
1 : RUNNING BEFORE YOU LEARN TO WALK

Starting at the beginning is no fun. This book propels you directly into programming and using PIC devices with an absolute minimum of preliminary and background information.

You will want to understand about binary and about hexadecimal numbering systems, about registers, memory, instruction sets and lots of other things, but all of that can come later.

What is a PIC?

'PIC' is short for 'Peripheral Interface Controller'. This is nice to know, but does not tell you much. A PIC device is an integrated circuit which contains an almost complete computer system, small but perfectly formed.

By programming the PIC, you can change its function as part of any electronic control circuit you want to design. PICs do not require complex support circuitry and are so easy to use that you will find real benefits in simplifying the design and construction of quite modest control circuits. PICs are not expensive and there are likely to be significant cost savings.

Microchip Technology Inc., based in Arizona, manufactures a large and growing family of PIC devices. It would be confusing to describe the range and variety of different devices when you are first learning about PICs. It is much better to focus on a single device. Once you have a working knowledge of one PIC, it is easier to understand the features and capabilities of other devices.

The PIC16F627 is a good choice for initial investigation. Look at the diagram below, which shows the pin connections for the PIC16F627.

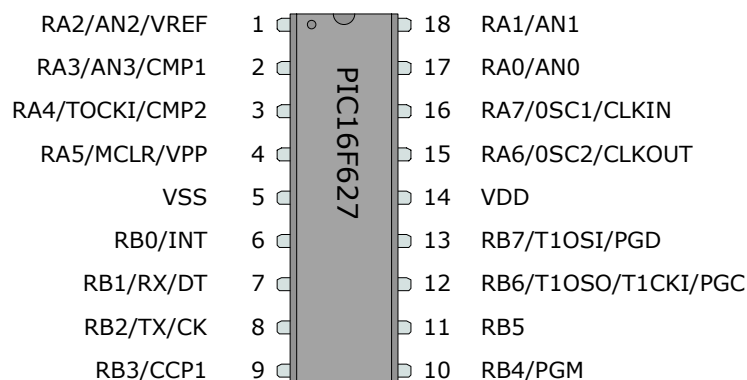


Fig. 1.1 Pin connections for PIC16F627.

As you can see, the PIC16F627 is an 18-pin integrated circuit, or 'beastie', with the pins numbered in the usual sequence, down the left-hand side and back up on the right.

It is surprising to find that most of the pins of the PIC can have two, three, or even four different functions! These functions are set when the PIC is programmed. The PIC16F627 is an extremely powerful and versatile device. Again, it is going to be easier to restrict the functions of the pins to the most obvious ones until these have been learnt and understood.

The next diagram shows the options which will be used for the first few projects described in this book.

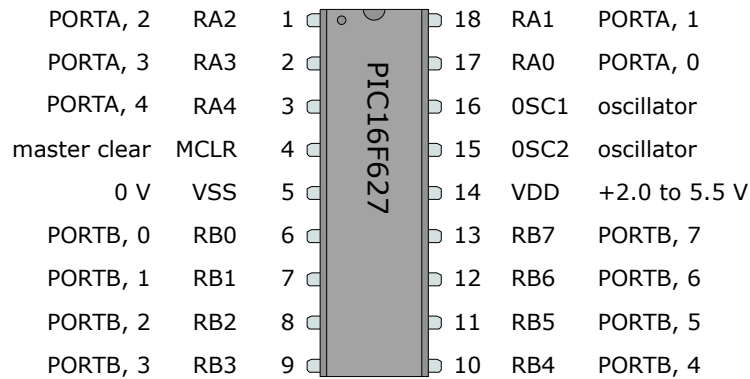


Fig. 1.2. Simplified pin connections for PIC16F627.

The **power supply** for the PIC is connected to pin 5, VSS 0 V, and pin 14, VDD +2.0 to 5.5 V. It is important to notice two things. First, these connections are different from the arrangement found in most logic integrated circuits, which have the power supply pins at opposite corners. Second, the *maximum* power supply voltage is just 5.5 V. If you connect it to a higher voltage, the PIC will fry.

Look at the PORTA and PORTB connections. PORTA accounts for five pins in this simplified arrangement, and PORTB for eight pins. Each of these pins can be individually programmed to act as an **input**, or as an **output**, giving tremendous flexibility in controlling an external circuit.

These are digital inputs and outputs which process HIGH and LOW voltages only. PORTA of the PIC16F627 can respond to varying voltage, or analogue signals as well. You will find out about this later.

To get the PIC to function, a **clock signal** is required. The PIC16F627 can be configured to produce this internally. Two different clock frequencies are available, 4 MHz and 37 kHz. In effect, you can allow the PIC to work at its usual speed, or you can slow it down to save power, or to see what is happening.

Sometimes a clock signal is generated by a connecting a circuit to the OSC1 and OSC2 pins.

The circuit diagram, Fig. 1.3 opposite, shows the essential connections for PIC operation. The 4.5 V power supply consists of three 1.5 V cells. (You can use AA cells in a suitable holder.) The 47 μ F and 100 nF capacitors provide power supply decoupling. That

is, they help to prevent the transmission of spikes and glitches along the power supply connections.

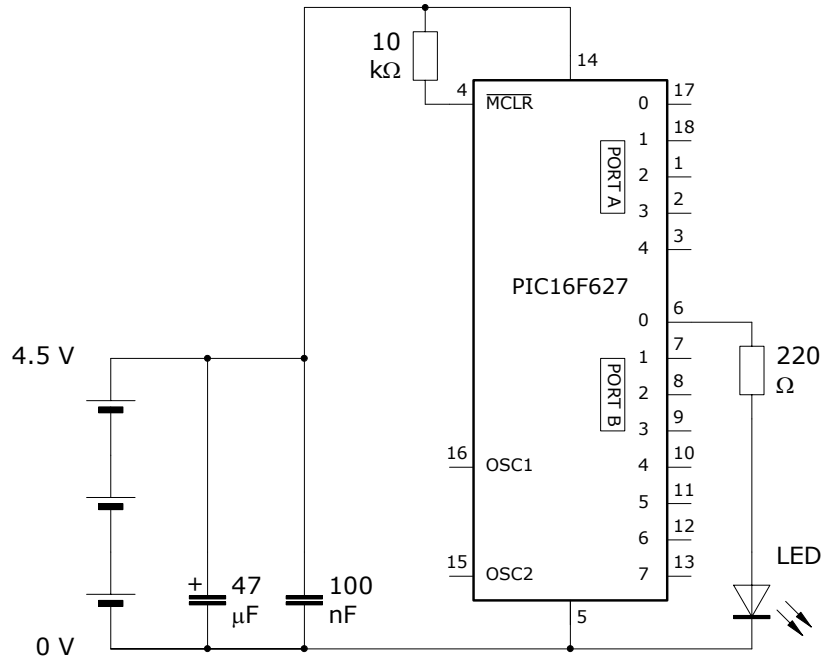


Fig. 1.3. Your first PIC circuit.

The master clear or reset pin, MCLR, of the PIC is wired to the positive power supply via a 10 kΩ resistor. This leaves the 220 Ω resistor and the LED connected to PORTB. Your mission in working through the rest of this chapter is to program the PIC to make the LED flash at a regular rate. This is a sensible initial target, and there is no doubt that you will be pleased when you succeed!

MPLAB®

MPLAB® is the *Microchip* name for the software application which supports PIC development work.

You get a copy of MPLAB as part of the PICSTART® Plus development package. In addition, the package contains the PICSTART® Plus development programmer, a suitable power supply, connecting leads and a data library on CD-ROM. Schools and colleges are strongly recommended to purchase this package, which offers a straightforward and reliable pathway into PIC programming.

MPLAB is also available for free download from the *Microchip* website at:

<http://www.microchip.com>

It is probably better to use the version of MPLAB shipped with the programmer initially, but you should aim to update to the most recent version of the software from time to time.

The downloaded file is in *.zip format, that is, it is compressed to save space, so that it downloads more quickly. Nevertheless, with a standard internet connection, the download may take a long time. Save the file to disk and double-click to start installation.

The default installation directory for recent versions of MPLAB is C:\Program Files\MPLAB IDE\. It is important for the files to be in their default locations if you want the program examples in the book to work without modification.

Launch MPLAB by clicking on its desktop icon, or by selecting the program from the Start menu.



Fig. 1.4. MPLAB desktop icon.

The application window looks like this:

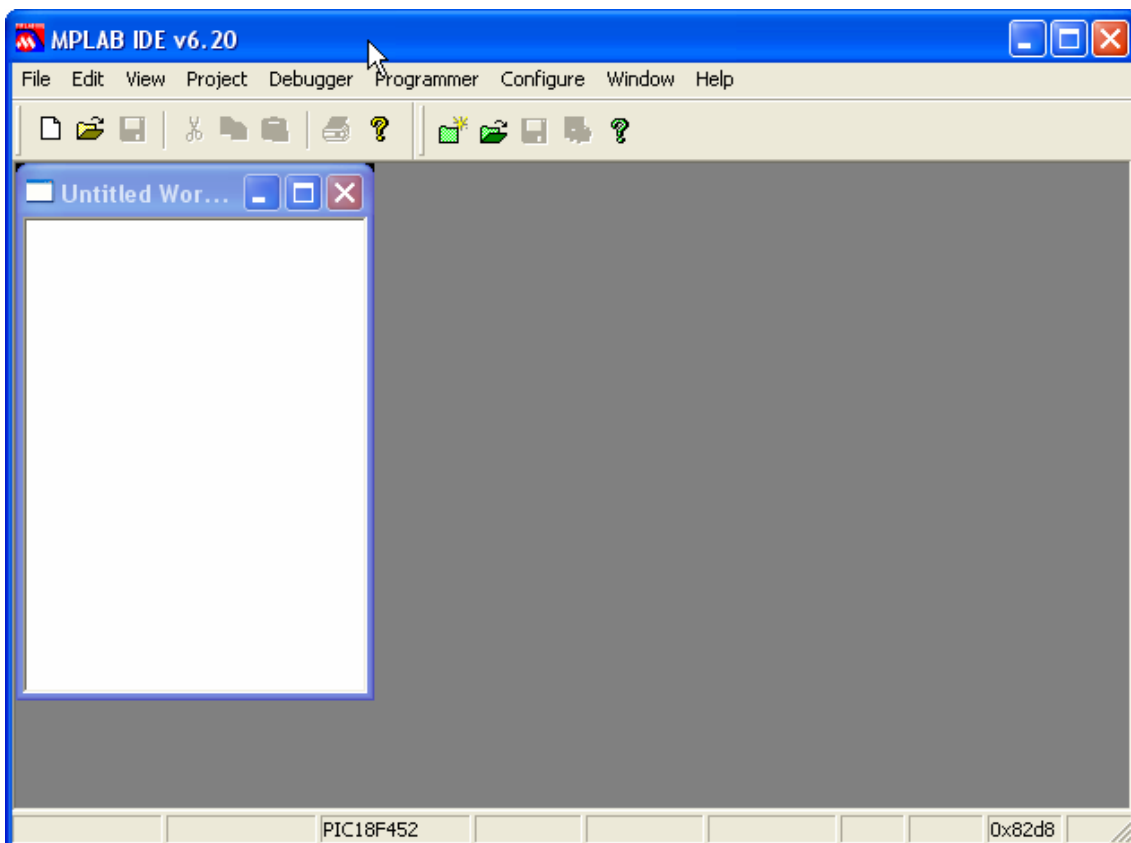


Fig. 1.5. MPLAB application window.

What do you need to do? The first step is to tell MPLAB which PIC you are going to use. From the menu bar, select Configure/Select Device... as follows:

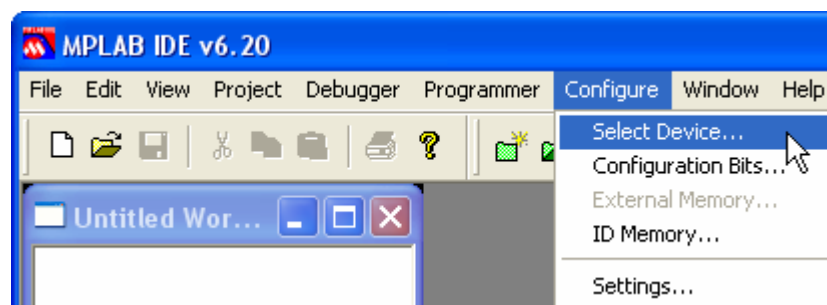


Fig. 1.6 Which PIC are you going to use?

This brings up a dialogue box:

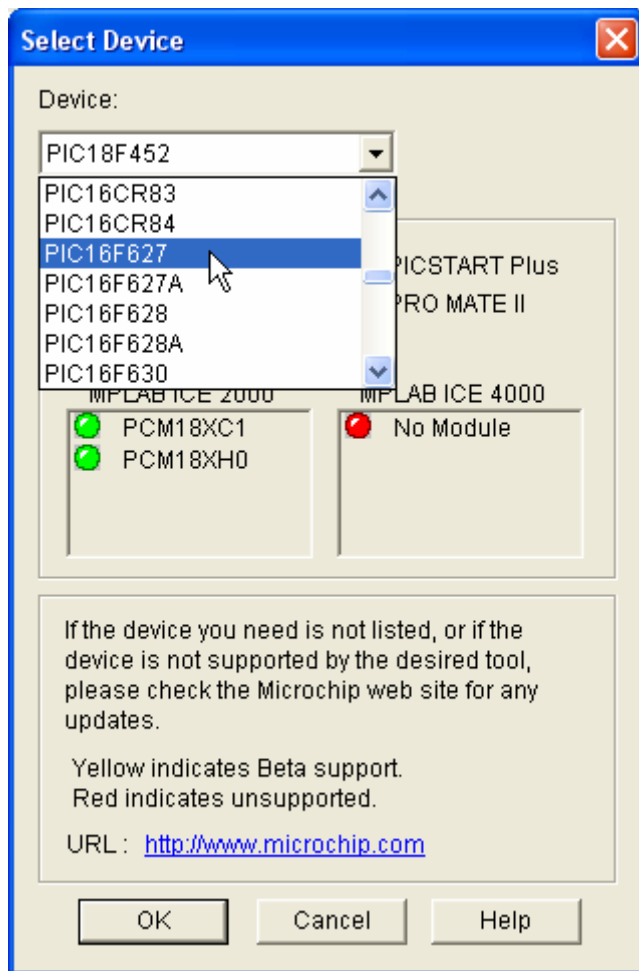


Fig. 1.7. Selecting the PIC.

Select the PIC16F627 from the drop down list and click OK. The selected device appears in the status bar at the bottom of the application window:

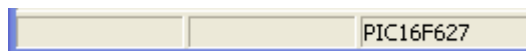


Fig. 1.8. Status bar display.

Next, you need to create a project which will contain the program files. Select Project/New... from the menu bar:

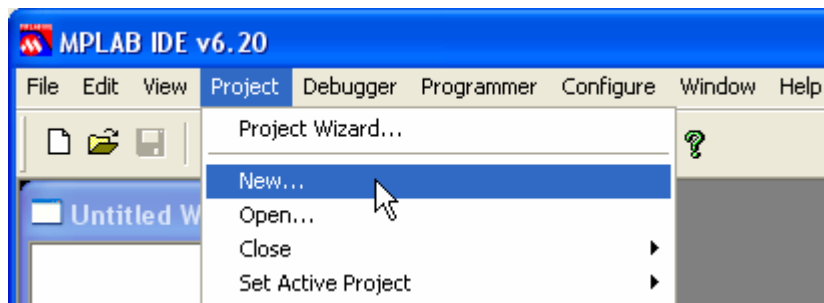


Fig. 1.9. Creating a project.

In the dialogue box, type a name for your project. Since you are going to make the LED flash, 'flash' seems appropriate:

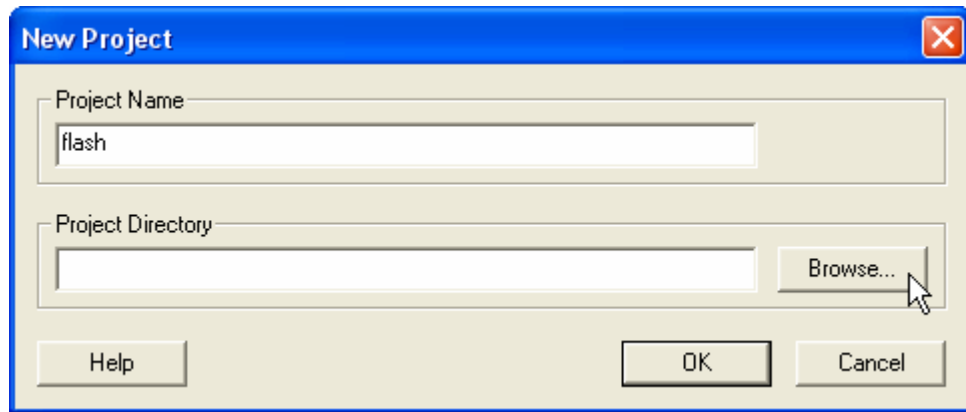


Fig. 1.10. *New Project* dialogue box.

The question of where to save your project files depends on whether you are using a stand alone computer, or one which is connected to a network.

With a stand alone machine, enter a directory name such as C:\PIC Projects\ or browse to another directory of your choice. If C:\PIC Projects\ does not exist, MPLAB will create it for you.

With a network computer, you will not be allowed to create a new directory on the C: drive. To get the files to save properly, you will need to map the network drive where your documents are stored so that a drive letter can be assigned to it.

To do this, open My Computer from the Start menu, or from the desktop and select Tools/Map Network Drive...

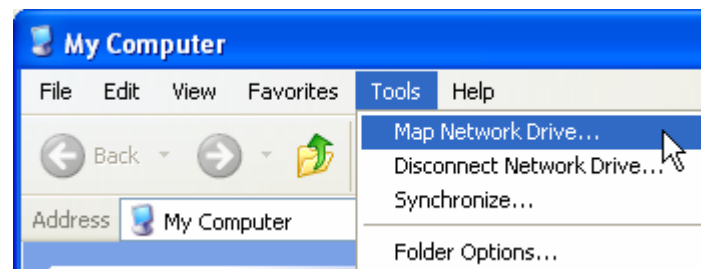


Fig. 1.11. *Mapping a network drive.*

In the dialogue box, Fig. 1.13 opposite, you can select a drive letter and browse to the location on the network where your files are stored. Return to the dialogue box and complete the process by clicking Finish.

A network drive icon appears in the My Computer window:

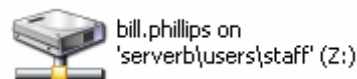


Fig. 1.12. *Network drive icon.*

Network drives are assigned letters from Z to A, and local drives, like floppy and hard drives, are assigned letters from A to Z.

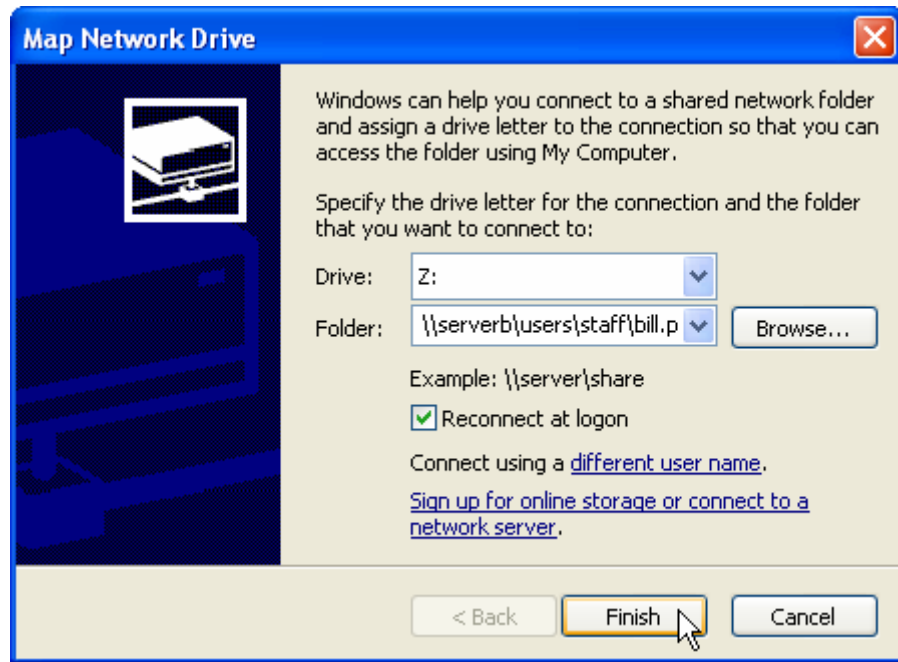


Fig. 1.13. Mapping a network drive.

For network users, the MPLAB New Project dialogue box will look something like this:

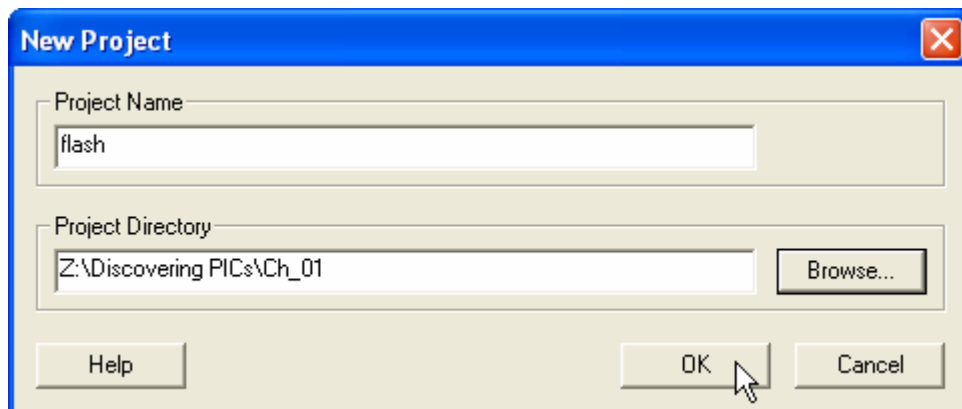


Fig. 1.14. Using a network drive.

You can tell that you have succeeded in setting up the new project because the MPLAB 'workspace' now shows a little directory tree ready to receive the project files:

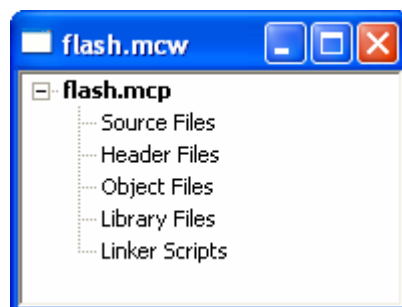


Fig. 1.15. Project directory tree.

Next, you are going to create a source file and add it to the project. The source file is the program which will control the PIC.


```

;-----;
;           DEFINE REGISTERS           ;
;-----;

loop      EQU      0x20                ;delay loop counter

;Program code starts here:
;-----;
;LABEL    OPCODE    OPERAND            ;COMMENT            ;
;-----;

init      BSF       STATUS, RP0        ;select bank 1
          BCF       PCON, OSCF         ;select 37 kHz
          MOVLW     0x00
          MOVWF     TRISB              ;PORTB all outputs
          BCF       STATUS, RP0        ;revert to bank 0

LED_on    BSF       PORTB, 0

delay_1   DECFSZ   loop, 1             ;delay time depends on
          GOTO     delay_1             ;oscillator frequency

LED_off   BCF       PORTB, 0

delay_2   DECFSZ   loop, 1
          GOTO     delay_2

          GOTO     LED_on

;-----;

          END

```

Fig. 1.18. Your first PIC program.

As you enter the program, note that anything which comes after a semicolon ; on the same line is interpreted as a comment, You can alter the details in the table at the beginning of the program in whatever way is appropriate. It is not a problem at this stage if most of what you type does not make obvious sense.

Be careful to distinguish between the number 0 and the letter O. Depending on the font used, these may be difficult to tell apart in the editor window.

Save the file frequently.

Try right-clicking over the editor window and select **Properties...** from the bottom of the resulting menu. This brings up a dialogue box from which you can change the text font and the TAB positions within the program text. With TABs at 10 character intervals, the program is easy to read and its separation into columns is obvious.

Save the file once more.

Adding the source file to the project

The source file, `flash.asm`, exists but is not yet part of the `flash.mcp` project. Right-click **Source Files** in the directory tree, as shown in Fig. 1.19, on the next page.

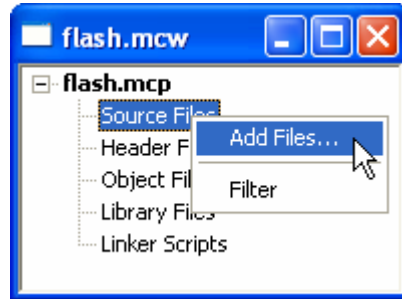


Fig. 1.19. Adding source files (right-click).

Select flash.asm from the dialogue box, click Open and check that the file is added to the directory tree:



Fig. 1.20. Directory tree showing the new file.

Assembling the program

You have written the program, but is it free from errors and will it work? To find out, you need to 'assemble' the program, converting the text file into code which can be transferred to the PIC. The code is a whole series of 0's and 1's which would be extremely difficult to work with directly.

To pass the program to the assembler, select Project/Build All, as follows:

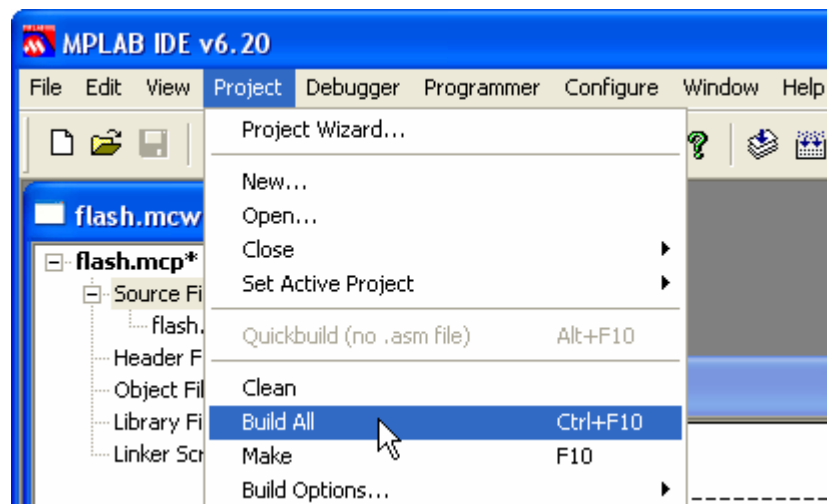


Fig. 1.21. Starting the assembler.

The assembler is a separate program within MPLAB, called MPASM. As well as generating the code, the assembler checks your program for errors and reports any which it finds.

The MPASM window appears briefly showing that the assembler is doing its stuff:

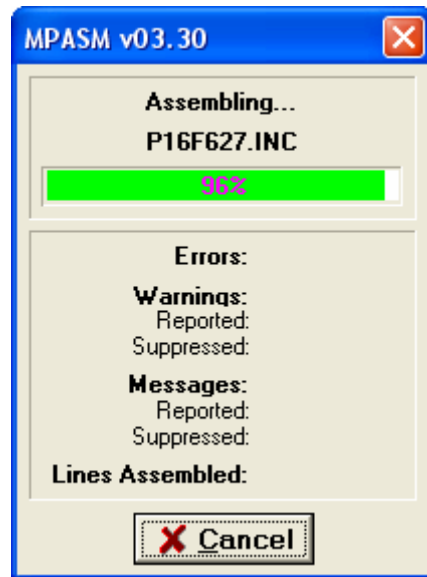


Fig. 1.22. MPASM window.

It is likely that the progress bar will turn red and that there will be errors. Double-clicking on the error message in the Output window directs you to the part of the program in which an error has been detected.

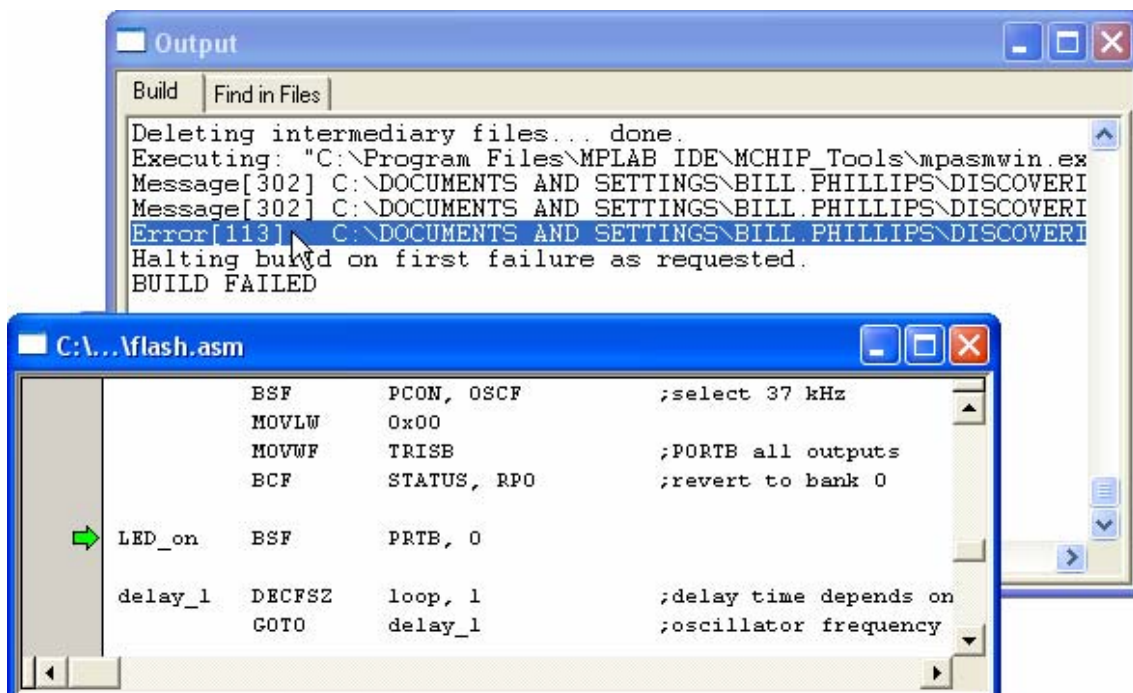


Fig. 1.23. Locating errors.

If there are lots of errors, it may be that MPASM has failed to find the P16F627.inc header file.

If necessary, check the location of the P16F627.inc file using My Computer. Modify the path so that it points to the file correctly or copy the file into the project directory.

Check carefully against the printed version of the program. Once everything is correct, the Output window will show the message BUILD SUCCEEDED on the last line.

This means that the program has assembled correctly and that the code is ready for transfer to the PIC.

PICSTART® Plus programmer

The diagram below shows the appearance of the PICSTART® Plus programmer:

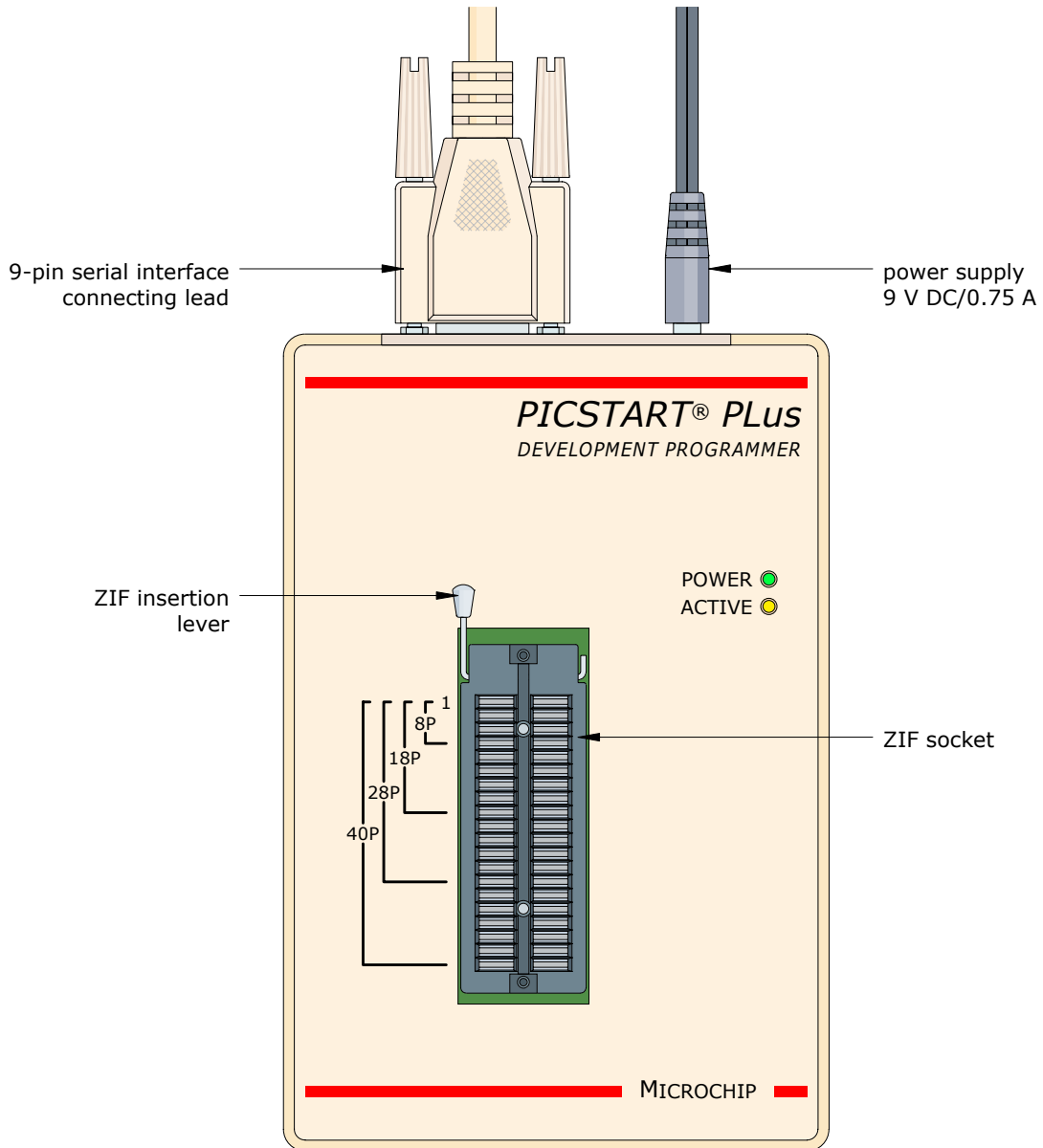


Fig. 1.24 PICSTART® programmer.

The power supply to the programmer, 9 V DC at 0.75 A, is provided by the self-contained transformer/regulator which is part of the PICSTART® Plus package. This is usually supplied with a separate mains lead. The low voltage output from the transformer/regulator is connected to the appropriate socket at the top of the programmer.

A 9-pin serial interface lead is part of the PICSTART® Plus package. The male end of this lead attaches to the programmer. The female end needs to be connected to a suitable serial interface at the back of your computer.

The details of how to do this depend on your computer. Most recent computers have a special small round PS/2 socket where the lead from the mouse is connected. In this case, you are likely to find a suitable 9-pin male connector where you can insert the serial lead. If the sockets are labelled, make the connection to COM 1.

In older machines, the mouse is usually connected to COM 1. There will be another serial connector, but this is likely to be a 25-pin connector. To solve the problem, you need to acquire a 9-pin male to 25-pin female adaptor (available from your electronics component supplier). The 25-pin connector is usually designated as COM 2.

Connect up your PICSTART® Plus programmer following these guidelines.

Next, you need to set up MPLAB so that it will communicate successfully with the programmer:

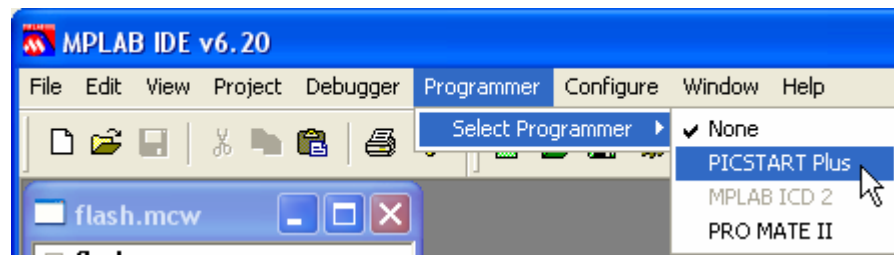


Fig. 1.25. Selecting the PICSTART® Plus.

Select options as shown and confirm that the programmer appears in the status bar:



Fig. 1.26. Status bar display.

Select Programmer/Enable Programmer, as indicated:

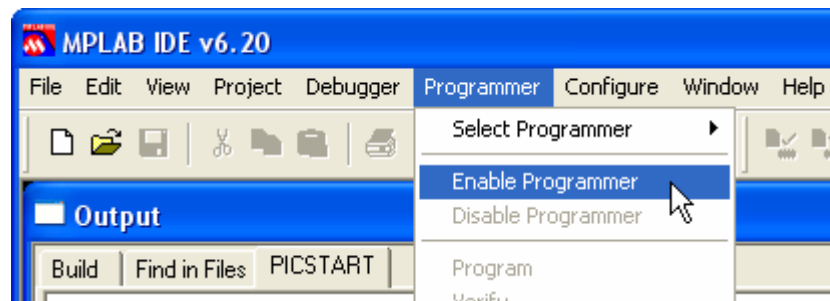


Fig. 1.27 Enabling the PICSTART® Plus.

If there is no error message in the Output window, the serial port and the programmer will have been correctly set up. This is good news!

If you do get an error message, check that the serial cable is correctly connected and that the green POWER indicator LED on the programmer is illuminated.

In some cases, you may need to change the port settings to establish communication between your computer and the programmer. Select **Programmer/Settings** and try changing the COM port, as follows:

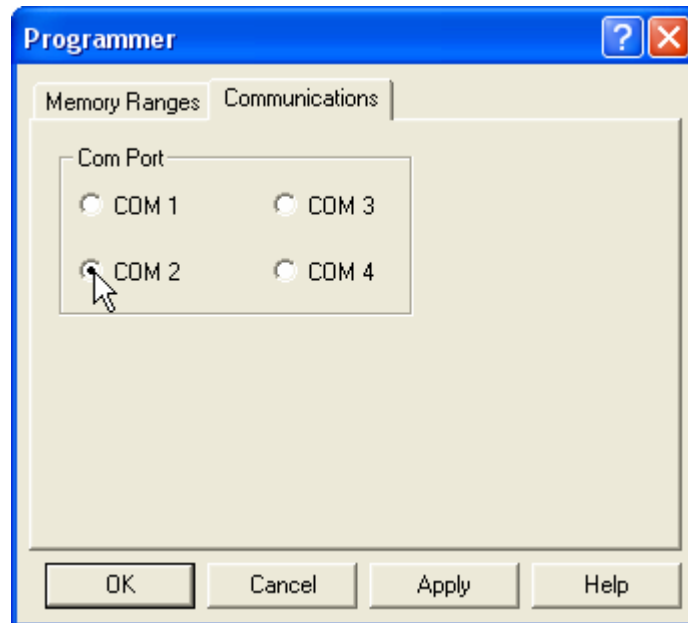


Fig. 1.28. Changing COM port settings.

If you are using an old PICSTART® Plus programmer with a new version of MPLAB, you may need to upgrade the operating system of the PICSTART® Plus to the current version.

You can do this following the instructions accessible from the PICSTART Plus Help menu, Fig. 1.29. Select Troubleshooting/Common Problems/Operating system upgrade needed. Print the help page to make things easier.

To carry out the upgrade, you need to buy a separate PIC17C44 device. This is a large PIC with EPROM memory which controls the operation of the PICSTART® Plus. The new version of the operating system is programmed into the PIC17C44, which is then used to replace the operating system inside the programmer.

The existing programmer is used to program the PIC17C44.

All this sounds complicated but is straightforward in practice. It is an advantage of buying a PICSTART® Plus instead of a different programmer that you *will* be able to update your system to program new PIC devices as and when they become available.

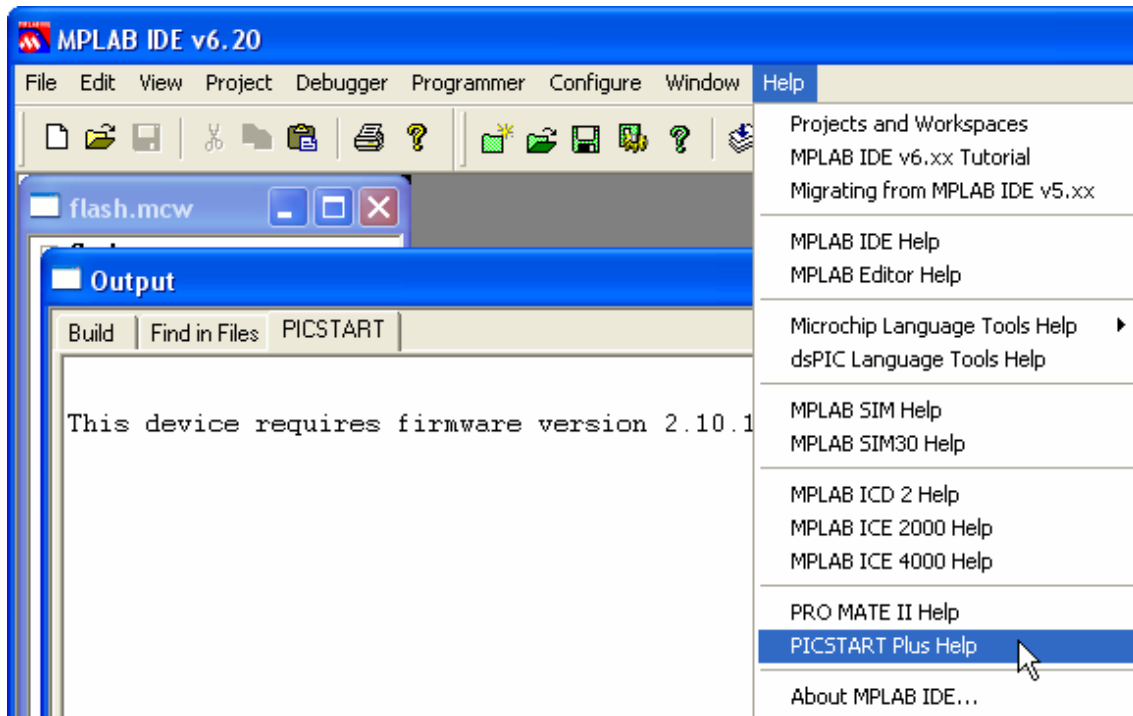


Fig. 1.29. Finding out about a PICSTART® Plus operating system upgrade.

Once it appears that the programmer has been set up correctly, select **Programmer/Enable Programmer** once again. If everything is in order, lots of new options appear in the **Programmer** menu:

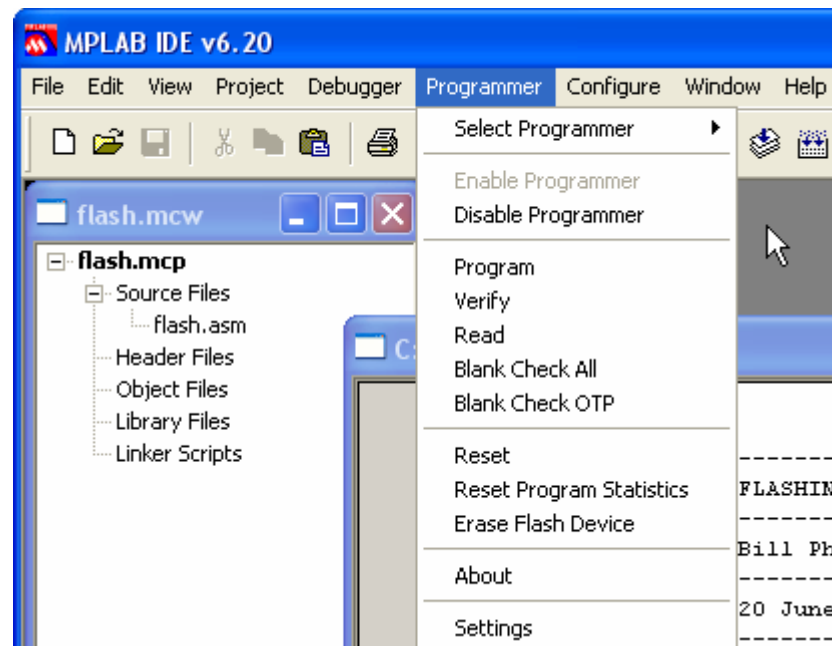


Fig. 1.30. Programmer menu options.

Most of these options were previously 'greyed out'. When they appear like this, you know that the PICSTART® Plus is working.

Programming the PIC

Insert a PIC16F627 in the ZIF (Zero Insertion Force) socket on the programmer. You need to move the ZIF insertion lever into one position, insert the integrated circuit, checking that pin 1 is correctly

located, and then push the lever back to its original position to grip the pins.

Not all ZIF sockets are the same. Sometimes the lever needs to be in a vertical position to grip the pins. Other ZIF sockets grip the pins with the lever in the horizontal position.

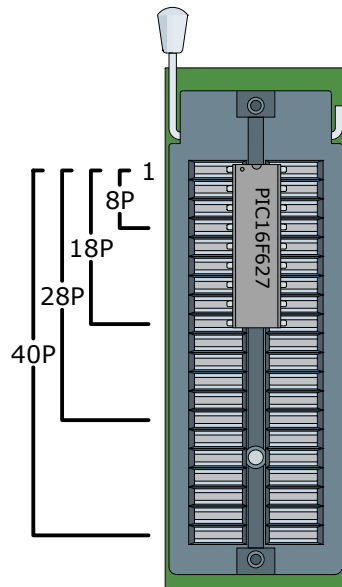


Fig. 1.31. Using the ZIF socket.

Once the PIC has been inserted, select Configure/Configuration Bits... , like this:

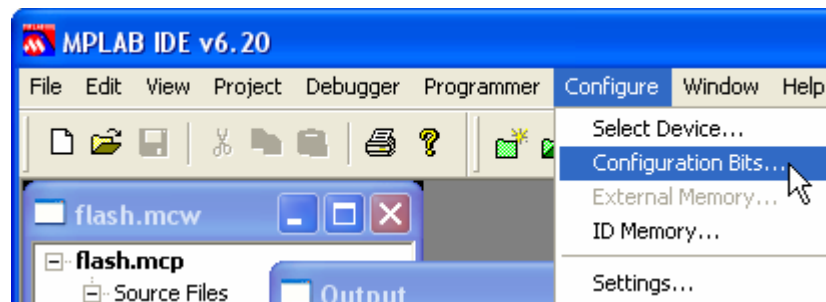


Fig. 1.32. Configuration bits.

In the dialogue box which appears, you can set various options which affect how the PIC will behave, as indicated in Fig. 1.33.

Click in the Setting column at the right-hand end of the first line. From the drop down list, select INTRC CLKOUT. This tells the PIC to use its internal oscillator to control program operation. This is a resistor/capacitor or RC oscillator which is part of the internal circuit of the PIC.

In the same way, set Watchdog Timer to Off, set Brown Out Detect to Disabled and Low Voltage Program to Disabled.

When you are finished, confirm that options selected in the dialogue box are the identical with those shown in Fig. 1.34. These options will be used for most of the programs described in this book.

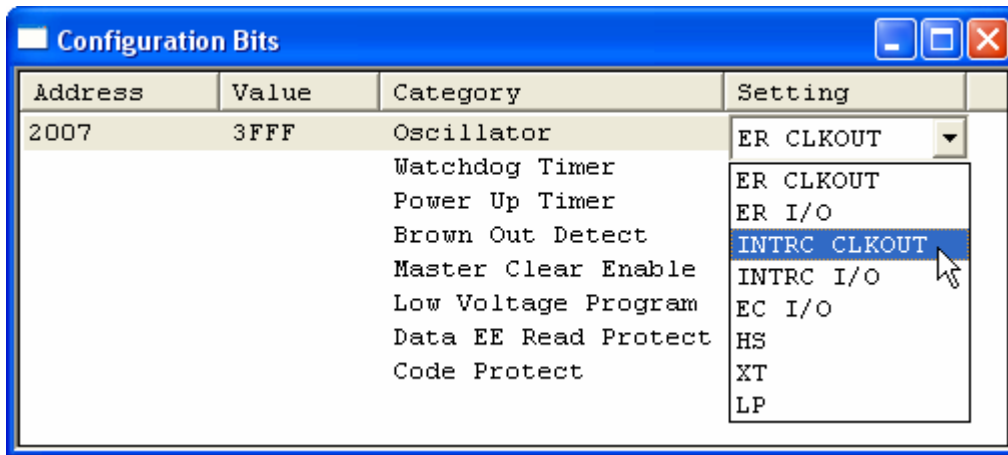


Fig. 1.33. Setting configuration bits.

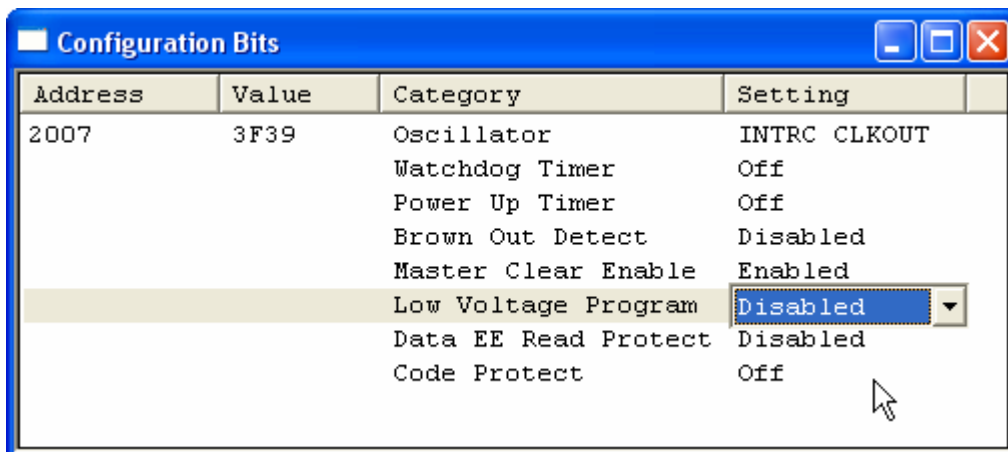


Fig. 1.34. Completed configuration changes.

Close the Configuration Bits dialogue box and click Project/Build All to confirm that flash.asm is present and correct. Next select Programmer/ Program, as follows:

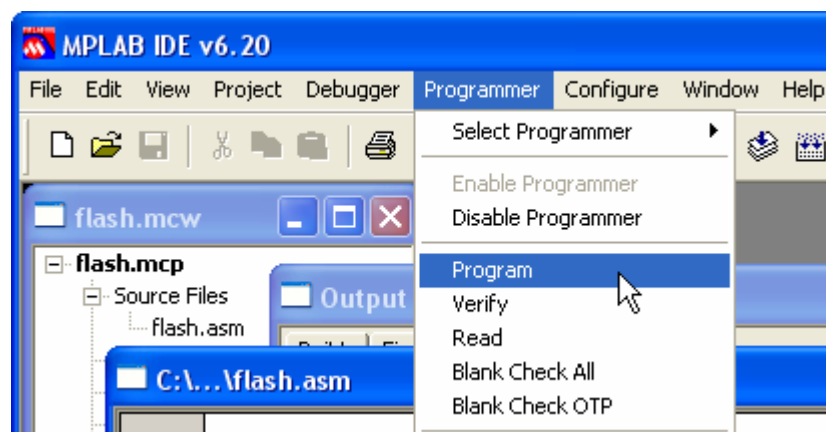


Fig. 1.35. Programming the PIC.

The status bar shows that programming is taking place.



Fig. 1.36. Programming in progress.

The orange ACTIVE LED on the PICSTART® Plus is illuminated.

If the PIC is correctly located in the ZIF socket, there should be no problems in transferring the program to its memory:

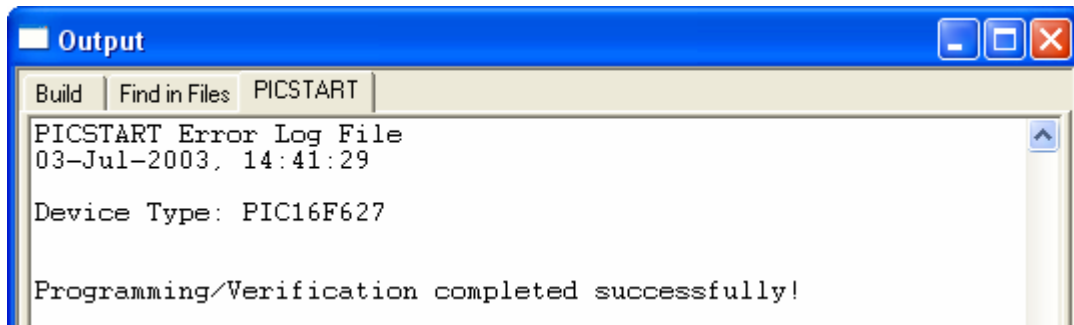


Fig. 1.37. Output message.

Select View/Program Memory:

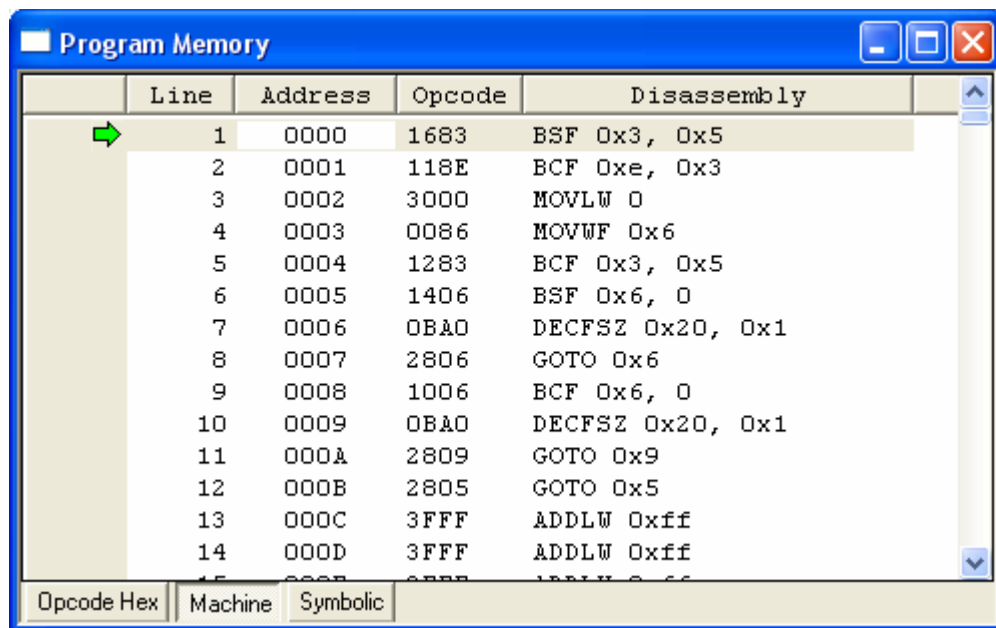


Fig. 1.38 Program memory.

You can tell that *something* has been programmed into the PIC because the first memory locations contain something other than 3FFF. In fact, the whole of the program to flash the LED is represented by the first 12 hexadecimal numbers in the Opcode column. 3FFF in the remaining locations indicates that this memory is blank. Scroll down to see how much empty memory there is.

Prototype testing

The PIC16F627 has been programmed, but does it work? To find out you need to build the prototype circuit shown on the next page.

Follow the prototype layout carefully, making sure that the PIC is inserted correctly, with pin 1 top left, and check the polarity of the polarised capacitor and the LED.

Once all the components and wire links have been inserted, connect the power supply. Three AA cells connected in series in a battery holder make an ideal power supply. The red lead from the

battery clip is the +4.5 V connection and must go to the right-hand side of the prototype board, while the black lead is the 0 V and must be inserted on the left.

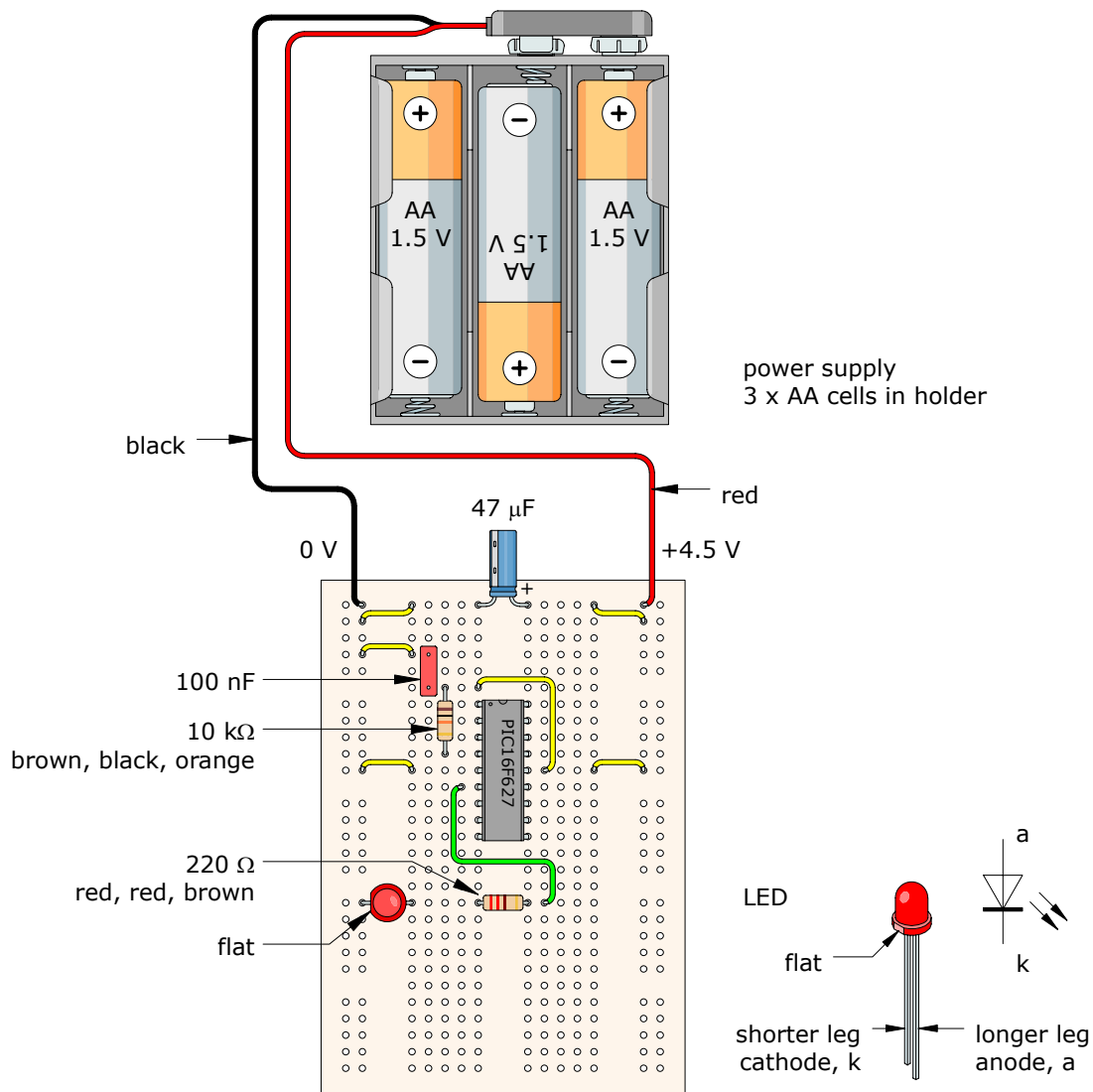


Fig. 1.39. Prototype circuit.

A regulated 5 V laboratory power supply designed for use with logic circuits can be used in place of the cells, if this is more convenient. It is probably better not to use a variable power supply to avoid the possibility of exceeding the 5.5 V maximum the PIC is able to tolerate.

If you have worked through this Chapter successfully, your LED will be flashing away happily. This is a small but pleasing achievement which sets the background for a whole range of more impressive results.

Although sketchy, you have gained an outline understanding of the stages involved in PIC programming and, assuming the LED is flashing, you know that your computer, the MPLAB software, the PICSTART® Plus programmer, the PIC16F627 and your prototype circuit are all working correctly.

Problems

It is difficult to anticipate exactly where problems might arise in working through this Chapter.

With older versions of MPLAB, the procedure for creating a Project and opening the editor window is somewhat different. In this case, check with the MPLAB User's Guide. This is included with the PICSTART® Plus, or can be downloaded from the *Microchip* website.

Establishing communication between your computer and the programmer should be straightforward. Sometimes COM 1 and COM 2 are incorrectly labelled on the back of the computer and it is certainly worth selecting a different port from Programmer/Settings/Communications to see whether this solves the problem.

Experiment and persevere until things do work as intended.

Check that the serial port leads are properly connected and that the power supply to the programmer is connected and switched on.

It may be that you will need to upgrade the operating system of the PICSTART® Plus, as described earlier.

Be careful to insert the PIC in the right place and the right way round in the ZIF socket.

Once programming has been completed in MPLAB, any residual errors are likely to be in your prototype board layout. Check carefully for any wiring errors. A single misplaced link is enough to prevent the circuit from working.