

# Lesson 11

## Programming a PIC

### Overview

<b>Introduction</b>	In this lesson, we will use the FPP program to load our code into a PIC.	
<b>In this section</b>	Following is a list of topics in this section:	
	<b>Description</b>	<b>See Page</b>
	Setting up FPP	2
	Reading a PIC	5
	Programming a PIC	7
	Troubleshooting	8
	Code Protected PICs	9
	Homework	9
	Wrap Up	10

## Setting up FPP

### Introduction

Before programming our PIC, we need to check the configuration of the FPP program.

### Making a shortcut accessible

When we installed FPP, we indicated that we could put the shortcut on the start menu, or on the desktop, or both. There are reasons why we would prefer one over the other, or perhaps want something different.

When there are a large number of programs on the start menu, it can be tough to find the right program. There may be so many programs that we need to wade through multiple levels of menus. Depending on how messy you keep your desktop, it can be just as challenging.

One trick is to place the FPP icon near the bottom of the screen, then size the MPLAB window so that the FPP icon is able to peek out from underneath MPLAB. This allows quick and easy access:



Another approach is to drag the FPP icon to the Quick Launch bar. This keeps FPP always readily available, but putting too many things in the Quick Launch bar sort of defeats the purpose:



Of course, if you are using Windows<sup>®</sup> XP, FPP will quickly show up in the recently used programs panel when you are working with it, so you may not need to fish it out of the menu quite so often.

### Checking the Configuration

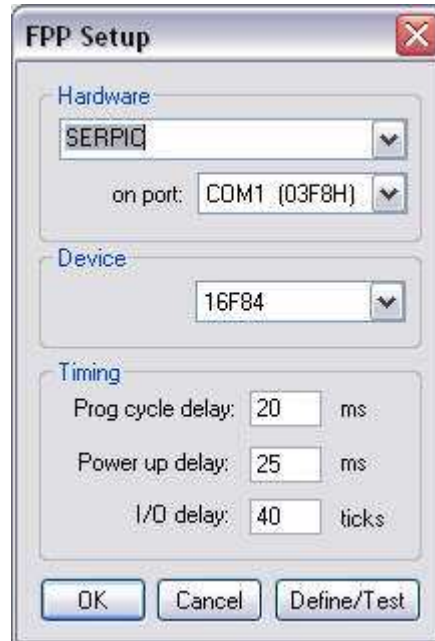
The first time you launch FPP, or if you have been making some configuration changes, you should check that the FPP configuration is properly set up. We initially did this in Lesson 10, but let's review.

Clicking on the 'Setup' button will bring up the setup window, where most of the interesting stuff lives:

*Continued on next page*

## Setting up FPP, Continued

### Checking the Configuration (continued)



The Hardware should reflect your programmer; SERPIC if you are using the PIC-EL. Similarly, the serial port address should reflect the *address* you discovered when you checked back in Lesson 10 where Windows had located your port. Notice that on some configurations, the COM port number may not reflect the address properly. It is the address that matters.

The Device should be the 16F84. A caveat here – for programming purposes, the 16F84 is no different than a 16F84A. However, this is not the case for all parts. FPP, for example, will program a 16F877, but not a 16F877A.

Generally, you should not have to make any changes in the timing section. However, if you have an especially long cable to your programmer you may find it advantageous to increase the times here. Data goes into the PIC at a speed determined by the programmer. There are no pre-set speeds like a modem. Stretching out the 'I/O delay' will slow programming, but could get around a little extra capacitance in your cable.

### The Define/Test Window

Clicking on 'Define/Test' will launch the Define/Test window. Once you have set up your programmer, you will likely not need to use this window (nor the setup window, for that matter). However, it provides a good way to test your hardware. There are a few things that are interesting here.

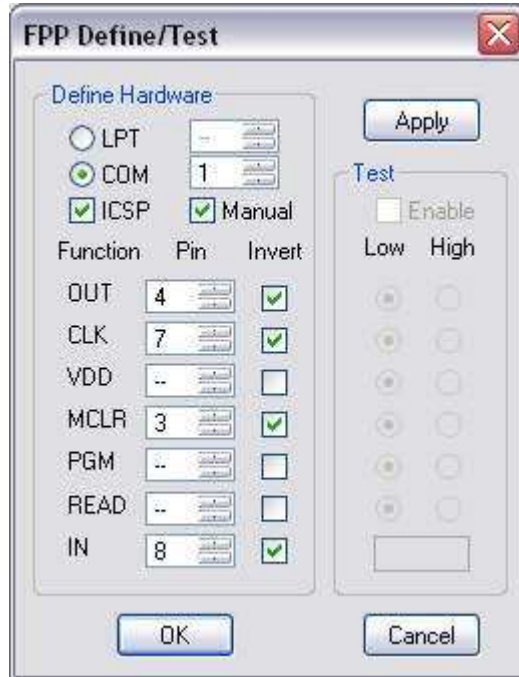
At the top of the window are displayed your COM or LPT port, no real surprises here. Underneath that are two check boxes. The ICSP box indicates that we will be programming in-circuit (as in the PIC-EL). The Manual box indicates whether there

*Continued on next page*

## Setting up FPP, Continued

### The Define/Test Window (continued)


is a switch on our programmer. All this does is control the generation of a popup prior to programming. You can un-check this box, but as will be described later, the popup is something of a convenience.



The section below Function/Pin/Invert describes how the PC is connected to the programmer. There are really only three pins required for programming; RB7, RB6, and MCLR, which are connected to **Functions** OUT, CLK and MCLR. In addition, the programmer may have logic that allows the state of RB7 to be fed back to the PC. This is the function IN. The **Pin** column contains the pin number on the PC connector to be used for this purpose. In most cases, the programmer will use a single transistor to drive the PIC pins. This will invert the signal to the PIC, thus, the **Invert** column allows this to be specified.

For the curious, the other Functions are not used on the PIC-EL, but might be on other programmers. The VDD function is used if the programmer controls the supply to the PIC. The PGM function is used if the programmer automatically operates the programming switch rather than having a physical switch. The read function is used by programmers that need to be commanded to send PIC data back to the programming software.

## Reading a PIC

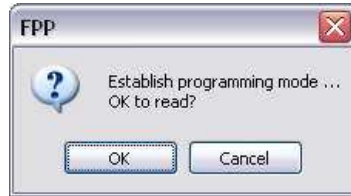
<b>Introduction</b>	With FPP, it is possible not only to program a PIC, but to read the contents of a PIC. In this section, we will examine how to read the PIC's program and save it to your hard drive.
<b>Connecting the Programmer</b>	<p>Before starting, the programmer should be connected to the appropriate port on your PC. In the case of a PIC-EL, this would be a serial port, and the cable would be a standard, straight through serial cable, with a DB9-P on one end for the PIC-EL, and the appropriate connector for your computer's serial port, usually a DB9-S, on the other.</p> <p>If you have not already done so, you should place the PIC to be read in the programmer socket.</p> <p>If the programmer requires power, you should apply power to the programmer now. (Some programmers may be powered from the serial or USB port, but many of these can be marginal with some ports.)</p> <p>Note that in the case of the PIC-EL, although most of the circuitry will run from a 9 volt battery, this is not the case with the programmer portion. You need at least 12 volts. The typical 13.8 volt station supply is ideal.</p>
<b>Launching FPP</b>	<p>If you have not already started FPP, you should do so by selecting it from your Start menu or double clicking the desktop icon, whichever you prefer. It is not necessary to have MPLAB running at this time, although it isn't a problem for MPLAB to be running, either.</p> <p>The title bar should show that you have selected the 16F84:</p>  A screenshot of a Windows window title bar. The title bar is light gray and contains a small yellow icon on the left, followed by the text 'FPP - [16F84]'. On the right side of the title bar, there are three standard Windows window control buttons: a minimize button (a horizontal line), a maximize button (a square), and a close button (an 'X' in a red square).

*Continued on next page*

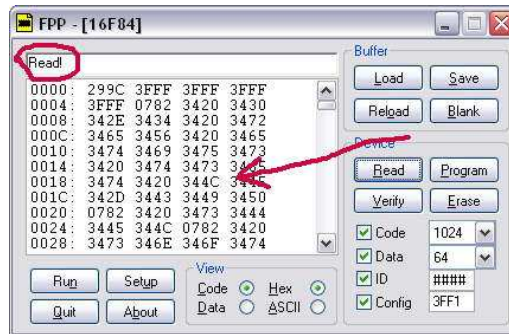
# Reading a PIC, Continued

## Reading the PIC

If you click on the 'Read' button, a dialog box will appear:



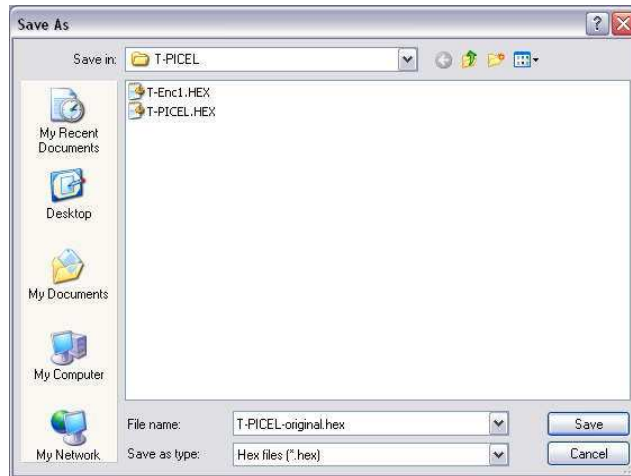
At this point, set your programmer to programming mode. In the case of the PIC-EL, moving the slide switch will cause the programming LED to illuminate. When you then click on OK, after a short wait, you should see the word 'Read!' appear in the upper white box, and a bunch of data to appear in the lower box. This is the data that was in the PIC:



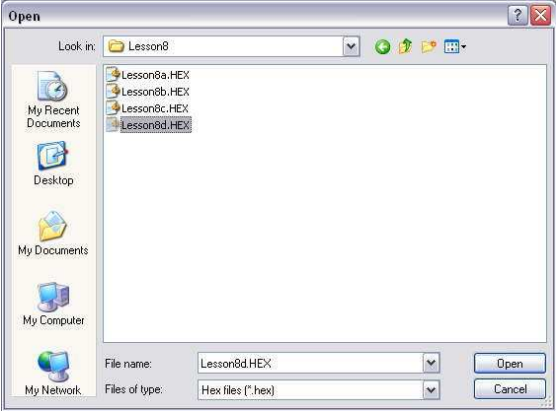
Flip the switch on the programmer back to the operate mode. Check to be sure that the data isn't all zeros or all 3fff. If it is, chances are there is a problem with your connection to the programmer.

## Saving the data

Now that you have the data from the PIC, you can save it to a file in case you want to reprogram the PIC later with the original data. Click on 'Save' and enter a name into the Save dialog box. The normal extension should be .hex.



## Programming a PIC

<b>Introduction</b>	To begin programming, be sure your programmer is connected as in the previous section. Again, launch FPP and check that the correct PIC is selected in the title bar.
<b>Loading your program into FPP</b>	At this point, click on the <u>L</u> oad button to load the desired program into FPP. You will get the normal file open dialog. For our initial experiment, load the homework assignment from Lesson 8: 
	As before, you will see the data appear in the FPP window.
<b>Programming the PIC</b>	Clicking the ' <u>P</u> rogram' button will open the same "Establish programming mode" dialog as the <u>R</u> ead button. Flip the programming switch and click OK. After a few seconds, the word Success! Should appear in the top window of FPP.  Set the switch back to run mode.
<b>Verifying the program</b>	Although not strictly necessary, it is good practice to verify the load. Click on the Verify button on FPP, flip the switch as you did earlier, and click OK. This will cause FPP to read the PIC contents and compare them to the loaded file. With luck, you will see 'Verified!' in the top box of FPP. Don't forget to set the switch back to the run mode.  This can also be handy if you have a PIC and you don't remember what you put in it. You can load the program into FPP and verify the PIC against the program without programming. Next time remember to stick a label on the PIC!
<b>Testing the program</b>	At this point, your PIC-EL should be sitting there not beeping (for a change). None of the LEDs should be on. The display will show whatever happened to be there when you flipped the programming switch.  Pressing the buttons should cause the corresponding LED to light. (If you used Lesson 8d as in the image above, only the top LED will respond).  If the program didn't work properly, it's time to go back to the Lesson 8 homework assignment and do some more debugging.

## Troubleshooting

<b>Introduction</b>	It is possible that when you clicked 'Program', FPP responded with "Failed to program code!" Now what?
<b>Voltages</b>	<p>The most common problem is having an adequate power supply. Be sure you have at least 12 volts supplied to the programmer. If you are not using the PIC-EL, check the documentation for your programmer. Some require as much as 24 volts.</p> <p>Next, revisit the "Testing the Programmer" section of Lesson 10. You should be suspicious if you can't get the voltage on pin 4 above 11 volts. This generally indicates that you aren't supplying the PIC-EL with enough voltage. In the low state, this voltage needs to be below one volt. In run mode, pin 4 should be between 4 and 5 volts.</p> <p>Some folks have reported marginal voltages at pins 12 and 13. These should swing from below one to above four. If they are close, it may be that they aren't able to get to voltage in the time needed for programming. You may try stretching out the time a little (see below).</p>
<b>Connection</b>	<p>Be sure you have the proper cable. If you are using the PIC-EL or a similar serial programmer, the cable should be reasonably short, preferably under three feet. By following the steps in Lesson 10 you should be able to discover if the cable is connected properly, however, you won't uncover excess capacitance in the cable.</p> <p>If the cable is long, or exhibits too much capacitance, you may be able to help by stretching out the times in the FPP Setup dialog.</p> <p>For those of you who are particularly interested in these things, you may try the following; Erase a PIC, program it with something long and measure the programming time, then adjust the I/O delay, and do it again. Graph the results. What you will find is that long times do stretch out the programming times, but so do short times. With short times, retries make up for the time gained in quicker programming. There is a sweet spot where the time is optimum. Erasing the PIC first allows you to get more consistent results, although more of the time will be taken up by PIC delays.</p>



## Code Protected PICs

<b>Introduction</b>	Some PICs have the code protection bit set. This prevents you from reading them or programming them.
<b>Erasing the PIC</b>	<p>If you wish to re-use a code protected PIC, you will need to erase the part first. (Note that we are still speaking of PICs with FLASH program memory – the CMOS parts cannot be reprogrammed.)</p> <p>Erasing the PIC is a simple matter of clicking the erase button. Once again, you will be prompted to put the programmer into programming mode.</p> <p>As a general rule, you don't need to erase a PIC prior to programming it.</p>
<b>Code protecting the PIC</b>	<p>There may be a case where you want to code protect a PIC. This generally isn't recommended because it can make debugging more difficult, and the hex code that can be read from a PIC usually isn't all that useful anyway. But it is a pretty simple thing to do. Just add <code>_CP_ON</code> to your configuration word:</p> <pre style="text-align: center;">__config    _XT_OSC &amp; _PWRTE_ON &amp; _WDT_OFF &amp; _CP_ON</pre> <p>After the PIC has been programmed you will get all zeroes if you attempt to read it. You will also be unable to verify it.</p>

## Homework

<b>Introduction</b>	<p>Ok, we couldn't very well stop here without giving you a homework assignment! Now that you know how to program the PIC, and you learned in Lesson 8 how to connect it to the circuit, let's revisit an earlier project. In Lesson 6 you wrote code to send TEST in CW on a bit in memory. Change that program to do it to the transmit transistor in your PIC-EL, program it into the PIC-EL, and test it (with a dummy load, of course).</p>
---------------------	--

## Wrap Up

### Summary

In this lesson, we have loaded a program into our PIC. We have seen how to erase a PIC, and how to read what is already there. We have used FPP's Verify function to validate that we properly programmed the PIC, and we have tested the program on actual hardware.

At this point, we have learned most of the PIC instructions, we have seen how to move from a thought to a program, and we have learned how to make our PIC influence the circuit where it lives. We have written a program, assembled it, loaded it into a PIC, and tested it. We're ready to rock and roll!

### Coming Up

In the next few lessons, we will start to do more experiments on the actual hardware. We'll start off slow, but we will investigate how to read an encoder, how to manipulate the LCD, and we will talk about how to put these things together into useful applications.

It is tempting to start doing other experiments. Well, what are you waiting for? We will do quite a number of different things in the coming weeks, but that is no reason for you to hold off. It's time to play!