<https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/>

**PIC16F877A – LCD 4Bit Interfacing**

In previous LCD tutorials we were used that LCD in 8 bit mode. But here we will see LCD 4 bit interfacing with PIC16F877A Microcontroller.

**Post Contents**[[hide](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/)]

* [1 LCD 4 bit interfacing with PIC16F877A](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#LCD_4_bit_interfacing_with_PIC16F877A)
* [2 Introduction](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Introduction)
* [3 LCD Initializing](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#LCD_Initializing)
* [4 Sending command](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Sending_command)
* [5 Sending Data](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Sending_Data)
* [6 Circuit Diagram](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Circuit_Diagram)
* [7 Code](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Code)
* [8 Output](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Output)
  + - [8.0.1 Share this:](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Share_this)
    - [8.0.2 Like this:](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Like_this)
    - [8.0.3 Related](https://embetronicx.com/tutorials/microcontrollers/pic16f877a/lcd-4-bit-interfacing-with-pic16f877a/#Related)

**LCD 4 bit interfacing with PIC16F877A**

**Introduction**

**8 bit mode** –  Using 8 data lines in LCD (totally 8 data lines are there)

**4 bit mode** –  Using only 4 data lines in LCD module

8 bit mode is already working and that is looks awesome. Then why we are going to 4 bit mode? This is the question comes in every mind whenever i said 4 bit mode. Yeah that 8 bit mode is nice. But but but… Just assume. I’m doing one project which requires more number of hardwares. But PIC16F877A have only 33 GPIOs. So in that time i can use this 4 bit mode and reduce the pin required for LCD module. Am i right? Great. That’s why 4 bit mode also important. Already we know the LED’s operation. If we want to enable 4 bit mode we have to do small modification in normal method. Let’s see that.

In initializing time we have to give 0x28 command. That’s all.

**LCD Initializing**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | void lcd\_init()  {      cmd(0x02);      cmd(0x28);      cmd(0x0e);      cmd(0x06);      cmd(0x80);  } |

**Sending command**

Here everything is same except way of data writing. Here we have only 4 bits. So we need to send nibble by nibble. So first we need to send first nibble then followed by second. See that code. I’m writing into the Port B’s last 4 bits. Because last 4 bits are connected into LCD.

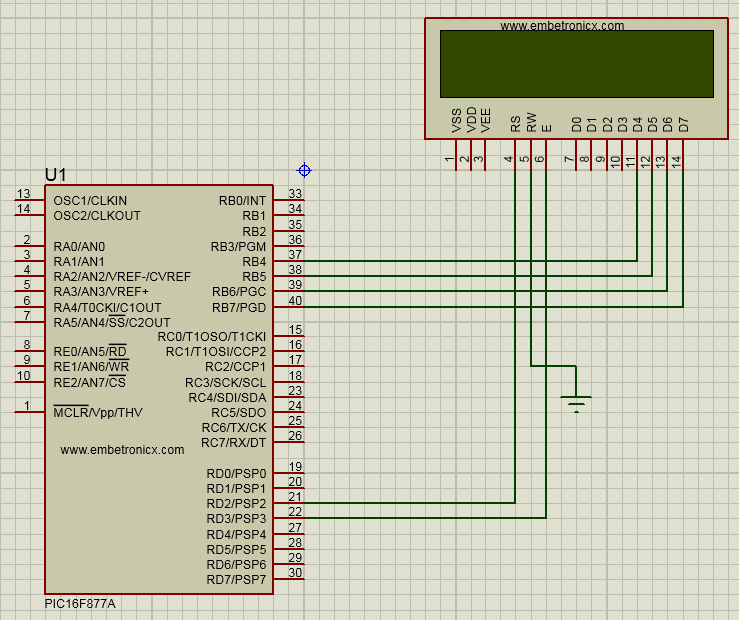
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | void cmd(unsigned char a)  {      rs=0;      PORTB&=0x0F;      PORTB|=(a&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();      PORTB&=0x0f;      PORTB|=(a<<4&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();  } |

**Sending Data**

Same as sending command.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | void dat(unsigned char b)  {      rs=1;      PORTB&=0x0F;      PORTB|=(b&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();      PORTB&=0x0f;      PORTB|=(b<<4&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();  } |

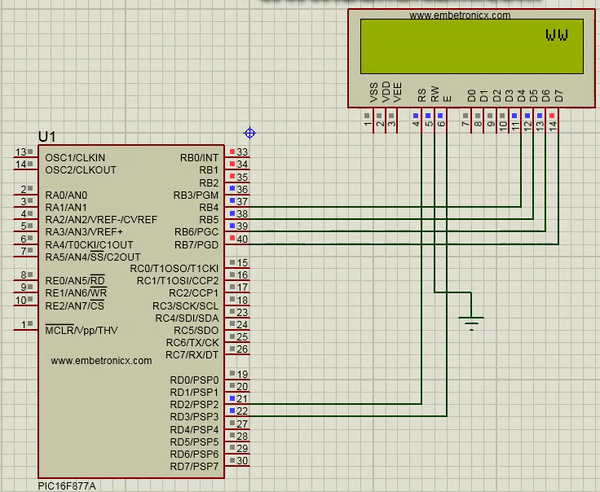
**Circuit Diagram**



**Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85 | #include <pic.h>    \_\_CONFIG( FOSC\_HS & WDTE\_OFF & PWRTE\_OFF & CP\_OFF & BOREN\_ON & LVP\_OFF & CPD\_OFF & WRT\_OFF & DEBUG\_OFF);      #define rs RD2  #define en RD3    void lcd\_init();  void cmd(unsigned char a);  void dat(unsigned char b);  void show(unsigned char \*s);  void lcd\_delay();    void main()  {      unsigned int i;      TRISB=TRISD2=TRISD3=0;      lcd\_init();      cmd(0x90);      show("www.EmbeTronicX.com");      while(1)      {          for(i=0;i<15000;i++);          cmd(0x18);          for(i=0;i<15000;i++);        }  }    void lcd\_init()  {      cmd(0x02);      cmd(0x28);      cmd(0x0e);      cmd(0x06);      cmd(0x80);  }    void cmd(unsigned char a)  {      rs=0;      PORTB&=0x0F;      PORTB|=(a&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();      PORTB&=0x0f;      PORTB|=(a<<4&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();  }    void dat(unsigned char b)  {      rs=1;      PORTB&=0x0F;      PORTB|=(b&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();      PORTB&=0x0f;      PORTB|=(b<<4&0xf0);      en=1;      lcd\_delay();      en=0;      lcd\_delay();  }    void show(unsigned char \*s)  {      while(\*s) {          dat(\*s++);      }  }    void lcd\_delay()  {      unsigned int lcd\_delay;      for(lcd\_delay=0;lcd\_delay<=1000;lcd\_delay++);  } |

**Output**

[](https://embetronicx.com/wp-content/uploads/2017/07/lcd-4bit-pic16f877a-output.gif)**[**[**Please find the output image Here**](https://embetronicx.com/wp-content/uploads/2017/07/lcd-4bit-pic16f877a-output.gif)**]**

Hope you learned something from this. You can download the whole project [Here](https://github.com/Embetronicx/Tutorials/tree/master/Microcontrollers/PIC16F877A/LCD_4bit). If you have any doubt please ask us. See you in next tutorials.