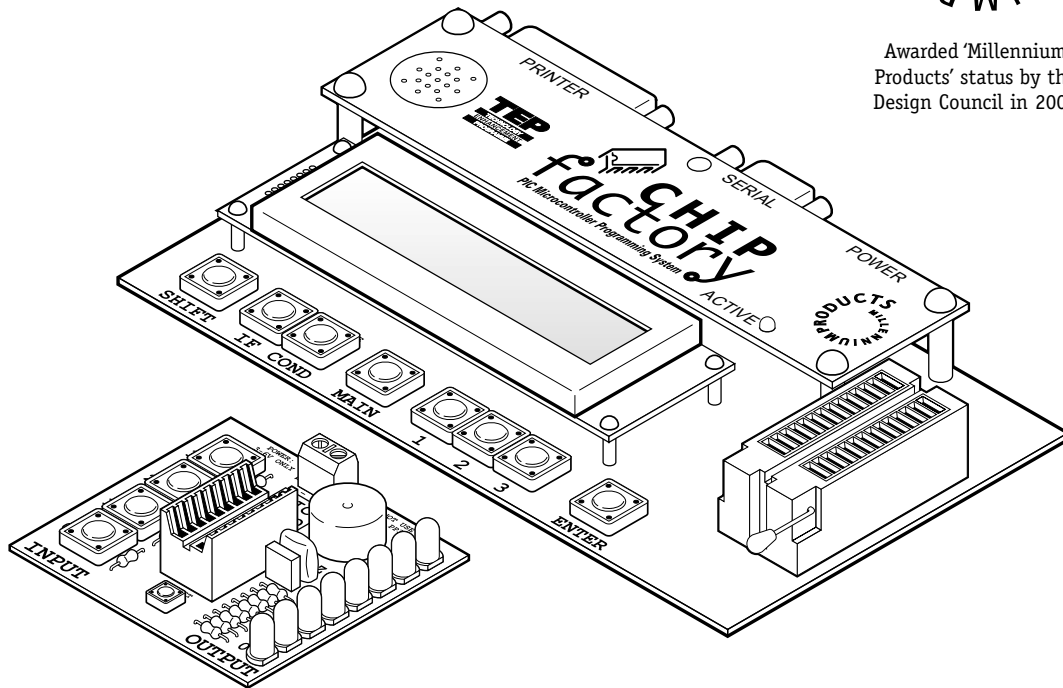


# CHIP FACTORY



Awarded 'Millennium Products' status by the Design Council in 2000



## Overview

The Chip Factory is an award-winning, easy to use system for programming PIC microcontrollers. Programming is achieved using the on-board switches to select commands that are clear and straightforward to understand. Commands are built up into a 'Basic' style program, which is then downloaded directly into the microcontroller. No computer is required! Once programmed, the microcontroller can be tested using the PIC demo board, then integrated into pupils' project work.

The system can be used with a range of microcontrollers, including 28 pin types that offer 8 digital inputs, 4 analogue inputs and 8 digital outputs.

While the programmer functions primarily as a 'stand alone' device, software is available that enables programs to be written and downloaded via a computer.

## QUICK-START TUTORIAL

This quick-start tutorial is designed to quickly show how a simple program is created, programmed and tested. Note that all the commands are described in more detail within this publication.

The program described below will simply flash output LED 1 on the demo board on and off every second.

1. Switch the programmer on.
2. Press [MAIN] to select the main-menu.
3. From the main-menu press [1] for 'Edit'.
4. From the edit-menu press [2] for 'New'.
5. Press [1] to select '18L' for PIC type 16F627.
6. Press [Enter] to confirm the memory erase. Wait until the erase is complete - once complete you will be able to directly enter the 'program'.
7. Repeatedly press [MAIN] until the word 'high' is displayed.
8. Press [3] until the digit '1' is displayed.
9. Line 00 should now read high 1. Press [Enter] to save the line.
10. Repeat these operations until the following program has been entered:

```
00  high  1
01  wait  010
02  low   1
03  wait  010
04  goto  00
```

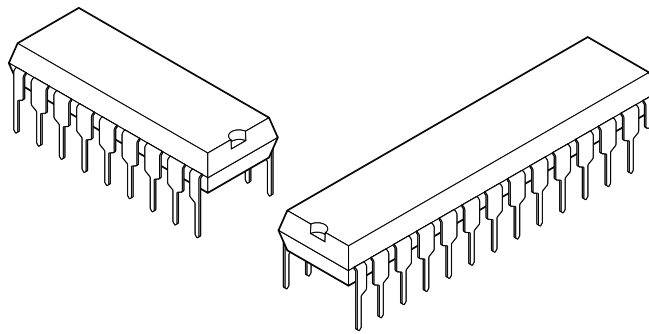
11. Make sure you press [ENTER] after programming line 04! (Forgetting to press [ENTER] after the last-line is a very common mistake!) If you need to move back up to a line to correct a mistake press [SHIFT] + [ENTER]
12. Press [SHIFT] + [MAIN] to return to the main-menu.
13. Press [3] to select 'Program'.
14. Place the PIC16F627 chip into the socket, ensuring pin 1 is top left.
15. Press [Enter] to start programming.
16. When the 'OK' message appears press [MAIN] to return to the main-menu.
17. Remove the PIC16F627 chip and place it in the demo board.
18. Connect the battery to the demo board. Output LED 1 should flash on and off every second.

	<b>Page</b>
<b>SECTION 1</b>	
<b>Introduction</b>	<b>5</b>
What is a PIC Microcontroller?	5
Programming PIC Microcontrollers	6
The PIC Demo Board	7
Using this booklet	7
<b>General Operation</b>	<b>8</b>
Free-Standing Mode	8
Serial Programmer Mode	10
<b>Programmer Module</b>	<b>11</b>
Creating a program	12
<b>Quick-start Tutorial</b>	<b>13</b>
<b>Programming Commands in Free-Standing Mode</b>	<b>14</b>
<b>Introduction to the Main Commands</b>	<b>14</b>
Substituting variables in main commands	14
Using high, low, wait, goto commands	15
<i>Sample Program 1</i>	
Using the out command	15
<i>Sample Program 2</i>	
Using the beep command	16
<i>Sample Program 3</i>	
<b>Introduction to the IF statements</b>	<b>16</b>
Using Inputs within if statements	17
<i>Sample Program 4</i>	
Waiting for Inputs using if statements	17
<i>Sample Program 5</i>	
Using analogue sensors	18
<i>Sample Programs 6 &amp; 7</i>	
<b>Introduction to Variables</b>	<b>19</b>
Using let statements with variables	19
Variable Arithmetic	19
Variable Storage	20
Using Variables with main commands	20
<i>Sample Programs 8-11</i>	
<b>Printing the Program</b>	<b>23</b>
<b>Programming the PIC</b>	<b>24</b>
Using the run-delay feature	25
Important Programming Notes	25

	<b>Page</b>
<b>SECTION 2</b>	
<b>Windows Chip Factory Programming Software</b>	<b>26</b>
Software Installation	26
Connecting the programmer to the computer	26
Using the software	27
Program Simulation	28
Chip Factory Software menus	30
<b>SECTION 3</b>	
<b>PIC Programmer Software</b>	<b>31</b>
Software Installation	31
Connecting the programmer to the computer	31
Using the PIC Programmer software	32
Software Features	33
<b>SECTION 4</b>	
<b>Interfacing to the PIC Microcontroller</b>	<b>34</b>
Control Systems	34
The PIC Project Board	35
Wiring Up the PIC Project Board	36
Minimum Hardware Configuration	41
Generating a 5V Voltage Supply	42
Output Devices that do not require interfacing	43
Using a transistor to interface outputs	44
Using a Darlington driver IC	45
Using a power MOSFET to interface outputs	45
Using a relay to interface outputs	46
Using a Motor controller IC to control motors	47
Interfacing switch inputs	48
Interfacing Analogue Inputs	48
Using the Low-Resolution Analogue Input	49
<b>APPENDICES</b>	
<b>Appendix A: Programming Quick Guide</b>	<b>51</b>
Main Commands Summary	52
If Conditions Summary	53
Let Statements Summary	54
<b>Appendix B: Programmer Power Supply</b>	<b>55</b>
<b>Appendix C: Upgrading your Programmer</b>	<b>56</b>
<b>Appendix D: Abbreviations</b>	<b>57</b>

# SECTION 1

## INTRODUCTION



### What is a PIC microcontroller?

A PIC microcontroller is often described as a 'computer-on-a-chip'. The Peripheral Interface Controller is a complete 'computer' that can be built into a device to make the product more intelligent and easier to use. Microcontrollers are programmed to perform a specific control task - for instance, a microwave oven may use a single microcontroller to process information from the keypads, display user information on the seven segment display, and control the output devices (turntable motor, light, bell and magnetron).

Microcontrollers are computers designed to control specific processes or products. The microcontroller is programmed with a specific software program to complete the desired task. By altering this software program, the same microcontroller can be used to complete different tasks. Therefore the same chip can be used in a range of different products by simply programming it with a different software program.

One microcontroller can often replace a number of separate parts, or even a complete electronic circuit. Some of the advantages of using microcontrollers in a product design are:

- *increased reliability and reduced stock inventory (as one microcontroller replaces several parts)*
- *simplified product assembly and smaller end products*
- *greater product flexibility and adaptability since features are programmed into the microcontroller and not built into the electronic hardware*
- *rapid product changes or development are possible by changing the program and not the electronic hardware*

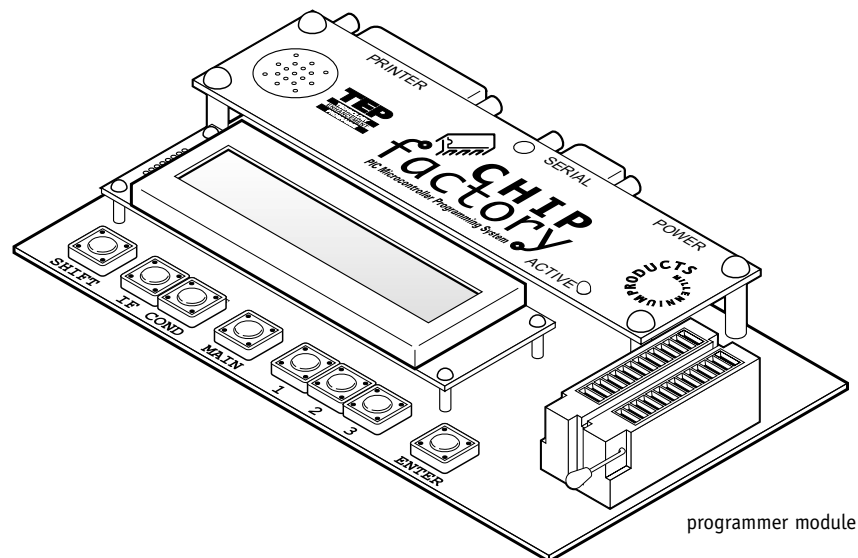
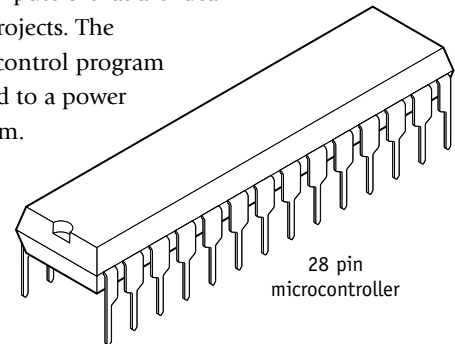
Applications that use microcontrollers include household appliances, alarm systems, medical equipment, vehicle subsystems, and electronic instrumentation. Although microprocessor systems tend to be more widely publicised (mainly via the popularity of personal computer systems), microcontroller manufacturers actually sell hundreds of microcontrollers for every microprocessor sold.

## Programming PIC Microcontrollers

PIC microcontrollers are low-cost single-chip 'computers' that are ideal controllers for many industrial and educational projects. The microcontroller is programmed with a dedicated control program (called the software), and whenever it is connected to a power supply it automatically 'runs' this software program.

When the software is 'programmed' into the microcontroller, the 'programmed chip' is called the system 'firmware'.

In industry the microcontroller program is usually created using 'C' or 'assembler' language routines. However these languages are usually too complicated for educational projects, and so the Chip Factory system has been designed to provide a easy to learn 'entry level' step into the world of microcontrollers.



The system is based around the programmer module, which is a stand-alone unit that can program without the use of a computer. A program, made up of simple 'BASIC' style commands, is entered into the programmer memory using the on-board keypad.

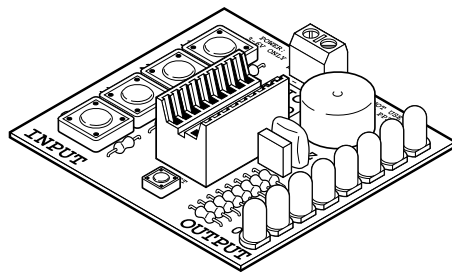
Once the program has been entered, the PIC is placed into the programming socket (ZIF socket) and the program is transferred to the PIC. When the programming cycle is complete, the PIC can be removed from the programming socket and transferred to the project board.

Modern PICs are constructed using Flash EEPROM memory, which means that the device can be re-programmed over 10,000 times. Therefore, if the firmware does not operate correctly the first time, the PIC can simply be re-programmed with new code.

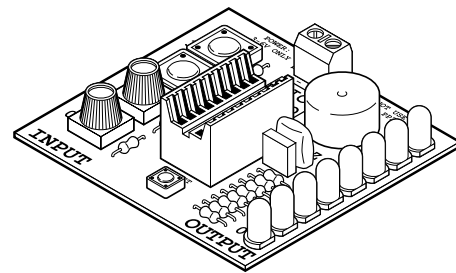
## The PIC Demo Board

The PIC demo board is a low-cost demonstration board which can be used to 'test' programs written with the Chip Factory system. The board uses push switches for inputs and LEDs for outputs, and also includes a piezo-transducer for sound generation. A second type of PIC demo board is also available that includes two variable resistors to simulate analogue inputs.

PIC demo board with  
4 x digital inputs and 8  
x digital outputs



PIC demo board with  
2 x analogue inputs,  
2 x digital inputs and 8  
x digital outputs



Use of the PIC demo board enables a student to quickly test and evaluate the program firmware without the need to build any hardware. Once the firmware program has been tested and debugged it can then be transferred to a pupil's own-designed project work board. This system also helps identify hardware faults - if the PIC firmware 'runs' on the demo board, but not the project board, there is obviously a hardware fault somewhere within the project board.

### Using this booklet...

The programming section of this booklet assumes that you are using the PIC demo board to 'run' your programs, and therefore makes no reference to building hardware. See the 'Interfacing...' section (section 4) for further details about suitable circuit designs for project work applications.

**Note:** The interfacing section in this booklet provides basic information for getting started. Additional information about interfacing components can be found in the 'Microcontroller Interfacing Circuits' booklet available separately.

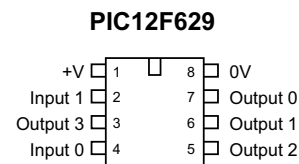
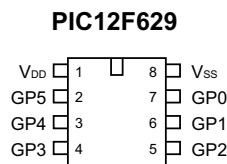
## GENERAL OPERATION

The Programmer has two modes of operation.

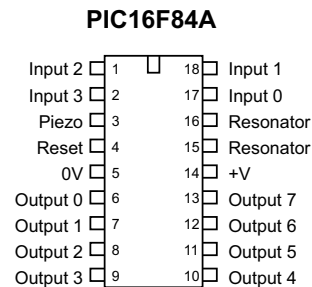
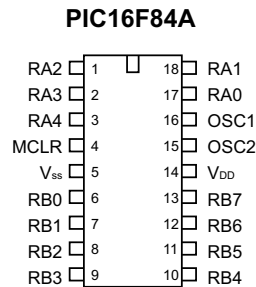
### Free-Standing Mode

In this mode PIC microcontroller programs can be created, edited, printed and programmed into the PIC *without* the need for a computer. The program listing is displayed on a LCD display, and the commands are entered via a simple push switch keypad configuration. In this mode five PIC types are supported. The PICs are known within the Programmer by their configuration type 8, 18, 18L, 18A or 28. These types are:

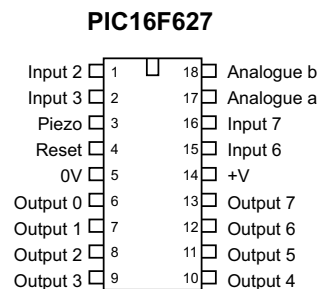
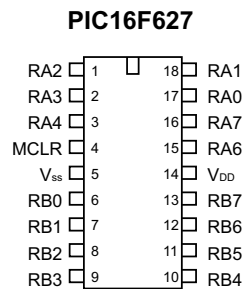
#### Type 8



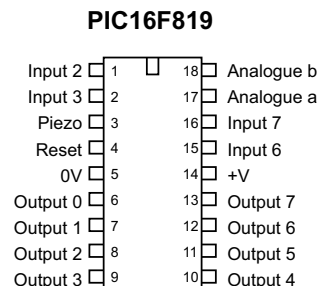
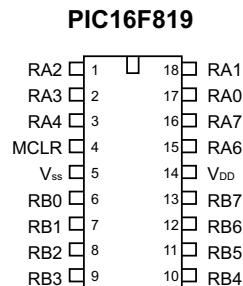
#### Type 18



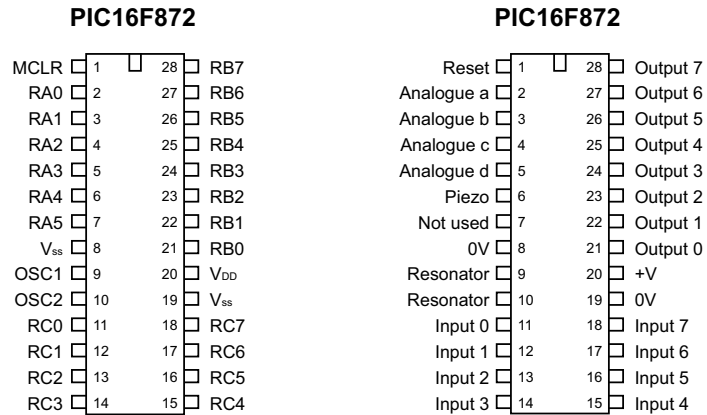
#### Type 18L



#### Type 18A



Type 28



**PIC12F629 (8 pin) - type 8**

- 2 digital inputs                   input 0 - input 1                   (pins GP 3,4)
- 4 digital outputs                   output 0 - output 3               (pins GP 0,1,2,5)  
(sound output combined on output 2)

**PIC16F84A (18 pin) - type 18**

- 4 digital inputs                   input 0 - input 3                   (pins RA 0 - 3)
- 8 digital outputs                   output 0 - output 7               (pins RB 0 - 7)
- 1 sound output                   piezo-sounder                   (pin RA 4)

**PIC16F627 (18 pin) - type 18L**

- 2 analogue inputs               analogue a - analogue b       (pins RA 0 - 1)
- 4 digital inputs               input 2,3,6,7                   (pins RA 2,3,6,7)
- 8 digital outputs               output 0 - output 7             (pins RB 0 - 7)
- 1 sound output               piezo-sounder                   (pin RA 4)

**PIC16F819 (18 pin) - type 18A**

- 2 analogue inputs               analogue a - analogue b       (pins RA 0 - 1)
- 4 digital inputs               input 2,3,6,7                   (pins RA 2,3,6,7)
- 8 digital outputs               output 0 - output 7             (pins RB 0 - 7)
- 1 sound output               piezo-sounder                   (pin RA 4)

**PIC16F872 (28 pin) - type 28**

- 4 analogue inputs               analogue a - analogue d       (pins RA 0 - 3)
- 8 digital inputs               input 0 - input 7               (pins RC 0 - 7)
- 8 digital outputs               output 0 - output 7             (pins RB 0 - 7)
- 1 sound output               piezo-sounder                   (pin RA 4)

## Serial Programmer Mode

In this mode the Programmer can be directly controlled from a host computer. Two Windows software applications are available to drive the Chip Factory programmer.

### Chip Factory Software

This (optional) software allows programming via the computer, rather than using the on-board switches and LCD display. The same programming commands are used as in the free-standing mode. On screen simulation for 'testing' programs is also provided. See section 2 for further details.

### PIC Programmer Software

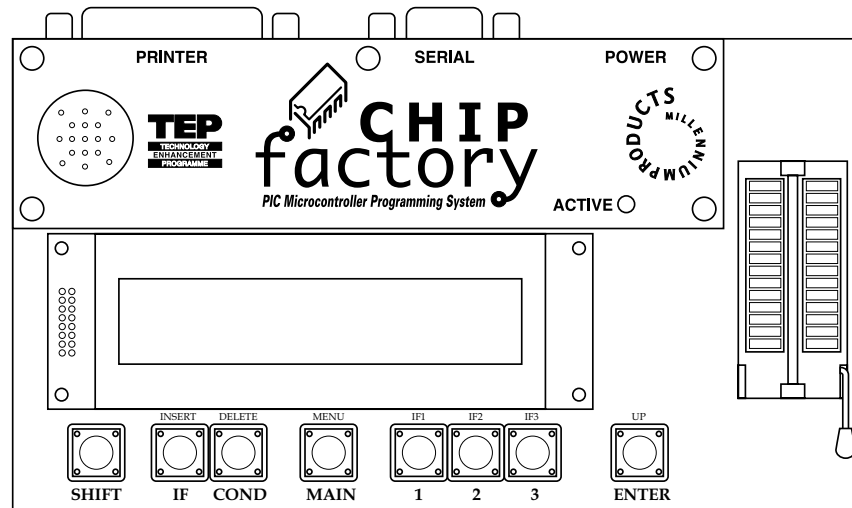
This free software allows the Programmer to be used as a conventional serial PIC programmer, for control programs written in other languages (e.g. assembler code). The Programmer is connected to the host computer via a serial cable and the software is used to control the programmer functions. See section 3 for further details.

PICs supported via the PIC Programmer software:

- *All 8 pin PICs in the 12F and 12C and 18F series.*
- *All 18 pin PICs in the 16F and 16C series (but not the old 16C5x devices).*
- *All 28 pin PICs in the 16F and 16C series (but not the old 16C5x devices).*

## PROGRAMMER MODULE

The Programmer is used to enter simple control programs for educational projects. It does not require the use of a computer, and programs can also be printed directly without the need for a computer.



programmer module

The general procedure for using the Programmer is:

1. Select the PIC device you wish to use.
2. Draw a flowchart and/or write the program out in rough.
3. Enter the program into the programmer memory.
4. Program the PIC microcontroller.
5. Place the PIC in the demo board and test the program operation.
6. Modify the program as necessary and re-program & test.
7. Print out the finished program.
8. Build the project hardware.
9. Place the PIC into the project hardware & test.

## Creating a Program

In the free-standing mode it is possible to enter (or 'edit') programs. To create a program:

switch the Programmer on and then press [MAIN] to select the main-menu.  
Select [1] for 'Edit' and then select [1] for 'Edit' again.

To leave the program once finished:

press [SHIFT] + [MAIN] to move back to the main-menu.

A program consists of a single listing, which has a maximum length of 100 lines (numbered 00 to 99). Each line can be programmed with a 'command' by using the on-board keypad buttons. Any unused lines are left blank.

The program is saved in the internal Programmer memory, even if the power supply is removed. If you wish to start a new program the internal memory can be erased for a 'fresh start'.

When the PIC 'runs' the program it carries out the commands sequentially from line 00. The program continues running until a blank line is reached, which causes the program to stop. If a 'goto' command is used to create a loop, the program will continue forever (or until the power is switched off!).

## QUICK-START TUTORIAL

This quick-start tutorial is designed to quickly show how a simple program is created, programmed and tested. Note that all the commands are described in more detail in later sections.

The program described in this tutorial will simply flash output LED 1 on the demo board on and off every second.

1. Switch the programmer on.
2. Press [MAIN] to select the main-menu.
3. From the main-menu press [1] for 'Edit'.
4. From the edit-menu press [2] for 'New'.
5. Press [1] to select '18L' for PIC type 16F627.
6. Press [Enter] to confirm the memory erase. Wait until the erase is complete - once complete you will be able to directly enter the 'program'.
7. Repeatedly press [MAIN] until the word 'high' is displayed.
8. Press [3] until the digit '1' is displayed.
9. Line 00 should now read high 1. Press [Enter] to save the line.
10. Repeat these operations until the following program has been entered:

```
00  high  1
01  wait  010
02  low   1
03  wait  010
04  goto  00
```

11. Make sure you press [ENTER] after programming line 04! (Forgetting to press [ENTER] after the last-line is a very common mistake!) If you need to move back up to a line to correct a mistake press [SHIFT] + [ENTER]
12. Press [SHIFT] + [MAIN] to return to the main-menu.
13. Press [3] to select 'Program'.
14. Place the PIC16F627 chip into the socket, ensuring pin 1 is top left.
15. Press [Enter] to start programming.
16. When the 'OK' message appears press [MAIN] to return to the main-menu.
17. Remove the PIC16F627 chip and place it in the demo board.
18. Connect the battery to the demo board. Output LED 1 should flash on and off every second.

Congratulations! That's your first program entered, programmed and tested. The following sections explain all of the commands in more detail, and provide further information on how the Programmer is used.

## PROGRAMMING COMMANDS IN FREE-STANDING MODE

The following sections explain the three main types of program line structure:

**Main Commands** - These are unconditional commands which are always carried out as the program runs.

**'If' Statements** - These are conditional commands that are only carried out if certain conditions are met.

**'Let' Statements** - These commands are used to assign values to the PIC variables.

## INTRODUCTION TO THE MAIN COMMANDS

There are six standard main commands:

**high** - This command switches an output high.

**low** - This command switches an output low

**goto** - This command causes program flow to 'jump' to a new line.

**wait** - This command creates a pause, in units of 0.1 seconds. Therefore the command wait 010 generates a pause 1 second long.

**out** - This command is used to switch ALL the outputs high/low simultaneously. The binary value (0-255) is transferred to the output pins.

**beep** - This command generates a sound. The value gives the 'pitch' of the sound - the higher the value the higher the frequency.

The different main commands are selected by use of the [MAIN], [1], [2] and [3] buttons. Note that the digit buttons are only active when they can be used to create legal values. Variables are selected by use of the [3] button.

### Substituting variables in MAIN commands

All commands except 'high' & 'low' can use a variable instead of a direct value. Variables can be selected by pressing [3] until the variable name is shown. When the program is run the value currently stored in the variable is used (instead of a direct number).

If you are not familiar with the concept of using variables see the 'Using Variables' section for more information.

Note that when a variable is used with the 'goto' command any variable value exceeding 99 (i.e. 100 to 255) will be executed as a 'goto 99' command.

## Using high, low, wait and goto commands

Sample program 1 demonstrates how to flash output pin 1 on and off every second. This could be used to control an LED as a very simple warning indicator - for instance to indicate when a burglar alarm system is switched on.

The **high** and **low** commands are used to switch the output on (high) and off (low). The **wait** command is used to add a time delay. **wait 010** gives a delay of 1 second, as the time is measured in units of 0.1 second ( $10 \times 0.1 = 1\text{sec}$ ). The final **goto** command causes the program flow to 'jump' back to the start (line 00). Therefore the program 'loops' forever.

### Sample Program 1

This program can run on any type of PIC. (Note that with all the example programs the default 'zero' run-delay (0.0s) is used).

```

00  high  1      ` Switch output 1 on
01  wait 010    ` Pause for 1 second
02  low   1      ` Switch output 1 off
03  wait 010    ` Pause for 1 second
04  goto 00     ` Loop back to line 00

```

## Using the out command

Sample program 2 demonstrates how to 'cycle' the lower 4 output pins on and off in a 'circular' pattern, for instance as a simple lights controller for a disco. This could be achieved by using multiple **high** and **low** commands, but it is quicker to use the **out** command instead. The **out** command controls *all* of the outputs *simultaneously*. Each output can be switched high or low according to the value calculated from the table below.

Output	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

The out value is the sum of the value of each output you want to be high (switched on). Therefore the command to switch on outputs 0, 2 & 4 would be out 21 (as  $21 = 1+4+16$ ). All the other outputs will be automatically switched off. The maximum out value is out 255 for the 18 pin PICs (out 15 on the 8 pin PIC), which will cause all the outputs to be switched on.

### Sample Program 2

This program can run on any type of PIC.

```

00  out 001     ` Switch output 0 on, others off
01  wait 010    ` Pause for 1 second
02  out 002     ` Switch output 1 on, others off
03  wait 010    ` Pause for 1 second
04  out 004     ` Switch output 2 on, others off
05  wait 010    ` Pause for 1 second
06  out 008     ` Switch output 3 on, others off
07  wait 010    ` Pause for 1 second
08  goto 00     ` Loop back to line 00

```

## Using the beep command

Sample program 3 demonstrates how to play a short 'tune' using the beep command. This could be used, for instance, to attract customers to a window display in a shop.

The **beep** command is followed by a number (0 to 255). Each number corresponds to a different pitch frequency - so each number creates a different sound. The higher the number the higher the pitch of the sound. Note that on the 8 pin PIC the beep acts on output 2, rather than a dedicated sound pin as with the other devices.

### Sample Program 3

```

00  beep 166      ` Make a sound
01  beep 189      ` Make another sound
02  beep 204      ` Make another sound
03  beep 212      ` Make another sound
04  beep 212      ` Do it again!
05  beep 204      ` Make another sound
06  beep 212      ` Make another sound
07  beep 212      ` Do it again!
08  beep 212      ` And once more!
09  wait 010      ` Pause for 1 second
10  goto 00       ` Loop back to line 00

```

## INTRODUCTION TO THE 'IF' STATEMENTS

An **if** statement involves a 'condition' that must be true for the main command within the statement to be completed. There are three different types of **if** condition:

**if** statements involving the digital inputs. For example:

if input 0 is on then...	if 0 on	(on = high)
if input 1 is off then...	if 1 off	(off = low)

**if** statements involving analogue sensors. For example:

if sensor a = 42 then...	if a=042	(equals)
if sensor b > 99 then...	if b>099	(greater than)
if sensor a < y then ...	if a< y	(less than)
if sensor b != 0 then...	if b!000	(not equal to)

**if** statements involving variables

(see the 'Introduction to Variables' section for more details). For example:

if variable x = 5 then....	if x=005	(equals)
if variable y > 28 then....	if y>028	(greater than)
if variable x < y then ...	if x< y	(less than)
if variable x != 0 then...	if x!000	(not equal to)

The **if** statement must be followed by one of the 'main' commands as described in the previous section. The 'main' command is ONLY carried out if the **if** condition is true. When the **if** statement is false the 'main' command is skipped and the program continues running at the next line.

The different **if** conditions are selected by use of the [IF], [COND], [SHIFT]+[1], [SHIFT]+[2] and [SHIFT]+[3] buttons. Note that the shift-digit buttons are only active when they can be used to create legal values. Variables are selected by use of the [SHIFT]+[3] button.

### Using inputs within if statements

Sample program 4 forces outputs 0 and 1 to 'follow' the condition of inputs 0 and 1. This could be used, for instance, in a bank to let the security guard know the condition of the bank vault doors. Two output LEDs would be connected to the security guard's monitor, and two input switches would be connected to the bank vault doors. Therefore when the door was opened or closed the LED would remotely indicate the condition of the door.

#### Sample Program 4

This program can be used with the PIC16F84A

```

00 if 0 off low 0      ` If input off switch output off
01 if 0 on high 0     ` If input on switch output on
02 if 1 off low 1     ` If input off switch output off
03 if 1 on high 1     ` If input on switch output on
04          goto 00   ` Loop back to line 00

```

Note that if you do not set a 'main' command after the 'if' command the program will stop running if the condition is true, as a 'blank' main command stops the program.

### 'Waiting' for inputs using if statements

Sample program 5 demonstrates how to jump to different lines when input switches are pressed. It also shows how to 'wait' until the input is released before continuing. This is achieved by repeatedly jumping to the same line until the input is switched off again.

This program could be used as a simple 'intercom' system to call two different assistants to a help desk in a shop.

#### Sample Program 5

This program can be used with the PIC16F84A.

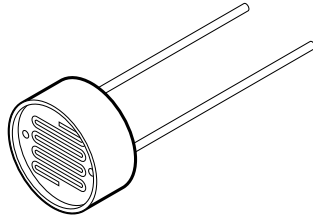
```

00 if 0 on goto 04     ` If input 0 on jump to line 04
01 if 1 on goto 08     ` If input 1 on jump to line 08
02          goto 00     ` Both off so loop to line 00
03
04 if 0 on goto 04     ` Loop here until input is off
05          beep 75     ` Make a sound
06          goto 00     ` Loop back to line 00
07
08 if 1 on goto 08     ` Loop here until input is off
09          beep 200    ` Make a different sound
10          goto 00     ` Loop back to line 00

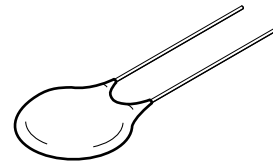
```

## Using Analogue Sensors

The PIC16F818 and PIC16F872 have 'analogue' inputs, which can respond to analogue quantities such as light and temperature. These microcontrollers contain on-board 'Analogue to Digital Converters' (ADC), which convert an analogue sensor value into a number between 0 and 255.



light dependent resistor (LDR)



thermistor

If an LDR (light dependant resistor) is used as the analogue sensor connected to analogue input 'a', the value of 'a' will vary from 0 in absolute darkness to 255 in bright light.

Sample program 6 switches output 0 on if the value of sensor 'a' is less than 150, and off if the value is greater than 150. If the value is exactly 150 a 'beep' sound is generated. This could be used to switch a street lamp on or off according to the light intensity falling on to the LDR sensor.

### Sample Program 6

This program can be used with any microcontroller with analogue inputs.

```

00 if a<150 high 0      ` If value is <150 switch on
01 if a>150 low  0      ` If value is >150 switch off
02 if a=150 beep 100    ` If value is =150 make sound
03                goto 00 ` Loop back to line 00

```

Sample program 7 demonstrates how to make a 'volume' type bar graph for an analogue sensor using the 8 outputs as a display. This could be used to show the volume of sound entering a microphone.

### Sample Program 7

This program can be used with any microcontroller with analogue inputs.

```

00                out 000    ` All outputs off
01                high 0     ` Switch output 0 on
02 if a>032 high 1         ` Output 1 on as well
03 if a>064 high 2         ` Output 2 on as well
04 if a>096 high 3         ` Output 3 on as well
05 if a>128 high 4         ` Output 4 on as well
06 if a>160 high 5         ` Output 5 on as well
07 if a>192 high 6         ` Output 6 on as well
08 if a>224 high 7         ` Output 7 on as well
09                wait 010    ` Pause for 1 second
10                goto 00     ` Loop back to line 00

```

## INTRODUCTION TO VARIABLES

A variable is a number that is stored in the memory of the PIC. Variables are commonly used as 'counters'. For instance, if you wanted to know how many cars were in a car-park, you could use a variable to store the total number of cars currently in the car park. Every time a car entered the park you would add 1 to the variable value, and every time a car left you would subtract 1 from the variable value.

It is also possible to perform simple mathematics using variables. Using the car park example you may know that a coach takes up four car spaces. Therefore every time a coach entered the park you would add four to the variable (instead of just one). This ensures the total number of spaces used in the car park is accurate at all times.

The Chip Factory system uses two variables, which are given the names 'x' and 'y'. Each variable can take any value in the range 0 to 255.

### Using Let Statements with variables

Let statements are used to set the variables x and y with specific values. The special let 'main' commands 'plus' (addition) and 'take' (subtraction) also allow for simple mathematics with the variable values. For example:

let x = 25	let x=025
let x = y	let x= y
let x = x + 1	let x= x plus 001
let x = x - 10	let x= x take 010
let x = x - y	let x= x take y

The different let statements are selected by use of the [IF], [MAIN], [1], [2], [3], [SHIFT]+[1], [SHIFT]+[2] and [SHIFT]+[3] buttons.

Note that the digit and shift-digit buttons are only active when they can be used to create legal values. Variables are selected by use of the [3] and [SHIFT]+[3] buttons.

### Variable Arithmetic

The PIC microcontroller uses single byte mathematics. This means that the variable can only contain a positive integer number between 0 and 255.

The number 'overflows' back to 0 if the maximum value of 255 is exceeded. This means the number 256 actually becomes 0, 257 becomes 1 etc. Likewise the number (-1) becomes 255, (-2) becomes 254 etc. For example:

let x = 5	give x the value 5	
let x = x + 5	give x the value 10	(5 + 5 = 10)
let x = x + 245	give x the value 255	(10 + 245 = 255 )
let x = x + 1	give x the value 0	(255 + 1 = 0)
let x = x + 5	give x the value 5	(0 + 5 = 5)
let x = x - 7	give x the value 254	(5 - 7 = 254)

## Variable storage

The variable values are stored in 'non-volatile' EEPROM memory. This means the values are NOT lost when the PIC is disconnected from the power supply (e.g. battery). Therefore when power is reconnected to the PIC the variable will have the same value as when the power was removed.

If you wish to always have the same starting value (e.g. 0) within a program, you must use a let command (such as let x = 000) at the start of the program.

## Using variables within main commands

Variables can be used instead of a direct number within the main commands wait, beep, out and goto. When the command is executed the value currently stored in the variable is used within the main command.

This is a very powerful technique that allows quite complicate programs to be written in just a few program lines. To select a variable when using a main command press button [3] until the variable name is displayed.

Sample program 8 demonstrates how to use variable x within the beep command. All 255 different sounds will be generated in turn.

### Sample Program 8

This program can be used with any 18 pin PIC.

```
00let x=000           ` Reset variable to zero
01      beep  x       ` Generate the sound
02let x=x plus 001   ` Make x 1 bigger than it was
03      goto  01      ` Loop back to line 01
```

This similar program cycles through all the possible output combinations. It can be used to check that all the outputs are operating correctly.

### Sample Program 9

```
00let x=000           ` Reset variable to zero
01      out   x       ` Output variable value
02let x=x plus 001   ` Make x 1 bigger than it was
03      goto  01      ` Loop back to line 01
```

Sample program 10 demonstrates how to use the variable as a counter, for instance to count the number of people entering a train carriage. The program waits for input 2 to be pressed (which could be a detector switch built into the step leading into the train carriage). A bar-graph type display (using all 8 outputs) then shows the train driver how full the train carriage is.

*Sample Program 10*

This program can be used with any 18 pin PIC.

```
00 if 2 on goto 04      ` If input 2 is on jump
01          goto 00      ` Else loop back to line 00
02
03
04 if 2 on goto 04      ` Wait for switch to be released
05let x= x plus 001     ` Add 1 to the value of x
06          out 000      ` All outputs off
07 if x>000 high 0      ` Switch output 0 on
08 if x>032 high 1      ` Output 1 on as well
09 if x>064 high 2      ` Output 2 on as well
10 if x>096 high 3      ` Output 3 on as well
11 if x>128 high 4      ` Output 4 on as well
12 if x>160 high 5      ` Output 5 on as well
13 if x>192 high 6      ` Output 6 on as well
14 if x>224 high 7      ` Output 7 on as well
15          goto 00      ` Loop back to line 00
```

Sample program 11 is a fairly complex program which shows, for instance, how to keep track of the number of people in a museum.

When switch 1 (connected to the 'entry' barrier) is pressed it adds 1 to the value stored in variable x, and then 'beeps' x number of times. Likewise when switch 2 (connected to the 'exit' barrier) is pressed it subtracts 1 from the value stored in variable x, and then 'beeps' x number of times.

Variable y is used to form a temporary 'loop' counter to keep track of the number of 'beeps'. This means the value of x is not altered during the 'beeps' cycle.

If you get bored with the large number of 'beeps' press input 3 to reset the value of x back to 0! Remember that the counter will not be cleared when the battery is removed as the variable values are saved in non-volatile EEPROM memory. Try it out!

#### Sample Program 11

This program can only be used with a PIC16F84.

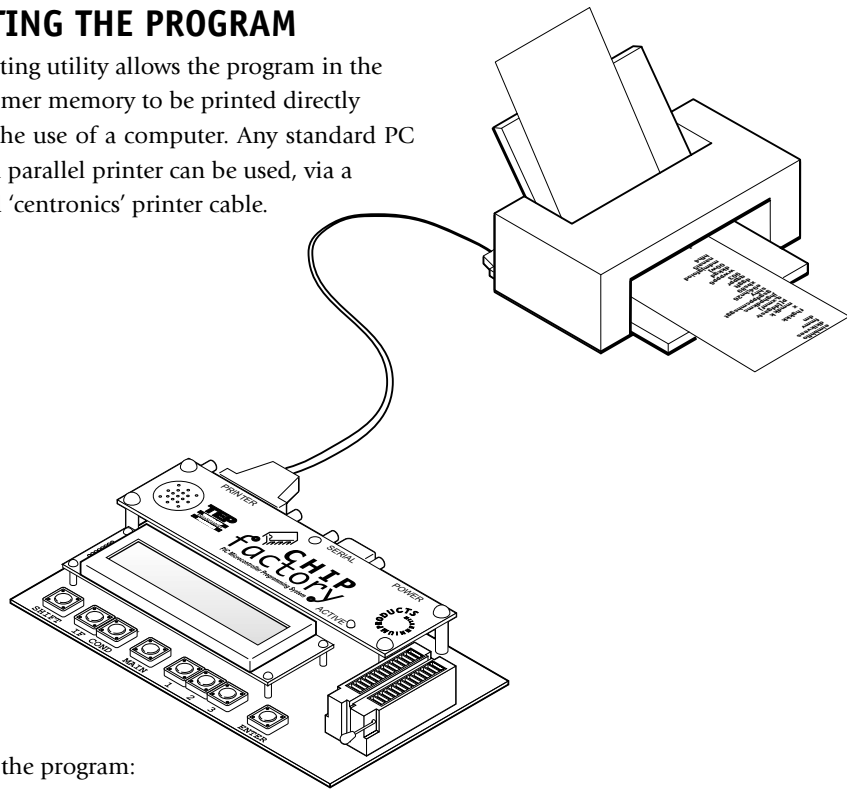
```

00 if 1 on goto 05      ` If input 1 is on jump
01 if 2 on goto 09      ` If input 2 is on jump
02 if 3 on goto 20      ` If input 3 is on jump
03          goto 00      ` Else loop back to line 00
04
05 if 1 on goto 05      ` Wait for switch to be released
06let x= x plus 001     ` Add 1 to the value of x
07          goto 13      ` Jump to line 13
08
09 if 2 on goto 09      ` Wait for switch to be released
10let x= x take 001     ` take 1 from the value of x
11          goto 13      ` Jump to line 13
12
13let y=000            ` Set y to zero
14          beep 100     ` Make a sound
15          wait 005     ` Wait a short while
16let y= y plus 001     ` Add 1 to y
17 if y= x goto 00      ` If y = x jump back to start
18          goto 08      ` y is not = x so loop again
19
20 if 3 on goto 19      ` Wait for switch to be released
21let x=000            ` Reset x back to zero
22          goto 00      ` Back to start

```

## PRINTING THE PROGRAM

The printing utility allows the program in the programmer memory to be printed directly *without* the use of a computer. Any standard PC or Acorn parallel printer can be used, via a standard 'centronics' printer cable.



To print the program:

1. Switch the printer off.
2. Remove the programmer power cable to switch it off.
3. Connect the printer cable to the printer and the programmer.
4. Switch the programmer on.
5. From the main-menu select [1] for 'Edit'.
6. From the edit-menu select [3] for 'Print'.
7. The default printer setting prints 21 lines (00 to 20). If desired the last line that is printed can be changed to any value between 10 and 99 by pressing [2] and [3]. Normally you should select the last line used in your program.
8. Switch the printer on.
9. Press [Enter] to start printing.
10. After printing has finished switch the printer off.
11. Press [MAIN] to return to the main-menu.

### Important Notes

Always switch both the printer and programmer off when connecting or disconnecting the printer cable.

The printing utility is quite basic in operation. It does not detect or report printer errors (such as no paper) and therefore the printer must be carefully checked before using the print utility.

It is not possible to alter the 'format' of the printout (font, margin spacing etc.).

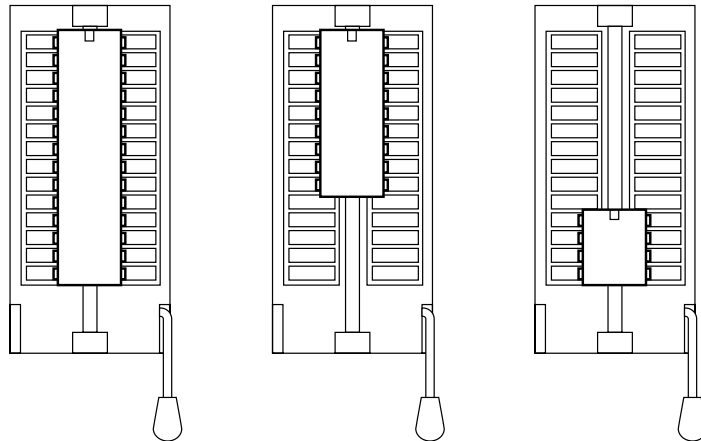
A line of 'garbage' characters may appear along the top line of the program listing if the printer is not switched off whilst it is connected to the programmer. To prevent this only switch the printer on when you are about to print and switch it off again as soon as printing has finished.

## PROGRAMMING THE PIC

The programming utility transfers the program in the programmer memory to the PIC microcontroller.

To program the PIC:

1. Insert the PIC into the ZIF socket ensuring pin 1 is correctly orientated.



2. From the main-menu select [3] for 'Program'.
3. The default run delay setting is off (0.0 seconds). If desired the run delay can be changed to any delay between 0.1 and 9.9 sec by pressing [2] & [3].
4. Press [Enter] to start programming.
5. Wait for the ACTIVE LED to go out.
6. Check that an 'OK' message is displayed on the LCD.
7. Remove the PIC.
8. Press [MAIN] to return to the main-menu.
9. If a 'FAIL' message appears the PIC program will not run correctly as the programming failed. Check that the PIC is correctly orientated in the programming socket, and that you are using the correct type of PIC. Also check that you are using a 9V DC power supply (e.g. NOT a 'weak' PP3 battery) to power the programmer. Before reporting an error also make sure that the first PIC is not damaged by trying to program another device.

## Using the Run-Delay feature

*The run delay feature has been removed from Version 6 of the Programmer due to code capacity limitations within the internal memory of the Chip Factory module.*

### **Important Programming Notes:**

Trying to program the wrong size of PIC (e.g. trying to program an 18-pin PIC when the program has been written for an 8-pin PIC) *could* permanently damage the PIC (although tests have shown that in practise the PIC normally survives!). It will not damage the programmer itself.

Accidentally programming an 18-pin 16F84A instead of the 18-pin 16F85 (or vice versa) will not generate a programming error. However the program will not run correctly if it contains any reference to input 0 or 1 (or sensor a or b).

Never remove the PIC from the ZIF socket when the ACTIVE LED is lit, as this could permanently damage the PIC.

Never try to program 16C or 12C series PICs via the free-standing programming method. (These devices can, however, be programmed when in the serial programmer mode). The Flash EEPROM programming algorithm used in the free standing mode could permanently damage these PICs. The code protect bit may also be set on windowed devices, making them un-erasable (an expensive mistake to make!).

## SECTION 2

### WINDOWS CHIP FACTORY PROGRAMMING SOFTWARE

#### Installing the software

Follow the installation instructions provide with the software disks.

#### Connecting the Programmer to the Computer.

##### PC (9 pin socket)

Connect the serial cable from the programmer to the serial port on the computer.

##### PC (25 pin socket)

Connect the serial cable to the programmer. Use a 9 to 25 way 'mouse' adapter to connect the other end of the cable to the computer.

##### Mac (Modem socket)

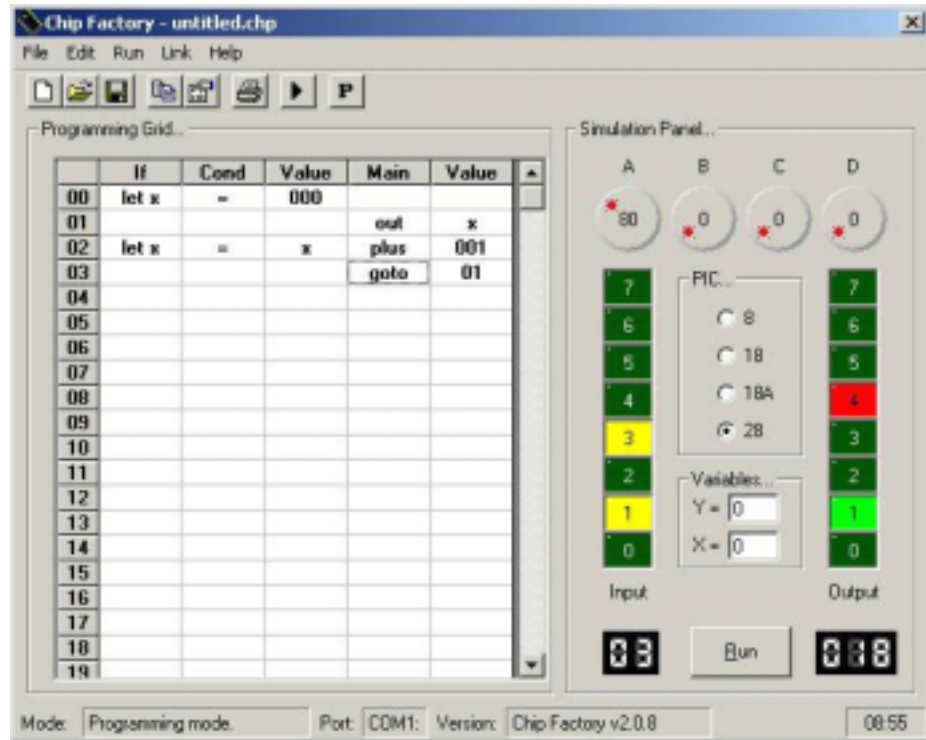
Connect the serial cable from the Programmer to the 'Modem' port on the computer. Note that you require SoftWindows™ to use the Windows software on a Mac.

#### Placing the Programmer in serial link mode

To place the programmer in serial mode select [2] 'Serial' from the main-menu.  
To leave serial mode press [MAIN].

## Using the Software

When you start the software the following window will appear:



### Menu Bar & Toolbar

These contains the usual menu options and 'shortcut' toolbar buttons.

### Programming Grid

The grid contains the actual program code. You can only see 20 of the program lines at once, and so the scroll-bar at the side of the grid must be used to move up and down the program.

### Simulation Window

This window shows the on-screen simulation of the Demo Board for 'trying out' programs.

### Status Bar

The status bar gives comments on how the software is configured.

### Programming

The programming grid operates in a very similar manner to the free standing mode System. Each program line is split into a number of sections - which correspond to the programmer buttons. To edit a line simply double-click on the part of the grid that you would like to change, and a pop-up menu of the available options will appear. Select the option you require by clicking on it

Note that the digits are 'linked' to the commands, and so you will not be able to select a number if no 'main' command has been selected. The computer will 'beep' if you try to select an unavailable digit in this manner.

### Downloading

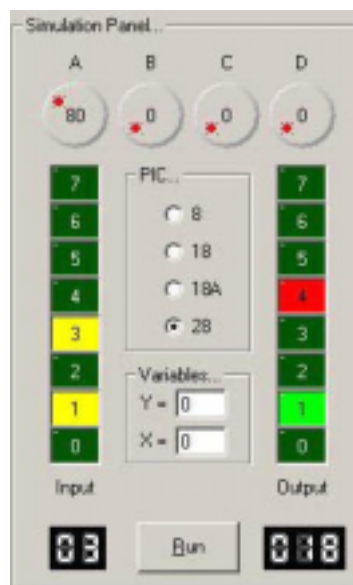
To download a program to a PIC microcontroller, make sure the Programmer is attached, and that the Programmer is in 'serial link' mode.

Click on the 'Download' button on the toolbar. The program will then be downloaded to the PIC microcontroller. If an 'error' message appears check that the cable and power supply are connected to the Programmer, that the PIC is inserted correctly, and that the Programmer is in 'serial link' mode.

Also check that you have selected the correct serial port via the 'Link' menu (The serial port currently selected is shown on the Status Bar)

Note that 'downloading' programs the PIC microcontroller directly. It DOES NOT alter the 'manual' program currently in the programmer memory. It is not possible to 'read' pre-programmed PICs as they are 'code-protected' to prevent illegal copying.

### Program Simulation



The simulation window allows you to 'try-out' programs without downloading them to the PIC microcontroller. This allows you to rapidly experiment with different programs. The on-screen simulation window provides a 'graphical' version of the PIC Demo board.

To start the simulation running click on the 'Start' button. The program will then run on-screen, starting from line 00. When the program reaches an un-programmed line it will stop automatically.

A 'looping' program can be stopped at any time by clicking on the 'Stop' button or by pressing the <Esc> or <Spacebar> keys on the keyboard.

To simulate an input click on the appropriate input LED to switch it on ("high") or off ("low"). To simulate an analogue input move the slider.

The speed the simulation runs at depends on your computer type. However no simulation will run as fast as a real PIC microcontroller, which can easily carry out all 100 program lines in under a second! If the simulation is running too fast to understand, you can add an extra time delay between each line to 'slow it down'. To do this select 'Simulation Speed' from the 'Edit' menu and select a delay between 0 to 5 seconds long.

**Tip!**

When a program is not being run you can click on the 'output' LEDs to switch them on and off. This alters the 'out' value shown beneath the LEDs. Therefore you can quickly calculate the out value to be used in a program by simply clicking on the LEDs until the correct output combination is shown. You cannot click on the output LEDs when a simulation is running (although you can still click on the input LEDs to change their value).

## Chip Factory Software Menus

Note that most menu commands also have a shortcut button on the toolbar.

### File Menu

New...	- start a new program
Open...	- open a file saved on disk
Save...	- save the current program
Save As...	- save the current program with a new name
Print...	- print the current program
Printer Setup...	- select the printer configuration
Exit	- exit the software

### Run Menu

Run	- run and on screen simulation
-----	--------------------------------

### Edit Menu

Insert Line...	- insert a new program line
Delete Line...	- delete the current program line
Simulation Speed	- alter the time delay between the simulation steps
Titles...	- add a title for printed / copied programs
Copy	- copy the program to the Windows clipboard

### Link Menu

Program PIC...	- program the PIC microcontroller
Read...	- read the program from the programmer memory
Download	- write the program from the programmer memory
Port...	- select the serial port to use

### Help Menu

Help Contents	- start the Help File
About...	- version details, programmer details and copyright notice

## SECTION 3

### PIC PROGRAMMER SOFTWARE

The starter kit includes the 'PIC Programmer' software which allows the programmer to be used as a conventional serial programmer for all 12F & 12C series 8-pin PICs, all 16F & 16C series 18-pin and 28-pin PICs.

#### Software Installation

Follow the installation instructions provide with the software disks.

#### Connecting the Programmer to the Computer

##### PC (9 pin socket)

Connect the serial cable from the programmer to the serial port on the computer.

##### PC (25 pin socket)

Connect the serial cable to the programmer. Use a 9 to 25 way 'mouse' adapter to connect the other end of the cable to the computer.

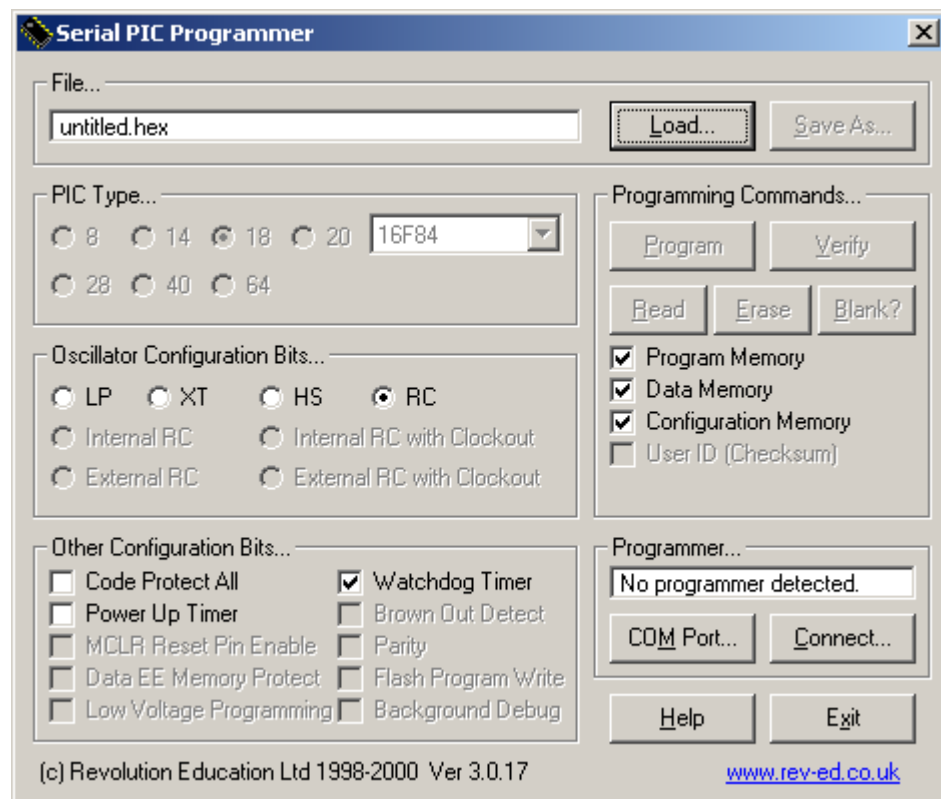
##### Mac (Modem socket)

Connect the serial cable from the Programmer to the 'Modem' port on the computer. Note that you require SoftWindows to use the Windows software on a Mac.

#### Placing the programmer in serial link mode

To place the programmer in serial mode select [2] 'Serial' from the main-menu.  
To leave serial mode press [MAIN].

## Using the PIC Programmer Software



When the PIC Programmer software is started the window above will be displayed. The general programming procedure is:

1. Select the appropriate PIC type.
2. Load the hex program file.
3. Check and set the oscillator configuration bits.
4. Check and set the other configuration bits.
5. Place the PIC in the programmer, ensuring correct polarity.
6. Program the PIC.

The software also supports the following features:

- Blank checking of a PIC.
- Verification of an unprotected PIC.
- Reading of an unprotected PIC.
- Erasing of Flash EEPROM memory PICs.

## PIC Programmer Software Features

### Load....

Loads any standard INTEL hex format file generated by MPASM (or any other assembler, C or BASIC compiler) to the internal buffer.

### Save As...

Saves the program in the internal buffer, and all fuse and configuration bits, to file.

### PIC Type...

Select the pin size of the PIC to be programmed, and then select the specific device from the drop-down list box.

### Oscillator Configuration Bits...

Use to select the oscillator type.

### Other Configuration Bits...

Use to disable and/or enable the other configuration bits.

### Program

Programs the PIC with the program in the internal buffer.

Program memory, configuration and data memory can be programmed individually or in any combination by checking the appropriate boxes.

### Verify

Reads the PIC and checks the reading against the internal buffer.

### Read

Reads the PIC and replaces the internal buffer with the reading.

### Erase

Erases the PIC. Only available for Flash EEPROM PICs. Use this command to 'clear' the code-protect configuration bit of a protected PIC.

### Blank?

Reads the PIC and confirms that it is blank.

### COM Port

Selects the serial port (COM1-4) used for communications.

### Connect

Use to 'connect' to the Programmer. This is not necessary if the Programmer is connected and switched on when the software is first started.

### Help

Provides on-line help.

### Exit

Exit from the PIC programmer software.

## SECTION 4

### INTERFACING TO THE PIC MICROCONTROLLER

#### Control Systems

All electronic control systems can be represented by block diagrams containing the following blocks - inputs, process and outputs.



When you use the PIC microcontroller as part of a control system it becomes the 'process' block, controlling the outputs according to the control program and information from the inputs. When using the PIC microcontroller within projects you will want to switch different types of output devices on and off (such as lamps, relays, motors, buzzers etc.) and will want to use sensors as well as switches to provide input information. It is possible to use the PIC microcontroller to do all of these things, but you will need to build the necessary interfacing circuits.

#### Important Note:

The PIC microcontroller outputs can be used to switch external devices on (high) or off (low). When 'High' they will be at 5 volts and when 'Low' at 0 volts.

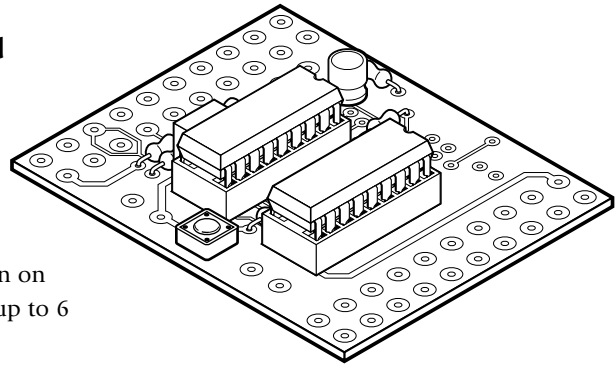
These output pins can only supply a limited amount of current. Each pin can **sink** ("absorb") 25mA and **source** ("give out") 20mA. The total sink and source for all 8 pins combined should never exceed **50mA sink** and **40mA source**. This means that you will need to use extra components to interface the PIC microcontroller to high current loads, e.g. motors or buzzers.

**Drawing excessive current from the PIC microcontroller will permanently damage the PIC microcontroller, which will then have to be replaced.**

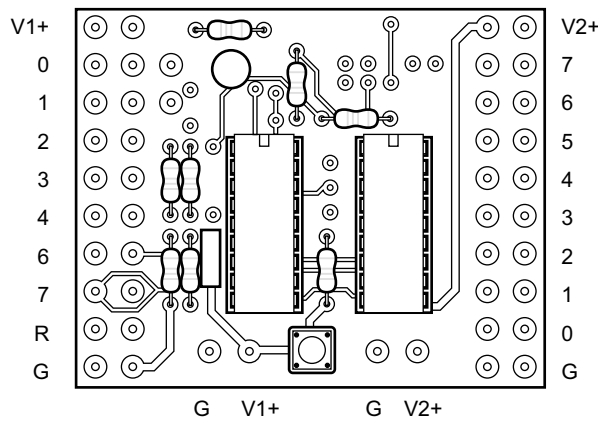
This section provides a brief introduction on how to interface simple components to the PIC microcontroller. More comprehensive interfacing details can be found in a separate publication 'Microcontroller Interfacing Circuits'.

### The 18 pin PIC Project Board

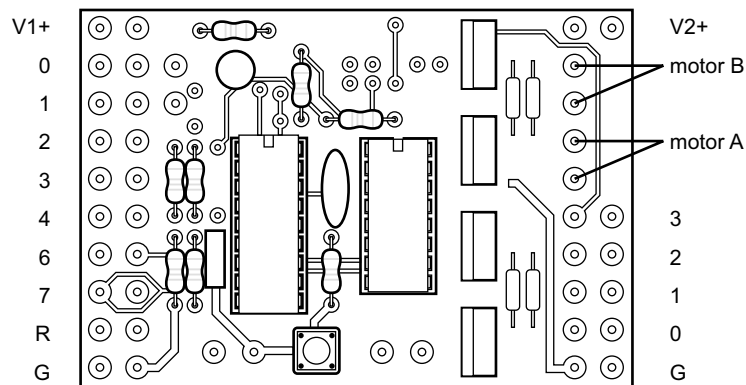
Project boards are available for the 8, 18 and 28 pin microcontrollers. The 18 pin microcontroller project boards are available in two configurations, standard and high-power. The input configuration on both boards is identical, providing up to 6 inputs.



The standard board uses a darlington driver IC to provide 8 digital (on/off) outputs. Each output is rated at 800mA.



The high power board uses 4 FETs to provide 4 high power digital outputs (rated at 1.5A each), and the option of a L293D motor driver IC to provide 2 reversible motor outputs, rated at 1A each. Please note that the L293D chip must be purchased separately.

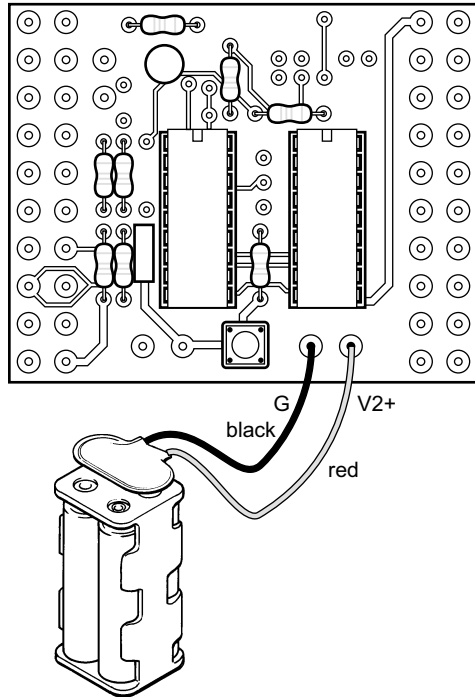


Both boards are supplied ready for use with a 16F627 microcontroller. However the boards may be modified as described below for use with a 16F84A (or any other type of 18 pin microcontroller).

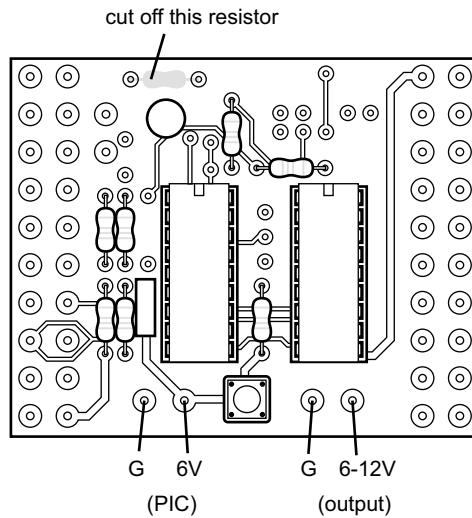
### Project Boards Power Supply

The projects board require a single 3-6V power supply to operate. We recommend this is supplied via AA cell battery packs, connected to the V2+ terminal connections beside the reset switch. This pack will then power the microcontroller and the output devices.

The black wire is connected to the G (ground) connection and the red wire to the V2+ connection.

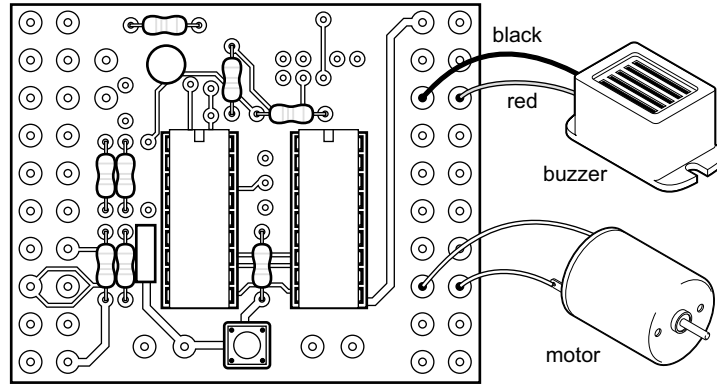


If a higher voltage (e.g. 12V) is required to drive the outputs, two separate power supplies may be used. In this case the second power supply only powers the output devices. The 3-6V power supply is connected to V1+ and the second 12V power supply is connected to V2+. When using two power supplies the resistor shown must be cut off the board to separate the two supplies.



### Output Devices

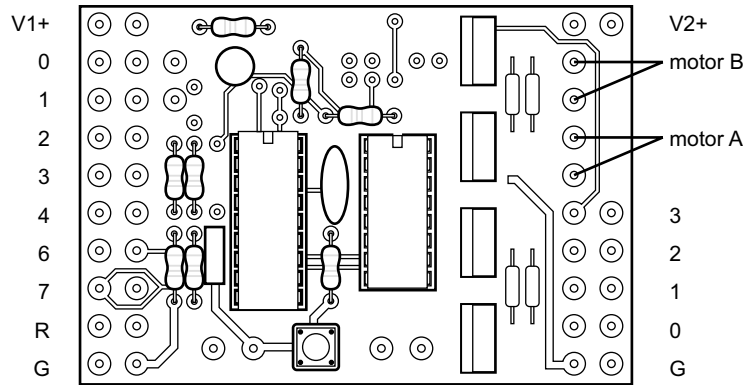
Output devices are connected between the pairs of holes on the pcb (pin and V2+) as shown below.



Note that motors should be suppressed by soldering a 220nF polyester capacitor across the motor terminals to prevent electrical noise affecting the circuit.

### Controlling Motors on the High Power Board

To control motors an L293D motor driver IC must be fitted to the board as shown below.



Note that there is a 1.5V voltage drop within the chip and so, for instance, if a 6V supply is used the motor voltage will be 4.5V.

The chip is designed to run warm in use. This is normal.

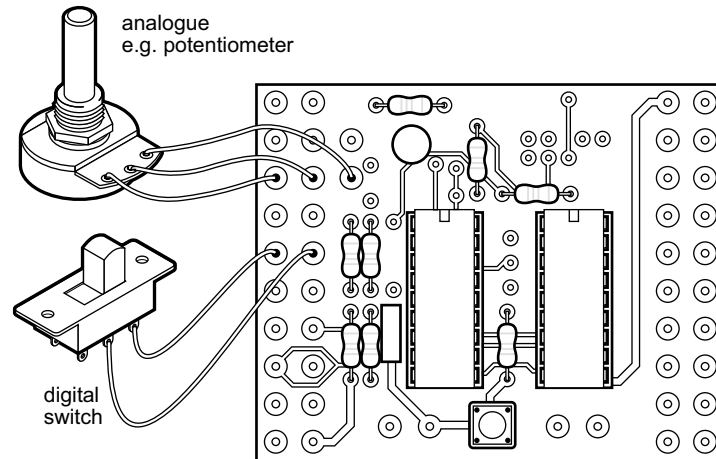
The direction of rotation of the motors is defined as follows:

Pin 4	Pin 5	Motor A
off	off	off
on	off	forward
off	on	reverse
on	on	off

Pin 6	Pin 7	Motor B
off	off	off
on	off	forward
off	on	reverse
on	on	off

## Input Devices

Analogue inputs are connected to input 0 (A) and input 1 (B) as shown.  
Digital inputs are connected between V1+ and the pin as shown below.



Note that input 5 does not exist. This is a characteristic of the microcontroller design.

### Reset Switch

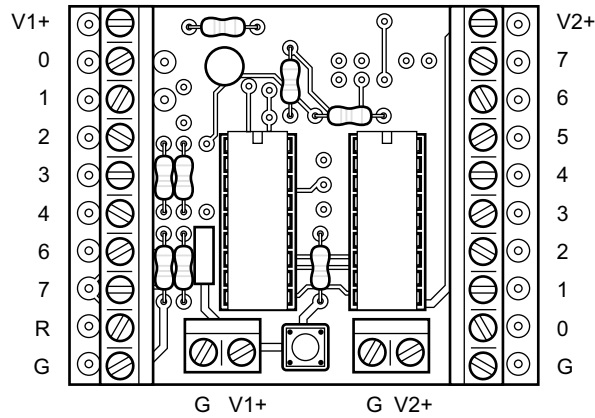
A small reset switch is provided on the board. If desired an external reset switch can be connected in parallel between the R input and G (ground, 0V).

### Piezo Sounder

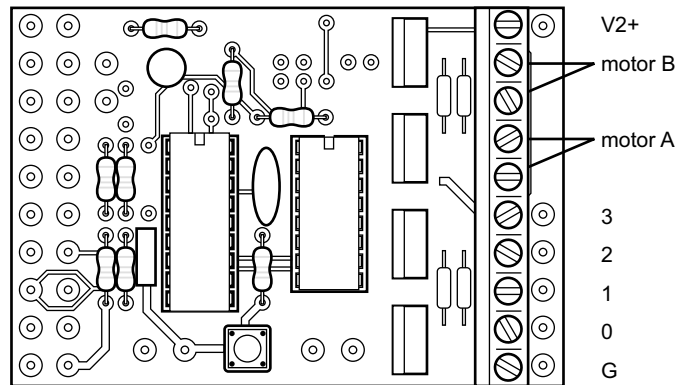
The Chip Factory system automatically re-configures input 4 as the sound output for 'beep' commands. The piezo sounder should be connected between V1+ and input 4 for correct operation.

### Connecting to the Project Boards

Inputs and outputs may be soldered via wires directly to the board. Alternately you may wish to purchase screw terminal blocks (5mm pitch) and solder these to the boards as shown below. This allows wires to be temporarily connected via the screw terminals.



Note that when using terminal blocks it is necessary to 'share' the V2+ output with all output pins and to 'share' the V1+ output with all inputs.

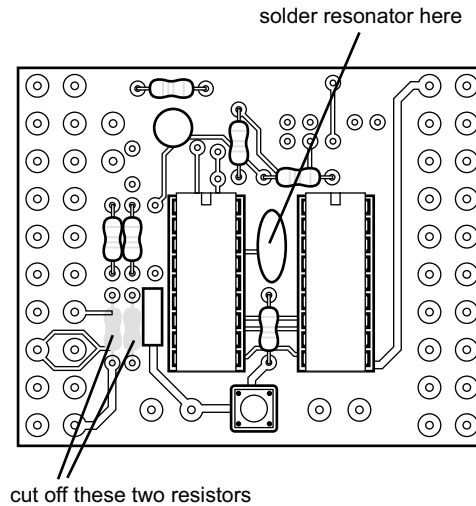


### Adapting the board for use with a PIC16F84A

The project board as supplied is configured for use with a PIC16F627 using the internal resonator.

To modify the board for use with a PIC16F84A, or any other 18 pin microcontroller that requires an external resonator, the following modifications are required.

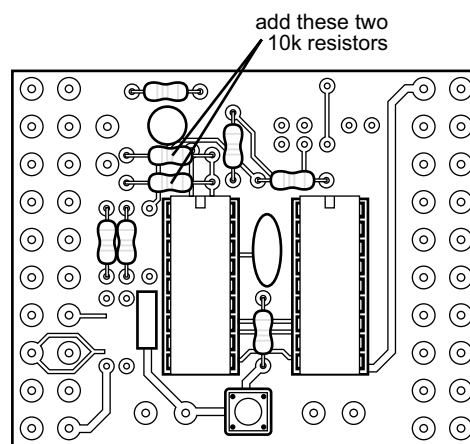
(Note that these changes apply to both boards, although the standard board is illustrated).



Cut off the two 10K resistors as shown.

Solder a 4MHz 3 pin ceramic resonator as shown.

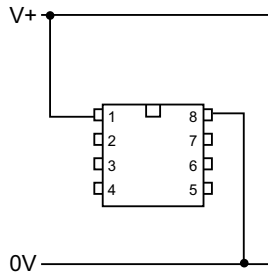
No external connection should be made to inputs 6 and 7, as these are now the resonator pins.



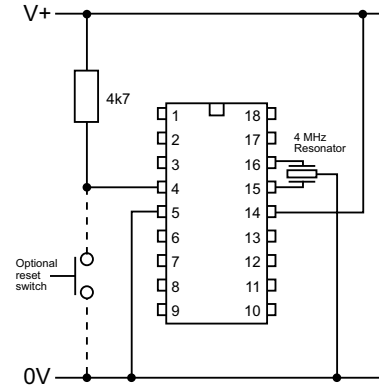
In addition inputs 0 and 1 now become digital inputs (rather than analogue inputs as with the 16F627). This requires the addition of two extra 10k pull-down resistors as shown.

**Minimum Hardware Circuit**

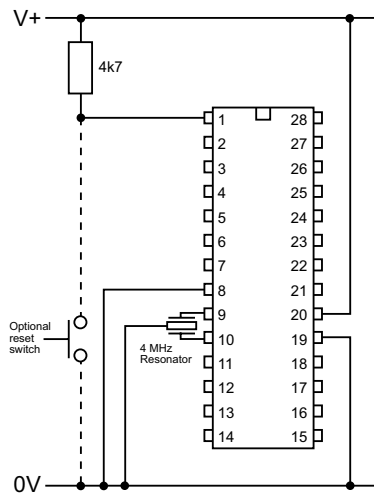
Minimum 8-pin Circuit



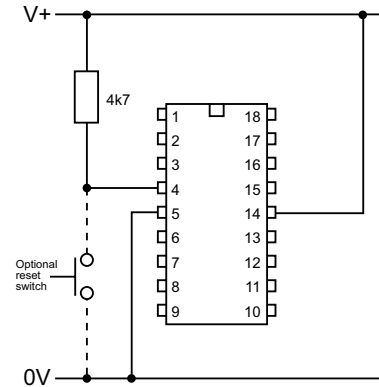
Minimum 18-pin Circuit (18)



Minimum 28-pin Circuit



Minimum 18-pin Circuit (18L/A)

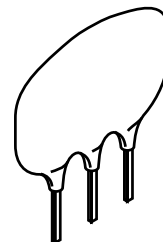


Both 18-pin and 28-pin PIC microcontrollers require ALL of the following connections to operate correctly:

	18	18A	18L	28	
Vdd	pin 14	pin 14	pin 14	pin 20	power supply, must be connected to V+
Vss	pin 5	pin 5	pin 5	pin 8/19	ground, must be connected to 0V
OSC1	pin 15	---	---	pin 9	clock, connect to one outer pin of resonator
OSC2	pin 16	---	---	pin 10	clock, connect to other outer pin of resonator
MCLR	pin 4	pin 4	pin 4	pin 1	reset, connect to V+ via 4.7 kilo-ohm resistor

**Notes:**

The centre pin of the ceramic resonator must be connected to ground (0V).  
If a manual reset switch is required it must be connected between MCLR and ground.

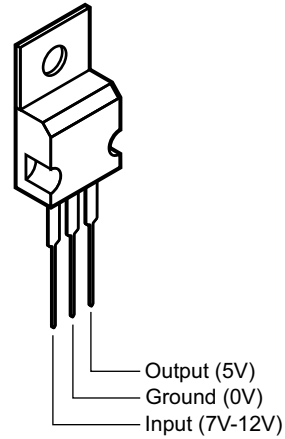


The 8 and 18L, 18A microcontrollers have an internal resonator. Therefore it is only necessary to connect the power rails.

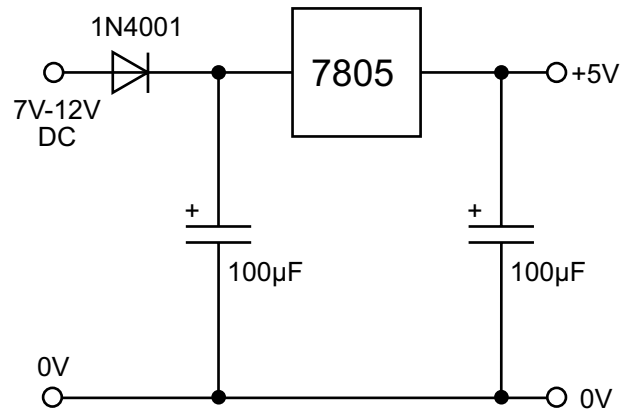
### Generating a 5V Voltage supply

PIC Microcontrollers are designed to work with a power supply of 3 to 6V. They can therefore be used directly with 'battery packs' made up of 2, 3 or 4 1.5V AA cells.

If you wish to use a higher voltage power supply for the circuit (e.g. 9 or 12V) you must use a 5V voltage regulator such as type 7805.



5V Voltage Regulator Circuit Diagram.

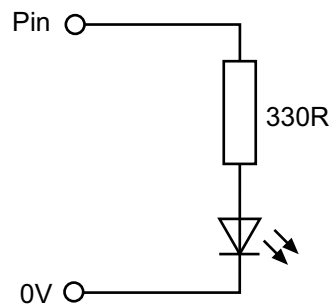


Note that the 1N4001 diode provides reverse polarity protection if the power supply is accidentally connected back to front. The electrolytic capacitors 'smooth' the power supply for more reliable use.

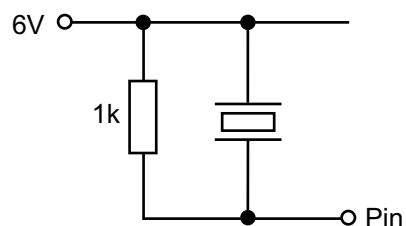
### Output Devices that do not Require Interfacing to PIC Microcontrollers

The PIC microcontroller can only supply a small amount of current to any device connected to an output pin (20mA maximum). Any output device that requires more current than this will need to be interfaced to protect the PIC microcontroller from damage.

LEDs can be connected directly to an output pin as long as a series resistor is used. A typical resistor value would be 330 ohms.



The piezo-transducer sounder can be connected between + 5V and pin 3 of an 18-pin PIC microcontroller or pin 6 of a 28 pin PIC microcontroller. Note that a 1 kilo-ohm resistor **MUST** also be connected in parallel to the piezo for the piezo to operate correctly. (This resistor is already included on the PIC project board)

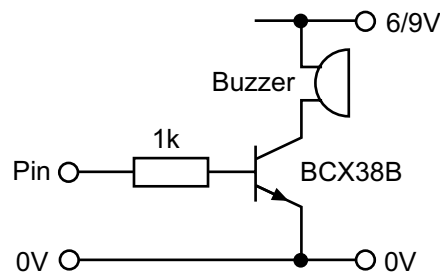


## Using a Transistor to Interface Outputs

Almost all output devices other than LEDs will require the addition of a transistor switching circuit. An ideal transistor to use is the BCX38B, connected as shown in the following circuits. The BCX38B is a Darlington device, and can switch currents up to 500mA.

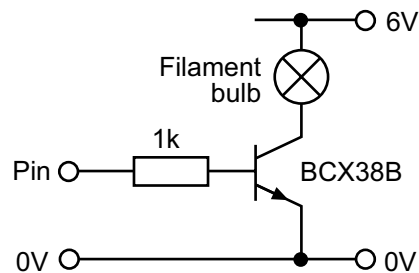
### Buzzer

Note that the buzzer is powered by an external power supply. When you use two power supplies like this it is essential to join the two 0 volt lines.



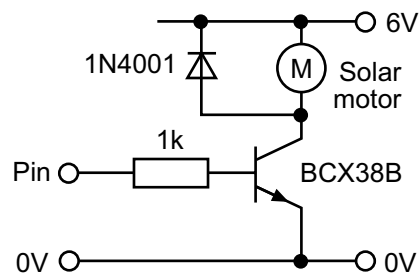
### Signal Lamp

Note that the signal lamp is powered by an external power supply. When you use two power supplies like this it is essential to join the two 0 volt lines.



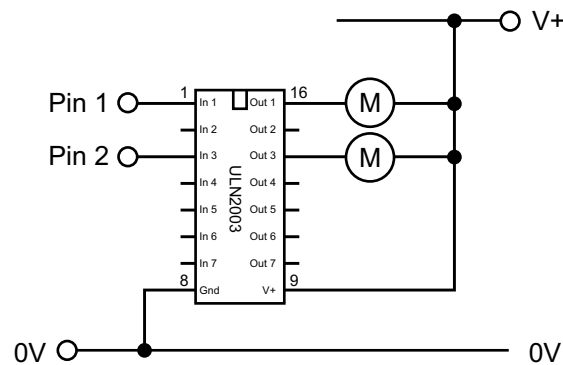
### Solar Motor

Note that the signal lamp is powered by an external power supply. When you use two power supplies like this it is essential to join the two 0 volt lines. Also note the use of a back EMF suppression diode across the motor contacts. This is to prevent damage to the transistor when the relay switches off. Diode type 1N4001 is suitable for this diode.



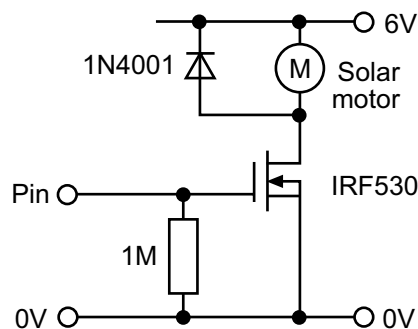
### Using a Darlington Driver IC to Interface Outputs

If you are controlling a number of output devices you may need to use a number of output transistors. In this case it will be more convenient to use an ULN2803 Darlington driver IC. This is simply an 18 pin 'chip' that contains 8 Darlington transistors similar in value to the BCX38B. The 'chip' also contains back EMF suppression diodes and so no external diodes are required.



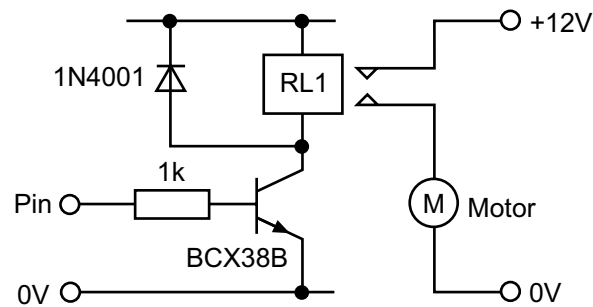
### Using a Power MOSFET to Interface Outputs

A power MOSFET can be used to replace the transistor switching circuit in some circuits. An ideal FET to use is the type IRF530, connected as shown in the following circuit.



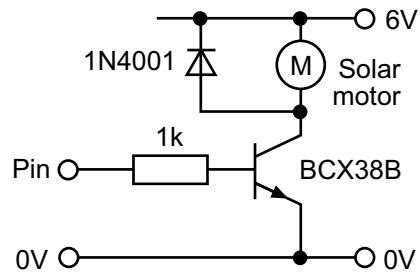
### Using a Relay to Interface Outputs

A relay can be used to switch higher power devices such as motors and solenoids. If desired, the relay can be powered by a separate power supply, so 12V solenoids, for example, can be controlled by the PIC microcontroller. When using two power supplies like this it is not necessary to join the two 0 volt lines as the two power supplies are kept separate from each other. Note the use of a back EMF suppression diode across the relay contacts. This is to prevent damage to the transistor when the relay switches off. Diode type 1N4001 is suitable for this diode.

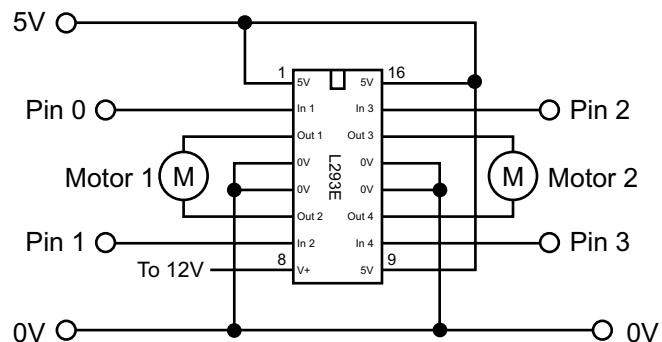


### Using a Motor Controller IC to Control Two Motors

The following circuit uses a Darlington transistor to switch a motor on and off.



This circuit will only allow the motor to run in one direction. In many cases you may want to run the motor in both directions (forwards and reverse). In this case it is convenient to use a motor driver IC, such as the L293D. This IC will allow control of two dc motors, using four outputs from the PIC microcontroller. Naturally if you only wish to control one motor you only need to use two output lines.



Both inputs low - motor halt

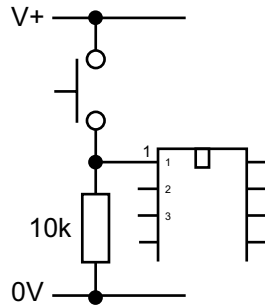
First output high, second output low - motor forward

First output low, second output high - motor reverse

Both inputs high - motor halt

### Interfacing Switch Inputs

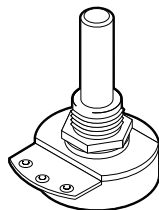
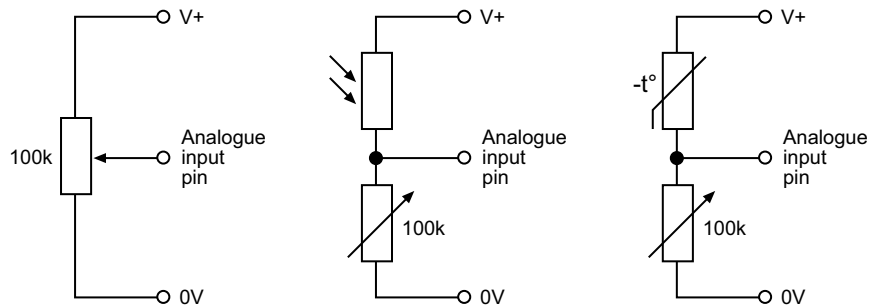
Different types of switch can be connected as shown to provide various input signals. Note that the 10 kilo-ohm 'pull-down' resistors are already included on the PIC project board.



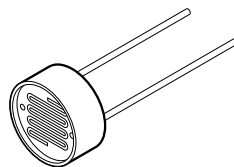
**Never apply a voltage that is greater than +5 volts to an input pin, or 'short-circuit' an input pin, as the PIC Microcontroller will be permanently damaged.**

### Interfacing Analogue Sensors

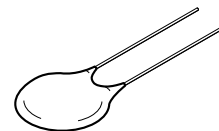
Digital inputs are input signals that are either on (0 volts) or off (+5 volts). In some projects you may wish to create a digital signal from an analogue input device such as a Potentiometer, Light Dependant Resistor, Thermistor or Moisture Sensor. These sensors can be used via the analogue inputs of the PIC16F627, PIC16F819 or PIC16F872 microcontrollers.



**Potentiometer**  
The simplest analogue sensor is simply a potentiometer connected as shown. When the potentiometer spindle is turned, the voltage applied to the analogue input will vary from 0V to +5V.



**Light Dependent Resistor (LDR)**  
A LDR and a variable resistor can give an analogue output voltage. The variable resistor allows the 'sensitivity' of the potential divider to be adjusted.



**Thermistor and Moisture Sensor**  
The LDR can be replaced by a thermistor or a moisture sensor. You may have to experiment with resistor values to produce a system that responds exactly as you wish.

### Using the Low-Resolution Analogue Input (16F627)

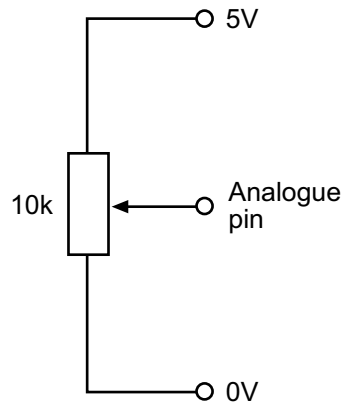
A standard analogue input will provide 256 different analogue readings (0 to 255) over the full voltage range (e.g. 0 to 5V) (PIC16F819 and PIC16F872). A low-resolution analogue input will provide 16 readings over the lower two-thirds of the voltage range (e.g. 0 to 3.3V) (PIC16F627). No readings are available in the upper third of the voltage range.

To ensure consistency between standard and low-resolution analogue input readings, the low-resolution reading will 'jump' in 16 discrete steps between the nearest standard readings, according to the table below.

Standard Reading Range	Low Resolution Reading
0-10	0
11-20	11
21-31	21
32-42	32
43-52	43
53-63	53
64-74	64
75-84	75
85-95	85
96-106	96
107-116	107
117-127	117
128-138	128
139-148	139
149-159	149
160-170	160
Values greater than 170 (170-255)	160

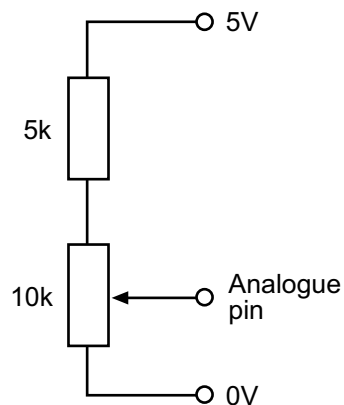
**Advanced technical information on using the low resolution ADC (PIC16F627)**

The low-resolution analogue pin within the microcontroller has an internal fixed resistor at the upper end of the voltage range. This results in an unavoidable 'dead-spot' between the standard values 160 and 255. Therefore if a simple potentiometer is used as the sensor (between 0 and 5V), this will result in the analogue value not changing above 160 for approximately the last third rotation of the potentiometer spindle.



For many projects, particularly when using LDR or thermistor sensors, this is of no consequence and the upper 'non-changing' area is simply ignored.

However, if desired, this area can be avoided by using an additional fixed resistor at the top of the potential divider circuit. This fixed resistor should have a value of 0.5 of the total resistance of the original potential divider circuit.



For example, when using a 10k potentiometer, a fixed 5k (5k1) resistor should also be included. In this case the highest analogue value read will still not exceed 160, but the changing values will be distributed over the full turning range of the potentiometer spindle (rather than two thirds of the turning range of the spindle).

# APPENDICES

## APPENDIX A: PROGRAMMING QUICK GUIDE

### Utilities...

Edit an existing program	From the main-menu select [1] for 'Edit' and then [1] for 'Edit'.
Start a new program	From the main-menu select [1] for 'Edit' and then [2] for 'New'.
Select a new PIC type	From the main-menu select [1] for 'Edit' and then [2] for 'New'.
Print a program	From the main-menu select [1] for 'Edit' and then [3] for 'Print'.
Program a PIC	From the main-menu select [3] for 'Program'

### Program Editing Functions...

Edit the main command	Press [MAIN].
Edit the main command digits	Press [1], [2], or [3].
Edit the if (or let) statement	Press [IF] and/or [COND].
Edit the if statement digits	Press [SHIFT] + ([1], [2], or [3]).
Save the line	Press [ENTER]
Move down one line	Press [ENTER]
Move up one line	Press [SHIFT] + [ENTER]
Insert a new line	Press [SHIFT]+[IF]
Delete the current line	Press [SHIFT]+[COND]
Return to main menu	Press [SHIFT]+[MAIN]

### Serial Programmer Mode...

Enter serial link mode	From the main-menu select [2] for 'Serial'.
Exit serial link mode	Press [MAIN]

## Main Commands Summary

Main Command	LCD Display		Values:
switch output N on	high	N	N =0-7 (see 1)
switch output N off	low	N	N =0-7 (see 1)
jump to line NN	goto	NN	NN = 99
jump to line in variable V	goto	V	V = x or y (see 2, 3)
wait for NN.N seconds	wait	NNN	NNN =000-255
wait for time in variable V	wait	V	V = x or y (see 3)
switch all outputs to value NNN	out	NNN	NNN =000-255 (see 4)
switch all outputs to value in V	out	V	V = x or y (see 3, 4)
make sound of tone NNN	beep	NNN	NNN =000-255 (see 5)
make sound of tone value in V	beep	V	V = x or y (see 3, 5)

### Notes:

1. With the PIC12F629 only outputs 0-3 are available
2. If the variable contains a number larger than 99 the program will jump to line 99. Therefore the variable value should be kept lower than 99.
3. With the PIC16F819 the variable can also be the sensor values a or b.
4. With the PIC12F629 only the lower four bits of the number (or variable value) are transferred to the outputs on the 'out' command.
5. With the PIC12F629 the 'beep' command acts on output 2.

## IF Conditions Summary

If Condition	LCD Display	16F84 (18 pin)	16F818 (18 pin)	16F872 (28 pin)
if input 0 is off...	if 0 off	Y	N	Y
if input 0 is on...	if 0 on	Y	N	Y
if input 1 is off...	if 1 off	Y	N	Y
if input 1 is on...	if 1 on	Y	N	Y
if input 2 is off...	if 2 off	Y	Y	Y
if input 2 is on...	if 2 on	Y	Y	Y
if input 3 is off...	if 3 off	Y	Y	Y
if input 3 is on...	if 3 on	Y	Y	Y
if x = number...	if x=NNN	Y	Y	Y
if x = variable...	if x= V	Y	Y	Y
if x > number...	if x>NNN	Y	Y	Y
if x > variable...	if x> V	Y	Y	Y
if x < number...	if x<NNN	Y	Y	Y
if x < variable...	if x< V	Y	Y	Y
if x != number...	if x!NNN	Y	Y	Y
if x != variable...	if x! V	Y	Y	Y
if y = number...	if y=NNN	Y	Y	Y
if y = variable...	if y= V	Y	Y	Y
if y > number...	if y>NNN	Y	Y	Y
if y > variable...	if y> V	Y	Y	Y
if y < number...	if y<NNN	Y	Y	Y
if y < variable...	if y< V	Y	Y	Y
if y != number...	if y!NNN	Y	Y	Y
if y != variable...	if y! V	Y	Y	Y
if a = number...	if a=NNN	N	Y	Y
if a = variable...	if a= V	N	Y	Y
if a > number...	if a>NNN	N	Y	Y
if a > variable...	if a> V	N	Y	Y
if a < number...	if a<NNN	N	Y	Y
if a < variable...	if a< V	N	Y	Y
if a != number...	if a!NNN	N	Y	Y
if a != variable...	if a! V	N	Y	Y
if b = number...	if b=NNN	N	Y	Y
if b = variable...	if b= V	N	Y	Y
if b > number...	if b>NNN	N	Y	Y
if b > variable...	if b> V	N	Y	Y
if b < number...	if b<NNN	N	Y	Y
if b < variable...	if b< V	N	Y	Y
if b != number...	if b!NNN	N	Y	Y
if b != variable...	if b! V	N	Y	Y

where NNN = a number between 000 and 255

V = the variable x or y (and a or b with 16F85)

**LET Statement Summary****LET statement**

let x = number  
 let x = variable  
 let x = number + number  
 let x = variable + number  
 let x = variable + variable  
 let x = number - number  
 let x = variable - number  
 let x = variable - variable  
 let y = number  
 let y = variable  
 let y = number + number  
 let y = variable + number  
 let y = variable + variable  
 let y = number - number  
 let y = variable - number  
 let y = variable - variable

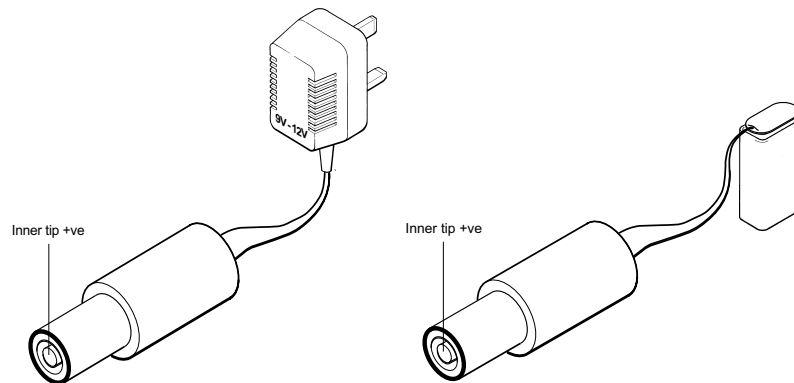
**LCD Display**

let x=NNN  
 let x= V  
 let x=NNN plus NNN  
 let x= V plus NNN  
 let x= V plus V  
 let x=NNN take NNN  
 let x= V take NNN  
 let x= V take V  
 let y=NNN  
 let y= V  
 let y=NNN plus NNN  
 let y= V plus NNN  
 let y= V plus V  
 let y=NNN take NNN  
 let y= V take NNN  
 let y= V take V

where NNN = a number between 000 and 255

V = the variable x or y (and a or b with 16F85)

## APPENDIX B: PROGRAMMER POWER SUPPLY



The programmer is designed to be run from a regulated 'plug-in' style mains transformer, rated at 300mA 9V DC. Supply is via a standard 2.1mm, tip positive, DC power plug.

It is not recommended that the programmer is run from batteries. However, due to the low power consumption of the unit, the programmer will operate correctly from an alkaline 9V PP3 battery for several hours. It must be noted that when the battery begins to 'fade' the programmer will appear to operate correctly, but PIC programming will always fail. This is because the high voltage programming power supply required during the programming process cannot be generated from a supply of less than 9V DC. Under these conditions the battery must be replaced.

## Appendix C: Upgrading your Programmer

As more PIC devices are released into the market it may be necessary to upgrade the programmer to support new programming protocols and more features and/or devices.

Two types of upgrade may be available in the future:

### Firmware Upgrade

In this case the programmer internal firmware is upgraded serially via the PIC Programmer software program running on a host PC. No physical adjustments are necessary to the programmer itself. The latest version of the PIC Programmer software can be downloaded free from [www.rev-ed.co.uk](http://www.rev-ed.co.uk).

### Hardware Upgrade

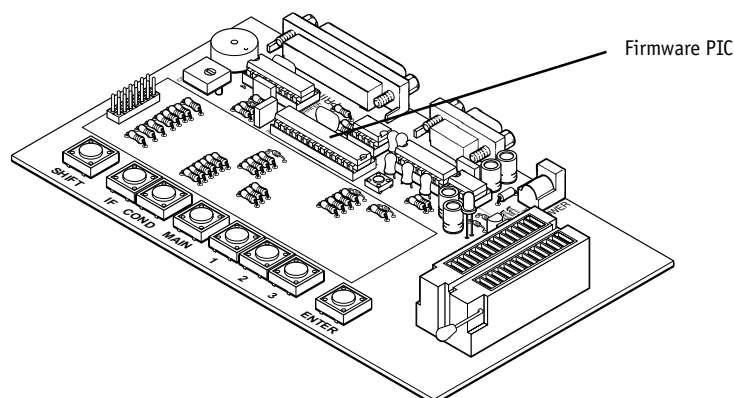
In this case the programmer internal microcontroller is replaced. This involves removing the top panel, replacing the chip and then replacing the top panel. This takes approximately five minutes and the only tool that is required is a small cross-head screwdriver.

#### Performing a Firmware Upgrade

1. Install the new PIC Programmer software on your computer.
2. Connect the programmer serial cable to the computer.
3. Switch the programmer on.
4. From the main-menu select [2] for 'Serial Link' mode.
5. Start PIC Programmer software on the computer.
6. Hold down the <SHIFT> key on the computer keyboard and press <U>.
7. Follow the on-screen instructions.

#### Performing a Hardware Upgrade

1. Disconnect the programmer power supply and serial/printer cables.
2. Remove the five screws holding the top panel in place.
3. Remove the top panel.
4. Carefully extract the long 28 pin firmware PIC (just above the LCD).
5. Install the replacement PIC, ensuring pin 1 (notched end) is to the right. Carefully check that all pins have entered the socket correctly.
6. Replace the top panel, ensuring the LED is correctly fitted through the hole.
7. Replace the five top panel screws.
8. Reconnect the power. The programmer should function correctly. If the LCD just shows 'squares' along the top row then the chip is incorrectly inserted.



## APPENDIX D: ABBREVIATIONS

<b>DC</b>	Direct Current
<b>PIC</b>	Peripheral Interface Controller
<b>PCB</b>	Printed Circuit Board
<b>EEPROM</b>	Electrically Erasable Programmable Read Only Memory