCD like **JHD162** which uses a **HD44780** controller and then describe its operation.



(/wiki/File:Pic16f877aLcdInterface.png)

| Pin Number | Symbol | Pin Function |
|---|---|---|

| | | |
|---|---|---|
| 1 | VSS | Ground |
| 2 | VCC | +5v |
| 3 | VEE | Contrast adjustment (VO) |
| 4 | RS | Register Select. 0:Command, 1: Data |
| 5 | R/W | Read/Write, R/W=0: Write & R/W=1: Read |
| 6 | EN | Enable. Falling edge triggered |
| 7 | D0 | Data Bit 0 |
| 8 | D1 | Data Bit 1 |
| 9 | D2 | Data Bit 2 |
| 10 | D3 | Data Bit 3 |
| 11 | D4 | Data Bit 4 |
| 12 | D5 | Data Bit 5 |
| 13 | D6 | Data Bit 6 |
| 14 | D7 | Data Bit 7/Busy Flag |
| 15 | A/LED+ | Back-light Anode(+) |
| 16 | K/LED- | Back-Light Cathode(-) |

Apart from the voltage supply connections the important pins from the programming perspective are the data lines(8-bit Data bus), Register select, Read/Write and Enable pin.

**Data Bus:** As shown in the above figure and table, an alpha numeric lcd has a 8-bit data bus referenced as D0-D7. As it is a 8-bit data bus, we can send the data/cmd to LCD in bytes. It also provides the provision to send the the data/cmd in chunks of 4-bit, which is used when there are limited number of GPIO lines on the microcontroller.

**Register Select(RS):** The LCD has two register namely a Data register and Command register. Any data that needs to be displayed on the LCD has to be written to the data register of LCD. Command can be issued to LCD by writing it to Command register of LCD. This signal is used to differentiate the data/cmd received by the LCD.
If the RS signal is **LOW** then the LCD interprets the 8-bit info as **Command** and writes it **Command register** and performs the action as per the command.
If the RS signal is **HIGH** then the LCD interprets the 8-bit info as **data** and copies it to **data register**. After that the LCD decodes the data for generating the 5x7 pattern and finally displays on the LCD.

**Read/Write(RW):** This signal is used to write the data/cmd to LCD and reads the busy flag of LCD. For write operation the RW should be **LOW** and for read operation the R/W should be **HIGH**.

**Enable(EN):** This pin is used to send the enable trigger to LCD. After sending the data/cmd, Selecting the data/cmd register, Selecting the Write operation. A HIGH-to-LOW pulse has to be send on this enable pin which will latch the info into the LCD register and triggers the LCD to act accordingly.

## Schematic

Below schematic shows the minimum connection required for interfacing the LCD with the microcontroller.

## Port Connection

This section shows how to configure the GPIO for interfacing the LCD.
The below configuration is as per the above schematic. You can connect the LCD to any of the PORT pins available on your boards and update this section accordingly

```
/* Configure the data bus and Control pins as per the hardware connection
   Databus is connected to P2_0:P2_7 and control bus P0_0:P0_2*/
#define LcdDataBus  P2

sbit LCD_RS = P0^0;
sbit LCD_RW = P0^1;
sbit LCD_EN = P0^2;
```

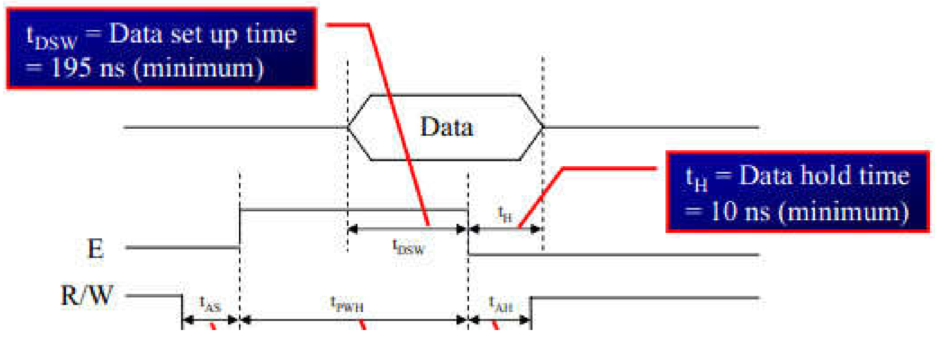
view raw (https://gist.github.com/SaheblalBagwan/1d7babfd4b67463ce5cf7b4a92f2805e /raw/e7816904d1e07ed474d91ee2f5b76d0e6621d9dd/8051Lcd8bitConnection.c) 8051Lcd8bitConnection.c (https://gist.github.com/SaheblalBagwan/1d7babfd4b67463ce5cf7b4a92f2805e#file-8051lcd8bitconnection-c) hosted with ❤ by GitHub (https://github.com)

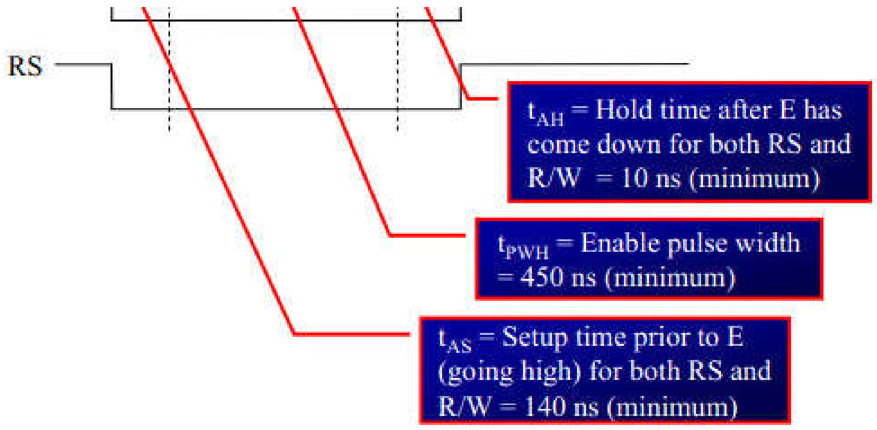
## LCD Operation

In this section we are going to see how to send the data/cmd to the LCD along with the timing diagrams. First lets see the timing diagram for sending the data and the command signals(RS,RW,EN), accordingly we write the algorithm and finally the code.

### Timing Diagram

The below image shows the timing diagram for sending the data to the LCD.
As shown in the timing diagram the data is written after sending the RS and RW signals. It is still ok to send the data before these signals.
The only important thing is the data should be available on the databus before generating the High-to-Low pulse on EN pin.


$t_{DSW}$ = Data set up time = 195 ns (minimum)
Data
$t_H$ = Data hold time = 10 ns (minimum)
$t_H$
$t_{DSW}$
E
R/W
$t_{AS}$
$t_{PWH}$
$t_{AH}$


RS
$t_{AH}$ = Hold time after E has come down for both RS and R/W = 10 ns (minimum)
$t_{PWH}$ = Enable pulse width = 450 ns (minimum)
$t_{AS}$ = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

(/wiki/File:LCD_CmdWrite.jpg)

## Steps for Sending Command:

- step1: Send the I/P command to LCD.
- step2: Select the Control Register by making RS low.
- step3: Select Write operation making RW low.
- step4: Send a High-to-Low pulse on Enable PIN with some delay_us.

```
/* Function to send the command to LCD */
void LCD_CmdWrite( char cmd)
{
  LcdDataBus=cmd;    // Send the command to LCD
  LCD_RS=0;          // Select the Command Register by pulling RS LOW
  LCD_RW=0;          // Select the Write Operation  by pulling RW LOW
  LCD_EN=1;          // Send a High-to-Low Pusle at Enable Pin
  delay_us(10);
  LCD_EN=0;
  delay_us(1000);
}
```

view raw (https://gist.github.com/SaheblalBagwan/9e8c6f42511355062789a69cace9f980/raw/0f51b5056571a50df3b1934eaab9b4748339f795/8051Lcd8bitCmdWrite.c) 8051Lcd8bitCmdWrite.c (https://gist.github.com/SaheblalBagwan/9e8c6f42511355062789a69cace9f980#file-8051lcd8bitcmdwrite-c) hosted with ❤ by GitHub (https://github.com)

## Steps for Sending Data:

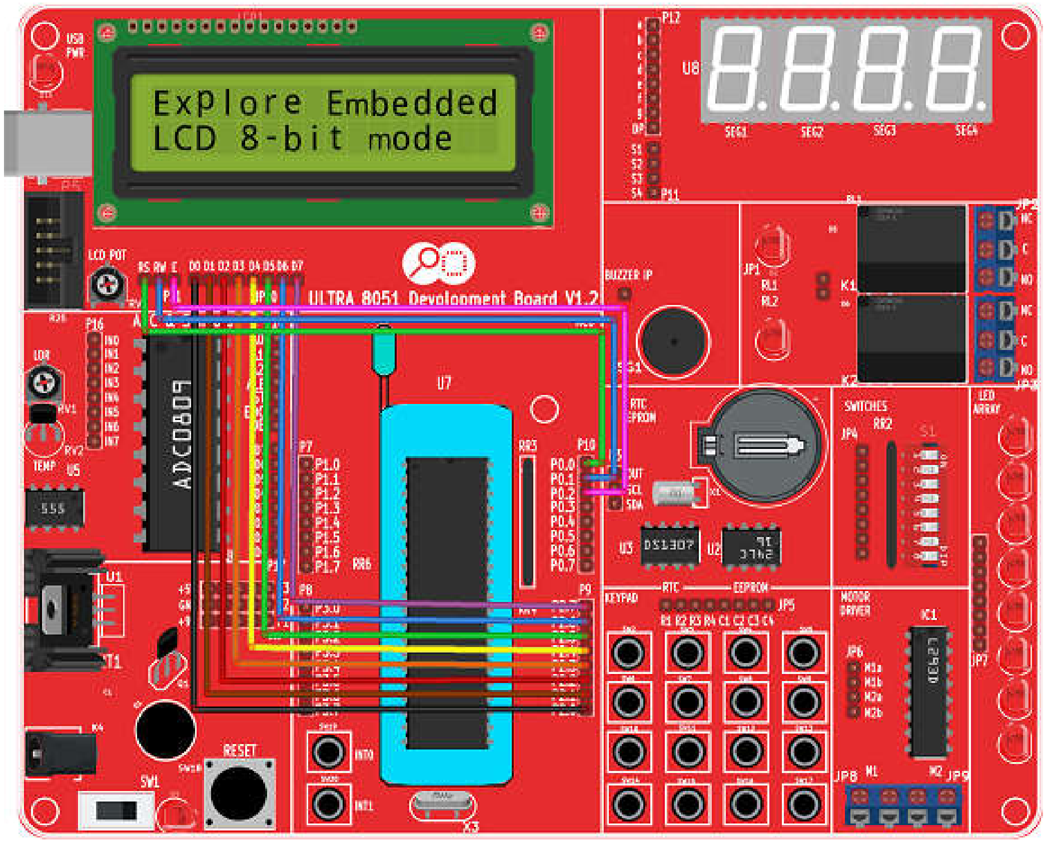
- step1: Send the character to LCD.
- step2: Select the Data Register by making RS high.
- step3: Select Write operation making RW low.
- step4: Send a High-to-Low pulse on Enable PIN with some delay_us.

The timings are similar as above only change is that **RS** is made high for selecting Data register.

```
/* Function to send the Data to LCD */
void LCD_DataWrite( char dat)
{
   LcdDataBus=dat;        // Send the data to LCD
   LCD_RS=1;          // Select the Data Register by pulling RS HIGH
   LCD_RW=0;          // Select the Write Operation by pulling RW LOW
   LCD_EN=1;          // Send a High-to-Low Pusle at Enable Pin
   delay_us(10);
   LCD_EN=0;
   delay_us(1000);
}
```

view raw (https://gist.github.com/SaheblalBagwan/fd021a189be20644c37541d75de144e4/raw/53a80c1b7a2048ad2238047a28e1eb2e0ce9490a/8051Lcd8bitdataWrite.c) 8051Lcd8bitdataWrite.c (https://gist.github.com/SaheblalBagwan/fd021a189be20644c37541d75de144e4#file-8051lcd8bitdatawrite-c) hosted with ❤ by GitHub (https://github.com)

## Hardware Connections


Explore Embedded
LCD 8-bit mode
ULTRA 8051 Development Board V1.2
RS RW E D0 D1 D2 D3 D4 D5 D6 D7
ADC0809
DS1307
L293D
RESET

(/wiki/File:Lcd8bit.png)

## Code Examples

Here is the complete code for displaying the data on 2x16 LCD in 8-bit mode.

```
#include<reg51.h>

/* Configure the data bus and Control pins as per the hardware connection
   Databus is connected to P2_0:P2_7 and control bus P0_0:P0_2*/
#define LcdDataBus  P2
sbit LCD_RS = P0^0;
sbit LCD_RW = P0^1;
sbit LCD_EN = P0^2;

/* local function to generate delay */
void delay_us(int cnt)
{
    int i;
    for(i=0;i<cnt;i++);
}

/* Function to send the command to LCD */
void LCD_CmdWrite( char cmd)
{
    LcdDataBus=cmd;    // Send the command to LCD
    LCD_RS=0;          // Select the Command Register by pulling RS LOW
    LCD_RW=0;          // Select the Write Operation  by pulling RW LOW
    LCD_EN=1;          // Send a High-to-Low Pusle at Enable Pin
    delay_us(10);
    LCD_EN=0;
    delay_us(1000);
}

/* Function to send the Data to LCD */
void LCD_DataWrite( char dat)
{
    LcdDataBus=dat;       // Send the data to LCD
    LCD_RS=1;         // Select the Data Register by pulling RS HIGH
    LCD_RW=0;          // Select the Write Operation by pulling RW LOW
    LCD_EN=1;         // Send a High-to-Low Pusle at Enable Pin
    delay_us(10);
    LCD_EN=0;
    delay_us(1000);
}

int main()
{
    char i,a[]={"Good morning!"};

    Lcd_CmdWrite(0x38);        // enable 5x7 mode for chars
    Lcd_CmdWrite(0x0E);        // Display OFF, Cursor ON
    Lcd_CmdWrite(0x01);        // Clear Display
    Lcd_CmdWrite(0x80);        // Move the cursor to beginning of first line

    Lcd_DataWrite('H');
    Lcd_DataWrite('e');
    Lcd_DataWrite('l');
    Lcd_DataWrite('l');
    Lcd_DataWrite('o');
    Lcd_DataWrite(' ');
    Lcd_DataWrite('w');
    Lcd_DataWrite('o');
    Lcd_DataWrite('r');
    Lcd_DataWrite('l');
    Lcd_DataWrite('d');

    Lcd_CmdWrite(0xc0);        //Go to Next line and display Good Morning
    for(i=0;a[i]!=0;i++)
    {
        Lcd_DataWrite(a[i]);
    }

    while(1);
}
```

view raw (https://gist.github.com/SaheblalBagwan/1ac04a9ae5112efdbe338f994d7a1af3/raw/5cd05af242da2f1993e15cd61e92ad50610c9796/8051Lcd8bitexample1.c)
8051Lcd8bitexample1.c (https://gist.github.com/SaheblalBagwan/1ac04a9ae5112efdbe338f994d7a1af3#file-8051lcd8bitexample1-c)
hosted with ❤ by GitHub (https://github.com)

# Using Explore Embedded Libraries :

In the above tutorial we just discussed how to interface 2x16Lcd in 8-bit mode.
Once you know the working of lcd, you can directly use the ExploreEmbedded libraries to play around with your LCD.
For that you need to include the lcd.c/lcd.h and the associated files(delay/stdutils).
After including these files, the only thing you got to do is to configure the PORTs in lcd.h as per your hardware connection.
The below sample code shows how to use the already available LCD functions.

## LCD 1x16

```
#include "lcd.h"

int main()
{
    /*Connect Ctrl Lines to PD0:PD2 and data lines to PORTB*/
    LCD_SetUp(PD_0,PD_1,PD_2,PB_0,PB_1,PB_2,PB_3,PB_4,PB_5,PB_6,PB_7);
    LCD_Init(1,16);

    LCD_DisplayString("Explore Lcd 1x16");

    while(1);

    return (0);
}
```

view raw (https://gist.github.com/SaheblalBagwan/778c519d1babc5659c71086e81da0381/raw/ad264f81a11e95c1730b05a0e63f81ce870ef4a2/pic16f877a_1x16Lcd8bit.c) pic16f877a_1x16Lcd8bit.c (https://gist.github.com/SaheblalBagwan/778c519d1babc5659c71086e81da0381#file-pic16f877a_1x16lcd8bit-c) hosted with ❤ by GitHub (https://github.com)

## LCD 2x16

```
#include "lcd.h"
int main()
{
    /*Connect Ctrl Lines to PD0:PD2 and data lines to PORTB*/
    LCD_SetUp(PD_0,PD_1,PD_2,PB_0,PB_1,PB_2,PB_3,PB_4,PB_5,PB_6,PB_7);
    LCD_Init(2,16);

    LCD_DisplayString("Explore Embedded");
    LCD_DisplayString("Lcd 8-bit Mode");
    while(1);

    return (0);
}
```

view raw (https://gist.github.com/SaheblalBagwan/f75119155d6e79321e222f8b611a6d34/raw/ed5b2a85fd0d50e8b26222bce0d11bfc0796b1a4/pic16f877a_2x16Lcd8bit.c) pic16f877a_2x16Lcd8bit.c (https://gist.github.com/SaheblalBagwan/f75119155d6e79321e222f8b611a6d34#file-pic16f877a_2x16lcd8bit-c) hosted with ❤ by GitHub (https://github.com)

## LCD 4x20

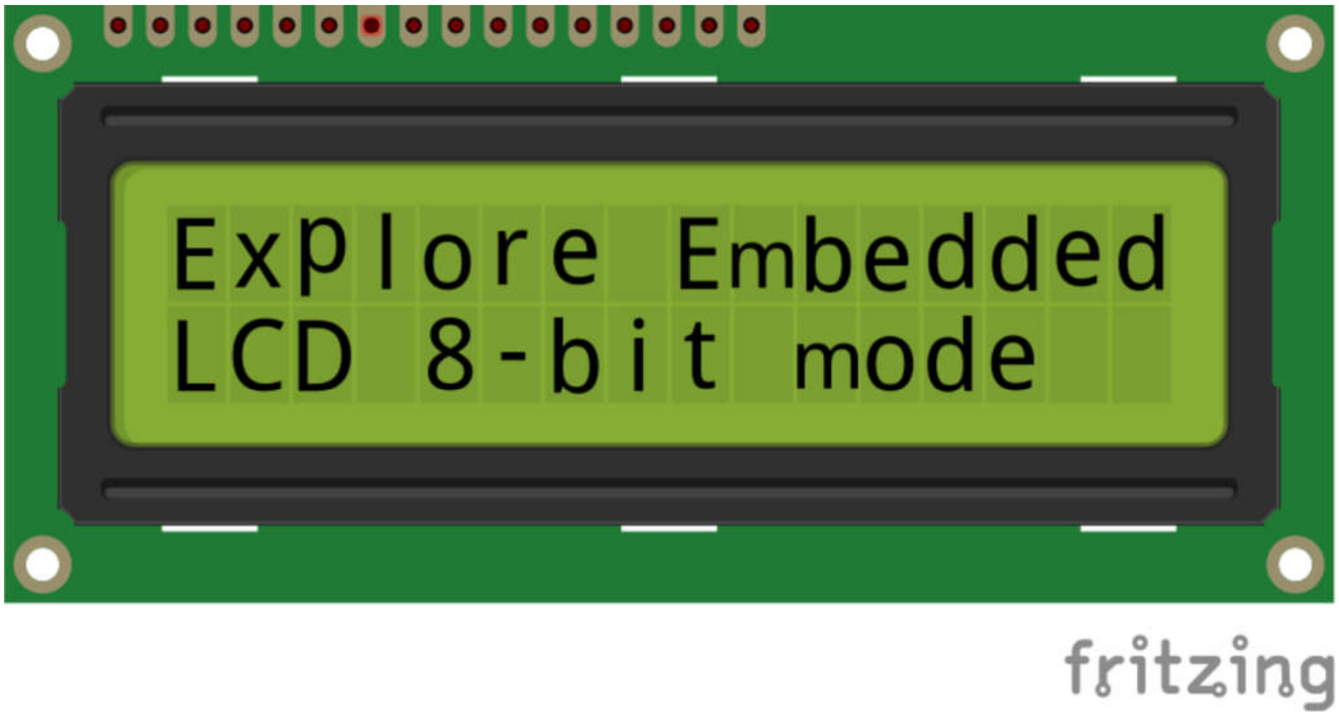
```
#include "lcd.h"
int main()
{
    /*Connect Ctrl Lines to PD0:PD2 and data lines to PORTB*/
    LCD_SetUp(PD_0,PD_1,PD_2,PB_0,PB_1,PB_2,PB_3,PB_4,PB_5,PB_6,PB_7);
    LCD_Init(4,20);

    LCD_DisplayString("Explore Embedded\n");
    LCD_DisplayString("LCD 8-bit Mode\n");
    LCD_DisplayString("20 x 4 \n");
    LCD_DisplayString(":)  :O");

    while(1);

    return (0);
}
```

view raw (https://gist.github.com/SaheblalBagwan/1b1a7846b1e0376f55bb5aedc8617dfe/raw/a1fb19eeb5d4fedf34ce1ead0b0c1cc518536386/pic16f877a_2x40Lcd8bit.c) pic16f877a_2x40Lcd8bit.c (https://gist.github.com/SaheblalBagwan/1b1a7846b1e0376f55bb5aedc8617dfe#file-pic16f877a_2x40lcd8bit-c) hosted with ❤ by GitHub (https://github.com)



(/wiki/File:01LCD_8bit.png)

## Downloads

Download the sample code and design files from this link (https://github.com/ExploreEmbedded/8051_DevelopmentBoard).

Have a opinion, suggestion , question or feedback about the article let it out here!

0 Comments exploreembedded.com/wiki 1 Login

Recommend 3 Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON EXPLOREEMBEDDED.COM/WIKI

**Overview of ESP32 features. What do they practically mean?**

3 comments • 2 years ago

Viraj — Hi..I am concerned about reliability of ESP32 in commercial product. Currently, I am working on prototype development. I have

**LPC2148 UART Programming**

2 comments • 2 years ago

Explorer — Install CP2102(Usb-to-Serial) drivers and try again.Check the below tutorial for installing the

**Serial Communication with PIC16F877A**

3 comments • 2 years ago

**6. Use of vTaskSuspend() and vTaskResume() functions**

Category (/wiki/Special:Categories): 8051 tutorials (/wiki/Category:8051_tutorials)

## Announcements

Software Engineer Recruitment

(https://www.exploreembedded.com/jobs/software_engineer)

**Subscribe to hear about our latest Explorations!**

name@example.com SUBSCRIBE

Now Shipping worldwide from India with ❤