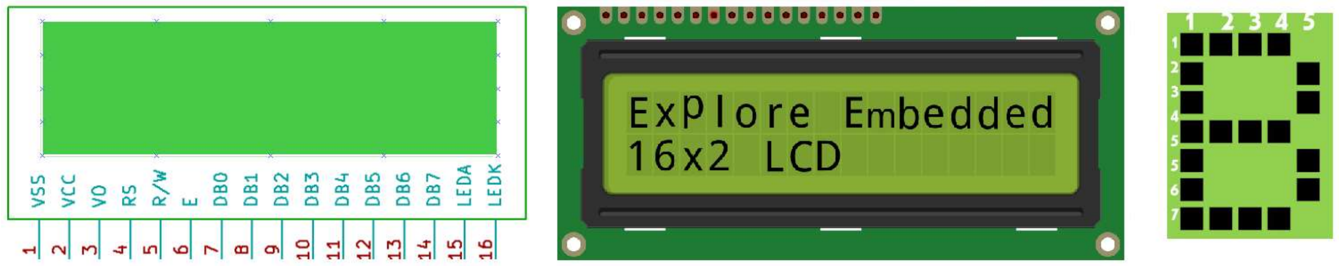
Interfacing LCD in 4-bit mode with 8051

Contents

As per the name the 2x16 has 2 lines with 16 chars on each line. It supports all the ASCII chars and is basically used for displaying the alphanumeric characters. Here each character is displayed in a matrix of 5x7 pixels. Apart from alphanumeric 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 (Not used in 4-bit operation)

8	D1	Data Bit 1 (Not used in 4-bit operation)
9	D2	Data Bit 2 (Not used in 4-bit operation)
10	D3	Data Bit 3 (Not used in 4-bit operation)
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 alphanumeric LCD has an 8-bit data bus referenced as D0-D7. As it is an 8-bit data bus, we can send the data/cmd to LCD in bytes. It also provides the provision to send 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. An HIGH-to-LOW pulse has to be sent 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.
As we are interfacing the LCD in 4-bit mode, only the higher 4 data lines are used as the data bus.

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 bus as per the hardware connection
   Databus is connected to P2.4:P2.7 and control bus P2.0:P2.2*/
#define LcdDataBus  P2
sbit LCD_RS = P2^0;
sbit LCD_RW = P2^1;
sbit LCD_EN = P2^2;
```

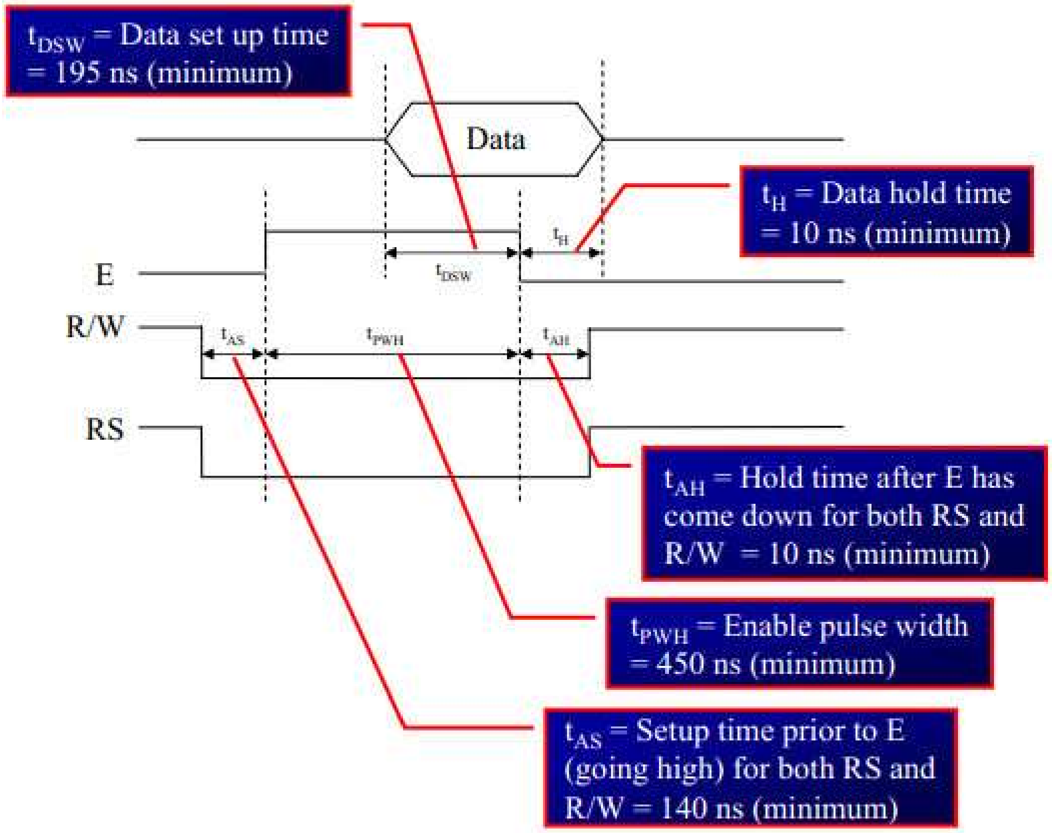
wan/2c051cee1f15cfd4329a945e2e3fa63e/raw/4704e1ac27a0312aa72478bc8886f8470b3740b5/8051_Lcd4bitConnection.c)
8051_Lcd4bitConnection.c (https://gist.github.com/SaheblalBagwan/2c051cee1f15cfd4329a945e2e3fa63e#file-8051_lcd4bitconnection-c) hosted with ♡ by **GitHub (https://github.com)**

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 let's 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.
   As it is 4bit mode, a byte of data is sent in two 4-bit nibbles */
void Lcd_CmdWrite(char cmd)
{
    LcdDataBus = (cmd & 0xF0);     //Send higher nibble
    LCD_RS = 0;   // Send LOW pulse on RS pin for selecting Command register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);

    LcdDataBus = ((cmd<<4) & 0xF0); //Send Lower nibble
    LCD_RS = 0;   // Send LOW pulse on RS pin for selecting Command register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);
}
```

van/4a521e33274ae73b7b492611ed20d762/raw/c12d802b957fc86e46366c96096c4cb713b6c351/8051_Lcd4BitCmdWrite.c)
8051_Lcd4BitCmdWrite.c (https://gist.github.com/SaheblalBagwan/4a521e33274ae73b7b492611ed20d762#file-8051_lcd4bitcmdwrite-c) hosted with ♡ by **GitHub (https://github.com)**

Steps for Sending Data:

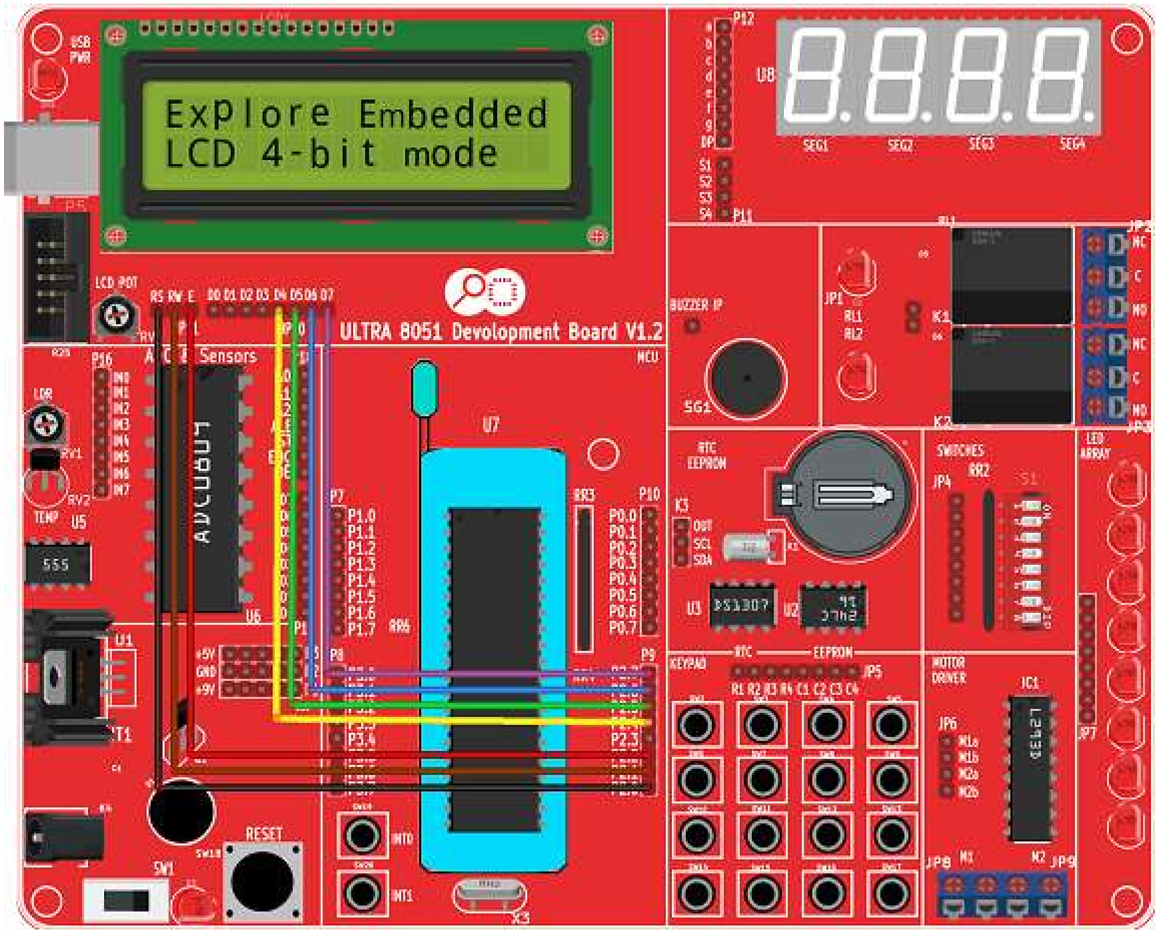
- step1: Send the character to LCD.
- step2: Select the Data Register by making RS high.
- step3: Select Write operation making RW low.
- step4: Send a High-to-Low pulse on Enable PIN with some delay_us.

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.
   As it is 4bit mode, a byte of data is sent in two 4-bit nibbles */
void Lcd_DataWrite(char dat)
{
    LcdDataBus = (dat & 0xF0);      //Send higher nibble
    LCD_RS = 1;   // Send HIGH pulse on RS pin for selecting data register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);

    LcdDataBus = ((dat<<4) & 0xF0);  //Send Lower nibble
    LCD_RS = 1;    // Send HIGH pulse on RS pin for selecting data register
    LCD_RW = 0;    // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;    // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);
}
```

van/19146e69651dc5465bdac99f75c31a29/raw/777921f2b8c0804936a3d8d5d0c73bb7020d7f9b/8051_Lcd4BitDataWrite.c)
8051_Lcd4BitDataWrite.c (https://gist.github.com/SaheblalBagwan/19146e69651dc5465bdac99f75c31a29#file-8051_lcd4bitdatawrite-c) hosted with ♡ by GitHub (https://github.com)

Hardware Connections



(/wiki/File:Lcd_4-Bit_Tutorial.png)

Code Examples

Here is the complete code for displaying the data on 2x16 LCD in 4-bit mode.

```
#include<reg51.h>
/* Configure the data bus and Control bus as per the hardware connection
```

```
   Databus is connected to P2.4:P2.7 and control bus P2.0:P2.2*/
#define LcdDataBus  P2
sbit LCD_RS = P2^0;
sbit LCD_RW = P2^1;
sbit LCD_EN = P2^2;


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



/* Function to send the command to LCD.
   As it is 4bit mode, a byte of data is sent in two 4-bit nibbles */
void Lcd_CmdWrite(char cmd)
{
    LcdDataBus = (cmd & 0xF0);     //Send higher nibble
    LCD_RS = 0;   // Send LOW pulse on RS pin for selecting Command register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);

    LcdDataBus = ((cmd<<4) & 0xF0); //Send Lower nibble
    LCD_RS = 0;   // Send LOW pulse on RS pin for selecting Command register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);
}


/* Function to send the Data to LCD.
   As it is 4bit mode, a byte of data is sent in two 4-bit nibbles */
```

```
void Lcd_DataWrite(char dat)
{
    LcdDataBus = (dat & 0xF0);      //Send higher nibble
    LCD_RS = 1;   // Send HIGH pulse on RS pin for selecting data register
    LCD_RW = 0;   // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;   // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);

    LcdDataBus = ((dat<<4) & 0xF0);  //Send Lower nibble
    LCD_RS = 1;    // Send HIGH pulse on RS pin for selecting data register
    LCD_RW = 0;    // Send LOW pulse on RW pin for Write operation
    LCD_EN = 1;    // Generate a High-to-low pulse on EN pin
    delay(1000);
    LCD_EN = 0;

    delay(10000);
}



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

    LcdDataBusDirnReg = 0x00;  // Configure all the LCD pins as output


    Lcd_CmdWrite(0x02);        // Initialize Lcd in 4-bit mode
    Lcd_CmdWrite(0x28);        // 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);
}
```

gwan/3827aa5155b70f2745ea53ef2dd44574/raw/79cc2b57123fc8a91908f8d4630539c1806677e3/8051_Lcd4BitExample1.c)
8051_Lcd4BitExample1.c (https://gist.github.com/SaheblalBagwan/3827aa5155b70f2745ea53ef2dd44574#file-8051_lcd4bitexample1-c) hosted with ♡ by **GitHub (https://github.com)**

Using Explore Embedded Libraries :

In the above tutorial, we just discussed how to interface 2x16Lcd in 4-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).

The below sample code shows how to use the already available LCD functions.

LCD 1x16

```
#include "lcd.h"

int main()
{
    /*Connect RS->P2.0, RW->P2.1, EN->P2.2 and data bus to P2.4 to P2.7*/
    LCD_SetUp(P2_0,P2_1,P2_2,P_NC,P_NC,P_NC,P_NC,P2_4,P2_5,P2_6,P2_7);
    LCD_Init(1,16);

    LCD_DisplayString("Explore Embedded\n");
    while(1);

    return (0);
}
```

aheblalBagwan/3329928508b185f1fad661a07f28d6c6/raw/59607f067c4c1f0297081043292ed0abff4862ff/8051_lcd_1x16.c)
8051_lcd_1x16.c (https://gist.github.com/SaheblalBagwan/3329928508b185f1fad661a07f28d6c6#file-8051_lcd_1x16-c)
hosted with ♡ by **GitHub (https://github.com)**

LCD 2x16

```
#include "lcd.h"

int main()
{
    /*Connect RS->P2.0, RW->P2.1, EN->P2.2 and data bus to P2.4 to P2.7*/
    LCD_SetUp(P2_0,P2_1,P2_2,P_NC,P_NC,P_NC,P_NC,P2_4,P2_5,P2_6,P2_7);
    LCD_Init(1,16);

    LCD_DisplayString("Explore Embedded");
    LCD_GoToLine(1);
    LCD_DisplayString("Lcd 4-Bit mode");
    while(1);

    return (0);
}
```

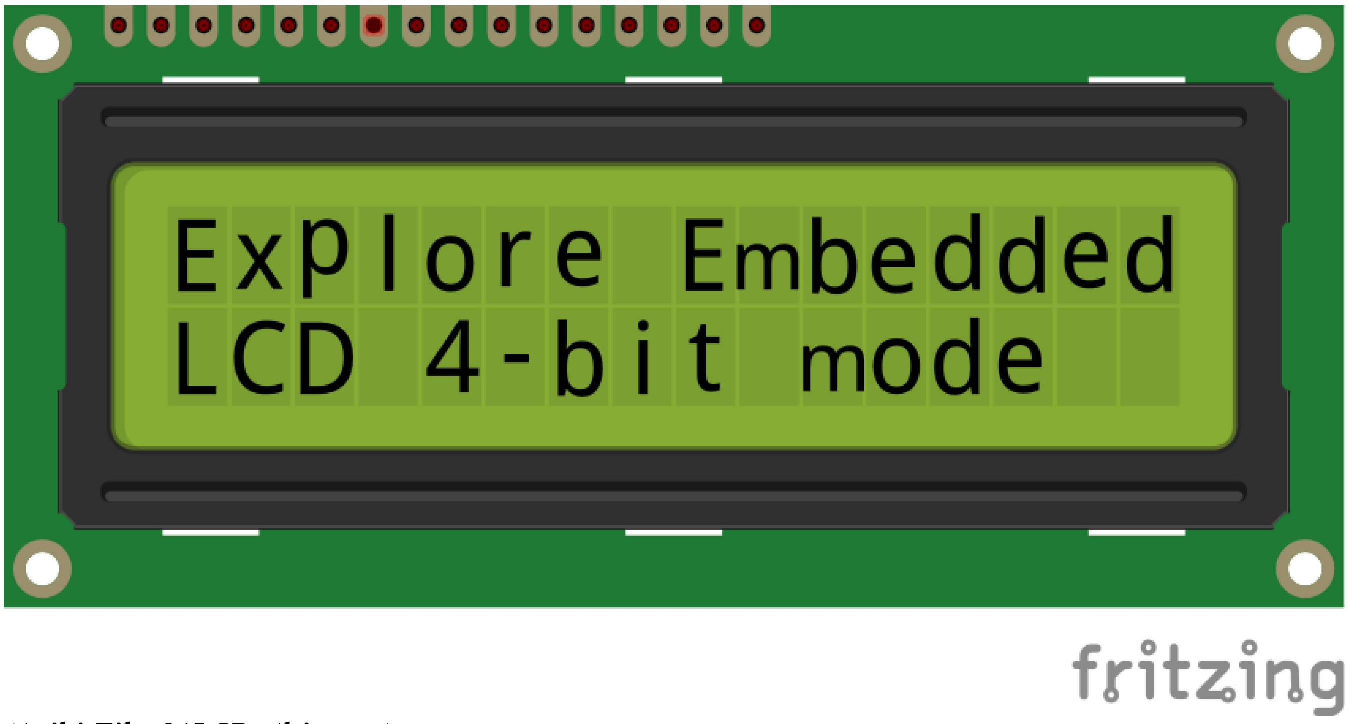
heblalBagwan/05cb28c2b0ee15b69bd630c498e1bcdb/raw/b1c87916efcb0d7ff8565d682d331eeec3e2f0ef/8051_lcd_2x16.c)
8051_lcd_2x16.c (https://gist.github.com/SaheblalBagwan/05cb28c2b0ee15b69bd630c498e1bcdb#file-8051_lcd_2x16-c)
hosted with ♡ by **GitHub (https://github.com)**

LCD 4x20

```
#include "lcd.h"

int main()
{
    /*Connect RS->P2.0, RW->P2.1, EN->P2.2 and data bus to P2.4 to P2.7*/
    LCD_SetUp(P2_0,P2_1,P2_2,P_NC,P_NC,P_NC,P_NC,P2_4,P2_5,P2_6,P2_7);
    LCD_Init(4,20);

    LCD_DisplayString("Explore Embedded\n");
          LCD_DisplayString("LCD 4-bit Mode\n");
```

```
        LCD_DisplayString("20 x 4 \n");
        LCD_DisplayString(":)  :O");
        while(1);

    return (0);
}
```

heblalBagwan/7c4da4bc35c7ffc5e964e1c31c673497/raw/582f1db625d0b5633e9550afbd2ef1bebe1635f6/8051_lcd_4x20.c)
8051_lcd_4x20.c (https://gist.github.com/SaheblalBagwan/7c4da4bc35c7ffc5e964e1c31c673497#file-8051_lcd_4x20-c)
hosted with ♡ by GitHub (https://github.com)



(/wiki/File:01LCD_4bit.png)

Downloads

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

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

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

Subscribe to hear about our latest Explorations!