PIC LOGICATOR™

A guide to using PIC-Logicator software and connecting to the PIC microcontroller

ECONOMATICS®

Contents

Section One:
PIC-Logicator Software

Section Two:
Connecting to the PIC Microcontroller

PIC-Logicator Pack

The PIC-Logicator Pack contains:

1. PIC-Logicator software for Windows (Site licence).

2. Programmer.

3. Serial lead to connect the Programmer to the computer.

4. Plug-in type power supply for use with the Programmer.

5. This book.

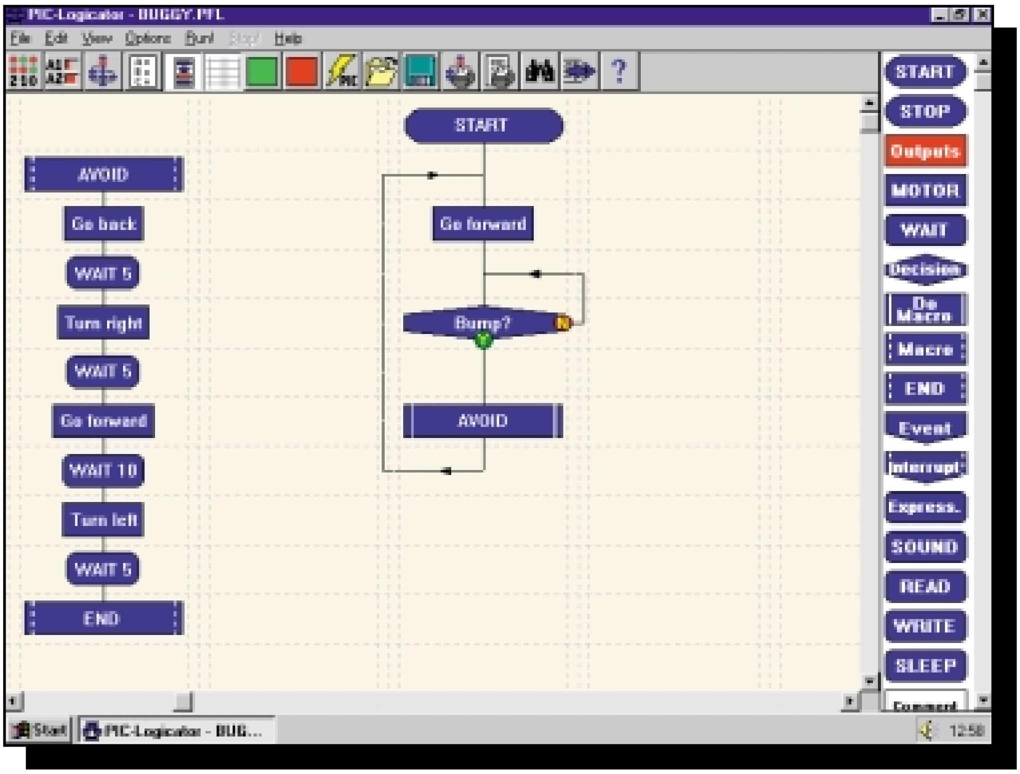
Getting Started

Install the PIC-Logicator software onto the hard drive of your computer using your normal method of installing software. Then see Section One for information on how to use the software and the Programmer. See Section Two for information on connection to the PIC microcontroller chip.

Website

The Economatics website provides the latest PIC-Logicator news and information. It includes: FAQs, projects and programming ideas, PCB designs to download, and an interactive demo of PIC-Logicator software.

www.economatics.co.uk/education



Section One:

PIC-Logicator Software

Overview

PIC-Logicator provides a graphical environment for designing, testing, editing and downloading control sequences for PIC microcontrollers.

The range of PIC-Logicator commands allows you to control output devices, such as motors and lamps, that are connected to the PIC microcontroller. You can switch devices on or off in sequences using: timing, counting, repetition, and decisions based on signals from digital and analogue sensors that are connected to the PIC microcontroller.

This section of the book explains how the software is used, giving examples of the various commands and techniques in the context of possible school projects. It is organised under the following headings:

1.How to build, edit and test run a PIC-Logicator flowsheet

2. Outputs

This section shows:

- how to switch output devices and motors connected to outputs of a PIC microcontroller, using Outputs, MOTOR, SOUND and OUT commands;
- how timing can be built into a control system using WAIT or SLEEP commands;
- how the SEROUT command can be used to output serial information from the PIC microcontroller.

3. Inputs

This section shows:

- how to check the state of digital sensors connected to a PIC microcontroller using the Digital Decision command;
- how to use Event and Interrupt commands for instant response to digital sensors;
- how to use the Compare Decision command to make use of readings from analogue sensors connected to a PIC microcontroller, in a control system.

4. Macros

This section shows the important technique of building a control system as a number of linked sub systems.

5. Variables

This section shows:

- how to create counting systems using INC and DEC commands;
- how timing can be built into a control system using the TIME variable;
- how Expression, IN and RND commands are used to give a value to a variable;
- how READ and WRITE commands are used to store and access values of variables using the PIC microcontroller’s EEPROM memory.

Quick Start

If you are unfamiliar with the Logicator approach to building control systems, it is a good idea to begin by familiarising yourself with the most commonly used commands which are: Outputs, WAIT, MOTOR and Digital Decision (see the Index of Commands page 8). Build and test run the Examples, using section 1 (“How to build, edit and test run a PIC-Logicator flowsheet”) as reference to help you.

Index of PIC-Logicator Commands

Command List

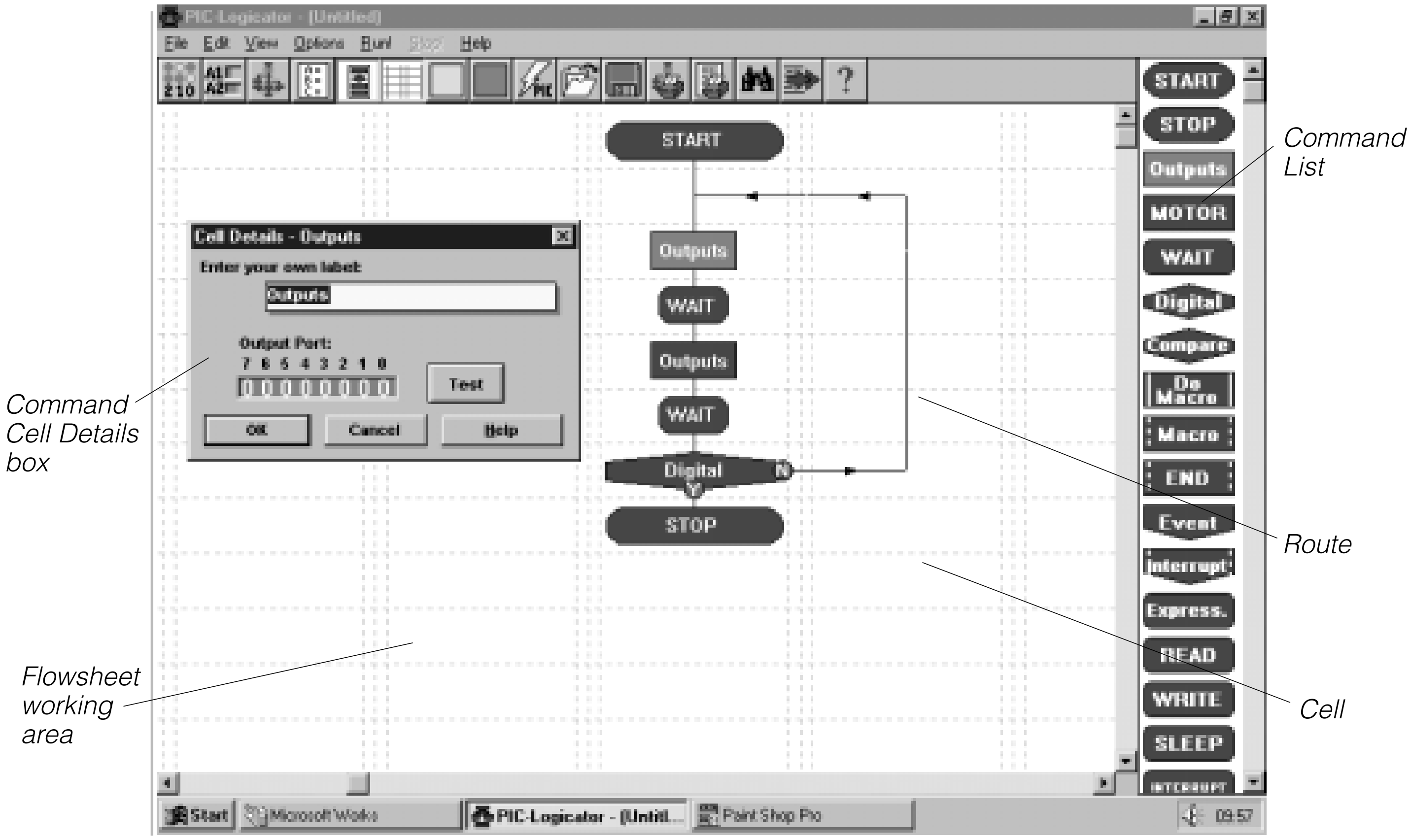
PIC-Logicator commands are accessed from the Command List on the right hand side of the screen. Scroll the list to see the full range of commands. You can use the Options>Setup Command List menu to customise the list, by changing the number of commands included, and the sequence in which they appear.

1. How to build, edit and test run a PIC-Logicator flowsheet

In PIC-Logicator, you create your control system in the form of a flowchart by dragging commands from the Command List and placing them in cells on the flowsheet working area (See fig. 1.1).

You can then use the commands' Cell Details boxes to fill in their details as required, and complete the flowsheet by drawing routes to connect the cells.

When the flowsheet runs, the flow of control follows the route you have drawn, carrying out the command in each cell as it passes through it.



Cell Details - Outputs

Enter your own label: Outputs

Output Port:
7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0

Fig. 1.1. Pic-Logicator screen

Options>PIC Type

Before you begin to build a flowsheet, you should decide which PIC microcontroller chip you intend to use in your project.

Select this chip from the Options>PIC Type menu. When you select a chip, the software automatically configures itself to display only the input, output and motor options available with that chip.

NOTE: The Options>PIC Type menu includes a "None" option. If you select this, you will find that the software configures itself to show 8 outputs, 4 motors, 8 digital inputs and 4 analogue inputs. You can build and test run a flowsheet using this option, but you cannot download it to the Programmer.

Commands

NOTE: This chapter deals only with drawing the flowsheet. Details of how to use the various PIC-Logicator commands are given elsewhere in Section One.
See the Index of commands on page 8.

Creating a command cell

Drag the required command from the Commands List and place it on an unoccupied cell.

Most commands have their own Cell Details dialog box which allows you to enter the command details. Double click on the command to open its Cell Details dialog box, and set the details of the command as required.

When you have set the necessary details, click OK to close the dialog box.

START and STOP commands

These two commands do not have Cell Details dialog boxes. Simply place them on the flowsheet working area.

A START command marks the point where the flowsheet starts running. When the PIC microcontroller is reset or powered up, the flowsheet starts at the START command. Every flowsheet must have a START command.

A flowsheet will stop running whenever a STOP command is reached.

Labelling a command

It can be useful to give a command a label which identifies what it is used for, e.g. “switches on lamp”. When you open a Cell Details box, the text in the yellow “label” box will be highlighted, so just type your label and click OK. This text does not affect the operation of a command; it is only a label.

Comment

Comment commands allow you to add short explanatory notes to a flowsheet. Although you can type up to 34 characters into the text box in the Cell Details box, the number of characters actually appearing in a Comment cell on the flowsheet will depend on factors such as the Zoom setting and screen setting. The default screen setting shows up to 16 characters in a Comment cell.

Comments have no effect on the operation of a flowsheet.

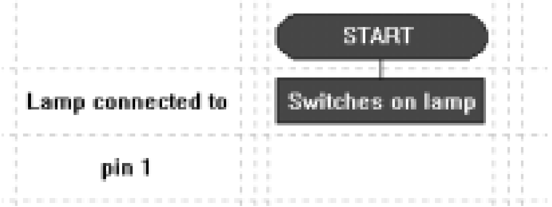


Fig 1.2 Explanatory information can be added to the flowsheet by using command labels and Comment commands.

Selecting a block of commands

Hold down the Ctrl key as you click at the top left hand corner of the block and drag down to the right to enclose the required commands and routes in the selection frame.

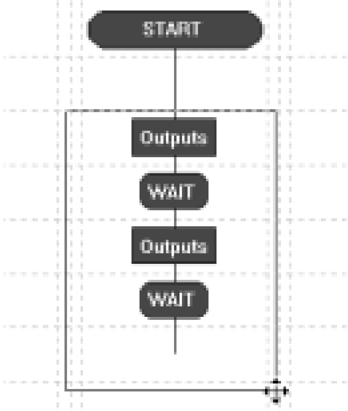


Fig 1.3. A block of commands in the selection frame.

Selected commands are coloured red. To deselect commands, click on another part of the flowsheet.

Deleting a command

Click on the command to select it. Selected commands are coloured red.

Press the Delete key to delete the selected command.

To delete a block of commands, select the block and press the Delete key.

Moving commands

To move a single command, drag it to its new position.

To move a block of commands, select them and drag them to their new position.

Cutting, Copying and Pasting

Use the Cut, Copy and Paste options from the Edit menu to cut or copy selected commands or blocks of commands and paste them either into another part of the same flowsheet or into a different flowsheet.

Alternatively, you can copy commands or blocks of commands within a flowsheet by first selecting them and then holding down the Ctrl key as you drag them to their new position. Remember that copied commands will retain their existing cell details.

Flowsheet working area

Cells are arranged in rows and columns. Each flowsheet has 20 columns and 100 rows. The default screen shows just 5 columns and 12 rows.

Use the View>Zoom menu if you want to change the number of cells visible on the screen.

Map

The Map option allows you to view the whole of the flowsheet at once. The red square marks the area currently displayed on the screen. If you click on any part of the Map, the red square will move to that area, and that area will be displayed on the screen.

Routes

Routes can be drawn through the middle of a cell, or in either one of the two rails between cells, as shown in fig 1.4.

Routes must be drawn in the direction that you want flow to take when the flowsheet runs.

Drawing routes

Click with the right mouse button on the command at the start of the route. The cursor will change to a pencil. Hold down the right mouse button and draw the route. Release the mouse button when you have completed the route.

Deleting routes

Click at the beginning of the route to be deleted, and press the Delete key.

When you draw a new route from a command, the existing route from the command will automatically be deleted.

To delete a route without deleting the command in which it starts: first click on the command to select it. Then hold down the Ctrl key as you press the Delete key.

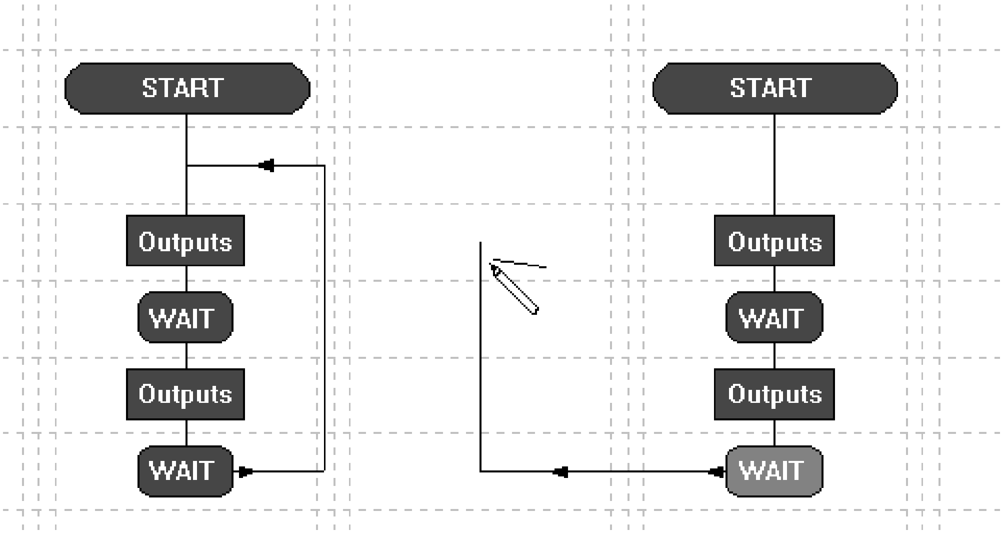


Fig 1.4. Routes can be drawn through cells or between rails.

How to test run a flowsheet

Before you download a flowsheet to a PIC microcontroller, it is useful to be able to check that it works as you intend it to. PIC-Logicator has a number of features that allow you to test run the flowsheet in the software.

1. The Digital Panel

As a flowsheet runs, the Digital Panel shows the changing state of outputs, motors and inputs as they would be if the flowsheet had been downloaded to a PIC microcontroller.

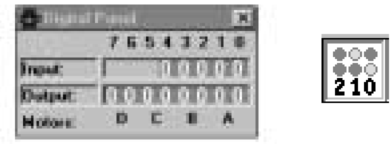


Fig 1.5. Digital Panel and its toolbar icon

To display the Digital Panel, select the View>Digital Panel menu. Alternatively, click the toolbar icon shown in fig 1.5.

2. Simulating digital inputs

The function keys on the computer keyboard are used to simulate inputs from digital sensors while a flowsheet is running.

Function keys F9 to F2 will simulate digital sensors connected to inputs 0 to 7 on a PIC microcontroller. Key F9 simulates input 0; key F2 simulates input 7.

Pressing the function key is equivalent to the sensor being “on” (1). When the key is not pressed, it is equivalent to the sensor being “off” (0).

3. Simulating analogue inputs

The Analogue Panel allows you to simulate the changing reading from analogue sensors while a flowsheet is running.

Identify the sensor (A0 to A3) which you wish to simulate, and use the slider in the panel to vary the simulated reading from 0 to 255.

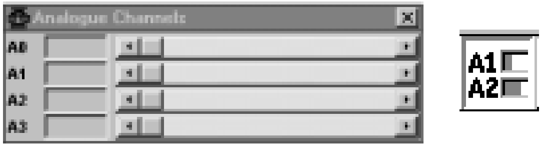


Fig 1.6. Analogue Panel and its toolbar icon

To display the Analogue Panel, select the View>Analogue Panel menu. Alternatively, click the toolbar icon shown in fig 1.6.

4. Run and Stop

To test run a flowsheet, either click the Run! menu or the green toolbar icon. To stop a flowsheet running, click the Stop! menu or the red toolbar icon.

As the flowsheet runs, the flow of control is highlighted so that you can follow it.

If you want to slow down the speed at which flow is highlighted, select the Options>Speed... menu, and use the dialog box to adjust the speed.

5. Variables and EEPROM display windows

If your flowsheet uses variables, it is useful to display the Variables window when you test run it. The changing values of any of the variables A to H that are used in the flowsheet will be displayed as the flowsheet runs. To see the changing value of the TIME variable, select the TIME window.

Fig 1.7.
The Variables window.

The EEPROM display window shows the value in each of the 16 addresses, when the flowsheet uses the READ and WRITE commands.

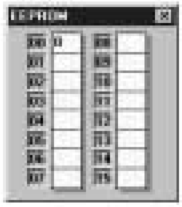


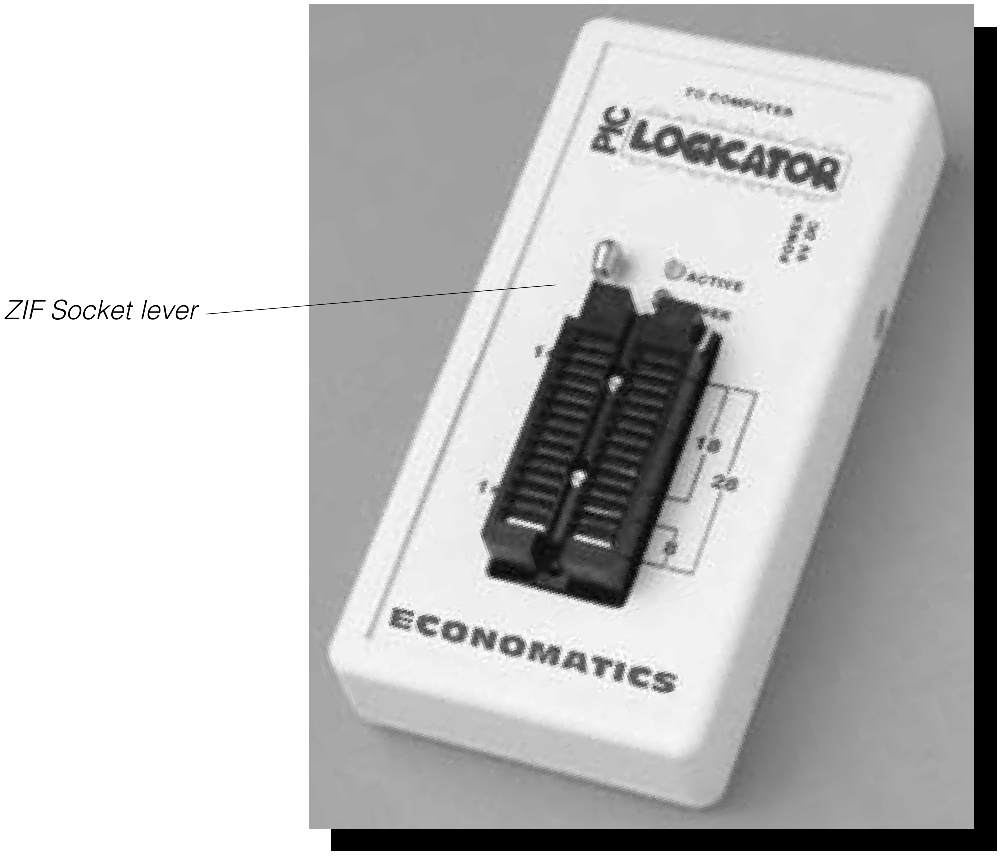
Fig 1.8.
The EEPROM window.

All of these display windows are selected from the View menu.

Downloading a flowsheet

See “Using the Programmer” (page 13) for instructions on downloading a flowsheet to a PIC microcontroller.

Using the Programmer



1. Use the serial lead to connect between the socket on the Programmer marked “TO COMPUTER” and the serial port socket of your computer.
2. Use only the power supply supplied in the PIC-Logicator Pack.

   Plug the power supply lead into the socket on the Programmer marked “POWER 9V DC”. Plug the power supply into a convenient mains socket. The green LED marked “POWER” will come on.
3. Lift the Zero Insert Force (ZIF) socket lever into its upright position.
4. The legend on the Programmer indicates correct positioning of 8,18 and 28 pin chips. Carefully insert a PIC microcontroller chip into the appropriate sockets, and return the lever to its down position.
5. PIC-Logicator software should be running, and the current flowsheet should be the one that you want to download. Click the “Program PIC” menu option or the toolbar icon.
6. The yellow LED marked “ACTIVE” will come on, and the Download bar will be displayed in the software. After about 20 seconds, the yellow LED will switch off and the Download bar will disappear. Downloading is now complete.
7. Lift the ZIF socket lever into its upright position. Carefully remove the PIC microcontroller chip which is now programmed with your control flowsheet.

To reprogram a PIC microcontroller chip, remove it from the project circuit and follow the same procedure to download the revised flowsheet. There is no need to erase the chip before reprogramming.

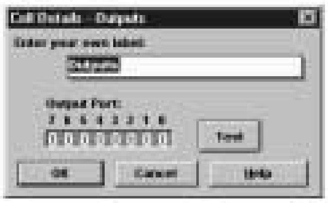
2. Outputs

Outputs command

Outputs

Use an Outputs command to switch on or off any output devices that are connected to the outputs of a PIC microcontroller.

The "Output Port" line of its Cell Details box (fig. 2.1) shows the number of outputs available for use.



Cell Details - Outputs

Enter your own label: Outputs

Output Port:

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

Fig 2.1. Outputs command Cell Details box for a 16F84 chip (eight outputs).

Each one of the digits in the Output Port represents one of the outputs on the PIC microcontroller. You can click each digit to set it to switch an output device on or off.

0 This means: switch this output off.

1 This means: switch this output on.

• This means: ignore this output. Leave it in the state in which it was set by the previous Outputs command.

Click OK.

WAIT command

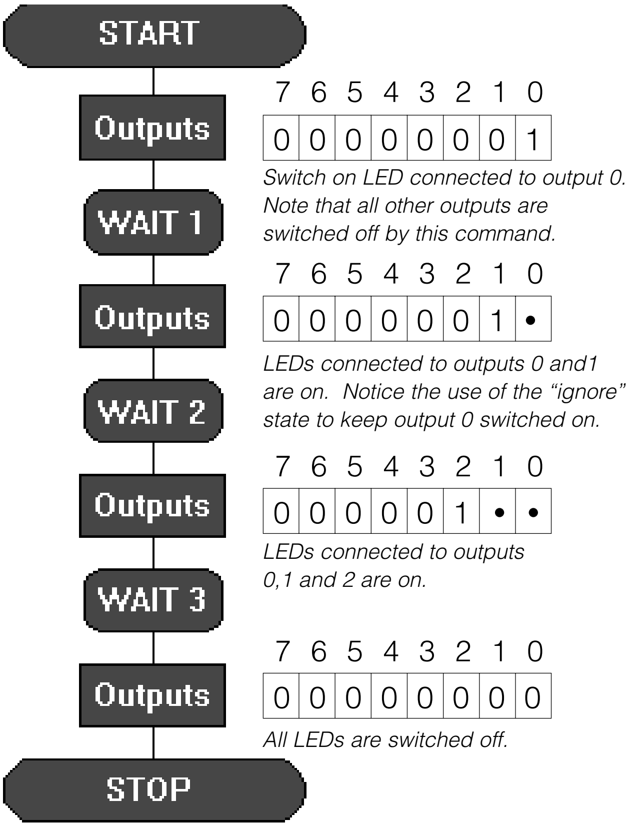
WAIT

A WAIT command makes a running flowsheet pause for the number of seconds specified before the next command is carried out. You can use it to keep output devices switched on or off for a set time.

Use its Cell Details box to enter a number of seconds (Max 10s. Min 0.1s). Click OK.

Example

A PIC microcontroller has 3 LEDs connected to outputs 0, 1 and 2. The flowsheet shown in fig. 2.2 will switch them on and off in a timed sequence. The sequence will begin as soon as the chip is powered and will stop at the STOP command, so it will do the sequence just once.



Switch on LED connected to output 0. Note that all other outputs are switched off by this command.

LEDs connected to outputs 0 and1 are on. Notice the use of the "ignore" state to keep output 0 switched on.

LEDs connected to outputs 0,1 and 2 are on.

All LEDs are switched off.

Fig 2.2.
Timed sequence, showing how the Output port is set to switch selected outputs on and off.

The flowchart shown in fig. 2.3 will continue to repeat the sequence until power to the chip is switched off. Notice that another WAIT command has been added to the repeating sequence. The PIC microcontroller operates so quickly that, without WAIT commands, the LEDs would switch on and off so quickly that you would not see it happening.

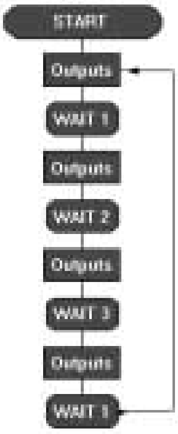


Fig 2.3.
Repeating timed sequence.

OUT command

When flow passes through an OUT command, the output port is set to the binary value of the number entered in the command (see fig 2.4). So, for example, if the command is set to 255, then all the outputs will be switched on.

output	7	6	5	4	3	2	1	0
binary value	128	64	32	16	8	4	2	1

Fig 2.4. The binary number equivalent of each bit in the output port.

If you are familiar with the binary system, this is a convenient way of switching combinations of outputs on and off

Fig 2.5 shows how OUT commands can be used to produce the same sequence as that shown in fig 2.2.

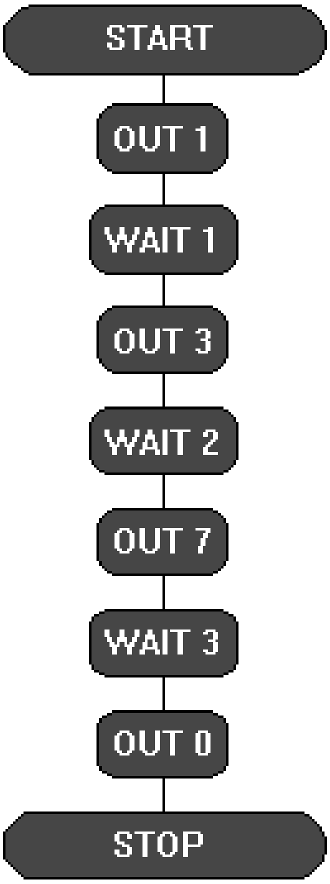
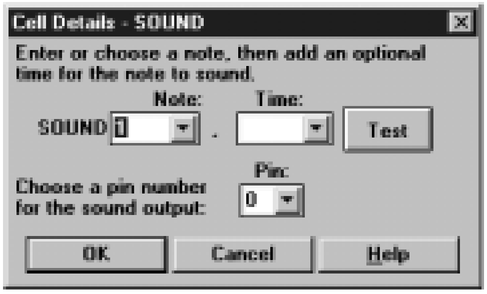


Fig 2.5. Using the OUT command.

SOUND command

Use the SOUND command to send a pulsed signal to a piezo sounder connected to an output of a PIC microcontroller. You can use a sequence of SOUND commands to play a tune.



Cell Details - SOUND

Enter or choose a note, then add an optional time for the note to sound.

Note: Time:

SOUND 0 .

Choose a pin number for the sound output: Pin: 0

Fig 2.6 SOUND command Cell Details box

Fig 2.6 shows the SOUND command Cell Details box. Use the “Note:” box to select the note you want the command to play. Fig 2.7 shows the musical note equivalent of each one of the numbers 1 to 49 which you can select in this box.

Use the “Time” box to select the length of time (in seconds) that you want the note to play for.

The piezo transducer will be connected to an output of the PIC microcontroller. Enter the number of that output in the “Pin” box. Make sure that the correct number is selected in every SOUND command on the flowsheet.

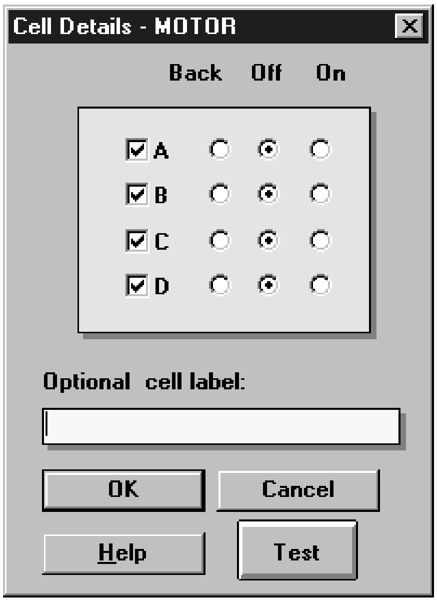
C	C#/D♭	D	D#/E♭	E	F	F#/G♭	G	G#/A♭	A	A#/B♭	B
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49											

Fig 2.7.
Musical note equivalent of SOUND numbers.

MOTOR command

The MOTOR command allows you to use just two outputs on a PIC microcontroller to switch a motor forward, reverse or off.

Use its Cell Details box to set the motor or motors to drive forward or reverse; or to stop. Remember that the direction in which a motor turns depends on which way current flows through it, and therefore on the way it is connected to power. For this reason, the terms "On" and "Back" indicate only that the directions will be different; not the actual direction in which motors in the project will turn.



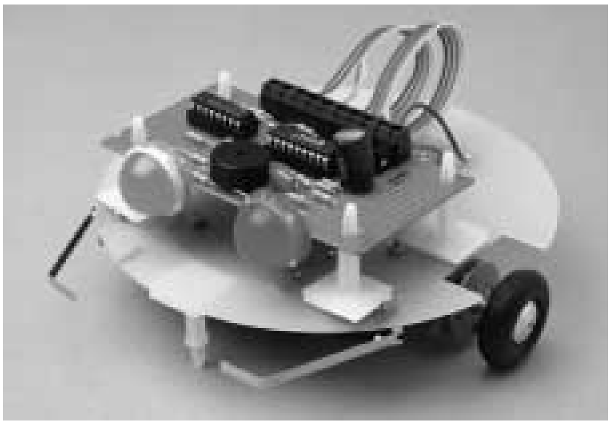
	Back	Off	On
☑ A	○	●	○
☑ B	○	●	○
☑ C	○	●	○
☑ D	○	●	○

Fig. 2.8. MOTOR command Cell Details box.

Motors are labelled A,B,C or D. Motor A is the motor controlled by outputs 0 and 1 of the PIC microcontroller. Motor B is the motor controlled by outputs 2 and 3, and so on. See "Connecting Motors" (page 53).

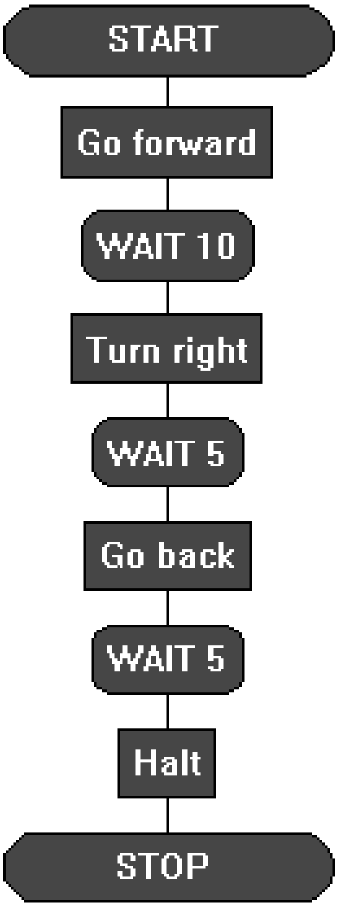
NOTE: Outputs and MOTOR commands both use the same output lines to switch the outputs of a PIC microcontroller. The default state of both commands is such that they will automatically switch off any outputs that are not set to be on.

So, to aviod inadvertently switching off an output device, click the check boxes of unused motors in a MOTOR command to disable them, and set unused outputs in a Outputs command to their "ignore" state.



Example

A steerable buggy is usually driven by two motors, one powering each driving wheel with a free-running jockey wheel to keep it stable. Fig 2.9 shows how a sequence of MOTOR commands can be used to drive a buggy which has one motor connected to outputs 0 and 1 (motor A) and the other motor connected to outputs 2 and 3 (motor B)



	Back	Off	On
A	○	○	●
B	○	○	●

	Back	Off	On
A	●	○	○
B	○	○	●

	Back	Off	On
A	●	○	○
B	●	○	○

	Back	Off	On
A	○	●	○
B	○	●	○

Fig 2.9. The MOTOR commands have been given labels to show what they do. The table beside each one shows how its Cell Details have been set.

SLEEP command

SLEEP

This command puts the PIC microcontroller into low power mode for a specified number of seconds. You can use this command to save battery power in your project.

All output devices will be left in their current condition, and signals from input devices will not be responded to while the chip is in sleep mode. The TIME variable will be reset to 0.

Use the Cell Details box to set the number of seconds of sleep mode you require. This can be any number between 1 and 255. A setting of 255 would give a sleep time of 4.25 minutes. Note that SLEEP times are not quite as accurate as WAIT times.

SEROUT command

SEROUT

This command allows you to output information from the PIC microcontroller to a device such as a serial printer or a serial LCD screen which is connected to an output of a PIC microcontroller. The information is sent serially at baud rate 2400.

The SEROUT Cell Details box contains two boxes. Use the first box to enter a number. This number will be the ASCII code equivalent of the single number, letter or character that you want to output. Remember to use the second box labelled “Pin” to enter the number of the output to which the device is connected.

Fig 2.10 shows a table of ASCII code equivalents of the most commonly used characters. Some numbers may be used as code for specific purposes such as “clear screen”, by the device you are using. Consult the device’s data sheet for this kind of information.

32		51	3	70	F	89	Y	108	l
33	!	52	4	71	G	90	Z	109	m
34	‘’	53	5	72	H	91	[	110	n
35	#	54	6	73	I	92	\	111	o
36	$	55	7	74	J	93	]	112	p
37	%	56	8	75	K	94	^	113	q
38	&	57	9	76	L	95	-	114	r
39	’	58	:	77	M	96	`	115	s
40	(	59	;	78	N	97	a	116	t
41	)	60	<	79	0	98	b	117	u
42	*	61	=	80	P	99	c	118	v
43	+	62	>	81	Q	100	d	119	w
44	,	63	?	82	R	101	e	120	x
45	_	64	@	83	S	102	f	121	y
46	.	65	A	84	T	103	g	122	z
47	/	66	B	85	U	104	h	123	{
48	0	67	C	86	V	105	i	124	:
49	1	68	D	87	W	106	j	125	}
50	2	69	E	88	X	107	k		

Fig 2.10. ASCII code equivalents of the most commonly used characters.

Example

The flowsheet shown in fig 2.11 will display the word HELP on an LCD screen connected to an output pin of a PIC microcontroller.

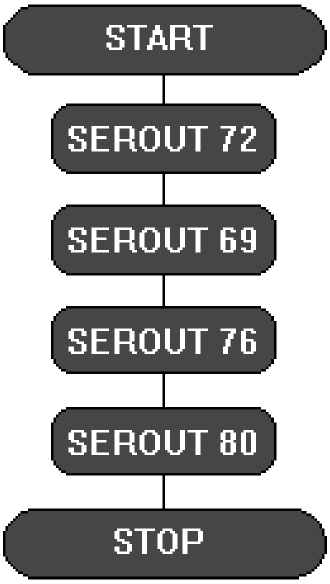


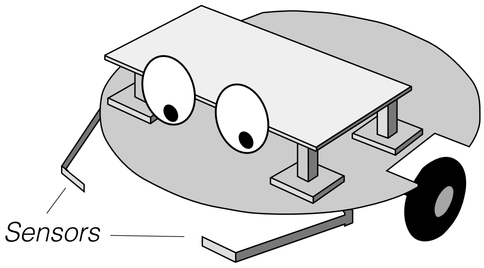
Fig 2.11. A sequence to display the word HELP.

3. Inputs

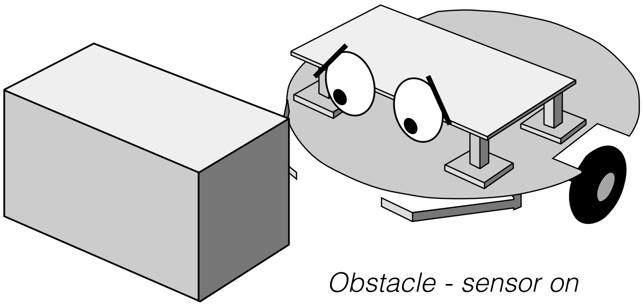
Input devices such as switches and sensors send information from the outside world into the control system. Output devices are switched on or off in response to the information provided by input devices.

Example one

A buggy is often fitted with microswitches so that if it approaches an obstacle, a microswitch will be pressed.



No obstacles - sensor off



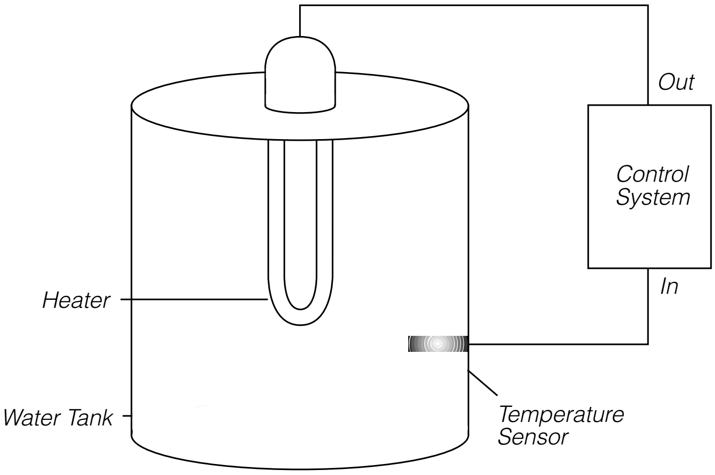
Obstacle - sensor on

The information that the switch has been pressed can be used in the system to switch off the motors driving the buggy, and start a sequence of movements to move around the obstacle.

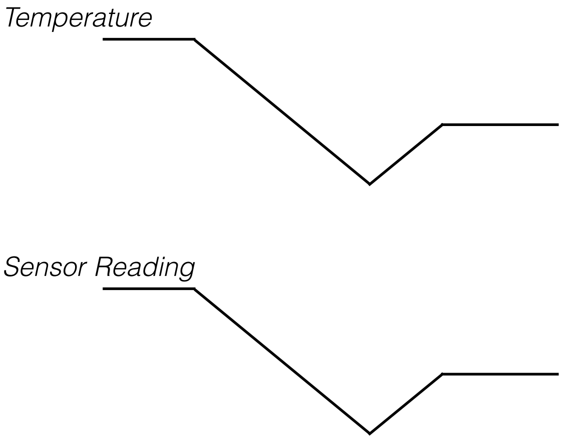
A microswitch is a digital sensor. It has only two states - “on” (or “closed”) and “off” (or “open”). These states are often labelled by the digits 1 and 0, which is why the sensors are called digital sensors.

Example two

A controlled hot water system includes a temperature sensor which constantly monitors the water temperature. The water heater is switched on and off in response to the information provided by the sensor. If the water temperature falls below a set level, the heater is switched on until it reaches that level again. Then the heater is switched off.



A temperature sensor is an analogue sensor. It provides a reading which changes in line with the changing level of whatever it is sensing.



Digital Inputs

Digital Decision command

Use this command to test the state of a digital sensor connected to a digital input of a PIC microcontroller.

When flow reaches a Decision cell, it continues in either the “Y” (yes) or “N” (no) direction depending on the result of the decision test. See fig 3.1.

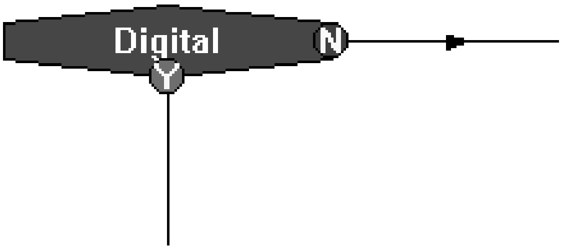
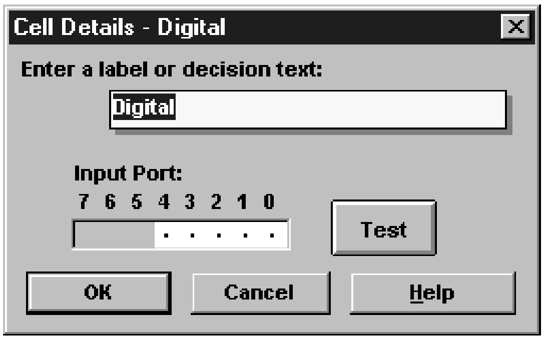


Fig 3.1. This Digital Decision command is testing the state of a microswitch. If the switch is pressed, flow will go in the Y route; if it is not pressed, flow will go in the N route.

The Cell Details box of the Digital Decision command is shown in fig 3.2. The Input Port line shows the number of digital inputs available for use on the PIC microcontroller you have selected.



Cell Details - Digital

Enter a label or decision text: Digital

Input Port:

7 6 5 4 3 2 1 0

· · · · ·

Fig 3.2. Digital Decision Cell Details box for a 16F84 chip (five digital inputs).

Each one of the digits in the Input Port represents one of the digital inputs on the PIC microcontroller. You can click each digit to set it to one of three states:

1 This means: is this sensor “on” (1)?

0 This means: is this sensor “off” (0)?

• This means: ignore this sensor.

When you have done this, click OK.

Drawing routes from a Decision command

The first line that you draw from a Decision command is the “Y” direction, and the second line is the “N” direction.

You can swap the “Y” and “N” routes by double-clicking on the Decision command with the right mouse button.

Example one

A PIC microcontroller is being used to control a security system. A buzzer is connected to one of the outputs. A pressure pad is connected to input 0, and a push switch is connected to input 1.

Fig 3.3 is a flowsheet for the control system, showing how the two Decision commands are set. When the chip is powered, the pressure pad is tested. If it is not pressed, flow will go in the N route and will continue to go round this loop until the pad is pressed. When the pad is pressed, flow will go in the Y route and the buzzer will be switched on. The buzzer will stay on until the push switch is pressed. When it is pressed, the buzzer will switch off and flow will return to testing the pressure pad.

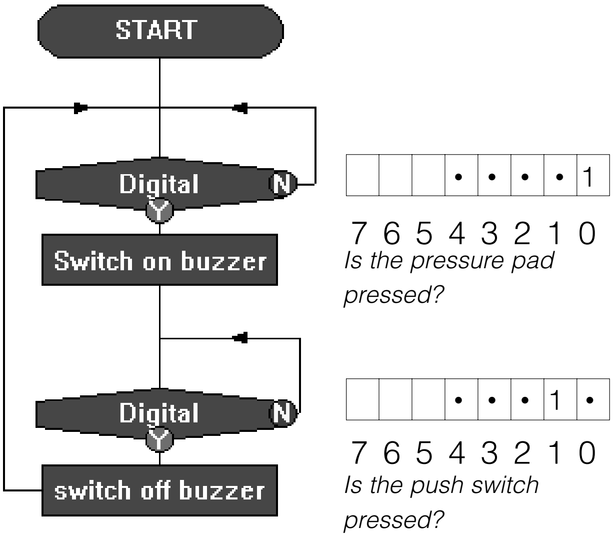
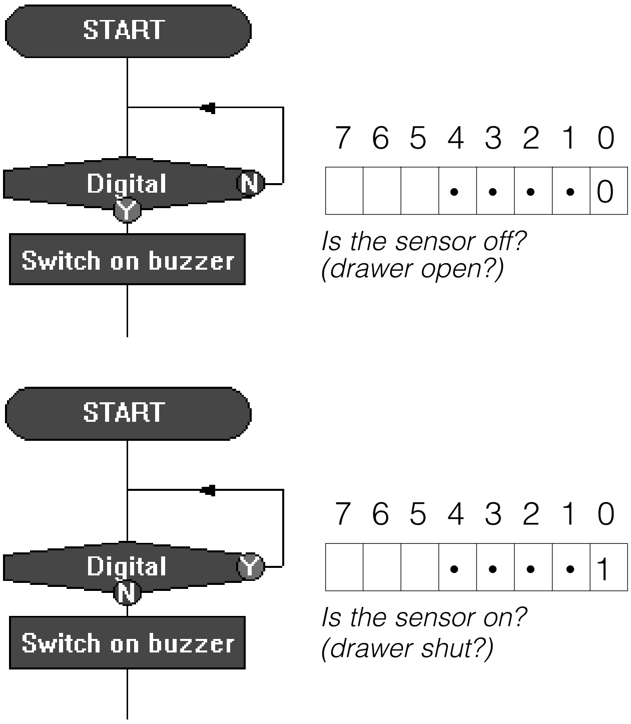


Fig 3.3. Security system.

A similar flowsheet could be used to control a security system for a drawer. In this case, the sensor could be a micro-switch which is kept closed (on) as long as the drawer is shut. If someone opens the drawer, the microswitch will be open (off).

Fig 3.4 shows two different ways of using a Decision command to test the micro-switch in this system.



7	6	5	4	3	2	1	0
			•	•	•	•	0

Is the sensor off? (drawer open?)

7	6	5	4	3	2	1	0
			•	•	•	•	1

Is the sensor on? (drawer shut?)

Fig 3.4.
Notice that the direction of flow depends on how the command is set.

Example two

Home security systems often have a number of sensors in different parts of the house. If any one of them is activated, the alarm is switched on. Fig. 3.5 shows a security system which has three sensors and a reset switch.

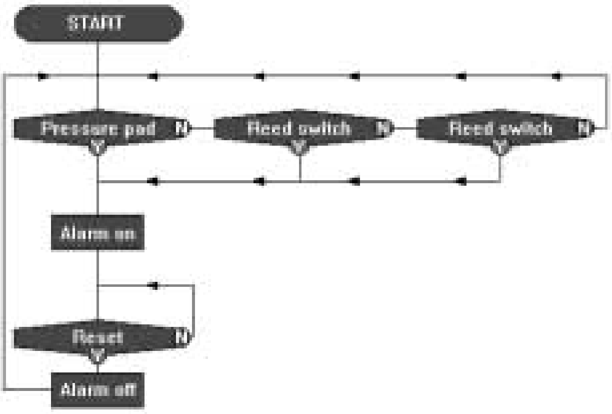
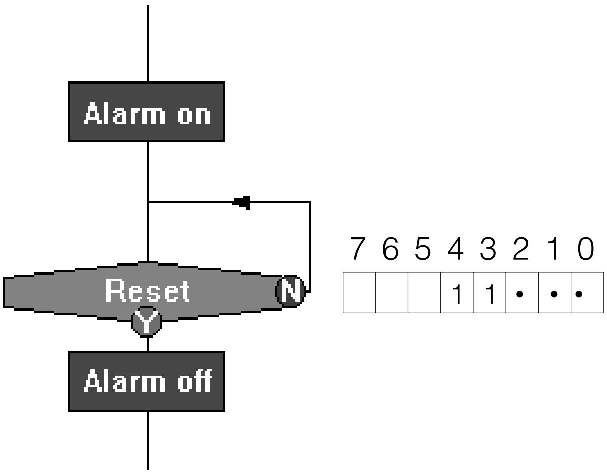


Fig 3.5. Security system with three sensors (OR function).

Two of the sensors are the magnetic type for windows which have the magnet fixed to the window frame and the reed switch fixed to the window. As long as the window is shut, the magnet keeps the reed switch contacts closed (“sensor on”). When the window is opened and the magnet is moved away from the switch, the contacts are open (“sensor off”). Therefore, the two Decision commands have been set to go in the Y route if the sensor is off (0).

The system shown in fig 3.5 is an OR function. Some security systems have two separate reset switches arranged in an AND function so that the system is reset only if both switches are pressed together. Fig 3.6 shows how you can set a Decision command to test two switches in this way.



7	6	5	4	3	2	1	0
			1	1	•	•	•

Fig 3.6.
Decision command set to check if two switches are pressed at the same time (AND function).

Example three

In the flowsheet shown in fig 3.7, the output is switched on when a push switch is pressed. When you stop pressing the switch the output switches off. In other words:
IF the input is on, THEN switch the output on, ELSE switch the output off.

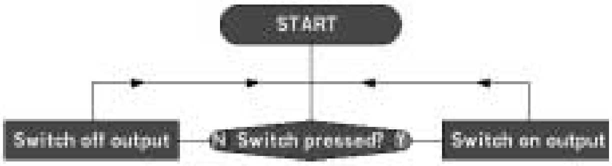


Fig 3.7 "Normally open" switch effect.

This is the equivalent of a simple electrical circuit containing a normally open push switch and an output device. The big difference is that you can change the way the system works in software, by simply changing over the Y and N on the Decision command as shown in fig 3.8.

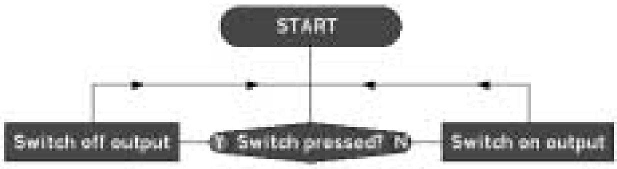


Fig 3.8 "Normally closed" switch effect.

Example four

A monostable device has only one stable state. It changes state when it is triggered by an input, and stays in that state for a certain time. It then goes back to its original state. Fig 3.9 shows how this function can be produced in PIC-Logicator.

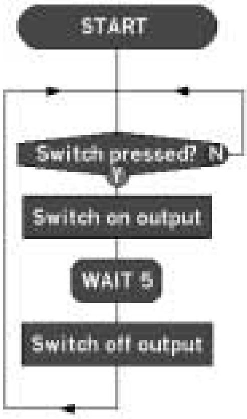


Fig 3.9 "Monostable" function.

Example five

A bistable device has two stable states. It changes state when it is triggered (set) by an input, and stays in that state until it is triggered (reset) by a second input. It then goes back to its original state. Fig 3.10 shows how this function can be produced in PIC-Logicator.

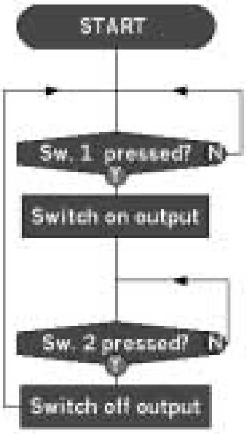


Fig 3.10 "Bistable" function using two switches for set and reset.

Fig 3.11 shows how you can use just one switch for both set and reset. In this case the Decision commands are used in pairs. The first one checks to see if the switch is pressed, and the second one checks for it to be unpressed before the output is switched. The program is processed so fast that, if you didn't include this feature, it would switch the output and start checking the switch again while you were still pressing it for the first time.

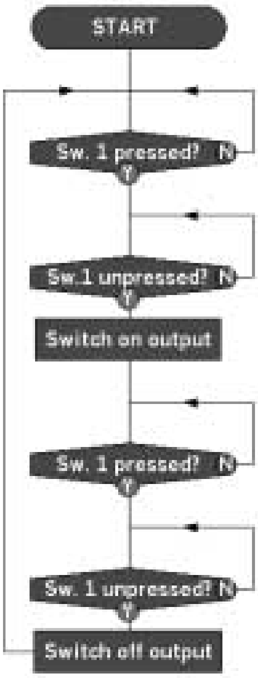


Fig 3.11 "Bistable" function using one switch for both set and reset

Events and Interrupts

Event and Interrupt commands instantly capture the flow of control whenever a pre-set digital input condition occurs to trigger it, e.g. when a switch is pressed.

Events

When an Event is triggered, flow jumps immediately to the Event command. The commands following the Event command are carried out and flow is routed back into the main routine at some chosen point.

The following activity is designed to show the difference between using a Digital Decision command and an Event command. Make sure that you are familiar with test running a flowsheet (see page 12) before you begin.

1. Build the flowsheet shown in fig 3.12. Set the first Outputs command to switch on a selection of outputs. Use the second one to switch them off.

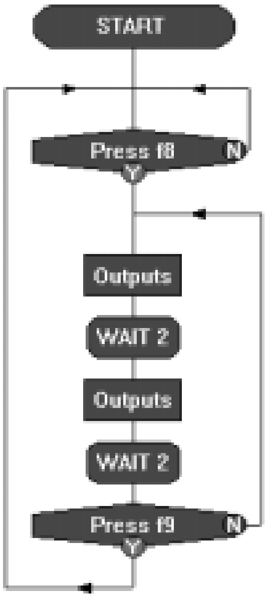


Fig. 3.12
This system is designed to start flashing the outputs on and off as soon as you press function key F8 (input 1), and go on flashing them until you press function key F9 (input 0) when it will stop flashing and wait for key F8 to be pressed again.

2. Run the flowsheet. When it is running, press function key F8 to start the outputs flashing. While they are flashing, press function key F9 and release it immediately. You will find that the flowsheet responds only if the key is pressed when flow is actually on the Decision command. If it is anywhere else, it will not respond.

3. Now edit your flowsheet to look like fig 3.13. Use the information below to help you.

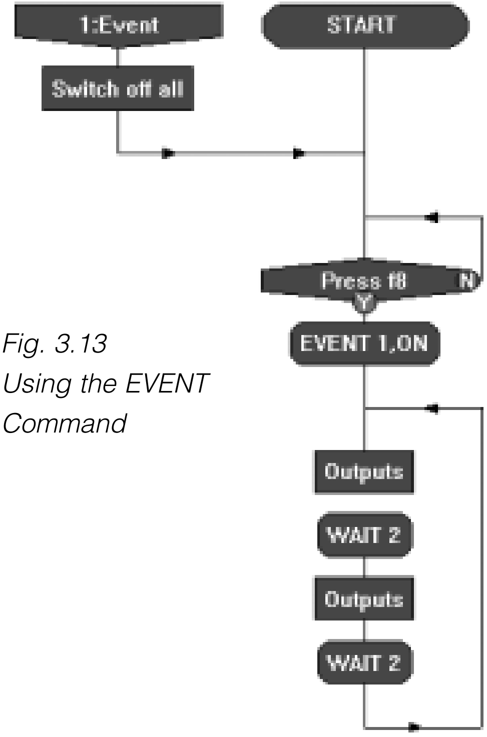


Fig. 3.13
Using the EVENT Command

1:Event

Drag an Event command onto the flowsheet. Double click on it to open its Cell Details box. It has the same Input Port line as a Digital Decision command. Set it to respond to function key F9 (input 0). Click OK. The command will automatically label itself 1: Event.

An Event will be triggered only if it has been enabled by an EVENT command.

EVENT

Drag an EVENT command onto the flowsheet. Double click on it to open its Cell Details box.

Select number 1 from the drop-down list called "Number". Select ON from the drop-down list called "State". Click OK.

When an Event is triggered, it must be re-enabled before it can be triggered again. So, flow is routed back through the EVENT 1, ON command to re-enable the Event.

Test run the flowsheet. This time it should respond immediately, whenever you press function key F9.

Interrupts

Interrupts work in a similar way to Events. The difference between them is that when the Interrupt is triggered, flow jumps immediately to the Interrupt command and then carries out any commands which follow until it reaches an END command. It then returns to the point which it was at when the Interrupt occurred.

The following activity illustrates the use of Interrupts. Make sure that you are familiar with test running a flowsheet (see page 12) before you begin.

1. Build the flowsheet shown in fig 3.14.

Use the following information to help you.

The Cell Details box of an Interrupt command has the same Input Port line as a Digital Decision or an Event command. Set each one of the three Interrupt commands on the flowsheet to respond to a different function key. Note that Interrupts automatically number themselves from 5 to 8.

Set the Outputs commands in the main routine so that all the outputs flash on and off every two seconds.

Set the Outputs command in Interrupt 5 to switch off all outputs.

Set the Outputs commands in Interrupt 6 to flash output 7.

Set the Outputs commands in Interrupt 7 to flash ouput 0.

Unlike Events, Interrupts have to be enabled only once. This is done with an INTERRUPT command.   Drag an INTERRUPT command onto the flowsheet. Double click on it to open its Cell Details box.

Select ALL from the drop-down list called “Number”. Select ON from the drop-down list called “State”. Click OK.

2. Test run the flowsheet. Press each one of the three function keys separately as the flowsheet runs, and watch the result in the Output line in the Digital Panel. Also follow the flow of control on the screen as the flowsheet runs to see how an Interrupt captures the flow. If you want to slow down the speed at which flow is highlighted, select the Options>Speed... menu.

You can use up to a maximum of four Events and four Interrupts in one PIC-Logicator flowsheet. Events are numbered 1 to 4; Interrupts are numbered 5 to 8.

Further examples of using Events and Interrupts can be found on pages: 27, 33, 36.

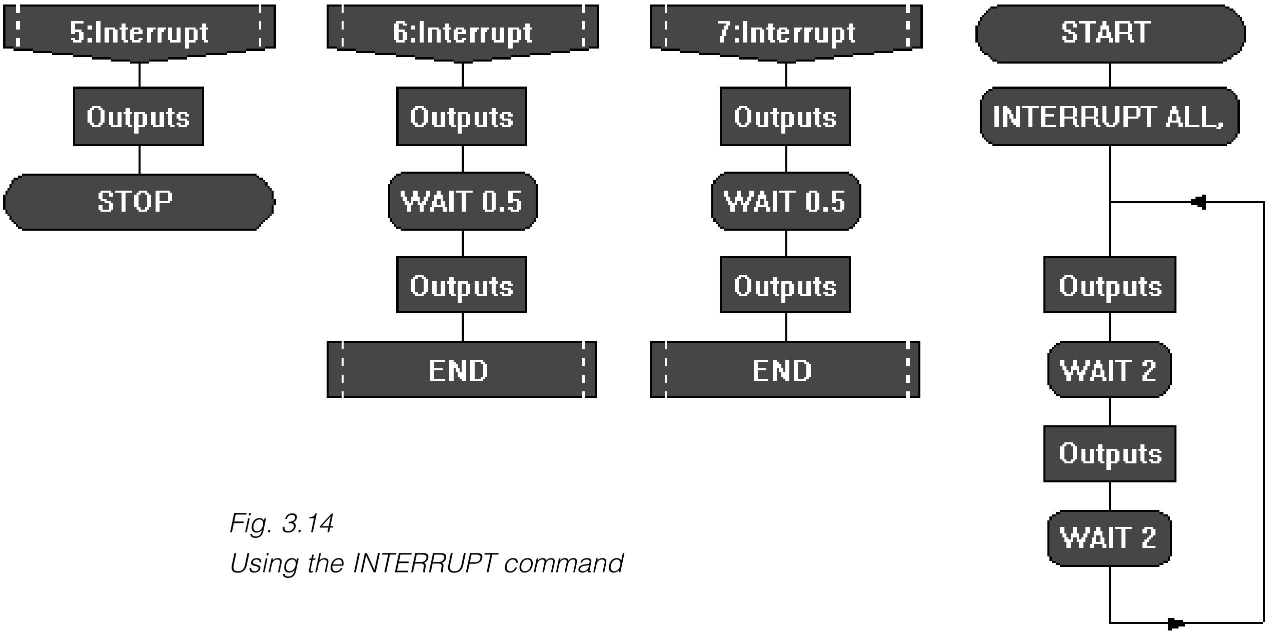


Fig. 3.14
Using the INTERRUPT command

Analogue Inputs

If you want to use analogue sensors in a project, you must use a PIC microcontroller that has analogue inputs (see page 46).

Calibrating sensors

All analogue sensors connected to a PIC microcontroller provide information on the conditions they are sensing, simply as a number between 0 and 255. So, when you have made a sensor as shown on page 56, you will need to calibrate it.

For example, if you are using a temperature sensor, you will need to know the equivalent in degrees centigrade of the numbers (0 to 255) that it provides. If you are using a light sensor in a system in which output devices are switched on and off at different light levels, you will need to know the number that the sensor provides when it is actually placed in each one of those light levels.

The PIC-Logicator Analogue Calibration Board allows you to calibrate your sensors in this way. Connect the sensor to the board as shown in fig 3.15, and make a note of the number between 0 and 255 that is displayed when the sensor is placed in different conditions.

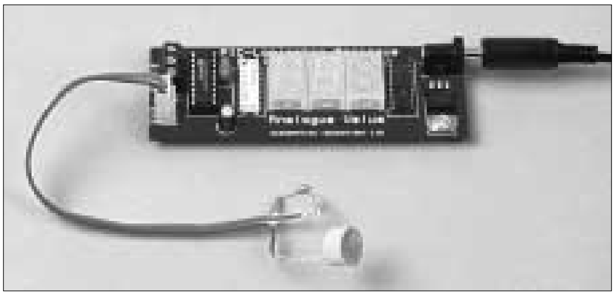


Fig 3.15 PIC-Logicator Analogue Calibration Board with light sensor connected.

Compare Decision command

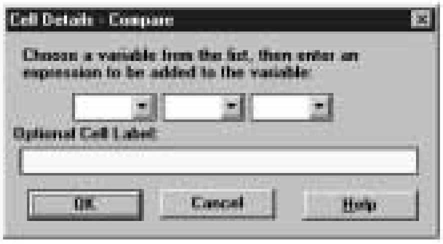
Compare

Use this command to check the reading from an analogue sensor connected to an analogue input of a PIC microcontroller.

The most common use of an analogue sensor in a control system is to switch output devices on or off when the reading from the sensor reaches a particular level. This level is sometimes called the "threshold".

When flow reaches a Compare Decision cell, the software checks the current reading from the specified sensor, and compares it with the threshold that you have set. Flow will continue in either the "Y" (yes) or "N" (no) direction depending on the result of the comparison.

The Cell Details box of the Compare Decision command is shown in fig 3.16.



Cell Details - Compare

Choose a variable from the list, then enter an expression to be added to the variable

Optional Cell Label:

Fig 3.16. Cell Details box of the Compare Decision command.

1. Use box one to select the sensor that you want the command to check. Analogue sensors are labelled A0 to A3 according to which pin on the chip they are connected to. Type in the number of the sensor you want the command to check, or select it from the drop-down box.

2. Use boxes two and three to complete the comparison. The drop-down list in box two contains a list of operators such as "greater than" (>), "less than" (<), and "equals" (=). Select the one that you require.

NOTE: It is usually better to use an operator such as "greater than or equals" (>=) instead of "equals", because analogue sensor readings can fluctuate rapidly, and you may find that the checking of the sensor reading never actually coincides with the exact threshold level.

Use box three to set the threshold level. Type in a number between 0 and 255, or select it from the drop-down list.

Example one

A PIC microcontroller is being used to control a lamp. A light sensor is connected to analogue input 0. The system will switch on the lamp automatically in dark conditions. Fig 3.17 shows a flowsheet for the system.

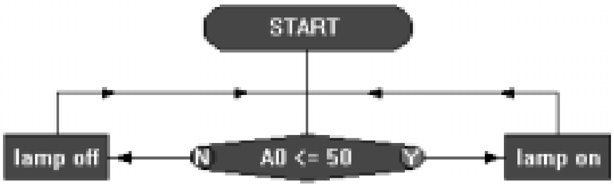


Fig 3.17 System to switch on a lamp automatically in dark conditions.

The Compare Decision command is checking the reading from the light sensor. If the reading is less than or equal to 50, flow will go in the Y route and switch on the lamp; if the reading is greater than 50, flow will go in the N route and switch off the lamp.

The system could be extended as shown in fig 3.18. This system controls three separate lamps. It automatically switches them on one by one as darkness falls, and switches them off in the same way as conditions grow lighter.

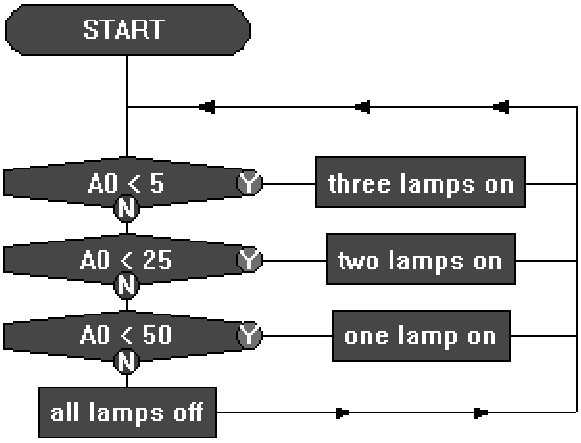


Fig 3.18. System to switch on three lamps in response to changing light levels.

Example two

A PIC microcontroller is used to make a light meter for use by cricket or tennis umpires to decide when to abandon play because of bad light. A light sensor is connected to analogue input 0. An LED is connected to each one of the eight outputs. In bright sunlight, all the LEDs will be lit. As the light level falls, the LEDs will switch off one by one. FIg 3.19 shows the flowsheet for the system. Notice the use of the OUT command to switch on combinations of outputs.

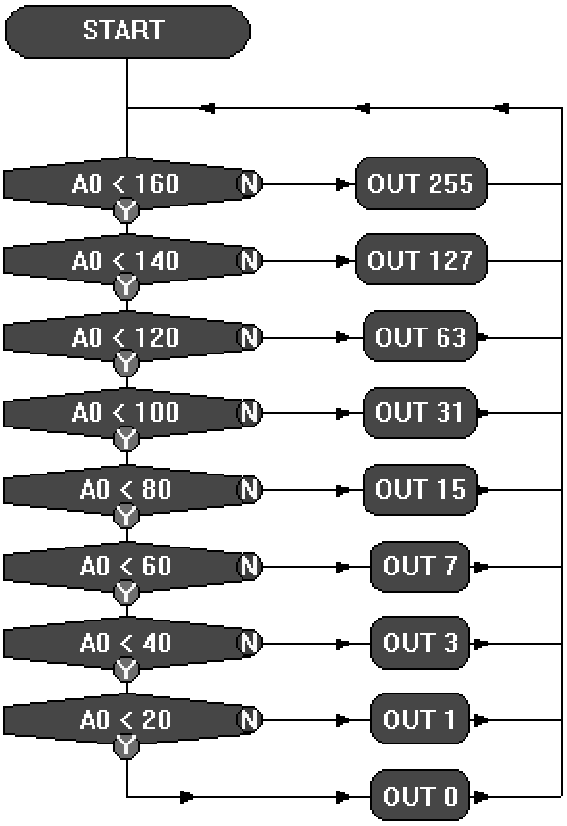


Fig 3.19. Light meter system.

4. Macros

PIC-Logicator software provides a clear, step-by-step method of building a complex control system, by creating a number of linked sub-systems called “macros”.

How to build a macro

Macro

1. Use a Macro command to begin the macro. Drag the command onto the flowsheet and place it separately from the START command as shown in fig 4.1. Double click on the command to open its Cell Details box. Type in any appropriate name, and click OK. The software automatically puts the name into capitals.

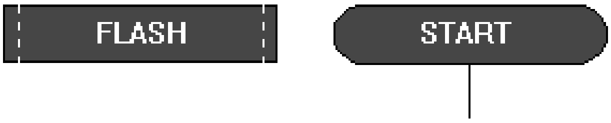


Fig 4.1. Placing the Macro command.

2. Use other commands as normal to create the macro. Place an END command at the end of the macro as shown in fig 4.2.

END

This command does not have a Cell Details box; simply place it on the flowsheet.

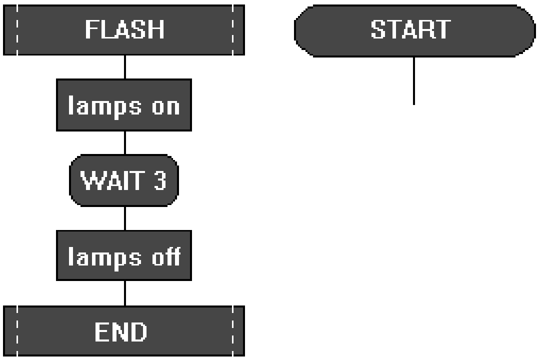


Fig 4.2. This macro, called FLASH will switch on selected lamps for 3 seconds and then switch them off.

3. When you have created a macro, you can test run it. Click on the Macro command to select it, and click Run!

How to use a macro

Once you have built a macro, you can call it into use whenever you like in the flowsheet by using the Do Macro command, as shown in fig 4.3.

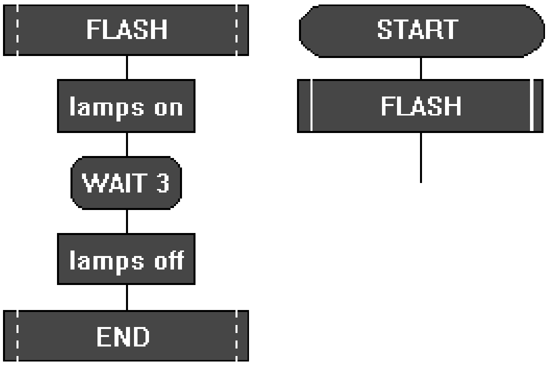


Fig 4.3. The Do Macro command calls the macro into use.

Do Macro

Drag a Do Macro command onto the flowsheet. Place it at the point where you want the macro to be called into use. Double click on the command to open its Cell Details box. Type in the name of the macro or select it from the drop-down list. Click OK.

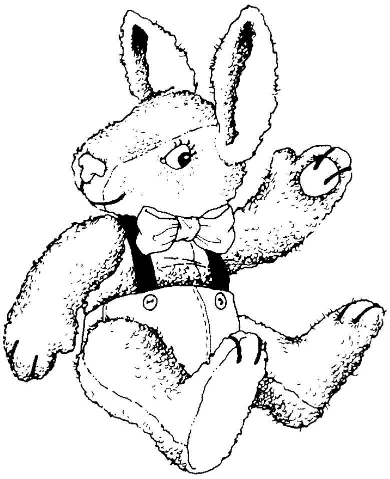
Note that all the macros that have been built in a flowsheet are automatically listed in the drop-down box.

When flow reaches a Do Macro command, it jumps to the Macro command with the same name. When the flow of control reaches an END command, the flow jumps back to the Do Macro command that called the macro.

Notice that after a macro has been test run, the Macro command is highlighted so that it can be run again if necessary. To test run the whole flowsheet, click on the START command to highlight it, and click Run!

Example one

A PIC microcontroller is used to to control a system in a child's toy which plays a tune when it is hugged. A piezo transducer is connected to an output pin, and a push switch is used to sense when the toy is hugged. The flowsheet for the system is shown in fig 4.4. The tune is created as a macro which can be tested and edited separately from the main routine.



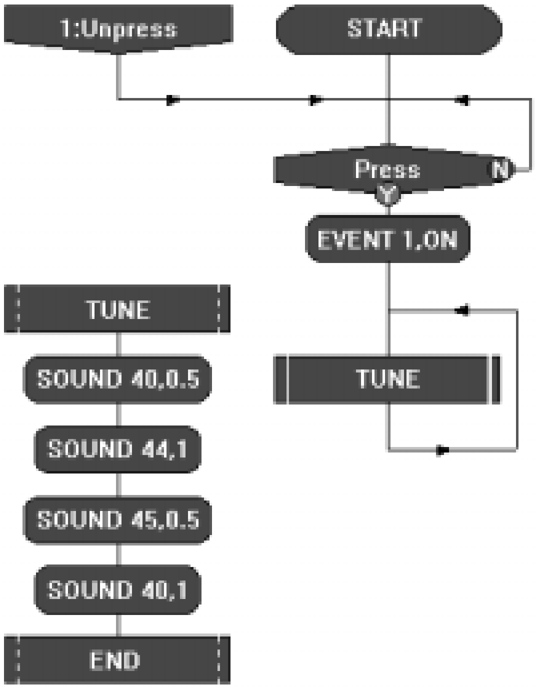
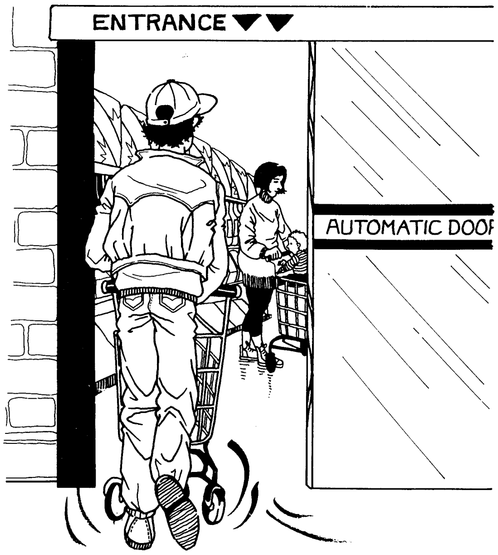


Fig 4.4. This is also a good example of the use of Event and SOUND commands.

Example two



The flowsheet shown in fig 4.5 is a control system for a sliding door. When a switch is pressed, the door opens. It stays open for ten seconds and then closes again. The system uses limit switches to sense when the door is fully open and fully closed. The motor is halted in response to the feedback from these microswitches.

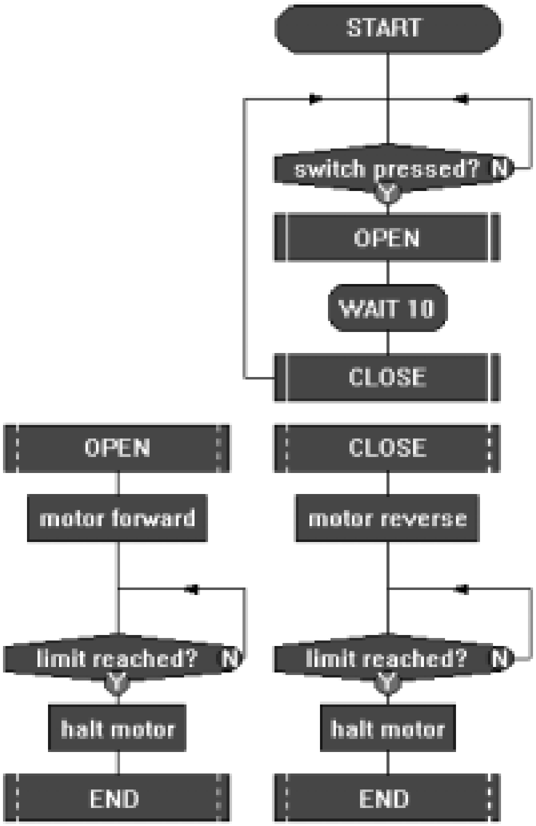


Fig 4.5. Sliding door control system using macros.

Example three

A keypad is a useful input device. This example shows how the PIC-Logicator software can be used to scan a keypad in a project in which a three digit number has to be entered to open a solenoid-operated lock.

Connect the keypad to a PIC microcontroller using inputs and outputs as shown in fig. 4.6. The flowsheet in fig. 4.7 shows how the scanning is done.

In this case, the code number uses a digit from each one of the first three rows (e.g. 357 or 268). Each row is scanned in turn using a macro.

To begin with, the row is made "live" by switching on the output to which it is connected. Then a Digital Decision command checks for the appropriate key in that row to be pressed, by testing for that input to be on. When the correct key is pressed, flow passes on to the next macro. When all three digits have been entered correctly, the solenoid is switched to unlock the door.

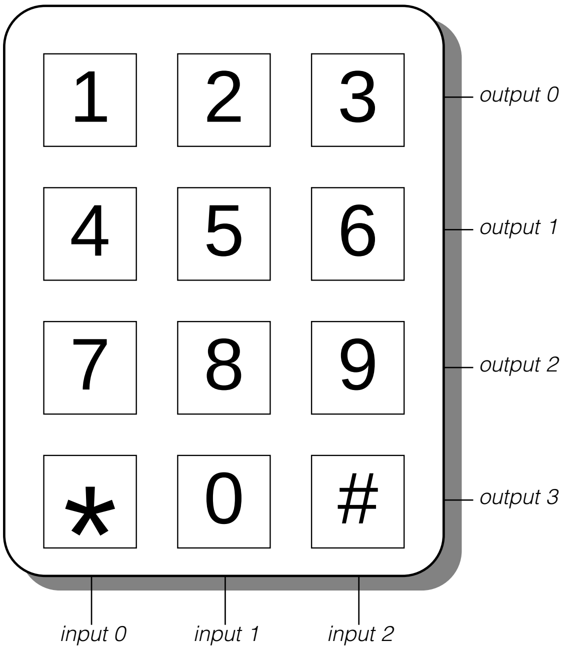


Fig 4.6. Keypad connections.

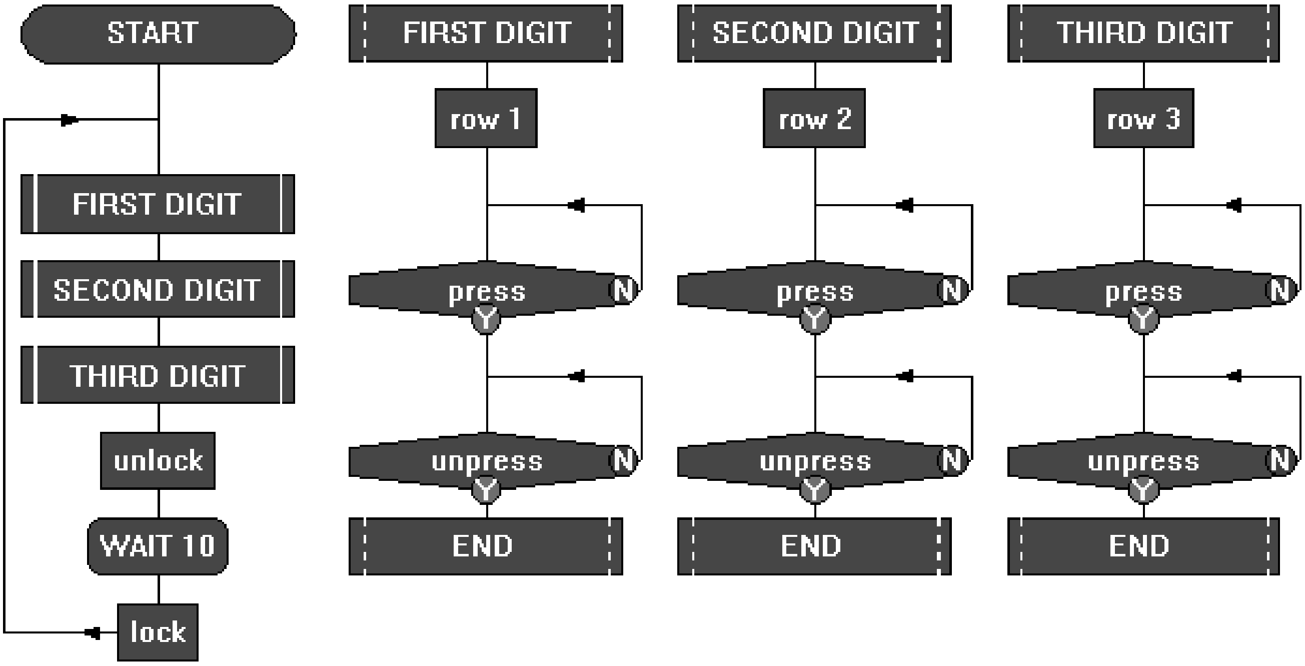


Fig 4.7. Flowsheet to scan the keypad.

Designing systems with macros

Using macros, you can design and test systems either “top-down” or “bottom-up”

Example one

The “top-down” approach.

This approach begins with an overall view of the system (the main routine), and then creates each part of it separately as a macro. The following sequence shows how it can be used to develop a control system for a buggy which is fitted with microswitches that are pressed if the buggy comes into contact with an obstacle. When this happens, the buggy sounds an alarm and moves round the obstacle.

1. The main routine is created as a series of Do Macro commands as shown in fig 4.8.

2. Then each part of the system is built as a separate macro as shown in fig 4.9. Each macro can be test run independently.

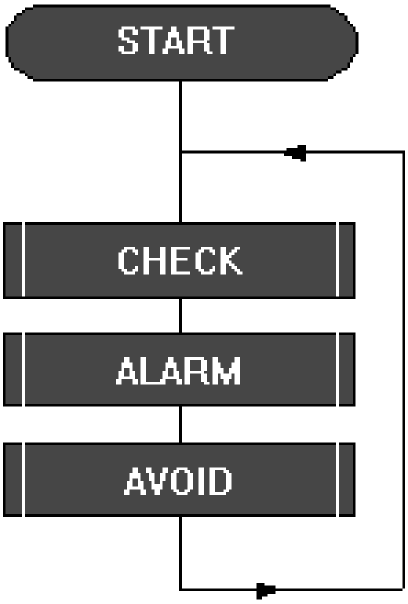


Fig 4.8. Main routine.

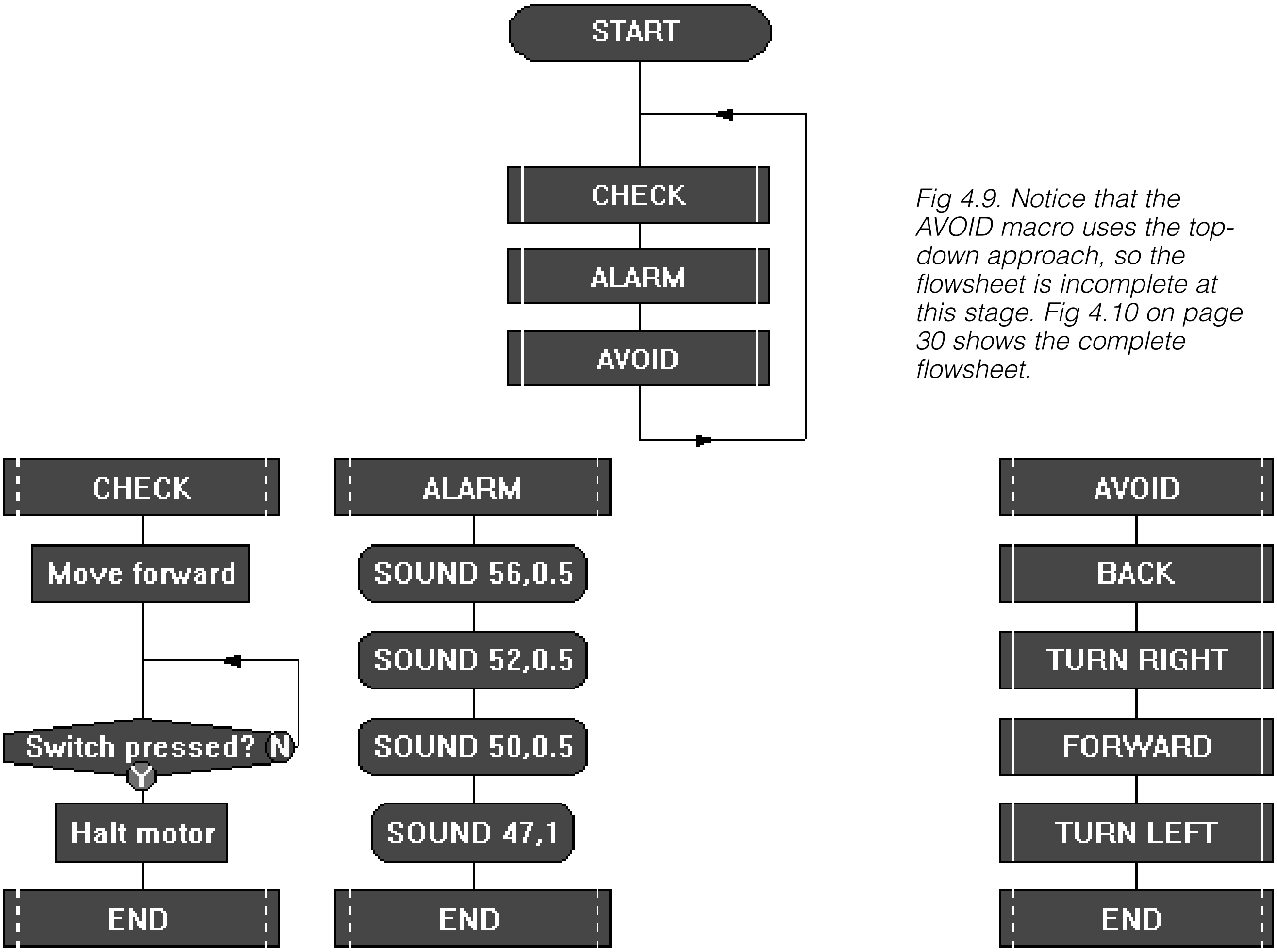
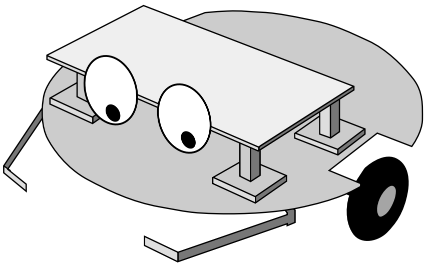


Fig 4.9. Notice that the AVOID macro uses the top-down approach, so the flowsheet is incomplete at this stage. Fig 4.10 on page 30 shows the complete flowsheet.

3. The AVOID macro shown in fig 4.9 has been built by using the top-down approach. To clarify the avoiding procedure, each movement is simply listed as a Do Macro command. Then the details required for the buggy to make each movement can be dealt with separately as shown in fig 4.10.



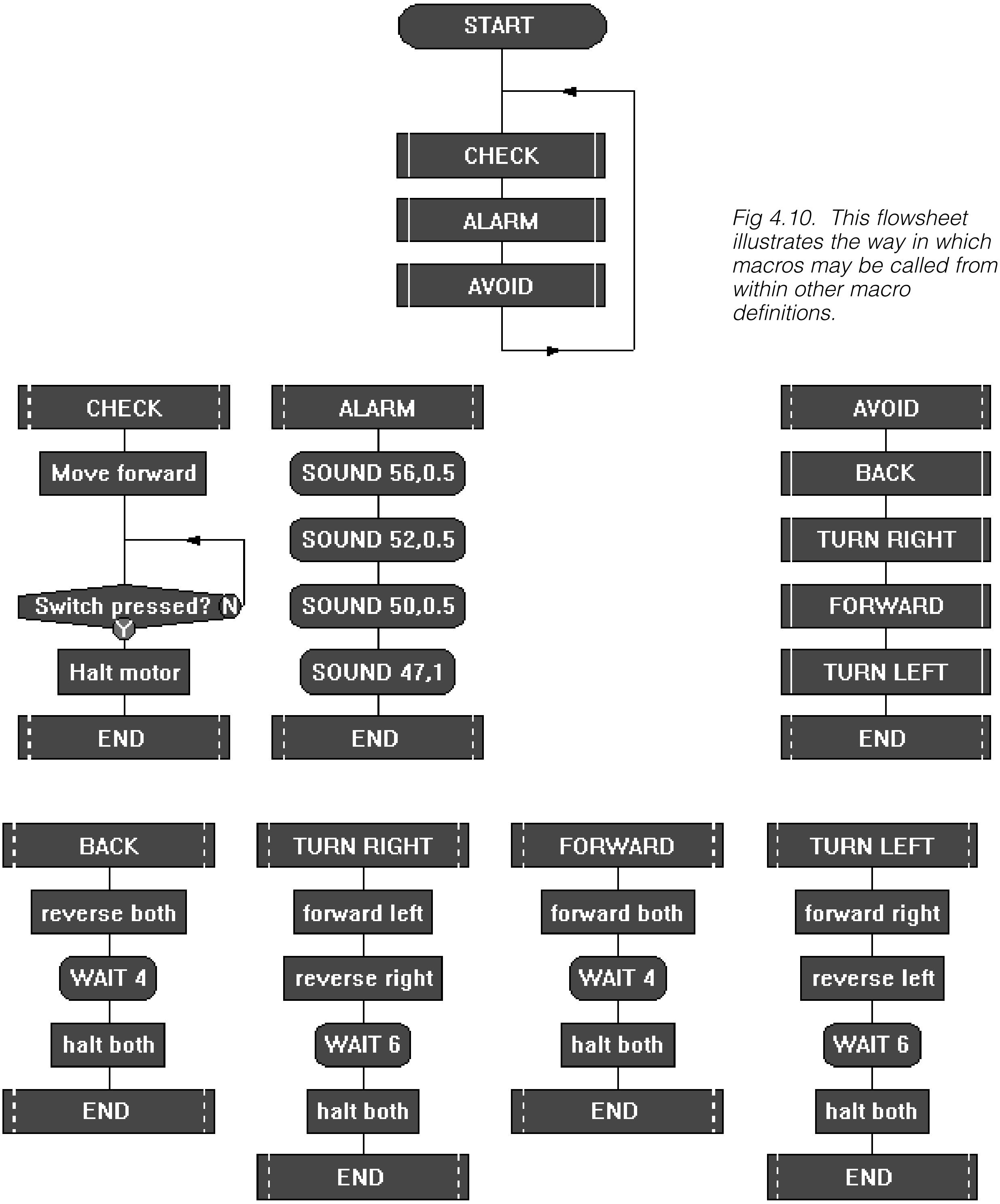


Fig 4.10. This flowsheet illustrates the way in which macros may be called from within other macro definitions.

Example two

The "bottom-up" approach.

This approach develops each part of the system separately as a macro, and then writes the main routine to link them. The following sequence shows how it can be used to develop a control system for an animated clown's head on which the eyes and nose light up and the hat rotates.

1. A separate macro is built and tested for each one of the three elements, as shown in fig 4.11.

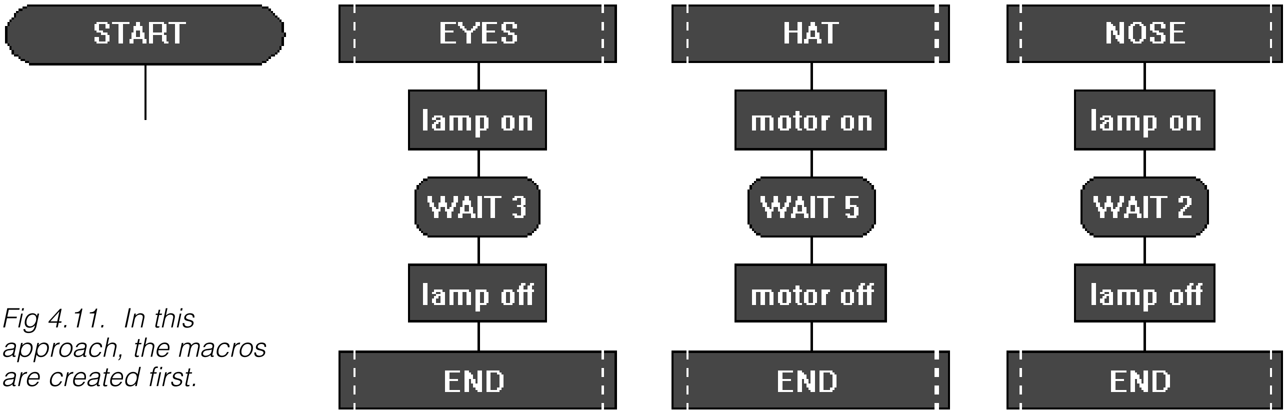


Fig 4.11. In this approach, the macros are created first.

2. A main routine is then written to call the macros into use in the required sequence whenever a switch is pressed (Fig 4.12).

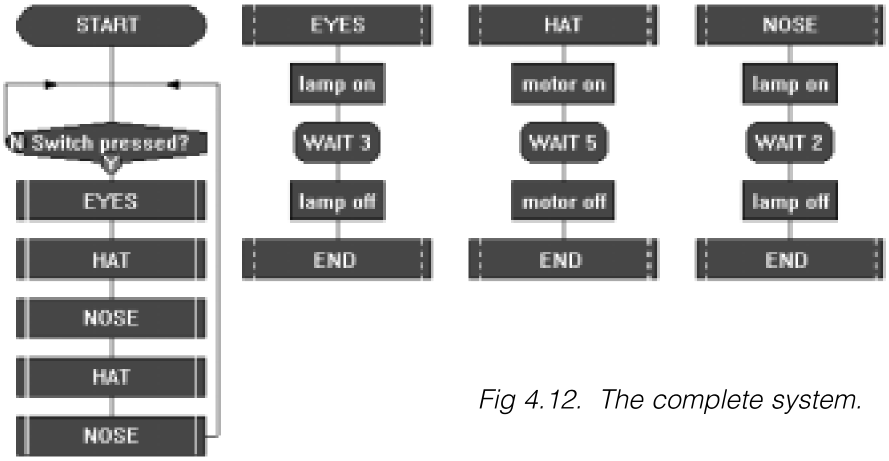


Fig 4.12. The complete system.

This flowsheet shows some of the advantages of using this approach. Once a macro has been created, it can be called into use as many times as you like within the flowsheet. Editing the sequence is easy.

The Do Macro commands can be moved around, deleted or copied to change the sequence as required. Macros can be cut, copied and pasted between flowsheets. Remember that copied commands will retain their existing cell details.

5. Variables

In PIC-Logicator a variable is a single letter or a keyword that can be given a value. The variables that can be used are: any one of the single letters A to H, and the keywords TIME and PORT. This section explains how they can be used for a variety of mainly counting and timing purposes.

Counting

The INC command INC

Each time flow passes through an INC command, one is added to the value of a selected variable (INC is short for increment). When you open the Cell Details box, simply select which variable you want to use, and click OK.

The flowsheet shown in fig 5.1 shows how it can be used to repeat a sequence three times.

Each time that flow goes round the loop, the FLASH macro is done, and one is added to the value of variable A.

A Compare Decision is used to check the value of A. When this value reaches 3, flow will go in the Y direction and stop the flowsheet.

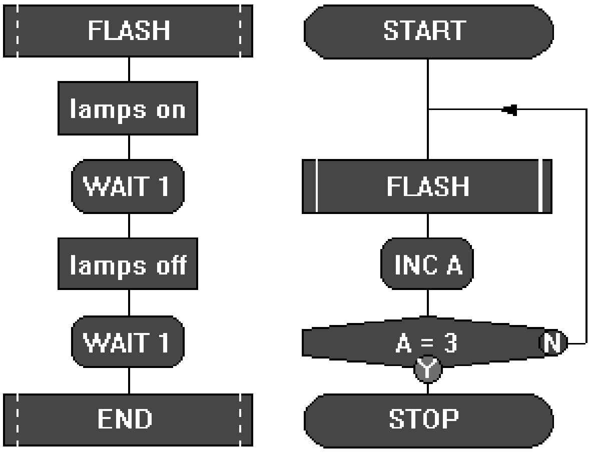
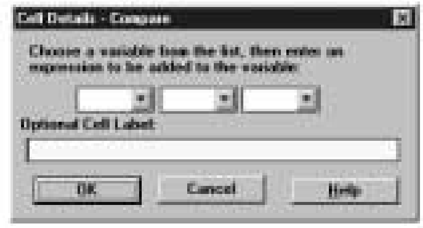


Fig 5.1. Repeating a sequence three times.


Cell Details - Compare

Choose a variable from the list, then enter an expression to be added to the variable:

Optional Cell Label:

Fig 5.2. Cell Details box of the Compare Decision command.

1. Use box one to select the variable that you want the command to check.

2. Use boxes two and three to complete the comparison. The drop-down list in box two contains a list of operators such as “greater than” (>), “less than” (<), and “equals” (=). Select the one that you require.

Use box three to set the number of times the sequence will repeat. Type in a number between 0 and 255, or select it from the drop-down list.

Another use of the INC command is to count the number of times something happens - the number of people passing through a gate or turnstile for example. This is often done by using a digital sensor such as a micro switch or a reed switch placed so that the sensor is “on” when a person passes. Fig 5.3 shows the three commands needed to do this. Notice that two Digital Decision commands are used to check the switch. The first command responds when the sensor is on. Then the sensor is immediately checked again to see that it is off before anything else happens. This ensures a clean signal for the INC command to count.

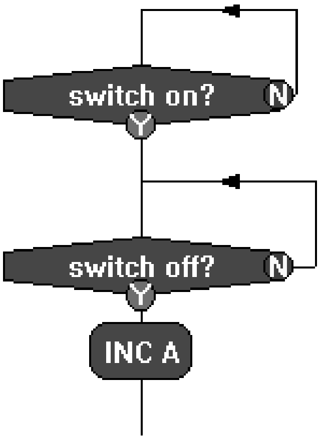


Fig 5.3. Ensuring a clean signal from a digital sensor.

You may well find that once it is downloaded into the chip, the flowsheet runs so quickly that even using the two Digital Decision commands does not give a clean count. If this is the case, you should include a short WAIT before the INC command, as shown in fig. 5.4. This flowsheet is for a system to count the number of people passing through a turnstile and to display the number in binary form, using LEDs connected to each one of the eight outputs on a PIC microcontroller.

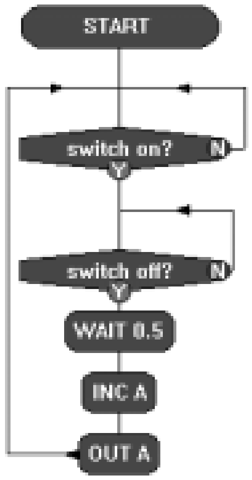


Fig 5.4.
Flowsheet for making and displaying a count.

Example one

A PIC microcontroller is used to control a system for counting cars entering and leaving a car park using two digital sensors. The outputs of the system are a red lamp lighting a "Full" sign, and a green lamp lighting a "Spaces" sign. The flowsheet for the system is shown in fig 5.5. When you test run this flowsheet, display the Variables window to see the changing value of A.

DEC

This system uses the DEC command which works in a very similar way to the INC command. The difference is that when flow passes through a DEC command, one is subtracted from the selected variable.

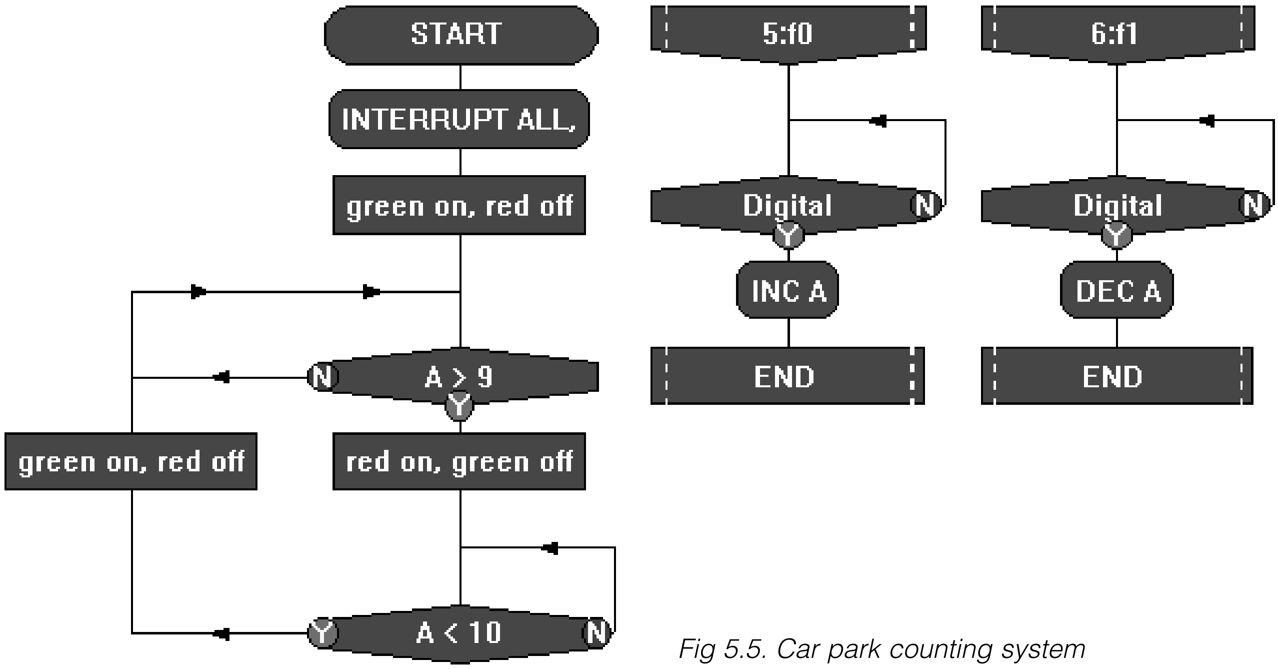


Fig 5.5. Car park counting system

Example two

A seven-segment display is a useful output device for displaying counting and timing. The flowsheet in fig. 5.6 is designed to control the kind of supermarket delicatessen-counter system in which customers take a ticket and then wait for their turn to be served when their number is displayed. When the assistant has served a customer, he or she presses a switch to display the next number.

The main routine uses an INC command to increment (add one to) the value of the variable A each time the assistant presses the switch. The DISPLAY macro makes an efficient way of translating the current value of A into an Outputs command which is set to switch on the appropriate number of outputs to display the number.

A similar approach could be used with an LCD screen. In this case, the DISPLAY macro would use a series of SEROUT commands as shown in fig. 5.7.

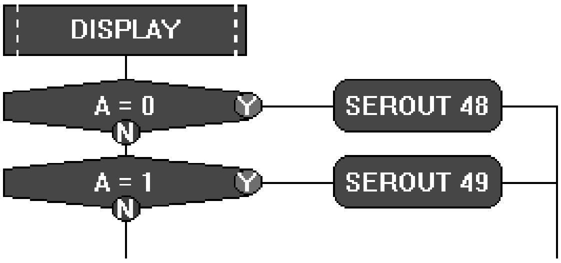
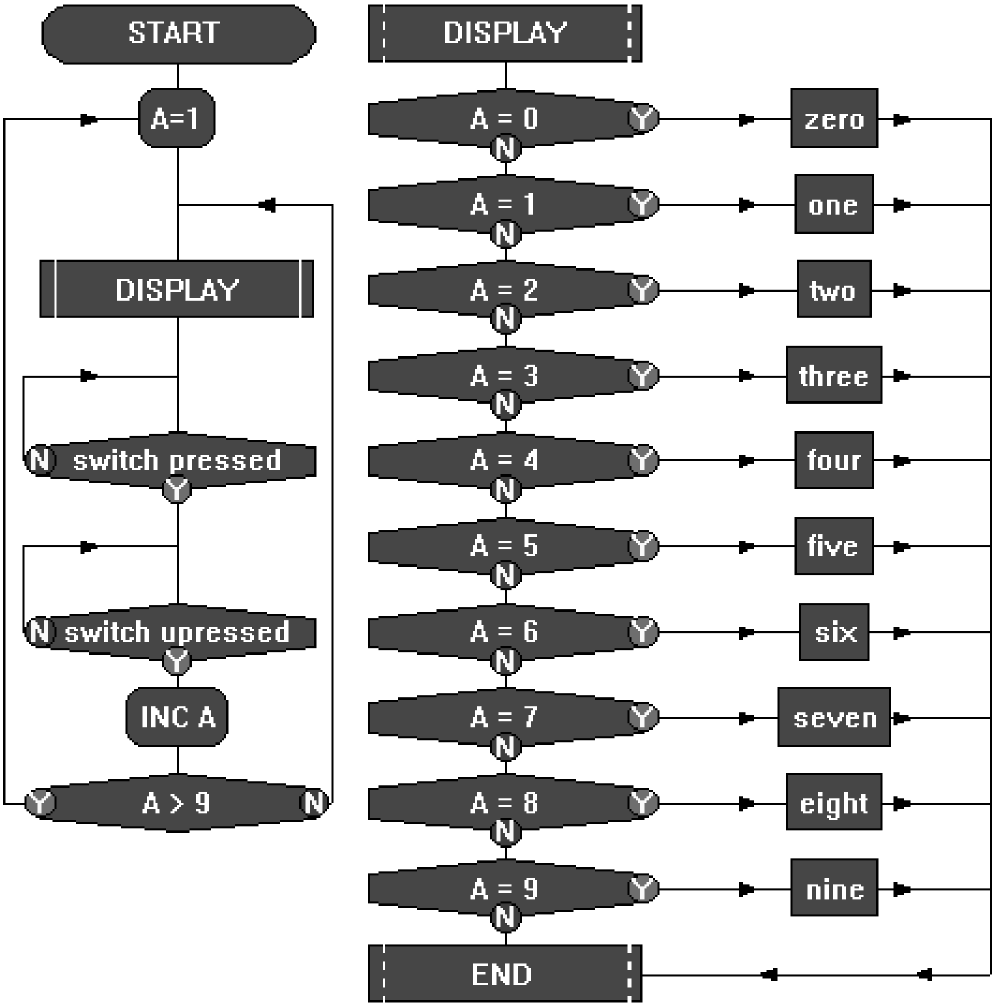


Fig 5.7. Part of an equivalent system that uses an LCD screen to display numbers.

Fig 5.6. A “Now Serving....” display system.



Timing

The variable TIME allows you to repeat a sequence for a set period of time. The flowsheet shown in fig 5.8 shows how it can be used to repeat a sequence for 10 seconds.

A Compare Decision is used to check the value of TIME. When this value reaches 10, flow will go in the Y direction and stop the flowsheet. Use the three boxes in the Compare Decision Cell Details box to enter the expression TIME > 10.

The maximum value that can be entered in the variable TIME is 255 (seconds).

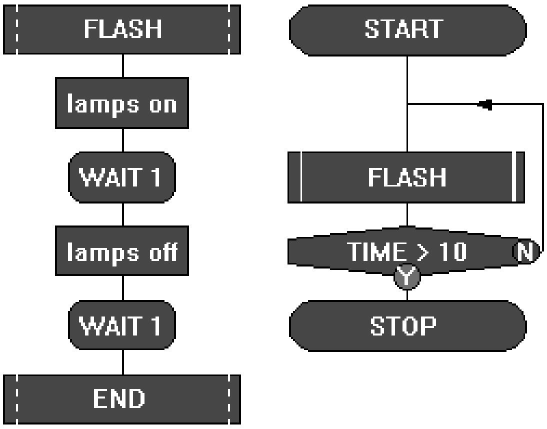


Fig 5.8.
Repeating a sequence for 10 seconds.

Setting the value of a variable

1: the Expression command

Expression

The Expression command is used to give a value to a variable as a flowsheet runs. The variable is given its value as flow passes through the command. The following examples show how it can be used.

Example one

The TIME variable works like a timer. It is reset to zero automatically at the start of a flowsheet run, and then continues to count up in seconds until the flowsheet stops.

There may be times when you want to reset the timer to zero during the flowsheet run. The system shown in fig 5.9 is designed to flash lights for 8 seconds and then play a tune for 6 seconds. If will do this only if the value of TIME is reset to zero as shown. Use the first two boxes in the Expression Cell Details box to enter the expression TIME = 0.

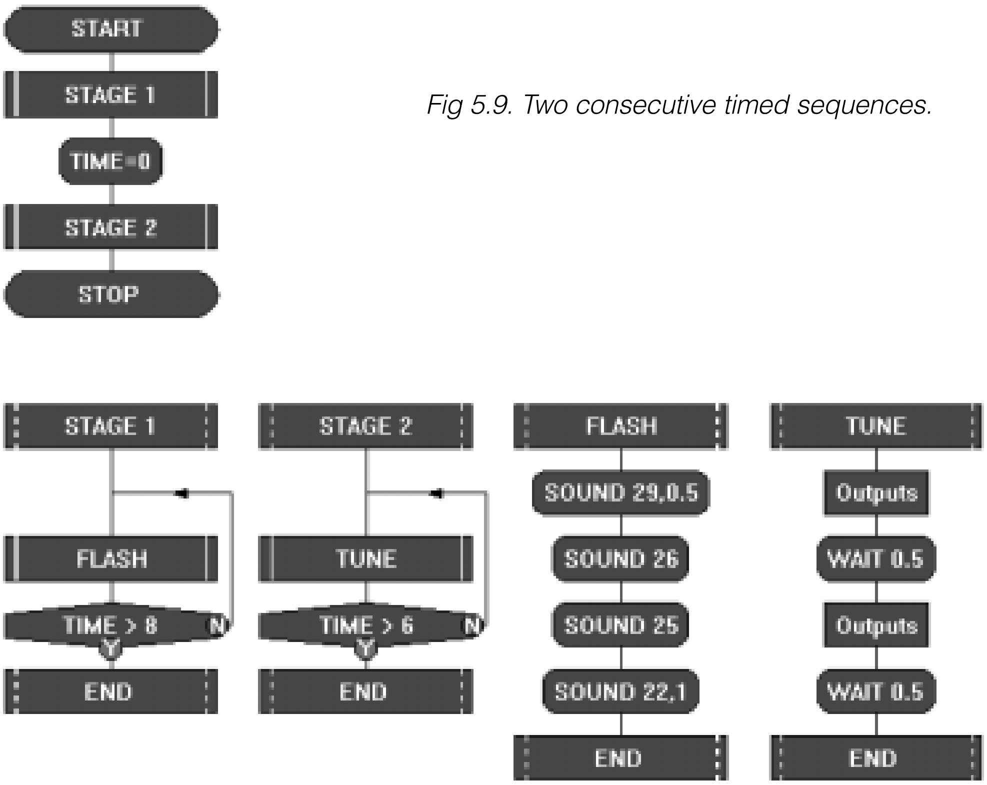


Fig 5.9. Two consecutive timed sequences.

Example two

A container in a warehouse is designed to hold ten packs of components. A system is needed to indicate the changing contents of the container as packs are removed. The flowsheet shown in fig 5.10 is designed to do this.

A digital sensor is used to indicate each time a pack is removed (notice the use of two Digital Decision commands to ensure a clean count). The number of packs in the container is displayed as a binary count using 8 LEDs connected to outputs of the PIC microcontroller.

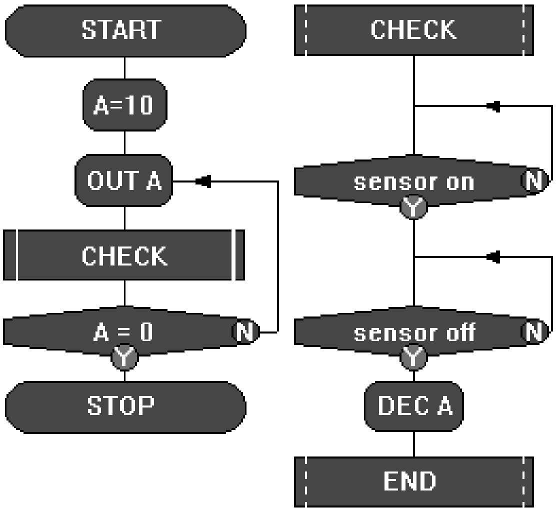
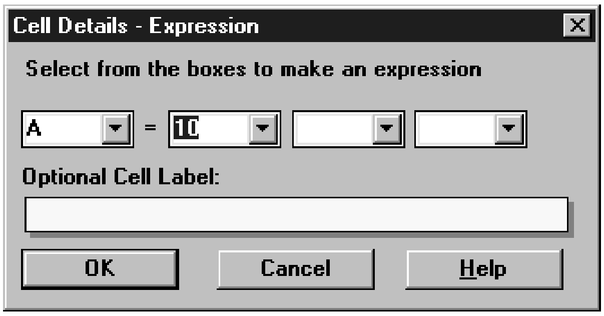


Fig 5.10. Counting down.

The DEC command counts down, so an Expression command is used to set the value of variable A to 10 at the start of the countdown when the container is full. The Expression command Cell Details box is shown in fig. 5.11. Use the first two boxes to enter the expression A = 10.



Cell Details - Expression

Select from the boxes to make an expression

A = 10

Optional Cell Label:

Fig 5.11. Expression command Cell Details box

Mathematical expressions

A value can also be given to a variable in the form of a mathematical expression as shown in the flowsheet in fig 5.12. This system counts the number of times that two separate switches are pressed, and displays the combined total. Use all four boxes in the Expression Cell Details box to enter the expression C=A + B.

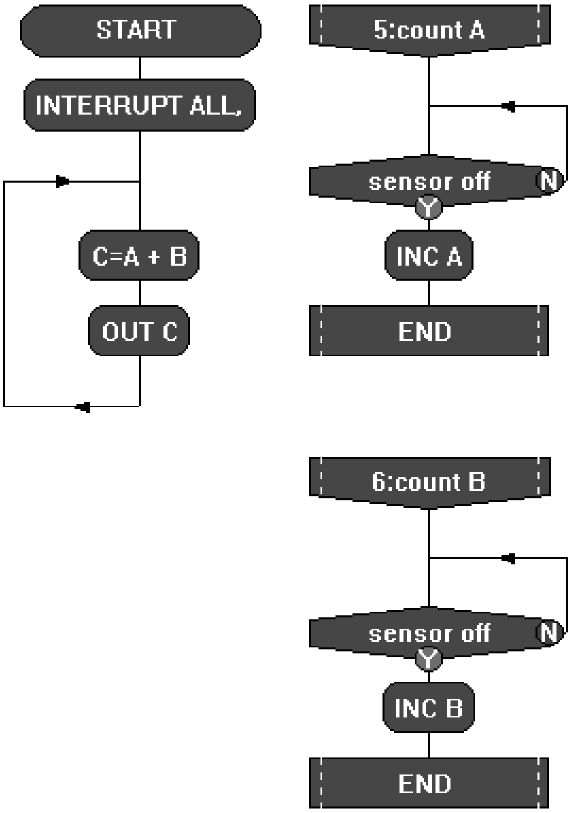


Fig 5.12. Displaying a combined count.

NOTE: the third box in the Expression Cell Details box contains a range of mathematical operators. These include SHL and SHR which are used to multiply and divide the values of variables in a limited way:

B = A SHR 1 means divide the value of A by 2, and put the result into variable B. SHR 2 means divide the value of A by 4, and so on. Use SHL in the same way to multiply the value of a variable.

Setting the value of a variable

2: IN,RND and PORT

IN

The IN command sets the value of a specified variable to the current binary value of the input port. So, for example, if switches connected to inputs 0 and 1 are pressed, then the value of the variable will be 3.

The flowsheet in fig 5.13 shows how this can be used to make a simple security system.

When switches connected to inputs 0 and 2 are pressed at the same time the binary value of the input port equals 5 (4+1), flow from the Decision command goes in the Y direction and a solenoid-operated lock is opened. If any other combination of switches is pressed, flow goes in the N direction.

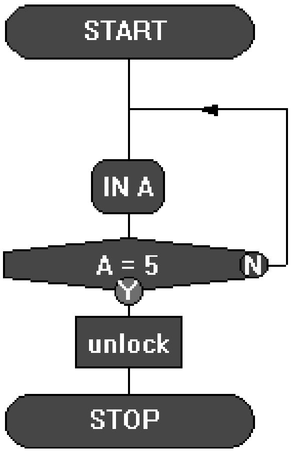


Fig. 5.13.
Simple security system that responds to pressing two switches.

The variable PORT can be used in a similar way to the IN command. The flowsheet in fig 5.14 serves the same purpose as that shown in fig 5.13. The Expression command A=PORT puts the binary value of the input port into the variable A.

Note that It is possible to transfer the binary value of the input port directly to the output port by using an Expression command PORT = PORT.

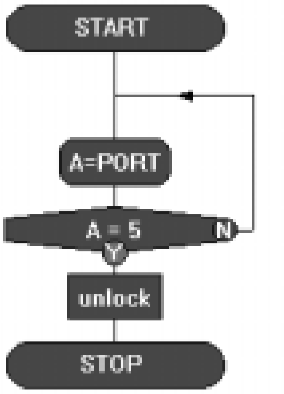


Fig 5.14
Using the variable PORT.

The RND command RND

Each time flow passes through this command, the selected variable is given a random value between 0 and 255. In the example shown in fig 5.15, a set of display lights for a small christmas tree are connected to 8 outputs of a PIC microcontroller. Every second the display will change at random.

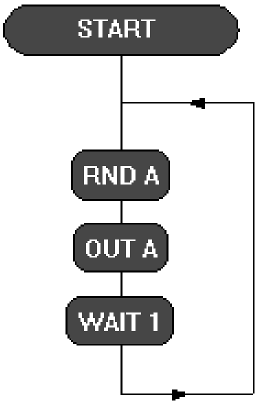


Fig 5.15.
Using the RND command to create a random display of lights.

READ and WRITE

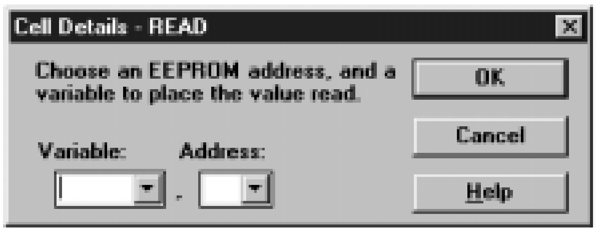
READ WRITE

When a flowsheet run is started, all variable values automatically reset to zero. So,when the PIC microcontroller is reset or powered up, all variable values are reset to zero.

If you want to retain variable values when the PIC microcontroller is powered up or reset, you can use the WRITE command to store values in the chip's EEPROM memory. The READ command is used to retrieve the values from the chip's memory.

The flowsheet in fig. 5.16 shows an example of how the commands can be used. The following information explains how this works.

1. The READ command takes the value which is currently stored in a selected address (in this case address 0), and puts it into the selected variable (in this case variable A). Use the READ command Cell Details box (fig 5.17) to enter the variable and the address from which the value is to be read.

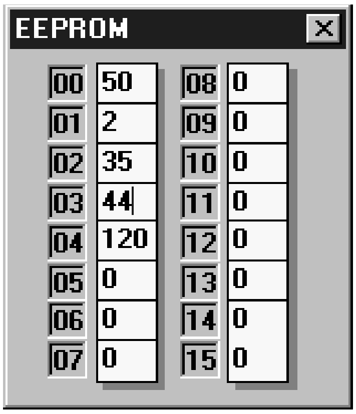

Cell Details - READ

Choose an EEPROM address, and a variable to place the value read.

Variable: Address:

Fig 5.17. READ command Cell Details box.

The PIC microcontroller's EEPROM memory has 16 separate addresses. Each one can store a number between 0 and 255. The EEPROM window displays the contents of the memory when you test run a flowsheet.


EEPROM

00	50	08	0
01	2	09	0
02	35	10	0
03	44	11	0
04	120	12	0
05	0	13	0
06	0	14	0
07	0	15	0

Fig 5.18 EEPROM window.

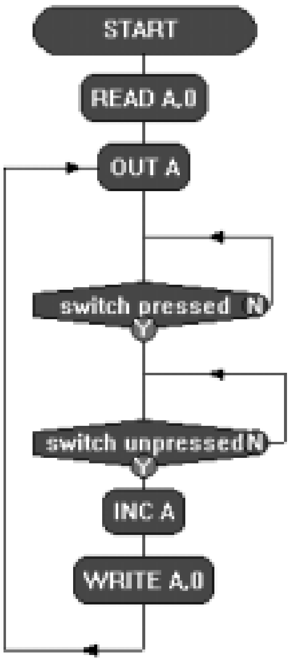


Fig 5.16. Using READ and WRITE commands to store a count.

2. The OUT A command in the flowsheet displays the current value of A using 8 LEDs connected to outputs of the PIC microcontroller.

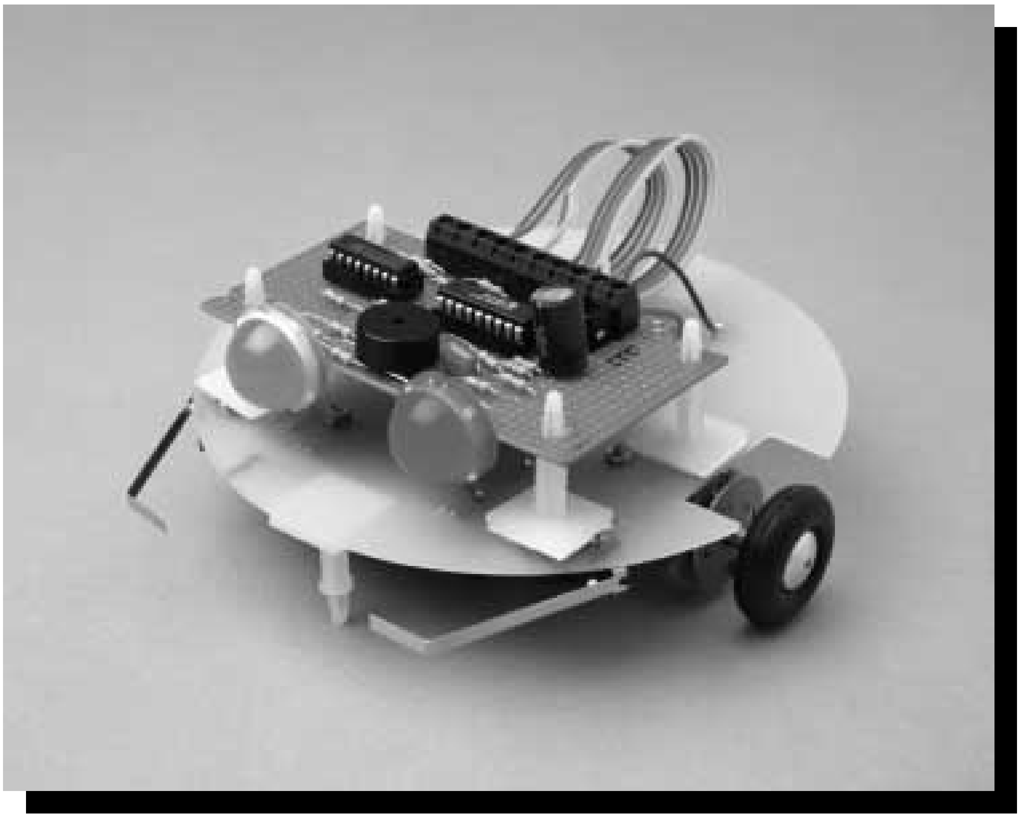
3. The INC A command increments (adds one to) the value of A each time a switch is pressed.

4. The new value of A is immediately stored in address 0 of the EEPROM memory by the WRITE command. The Cell Details box of this command is used in the same way as for the READ command.

When the PIC microcontroller is powered down, the value of A is stored in the chip's memory. When it is powered up, the first thing that happens is that the READ A,0 command retrieves the value of A which has been stored in address 0.

The EEPROM window gives an accurate simulation of the way these commands work when the flowsheet is downloaded. Any values shown will be saved when you save the flowsheet. They will also be downloaded with the flowsheet.

Therefore, when you have test run the flowsheet, make sure that you edit the values shown in this window back to zero before you download the flowsheet - unless, of course, you want to download the current values for some reason.



Section Two:

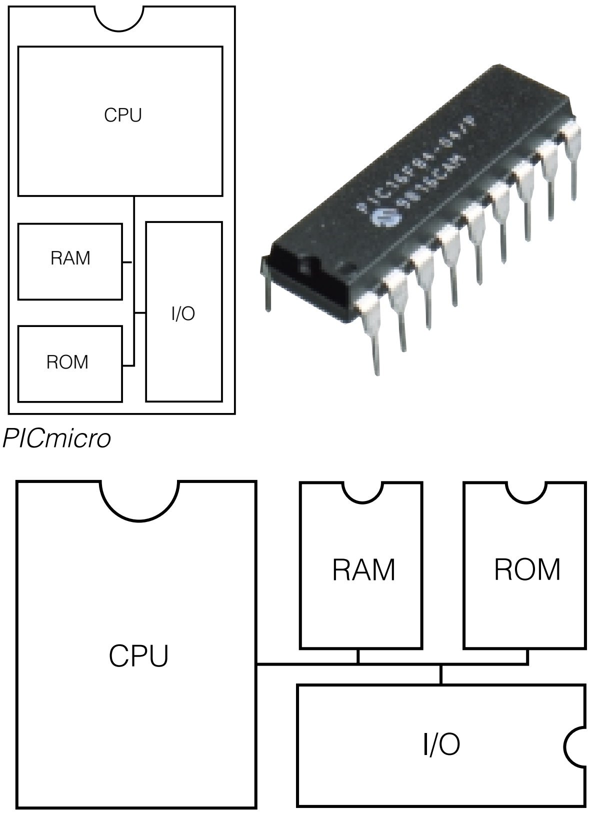
Connecting to the PIC Microcontroller

1. PIC Microcontrollers

Note: definitions of terms used in connection with PICs are in the "Definitions" section on page 44.

A PIC microcontroller is a single chip that can be programmed to switch output devices on and off in sequences and in response to input from sensors.

A microcontroller contains all the elements of a microprocessor system. This system, which is the basis of a computer, consists of a number of separate elements: CPU, RAM, ROM and I/O. In a microprocessor system, each of these elements will be in the form of one or more individual chips. However, the PIC microcontroller combines all of these elements in just one chip, that processes instructions as well as controlling devices.



PICmicro

Microprocessor System

Fig 1.1 The difference between a PIC microcontroller chip and a microprocessor system.

A PIC microcontroller is a programmable device, which means that it is able to store sets of instructions in the form of a program, and carry out the instructions whenever the program is run. The code that the chip uses internally to do this, is called "machine code". It is written in hexadecimal and is difficult for anyone other than a trained programmer to use. So, a programming language is used to allow designers to create, edit and download instructions to the chip. The machine code is generated automatically from the programming language.

Programming languages

PIC programming languages come in many different formats. High level languages such as PIC-Logicator and BASIC require very little knowledge of how PICs work and use commands that are easy to understand because they tend to use recognisable, everyday English words. The disadvantage of high level languages is that programs require more memory space and run more slowly than low level ones.

Low level languages (for example, assembler code) require a much deeper level of understanding of how the chips actually work. Their commands are complex and sometimes obscurely titled. The lower the level of the language, the nearer it is to the machine code and the harder it is to understand. On the other hand, the benefit of using a low level language is that the finished code is normally very concise and therefore runs very efficiently and faster than a higher level language can.

The panel on page 42 shows how the same simple program would appear in machine code and in three different programming languages.

Code examples

These sections of code simply blink an LED connected to output 4 once a second.

Machine Code Example (.HEX)

```
:10000000031E080086015F308C000D3084008001E3
:10001000840A8C0B07282C28212084000739840 6A9
```

BASIC Example (.BAS)

```
loop:        High4
             Pause 500

             low 4
             Pause 500

             Goto loop
             End
```

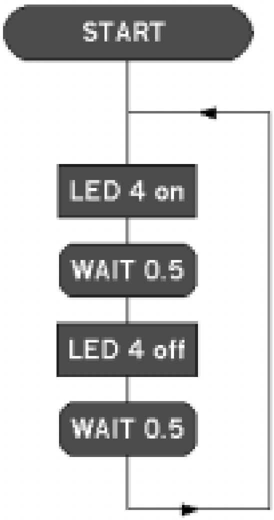
Assembler Code Example (.ASM)

```
PBCX         equ   1

             include

_loop        movlw 004h
             call  high@
             movlw 001h
             movwf R0+1
             movlw 0F4h
             call  pause@XW
             movlw 004h
             call  low@
             movlw 001h
             movwf R0+1
             movlw 0F4h
             call  pause@XW
             @GOTO _loop
             call  end@
```

PIC-Logicator Example (.PFL)



Different types of PIC

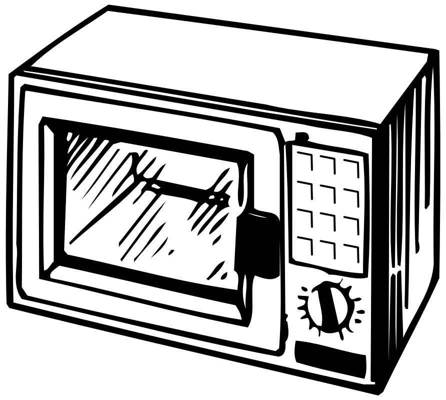
PIC chips are available in many different sizes to suit different applications. Chips in sizes from 8 pins to 40 pins are available to provide different numbers of outputs and inputs. Some provide digital inputs only. Some have ADC (analogue to digital conversion) built into them so that analogue sensors can be connected directly to them. Obviously the larger, higher specification PICs are more expensive, so it is important for designers to select the most appropriate chips for their purposes.

PICs also vary in terms of how they are programmed and erased. Some PICs are erasable by UV, some electrically and some not at all. UV light erasable chips are not widely used because they need a special UV light source to erase them and because of the health and safety problems associated with using UV light. Most PIC chips are FLASH reprogrammable which allows them to be overwritten electrically as many times as required. These chips have an “F” in their reference number. The third type of chip is called OTP (one time programmable) and, as the name suggests, can only be programmed once. These chips have a “C” in their reference number.

Applications

Millions of PIC chips are used each year in: electronic consumer goods, mobile telephones, medical equipment, computer products, and industrial applications. There will be, on average, 35 programmable chips in every car, used in a range of sub-systems from engine management to remote locking.

Up to 225 chips can be found in every home, where a typical use would be in a security system in which a PIC might be used to monitor the sensors placed around the house and switch an alarm; as well as enabling features such as the use of a code, input by a keypad, to set and reset the system. A microwave oven may use a single PIC to process information from the keypad, display user information on a seven segment display, and control the output devices (turntable motor, light, bell, magnetron).



The complex control system for an automatic washing machine is likely to use PICs. The system takes in information from devices used to select different washing programs, and also from sensors that monitor temperature and water levels. It is programmed to make use of this information and switch output devices such as motors, pumps and heaters on and off in appropriate timed sequences.

Advantages and disadvantages

One PIC chip can replace a wide variety of traditional discrete electronic components such as transistors, logic circuits, 555 timer chips. This means that products based on PICs can be smaller and cheaper. They will have fewer separate parts so they are likely to be more reliable. The company manufacturing the products will have reduced stock levels. Product-assembly will be simpler and therefore quicker and cheaper.

Using PICs can make products more flexible. Their features are programmed into the chip, not built into electronic hardware, so they can be developed and changed quickly and easily. If the manufacturer of a product wants to change a feature of the product, a simple change to the PIC program can achieve what is required without the need to alter any of the components on the PCB.

The main disadvantage of PICs is that they have only a very low power output, of a few milliamps. They therefore require interfacing circuitry to drive higher current loads.

Interfacing to a PIC

A PIC chip does not provide the complete control system. It should be seen as the “process” section of the system. It is programmed with instructions to use information from the devices in the “input” section, and switch on and off the devices in the “output” section.

INPUT	PROCESS	OUTPUT
	PICmicro	

Once the chip has been programmed it is useless until it is interfaced to the real world. Interfacing involves providing power to the circuit, and using standard interfacing circuits for input and output devices. These include potential dividers for sensors, and transistor drivers for output devices. This section of the book shows the basic interfacing circuits that are likely to be required for most school projects.

Definitions

PIC The letters PIC stand for “peripheral interface controller”. The full name of this type of chip is “PIC microcontroller”, but this is often shortened to “PICmicro” or just “PIC”.

CPU Central Processing Unit

ALU Arithmetic Logic Unit

A separate processor inside the chip to specifically handle mathematical and logic operations. This is more efficient than having the main processor perform these operations.

RAM Random Access Memory

Memory that can be accessed as required, with ability to write and read as necessary. Data stored in RAM is not held if the power to the chip is turned off.

ROM Read Only Memory

Memory that can only be read from and is programmed only once.

I/O ports Input / Output ports

Connections on the PIC chip that either connect to input devices such as switches or output devices such as bulbs through an appropriate driver. The I/O ports on PICs can be configured as either inputs or outputs, but in PIC-Logicator these have been preset to simplify programming.

PICs commonly have two I/O ports referred to as PORTA and PORTB. Within each port there are a number of bits (designated RAx or RBx (where x is the number of the bit)) which are in effect the physical pins on the PIC.

CLOCK All microprocessors operate at a fixed speed which is referred to as speed or clock frequency and is quoted in Hertz (Hz). PIC chips normally run at 4Mhz (Mega Hertz) and in some cases require an external resonator to regulate this speed. Some newer type PIC chips have these resonators built in.

EEPROM Electronically Erasable Programmable Read Only Memory

This is memory in the chips sometimes called FLASH memory that can be programmed with data and read back. This memory is still stored even if power to the device is switched off and is what makes up most of the data memory in PIC chips.

ADC Analogue to Digital Converter

Analogue sensors provide information in the form of varying voltages, usually between 0 and 5 volts. Analogue to Digital Converter chips sample these voltages at regular intervals and convert them to a digital information which is then sent out from the chip using serial data transmission.

Some PICs contain the ADC function. For example, the PIC16F872 has 4 analogue inputs which, after conversion, each give a digital value of between 0 and 255. So, a temperature sensor connected to the chip would give a range of readings in which 0˚C (0 volts) is converted to 0, and 100˚C (5 volts) is converted to 255. So, there are 2.5 ‘steps’ per degree change. The range is 0 to 255 because 8 bits are used to make the conversion (255 decimal = 11111111 binary). The higher the number of bits used for sampling, the higher the accuracy of the conversion.

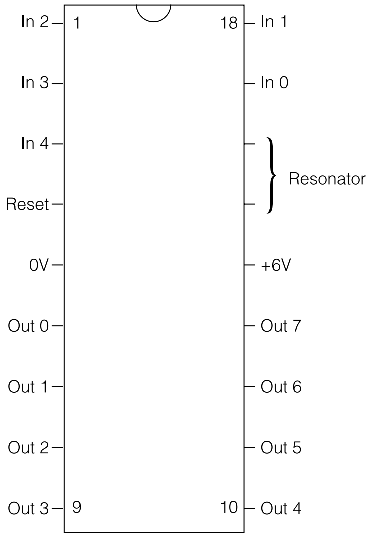
2. PIC Microcontroller Pin-out Diagrams

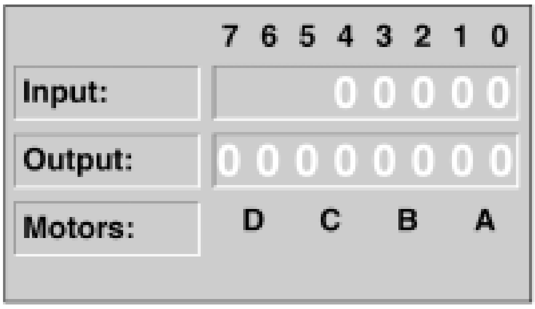
NOTE:

When you select a chip from the Options>PIC Type menu in PIC-Logicator software, the software automatically configures itself to display only the input, output and motor options available with that chip. This is illustrated in the table beside each pin-out diagram.

18 pin PIC microcontrollers

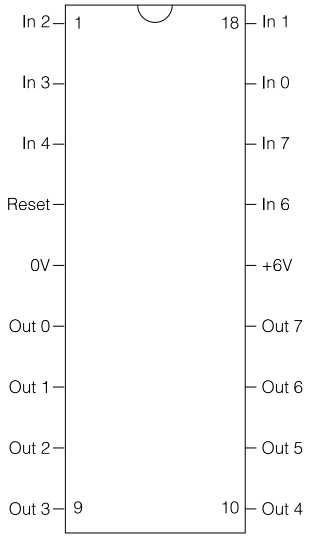
16F84, 16F84A



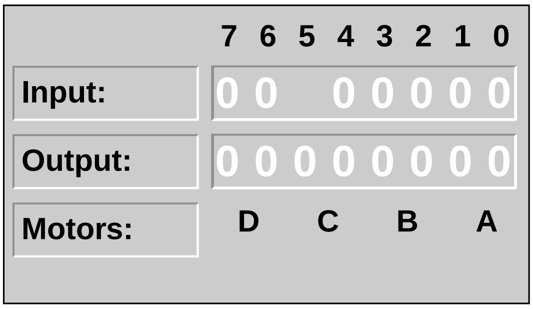


	7	6	5	4	3	2	1	0
Input:				0	0	0	0	0
Output:	0	0	0	0	0	0	0	0
Motors:	D		C		B		A	

16F627

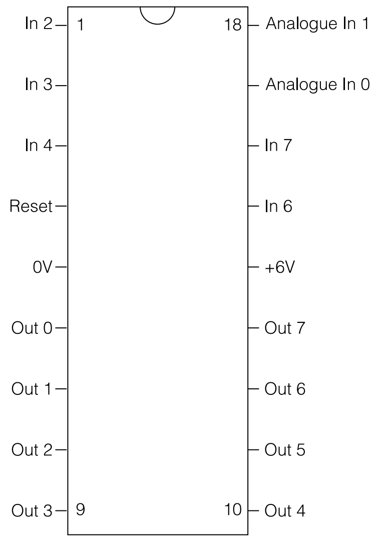


This chip has an internal resonator.

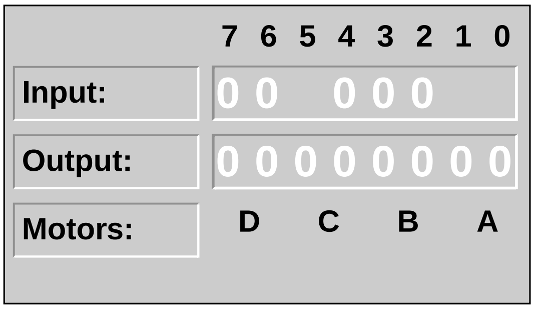


	7	6	5	4	3	2	1	0
Input:	0	0		0	0	0	0	0
Output:	0	0	0	0	0	0	0	0
Motors:	D		C		B		A	

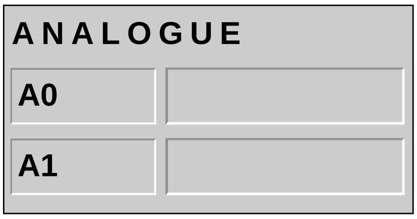
16F628



This chip has an internal resonator.



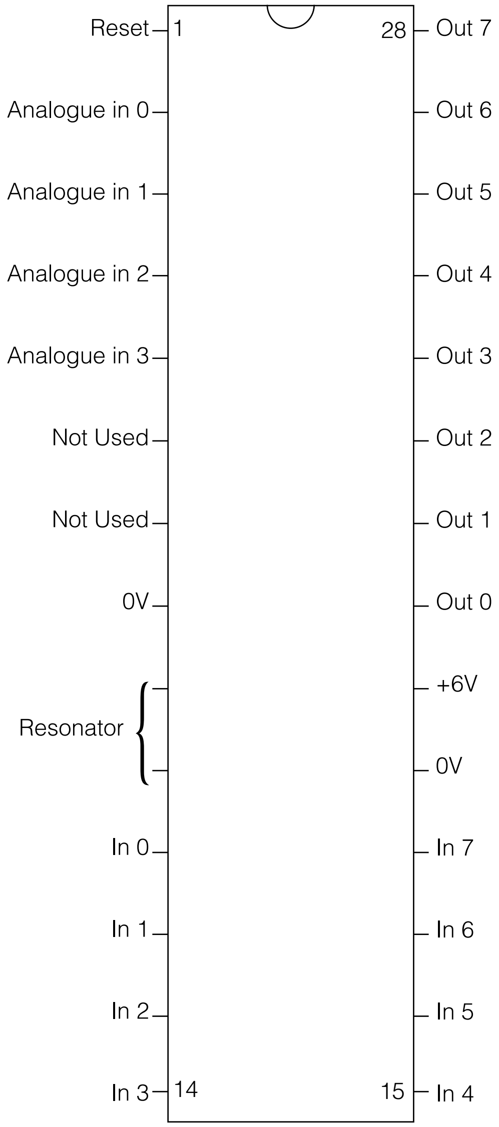
	7	6	5	4	3	2	1	0
Input:	0	0		0	0	0		
Output:	0	0	0	0	0	0	0	0
Motors:	D		C		B		A	

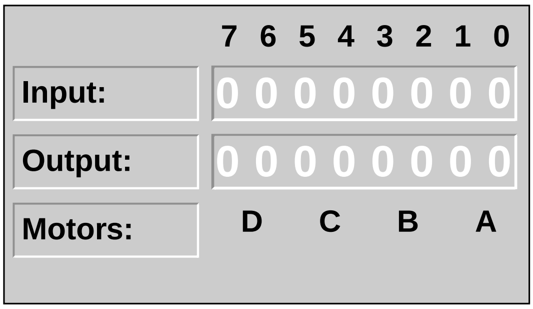


ANALOGUE	
A0	
A1	

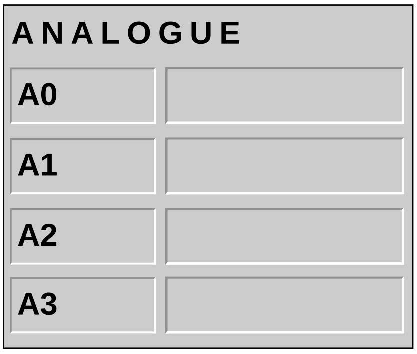
28 pin PIC microcontrollers

16F872, 16F873, 16F876





	7	6	5	4	3	2	1	0
Input:	0	0	0	0	0	0	0	0
Output:	0	0	0	0	0	0	0	0
Motors:	D		C		B		A	

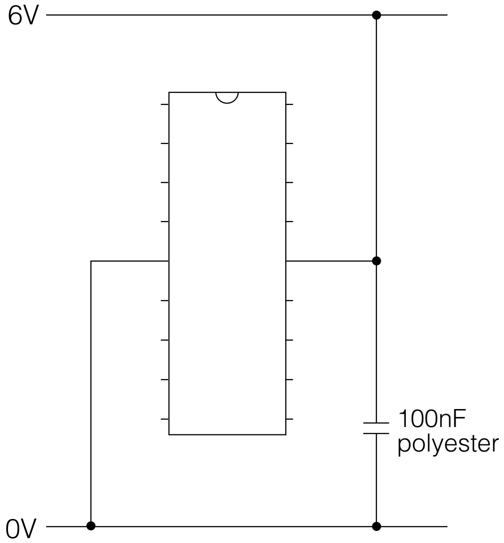


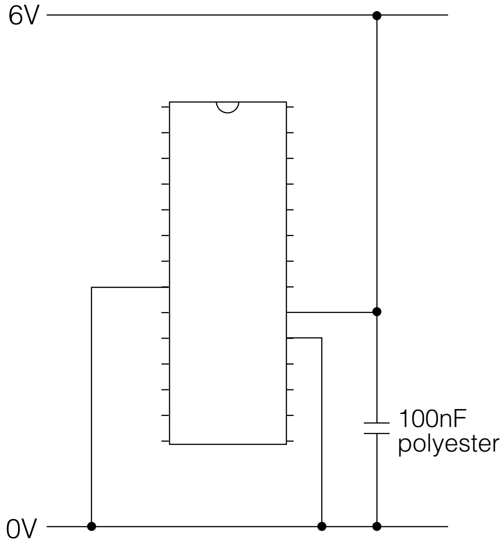
ANALOGUE	
A0	
A1	
A2	
A3	

3. Basic Connections

Power

The PIC microcontroller requires a 6V DC supply. This can be provided by 4 x AA cells. The following diagrams show connection to 18 and 28 pin chips. The 100nF polyester capacitor is used to decouple the PIC microcontroller power supply for reliable operation.





Using a low-voltage power supply

In some situations it might be more convenient to power your project from a low voltage DC power supply, rather than a battery. If you wish to do this, you should use a 7805 voltage regulator as shown in fig. 3.1.

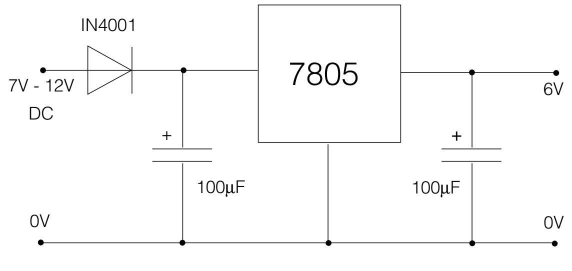
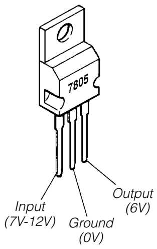


Fig 3.1 Using a 7805 voltage regulator



Using two power supplies

If your project includes an output device which requires a voltage greater than the 6V that you are using to power the PIC microcontroller circuit, use two separate power supplies as shown in fig. 3.2. This is more efficient and economical than using a relay.

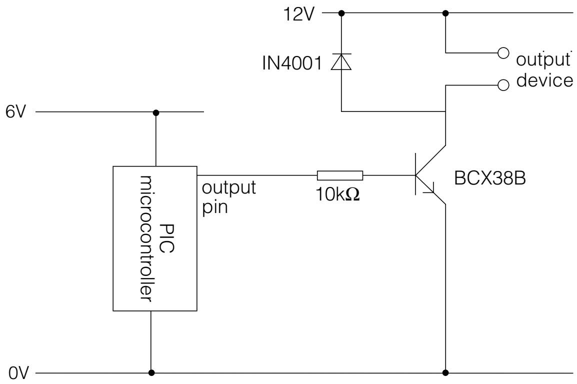
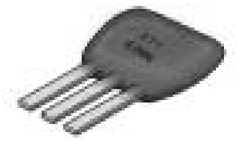


Fig 3.2 Using separate power supplies to power the process and output sections of a project.

Resonator

The PIC microcontroller provides a clock pulse internally. The resonator regulates the speed of the clock pulse, or in other words, sets the speed at which the PIC microcontroller works (4MHz). Some PIC microcontrollers have an internal resonator. Others need to have a 4MHz ceramic resonator connected as shown in fig. 3.3.



4Mhz ceramic resonator

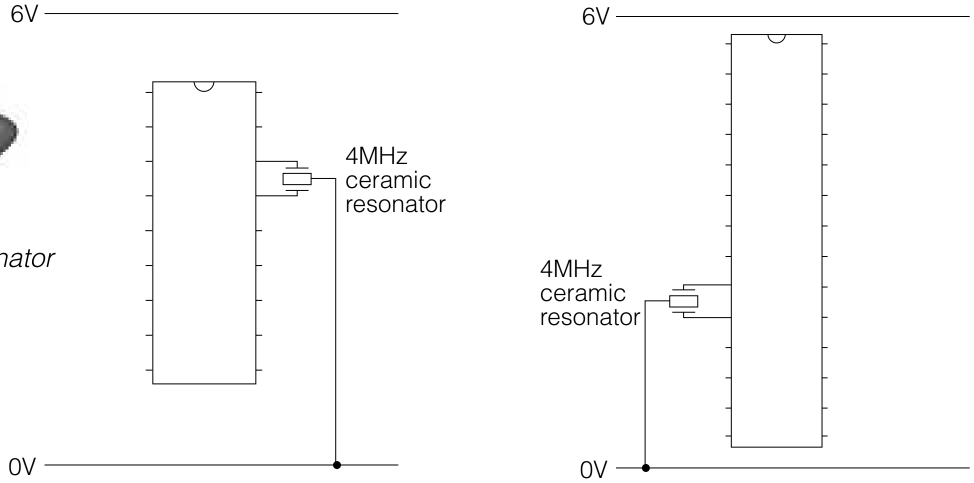


Fig. 3.3. Connecting a resonator to 18 pin and 28 pin microcontrollers

Reset

This pin must also be connected to the 6V rail through a 4k7 resistor as shown in fig. 3.4. You can also include a reset switch as shown in fig. 3.5, if you like. When you press this switch, the flowsheet programmed into the chip will restart at the START command.

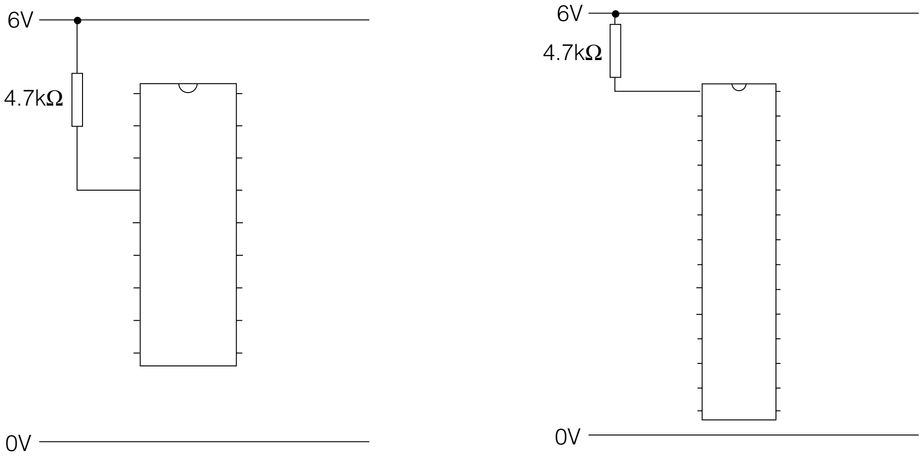


Fig. 3.4. Connecting the reset pin

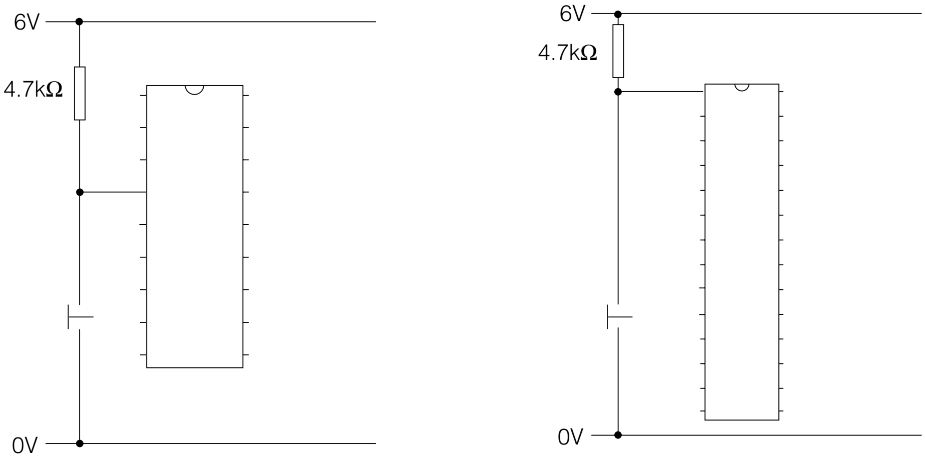


Fig. 3.5. Connecting the reset pin to include a reset switch

4. Connecting Output Devices

Note that diagrams shown in this section show how to connect the output device only. For simplicity, the PIC microcontroller is simply shown as a block. In each case, the PIC microcontroller must be connected to power as shown in “3. Basic Connections” (page 47).

The position of the output pins on the block diagram does not necessarily indicate the position of an output pin on a PIC microcontroller chip. Consult the pin-out diagrams in “2. PIC Microcontroller Pin-out Diagrams” (page 45) to identify the correct pin to use.

A: Components that can be connected directly to an output pin

The following low current output devices can be switched directly by the PIC microcontroller.

A1. Light Emitting Diode (LED)

An LED can be connected directly to an output pin of the PIC microcontroller, as shown in fig. 4.1. The resistor is required to limit the current through the LED.

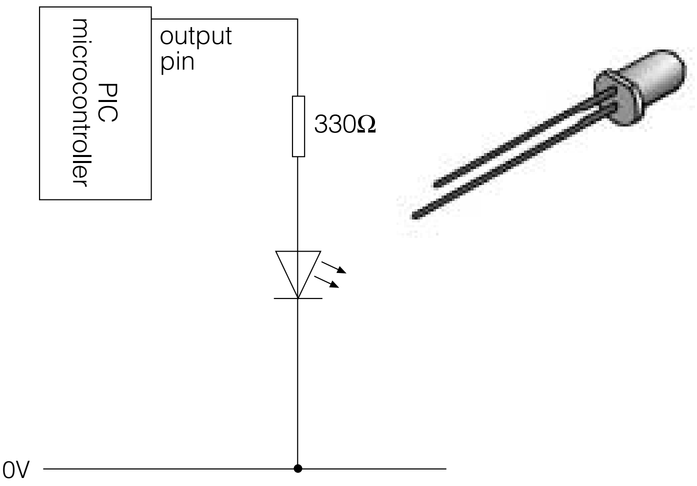


Fig 4.1 Connection of an LED

A2. Seven Segment Display

A seven segment display is made up of seven LED bars, which can be connected directly to output pins of the PIC microcontroller, using series resistors, as with individual LEDs.

The bars of the seven segment display can be lit in different combinations to show the ten digits 0 to 9, as well as some letters of the alphabet.

A number of different types of seven segment display are available. You should use a common cathode type. The common pin connects to 0V.

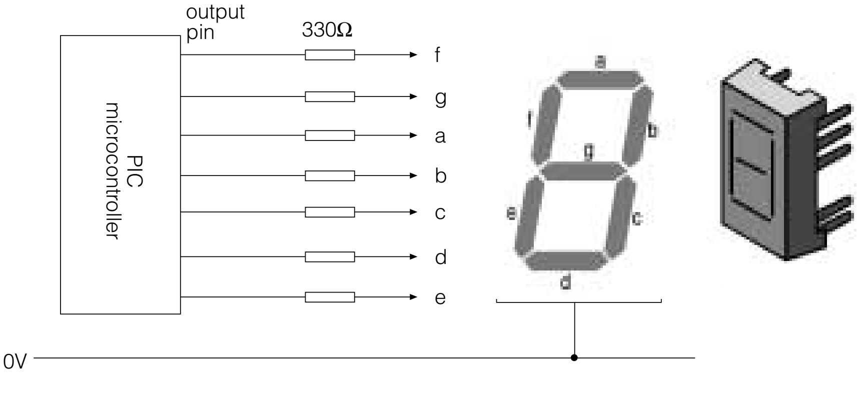
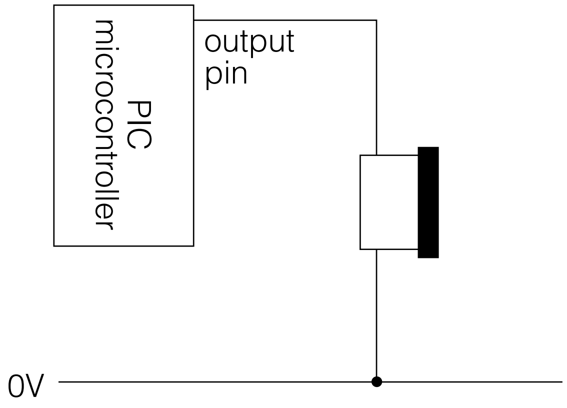


Fig 4.2 Connection of a seven segment display. Each LED bar has an identifying letter. Consult the pin-out diagram of the seven segment display that you are using, to identify the appropriate pins. This may be supplied with the component or it may be in the supplier’s catalogue.

A3. Piezo Sounder

A piezo sounder can be used to produce a range of different sounds. The sound is created by a pulsed signal. This signal can be sent from any one of the output pins on the PIC microcontroller. Use the SOUND command in PIC-Logicator software to select the sound and the pin to be used. Because the software is generating the pulsed signal, use a piezo sounder (sometimes called a "piezo transducer") without drive circuit.



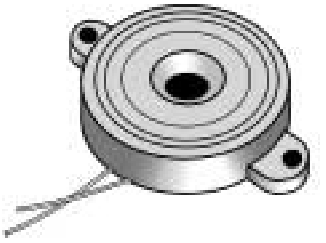


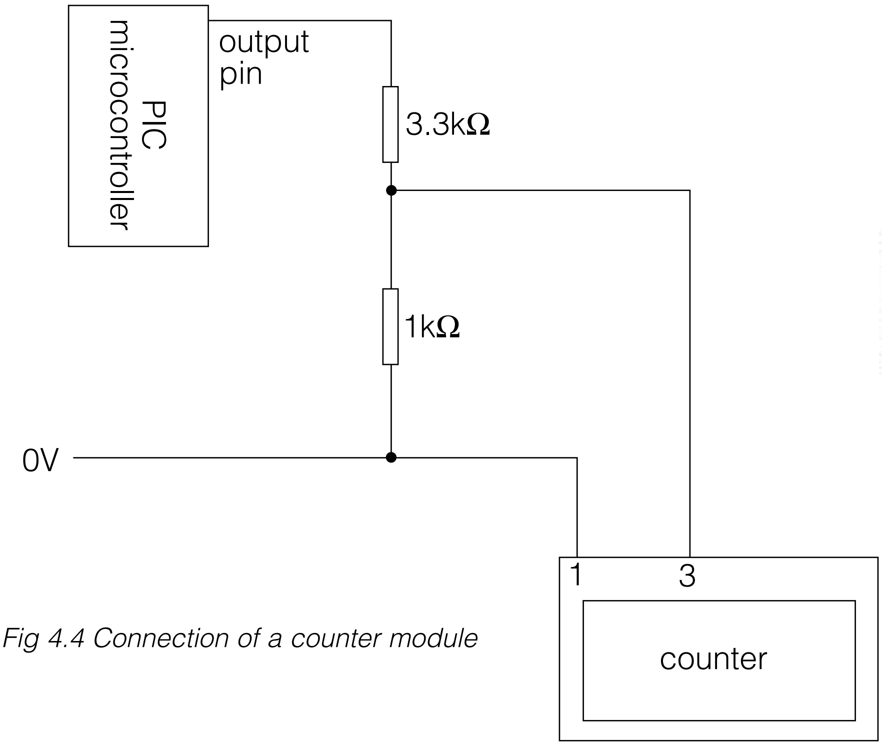
Fig 4.3 Connection of a piezo sounder

A4. Counter Module

A counter module is an LCD numerical display which can be used to show a changing count value. It is powered by its own internal 1.5V battery.

The counter is incremented by a pulse. Use two Outputs commands to create a pulse - one to switch the output on; the other to switch it off.

Connect the counter module as shown in fig. 4.4. A potential divider, formed from two resistors, is used to reduce the PIC microcontroller output signal to the 1.5V required by the counter module.



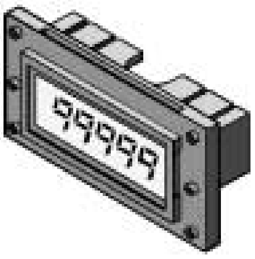
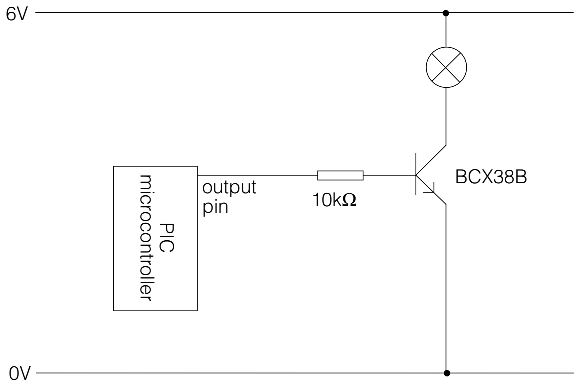


Fig 4.4 Connection of a counter module

B: Components that require a transistor switching circuit

Higher current output devices cannot be switched directly by the PIC microcontroller chip. They require a transistor switching circuit. A device that is commonly used for this purpose is a BCX38B Darlington driver, which is actually two transistors in a single package. This device can switch currents up to 800mA. If you want to switch higher currents than this, you should use a MOSFET as shown on page 52.

B1. Figs 4.5 and 4.6 show how a BCX38B Darlington driver is used to connect a signal lamp and a buzzer to a PIC microcontroller.



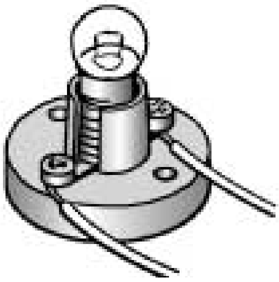
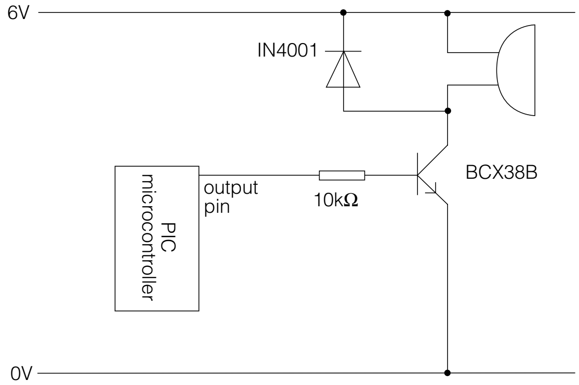


Fig 4.5 Connection of a signal lamp



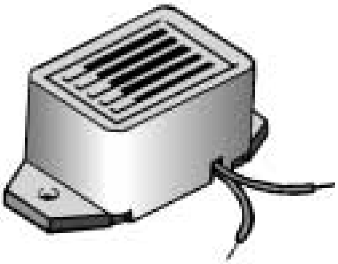
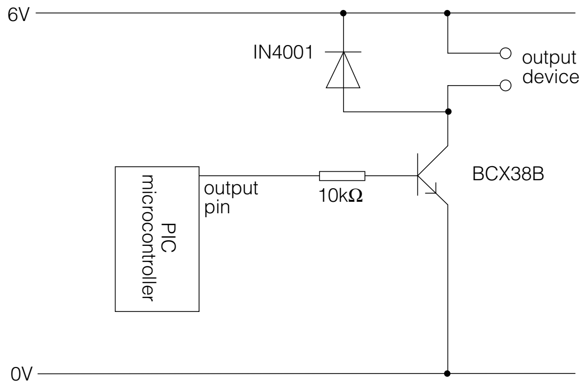


Fig 4.6 Connection of a buzzer

B2. Output devices such as relays, solenoids and solenoid valves create a back emf when power is switched off. When you are using one of these devices, it is essential that you connect a back emf suppression diode across the output device as shown in fig. 4.7.



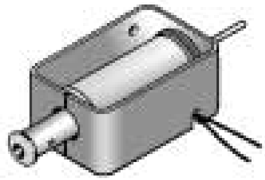


Fig. 4.7 Connection of an output device such as a relay, solenoid or solenoid valve

B3. If you need to connect a number of output devices to the PIC microcontroller, it will be more convenient to use a ULN2003A Darlington driver IC as shown in fig. 4.8. This chip contains seven Darlington transistors similar in value to the BCX38B. It also contains internal back emf suppression diodes, so no external 1N4001 diodes are needed.

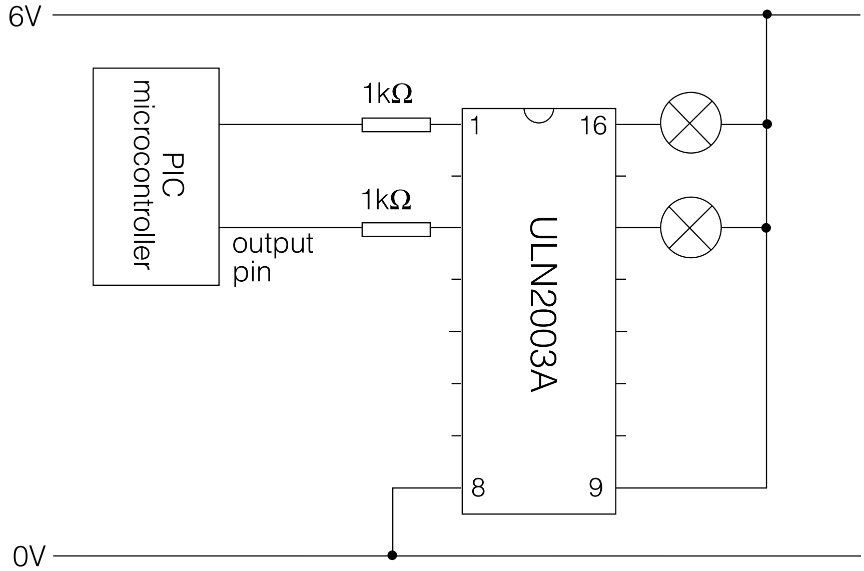


Fig. 4.8 Connection of a number of output devices using a ULN2003A Darlington driver IC

B4. The BCX38B can switch currents up to 800mA. If you want to switch higher currents than this, you should use a MOSFET as shown in fig. 4.9. The IRF530 is a suitable power MOSFET to use for this purpose. Note that when using two separate power supplies, it is necessary to link the 0V rails as shown.

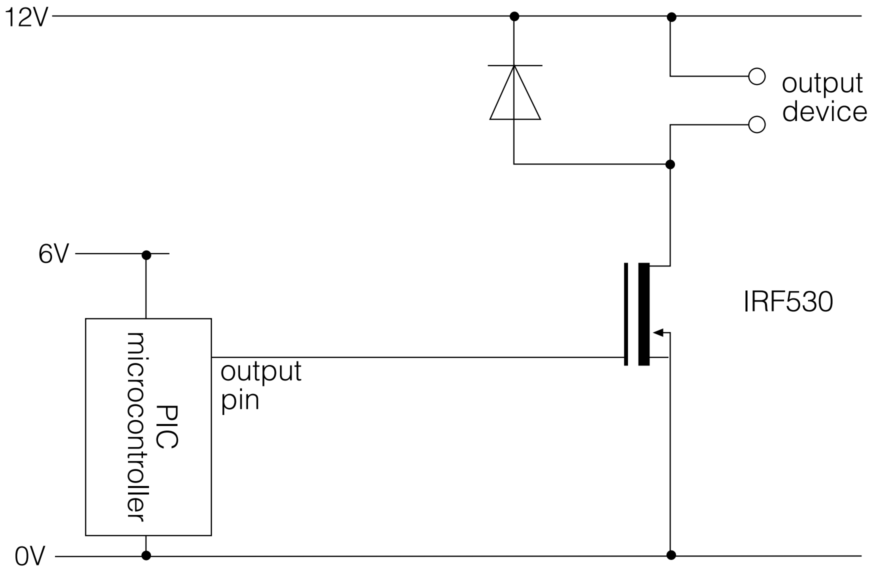


Fig. 4.9 Standard MOSFET circuit

C: Motors

C1. Low-voltage DC motors

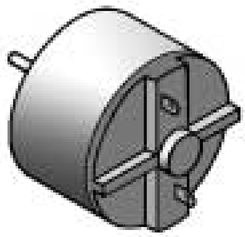
The higher quality low voltage DC motors often called “solar motors” are recommended.

The PIC-Logicator system simplifies the control of motors by using the MOTOR command (see page 16).

The dialog box of this command allows you to select the motor or motors you want to switch, and set them to drive forward or reverse.

Motors are labelled A,B,C or D. Motor A is the motor controlled by outputs 0 and 1 of the PIC microcontroller. Motor B is the motor controlled by outputs 2 and 3, and so on. See the “PIC Microcontroller Pin-out Diagrams” section (page 45) for information on how this works with the different chip options.

The simplest way to connect motors to the PIC microcontroller is to use a motor driver IC such as the L293D. This allows you to control either one or two motors. Fig 4.10 shows how to connect one motor. If you use the two PIC microcontroller outputs 0 and 1 as shown, then you would use Motor A in the software MOTOR command to control it.



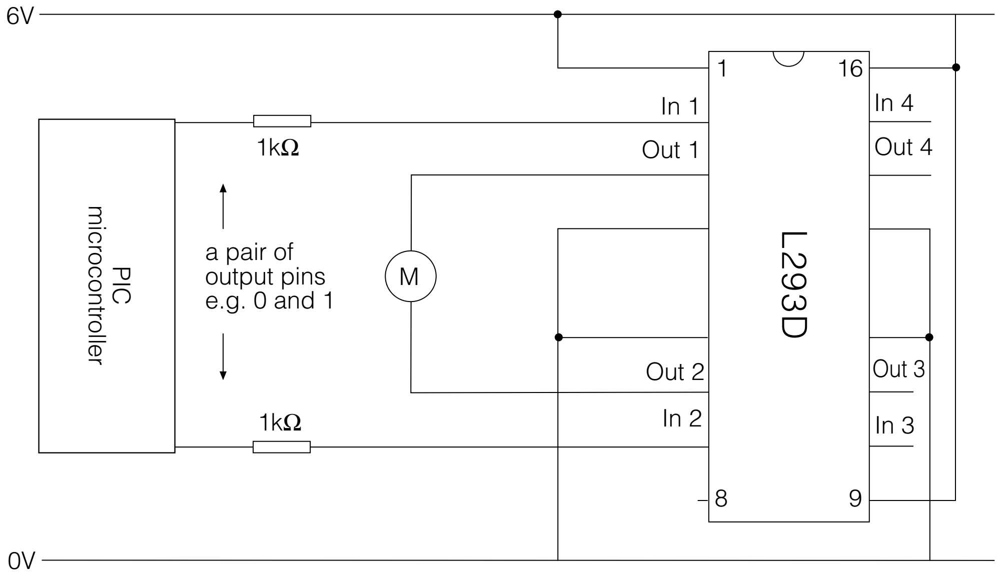


Fig 4.10 Using the L293D motor driver with a PIC microcontroller

NOTE: Pin 8 of the L293D should be connected to 6V if this is the appropriate voltage for the motor that you are using. If you are using a higher voltage motor e.g. 12V, then this pin should be connected to a power supply of this voltage. Maximum current is 1A.

You could connect another motor to pins “Out 3” and “Out 4” of the L293D. It would be controlled by connecting pins “In 3” and “In 4” to another pair of output pins on the PIC microcontroller. If you were to use PIC microcontroller outputs 2 and 3, then you would use Motor B in the software MOTOR command to control it.

C2. Stepper motors

Stepper motors are very accurate motors that are used in devices such as printers and computer disk drives. Unlike DC motors which spin freely, a stepper motor moves round in accurate steps. Each step is usually 7.5 degrees, so that 48 steps make one complete revolution.

There are two main types of stepper motor - unipolar and bipolar. The unipolar is most commonly used in school projects and so this is the type shown here.

Unipolar motors usually have four coils which must be switched on and off in a particular sequence to make the motor turn. The table in fig. 4.11 shows the four-step sequence.

Step	Coil 1	Coil 2	Coil 3	Coil 4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Fig 4.11 Four step sequence to drive a stepper motor

Use an Outputs command in PIC-Logicator software for each step. Set each command to switch on or off the appropriate outputs from the PIC microcontroller. The time between each Outputs command will determine the speed at which the motor turns. The minimum WAIT time is 0.1s, so if you use this command the fastest speed possible is only about 5s per revolution. A better solution is shown in fig. 4.12.

The TIMER macro in this flowsheet provides a very brief pause between each Outputs command. It will take the PIC about 0.5ms to process the macro. The Expression command that sets the value of A will therefore set the speed of the motor. See pages 32 and 35 for more information on the commands used.

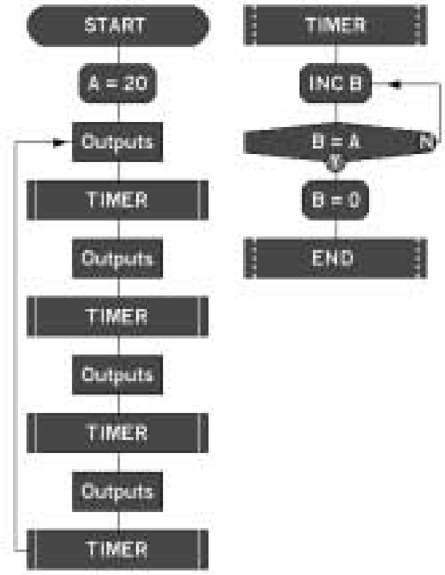


Fig. 4.12.
Accurate speed control of a stepper motor.

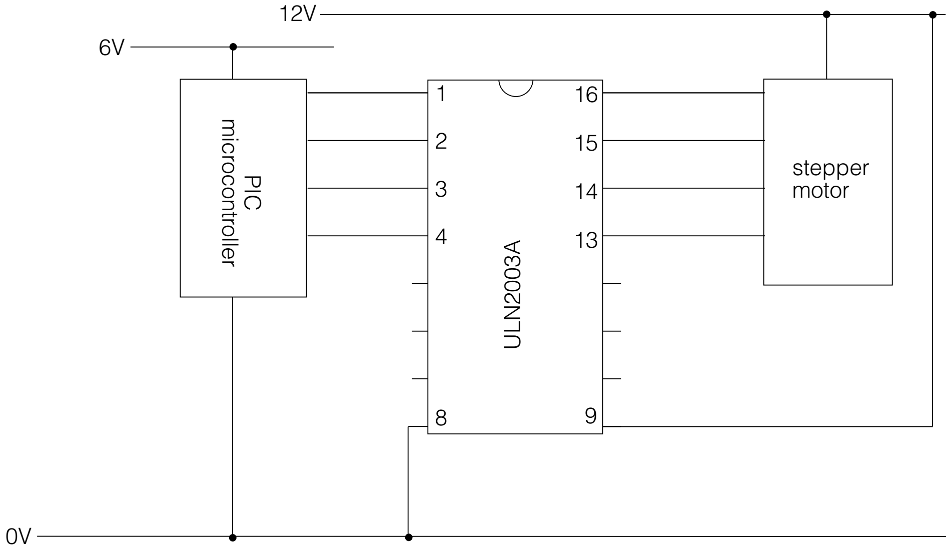
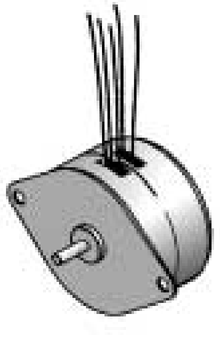


Fig 4.13 A ULN2003A Darlington Driver is used to drive the stepper motor.



5. Connecting Input Devices

Note that diagrams shown in this section show how to connect the input device only. For simplicity, the PIC microcontroller is simply shown as a block. In each case, the PIC microcontroller must be connected to power as shown in “3. Basic Connections” (page 47).

The position of the input pins on the block diagram does not necessarily indicate the position of an input pin on a PIC microcontroller chip. Consult the pin-out diagrams in “2. PIC Microcontroller Pin-out Diagrams” (page 45) to identify the correct pin to use.

A. Digital Sensors

The most-commonly used digital sensor is a switch, such as a microswitch, push switch or reed (magnetic) switch. All of these devices have two contacts which are either open (0) or closed (1). Fig 5.1 shows how to connect a digital sensor such as a switch to a PIC microcontroller

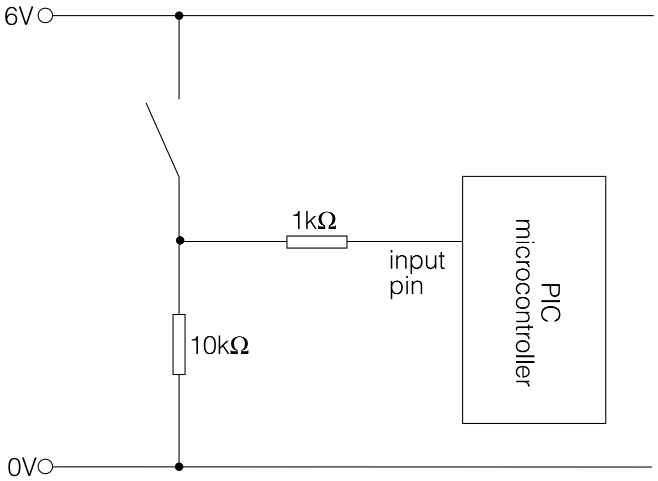
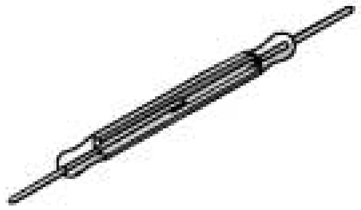


Fig 5.1 Connection of a switch

Microswitch

Push Switch



Reed (Magnetic) Switch

B. Analogue Sensors

B1. Potentiometer

Connect the wiper directly to an analogue input pin on the PIC microcontroller as shown in fig. 5.2.

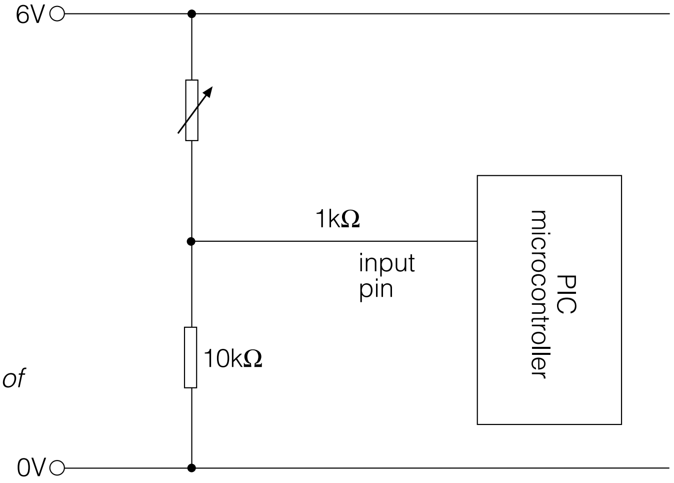
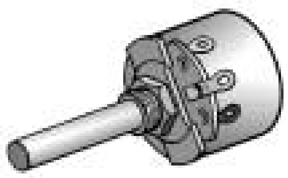
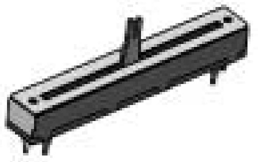


Fig 5.2 Connection of a potentiometer.



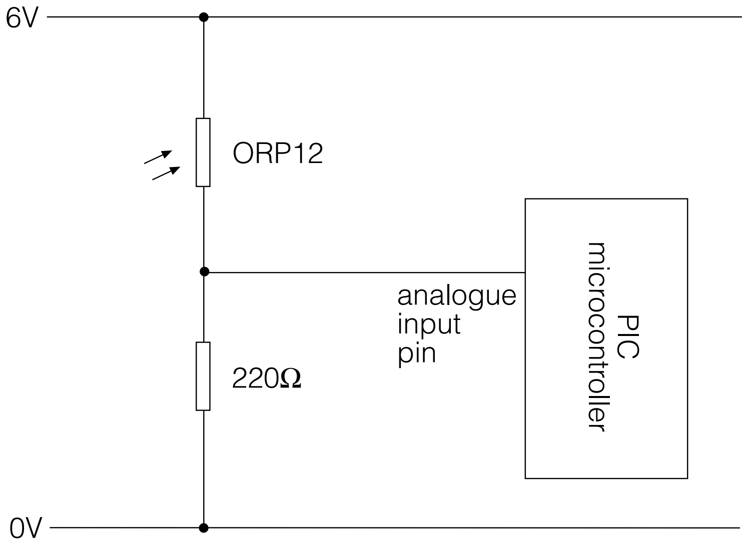
Rotary potentiometer

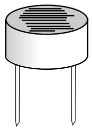


Linear potentiometer

B2. Light sensor

Fig 5.3 shows how a light sensor can be made by connecting a light dependent resistor (LDR) and a fixed resistor to a PIC microcontroller. For this circuit, ambient light (inside) will be around the 80 level on the scale, with dark being 0 and bright sunlight taking the reading up to 255.



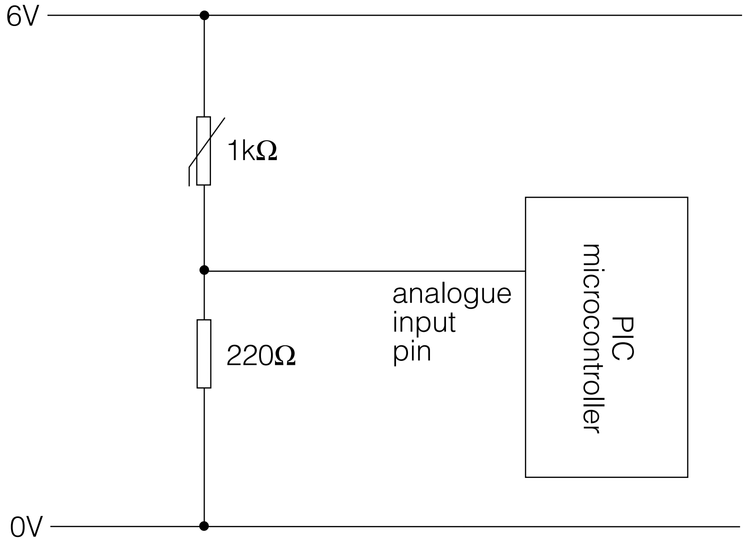


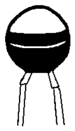
Light dependent resistor

Fig 5.3 Connection of a light sensor

B3. Temperature sensor

A temperature sensor is made by connecting a thermistor and fixed resistor to a PIC microcontroller as shown in fig 5.4. In this circuit, a temperature of about 20˚C will give a reading of about 30 on the PIC-Logicator scale and will drop to nearly zero at 0˚C.





Thermistor

Fig 5.4 Connection of a temperature sensor

To avoid possible interference from sensors to other analogue inputs, it is a good idea, when designing your circuit, to connect any unused analogue inputs to 0 volts (ground).

In this type of simple circuit, the sensors do not respond linearly. Some experimentation is required for your own application to ensure that your PIC microcontroller is responding at the correct levels. To be sure of the actual level that the PIC microcontroller is reading from the input, use the PIC-Logicator Analogue Calibration Board (see page 24) which is a valuable development tool because it displays the actual reading that is being obtained from the sensor.