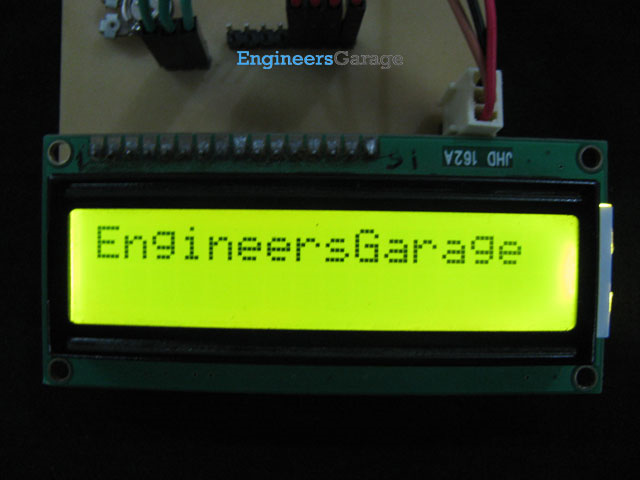
**[How to interface 16x2 LCD in 4-bit mode with PIC Microcontroller (PIC18F4550)](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/lcd-interfacing-4bit-mode-circuit" \o "How to interface 16x2 LCD in 4-bit mode with PIC Microcontroller (PIC18F4550))**



The [16x2 character LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet) can work in two modes, namely, 8-bit and 4-bit. These modes basically correspond to the number of data pins used in interfacing LCD. 8-bit mode uses all the data lines and has been explained in [LCD interfacing with PIC18F4550](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-lcd-circuit). In 4-bit mode, only four data pins of LCD are connected to the controller. This mode, thus, saves four pins of the controller unlike 8-bit mode. The configuration and display method of LCD in 4-bit mode has been explained here.

The [8-bit mode of LCD interfacing with PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-lcd-circuit) has been explained earlier. In the 4-bit mode the (8-bit) data/command is sent in nibble (four bits) format to [LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet). The higher nibble is sent first followed by the lower nibble. In 4-bit mode only four data pins (D4-D7) of LCD are connected to the controller. The control pins (RS, RW and EN) are connected the same way as in 8-bit mode. The connections of LCD with [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) are shown in the adjoining circuit diagram. Please note that here only PortB is used to connect data lines as well as control lines unlike in 8-bit mode. Refer [LCD interfacing with PIC in 8-bit mode](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-lcd-circuit).

[LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet) is configured for 4-bit mode by sending appropriate instruction known as Function Set. The Function Set is hexadecimal instruction for LCD MPU unit which selects the working modes of LCD. The Function Set is given below along with its description.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RS** | **RW** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| **Function Set** | 0 | 0 | 0 | 0 | 1 | **DL** | **N** | **F** | - | - |

**Description:**

**DL**       -           Data Length

**N**       -           No. of Lines

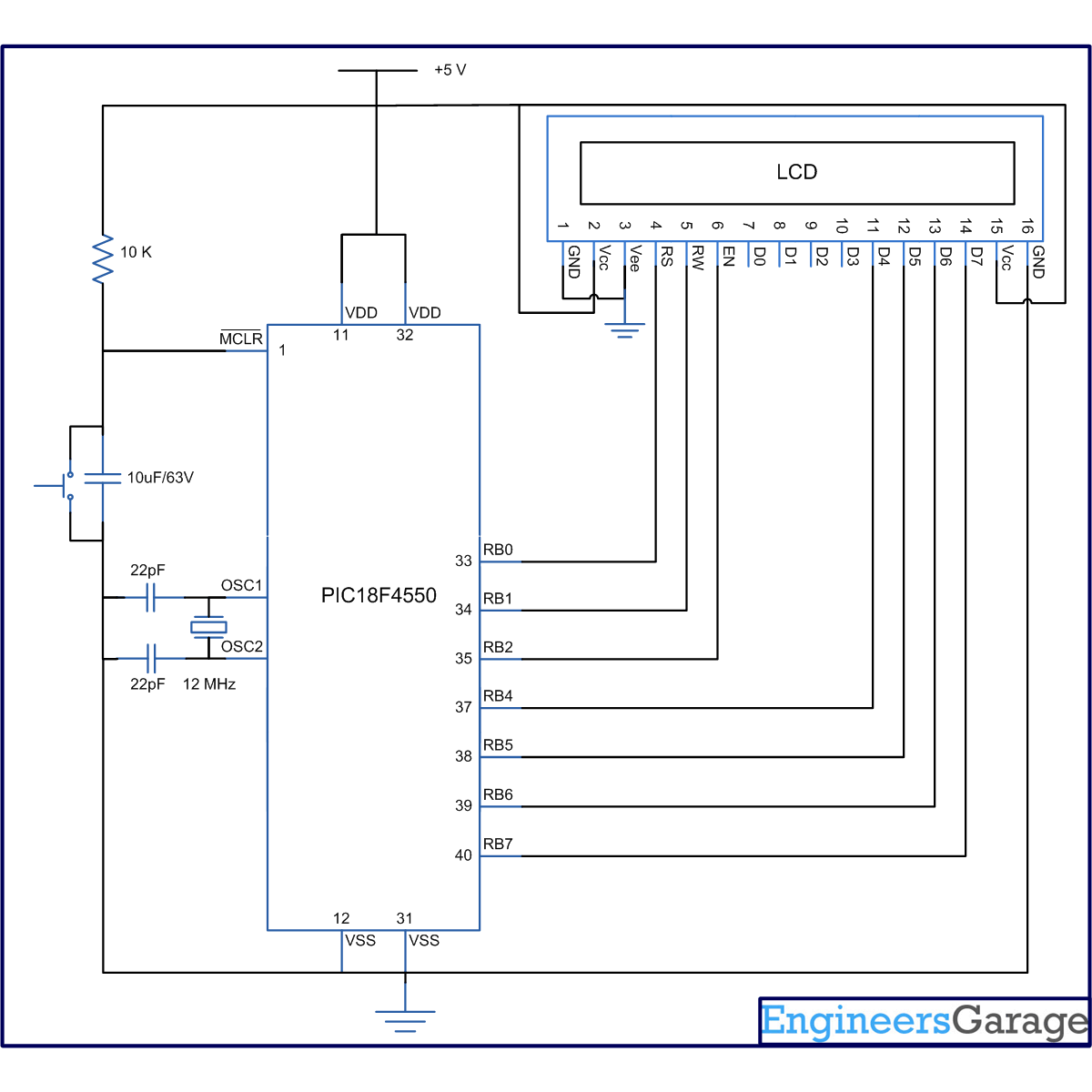
**F**        -           Font

|  |  |  |  |
| --- | --- | --- | --- |
| **Value** | **DL** | **N** | **F** |
| **1** | 8 bit | 2 lines | 5x10 dots |
| **0** | 4 bit | 1 line | 5x7 dots |

According to the table, the value of Function Set for 4–bit mode will be [ 0010 0000 ] 0x20. The value of Function Set for the LCD configuration : 2 line (N=1), 5x7 dots (F=0) and 4-bit (DL=0) mode will be [ 0010 1000 ] 0x28.

It is important to note that when the power supply is given to LCD, it remains in 8-bit mode. In this state if 0x20 is sent, lower nibble will not be received by LCD because only four data lines (D4-D7) are connected, so 0x02 is sent instead of 0x20.

For more details on nibble arrangement and bit shifting, refer [LCD 4-bit mode with AVR](http://www.engineersgarage.com/embedded/avr-microcontroller-projects/interface-lcd-4bit-mode-circuit).



// Program to interface 16x2 LCD with PIC18F4550 Microcontroller using 4-bit mode  
  
// Configuration bits  
/\* \_CPUDIV\_OSC1\_PLL2\_1L, // Divide clock by 2  
 \_FOSC\_HS\_1H, // Select High Speed (HS) oscillator  
 \_WDT\_OFF\_2H, // Watchdog Timer off  
 MCLRE\_ON\_3H // Master Clear on  
\*/  
  
//LCD Control pins  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
  
//LCD Data pins  
#define lcdport LATB  
  
void lcd\_ini();  
void dis\_cmd(unsigned char);  
void dis\_data(unsigned char);  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);  
  
void main(void)  
{  
 unsigned char data0[]="EngineersGarage";  
 unsigned int i=0;  
 TRISB=0; // Configure Port B as output port  
 LATB=0;  
 lcd\_ini(); // LCD initialization  
 while(data0[i]!='\0')  
 {  
 dis\_data(data0[i]);  
 Delay\_ms(200);  
 i++;  
 }  
}  
void lcd\_ini()   
{  
 dis\_cmd(0x02); // To initialize LCD in 4-bit mode.  
 dis\_cmd(0x28); // To initialize LCD in 2 lines, 5x7 dots and 4bit mode.  
 dis\_cmd(0x0C);  
 dis\_cmd(0x06);  
 dis\_cmd(0x80);  
}  
  
void dis\_cmd(unsigned char cmd\_value)  
{  
 unsigned char cmd\_value1;  
 cmd\_value1 = (cmd\_value & 0xF0); // Mask lower nibble because RB4-RB7 pins are being used  
 lcdcmd(cmd\_value1); // Send to LCD  
 cmd\_value1 = ((cmd\_value<<4) & 0xF0); // Shift 4-bit and mask  
 lcdcmd(cmd\_value1); // Send to LCD  
}  
  
  
void dis\_data(unsigned char data\_value)  
{  
 unsigned char data\_value1;  
 data\_value1=(data\_value&0xF0);  
 lcddata(data\_value1);  
 data\_value1=((data\_value<<4)&0xF0);  
 lcddata(data\_value1);  
}  
  
void lcdcmd(unsigned char cmdout)  
{  
 lcdport=cmdout; //Send command to lcdport=PORTB  
 rs=0;   
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}  
  
void lcddata(unsigned char dataout)  
{  
 lcdport=dataout; //Send data to lcdport=PORTB  
 rs=1;  
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}