A guide to using PIC-Logicator Version 2 software and connecting to a PIC microcontroller

© Copyright Economatics (Education) Ltd. 1999-2004. PICAXE technology and portions of this document are © Copyright Revolution Education Ltd. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form by any means, without prior permission of the copyright holder. Copyright is waived in the following circumstances: small number of copies may be made for use only in the purchaser's school. These copies may not be sold or made available outside the purchaser's school.

L300-01

Contents

Getting Started

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

Version 2 of the PIC-Logicator software is built on the Microsoft.NET framework and as such will require this to be installed prior to running the software. The .NET framework is included on the PIC-Logicator CD and will normally install automatically. Please note that PIC-Logicator version 2 is not compatible with Windows 95.

RM CC3 Installation

On the CD you will find a folder called 'RM CC3 Package' and within this, a zip file and instructions.

Because PIC-Logicator requires the .NET framework, it is necessary to allocate both the PIC-Logicator package and the .NET framework package to every workstation.

Website

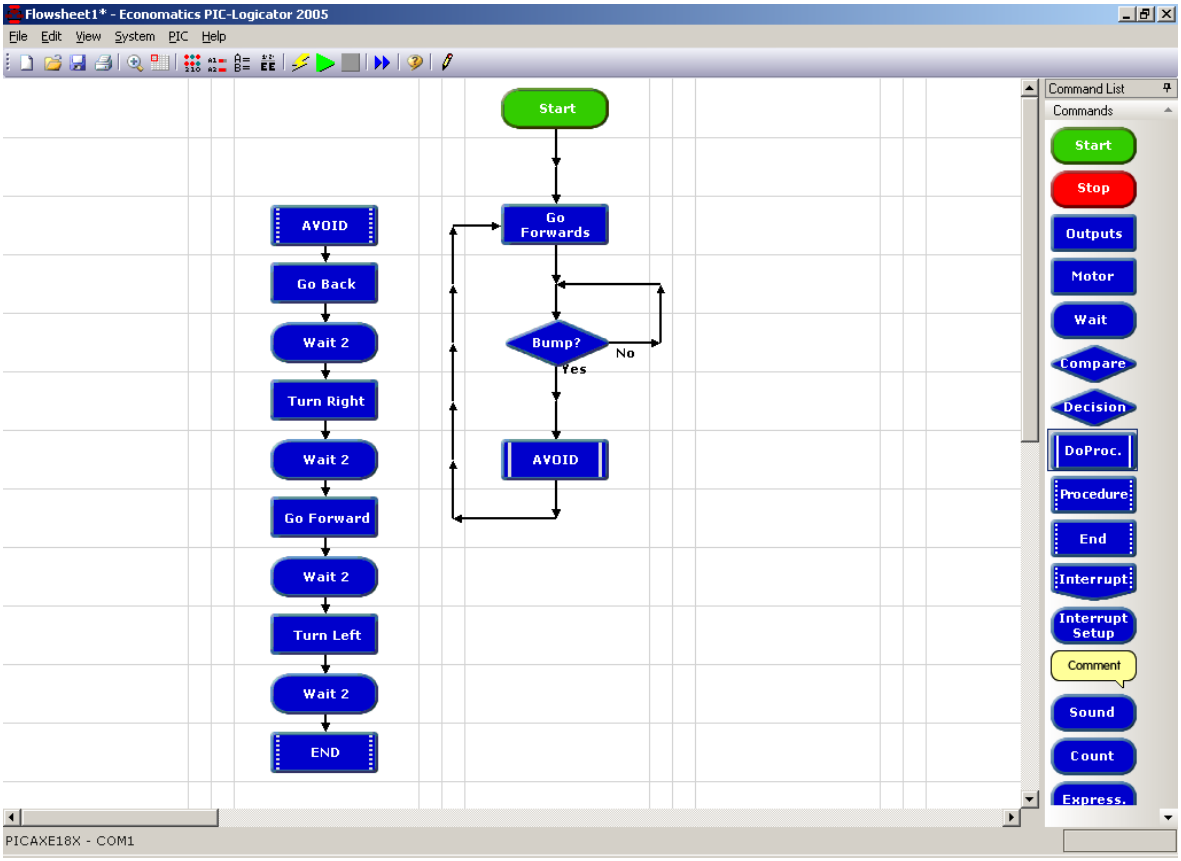
The Economatics website provides the latest PIC-Logicator news and information. It includes: FAQs, projects and programming ideas, PCB designs to download, and an interactive demo of PIC-Logicator software.

www.economatics.co.uk/education

Support

The PIC-Logicator user group on the Economatics Education website contains most of the information that you will need in order to solve any technical issues. Once registered, you can post and contribute information to the group to help you and other users of the products gain help.

The PIC-Logicator help files contain frequently asked questions and solutions to common problems. The help can be found by pressing F1 in the PIC-Logicator software or from the Help>Contents menu.



Section One:

PIC-Logicator Software Version 2

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 Decision command;
how to use the Interrupt command for instant response to digital sensors;
how to use the Compare command to make use of readings from analogue sensors connected to a PIC microcontroller, in a control system.

4. Procedures

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; 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 Decision (see the Index of Commands page 6). 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 Setup Command List menu to customise the list (right click on the list), by changing the number of commands included, and the sequence in which they appear.

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

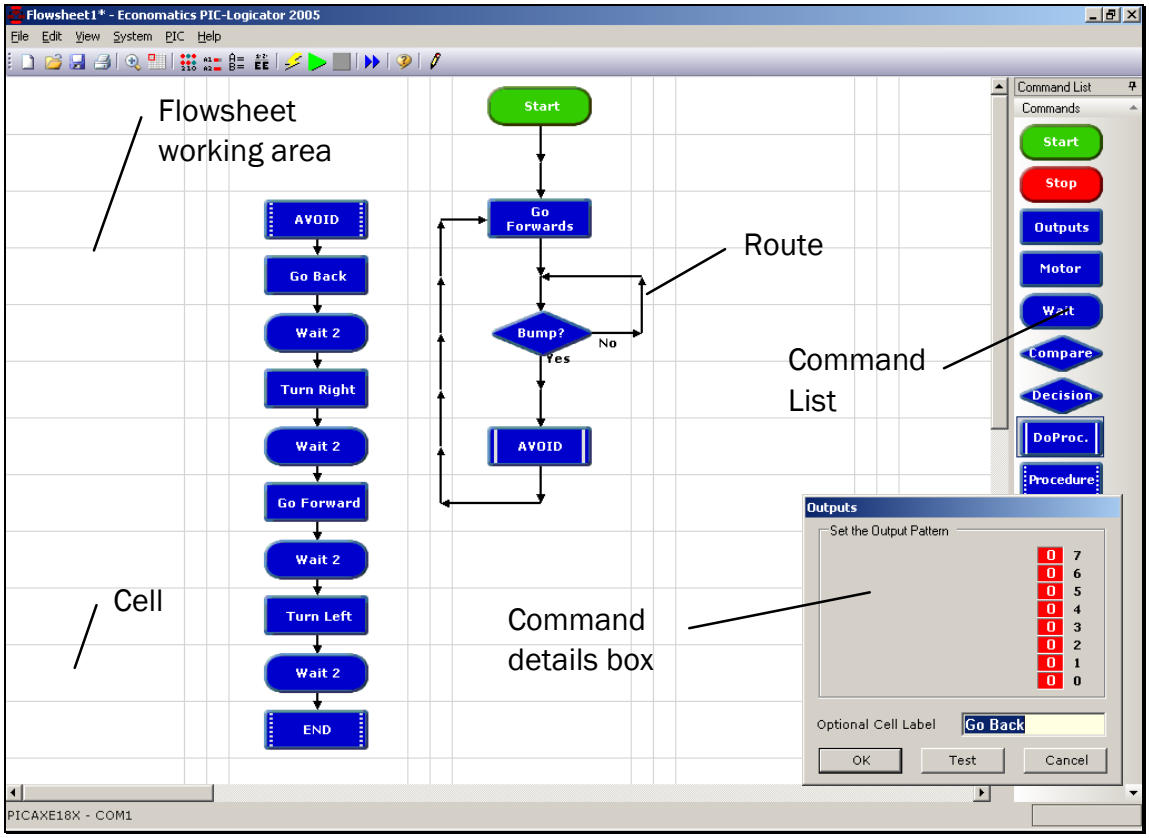


fig. 1.1 The PIC-Logicator screen.

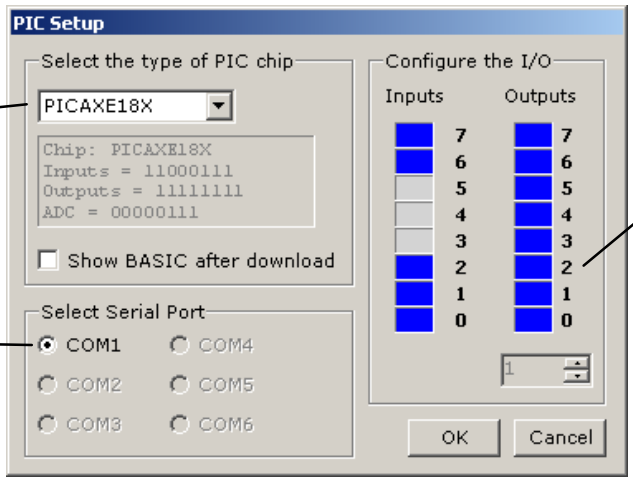
In PIC-Logicator, you create your control system in the form of a flowchart by dragging commands from the Command List and placing them in cells on the flowsheet working area (See fig. 1.1).

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.

Select PIC Type

Before you begin to build a flowsheet, you should decide which PIC microcontroller chip you intend to use in your project. Select the chip from the PIC> Select PIC Type menu.



PIC Type

Set the COM port that your chip project or programmer is attached to

Available inputs & outputs are shown in blue

For 8 pin PICAXE devices, you must also configure the input/output options using the up/down selection box. Because 8 pin chips actually only have 5 pins that are available to use as inputs or outputs, these can be configured as such to suit your project. Pin 4 and Pin 7 are fixed as Input 3 and Output 0 respectively, but all other combinations are available. Note that on the PICAXE08, the only analogue input is on Input 1, so if you wish to use analogue inputs with your PICAXE08 you must setup Input 1 to an input.

When you select a chip, the software automatically configures itself to display only the input, output and motor options available with that chip.

Memory Use

The amount of memory available in the PIC chip you have chosen for your project is an important consideration when designing a flowsheet.

Most commands use similar amounts of memory, but this does vary. PIC-Logicator provides two helpful tools to help you understand how much memory your flowsheet has used.

While you are designing a flowsheet, clicking PIC>Update Memory Use (ALT-F3) will re-calculate an *estimate* of the percentage memory used by your flowsheet. This is displayed as a bar graph in the lower right corner of the PIC-Logicator window.

The bar fills with colour from left to right, for example:

 Less that 5% memory use.

 Approximately 40% memory use

 Memory Full.

When using PICAXE type chips, the actual memory used *after* download is available, and is show on the status bar, below the flowsheet area in PIC-Logicator. Note that you must download your flowsheet into a PICAXE to get this information displayed.

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 6.

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.

You can only use one Start and one Stop command in any flowsheet.

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. his 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.

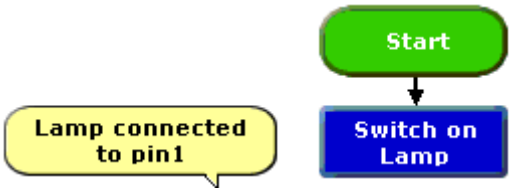


Fig 1.2 Explanatory information can be added to the flowsheet by using command labels and Comment commands.

Selecting a block of commands

Click on the top left corner of the block of cells. Hold down the Control Key (CTRL) and click on the lower right corner of the range of cells.

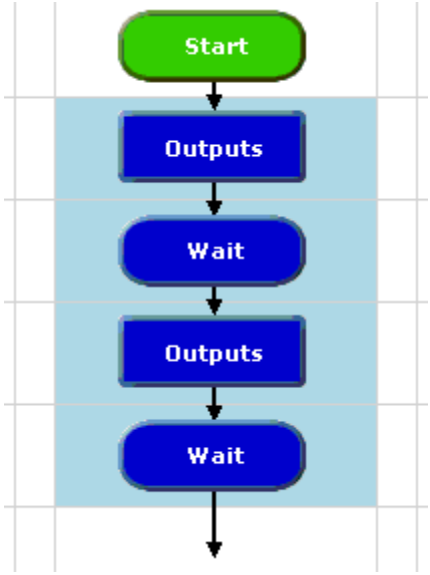


Fig 1.3. A block of commands in the selection frame.

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

Deleting a command

Click on the command to select it. Selected commands are coloured light blue. 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 or a block of commands, select the area and drag it to its 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 22 columns and 25 rows. The default screen shows just 12 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.

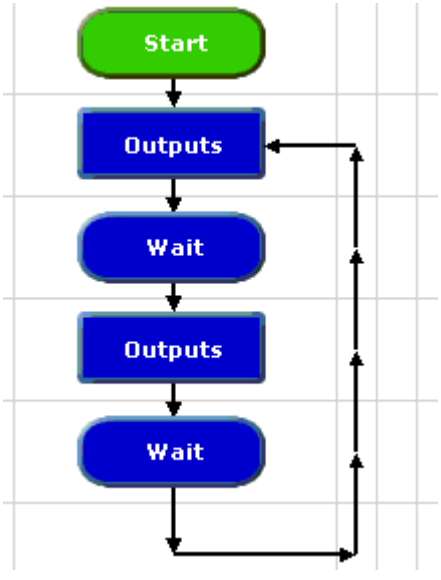


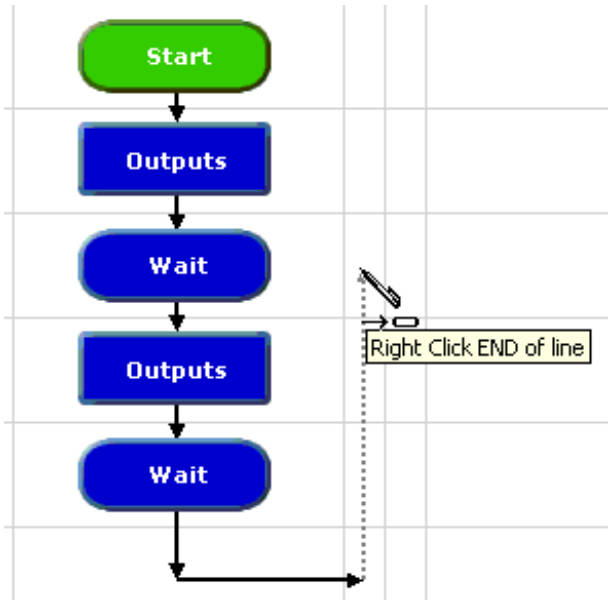
Fig 1.4. Routes can be drawn through cells or between rails.

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 Lines

Click on the Line Drawing icon on the toolbar.  The mouse cursor changes to a pen icon. Click with the right mouse button where the line should start. Right click at the end point of the line.



Lines can only be drawn vertically or horizontally. Always draw the line in the direction of the flow, as indicated by the arrows.

By holding down the Control key, the arrow keys can also be used to draw lines.

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.

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

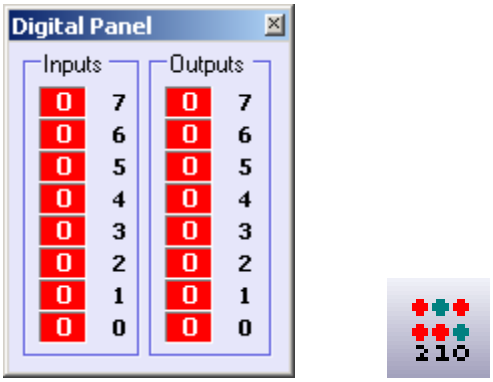


Fig 1.5. Digital Panel and its toolbar icon

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).
Clicking on the corresponding input or output on the digital panel will also have the same effect.

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.

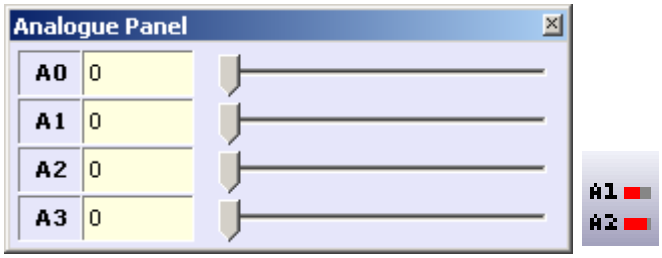


Fig 1.6. Analogue Panel and its toolbar icon

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 System>Run menu or the green toolbar icon. To stop a flowsheet running, click the System>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>Run 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.

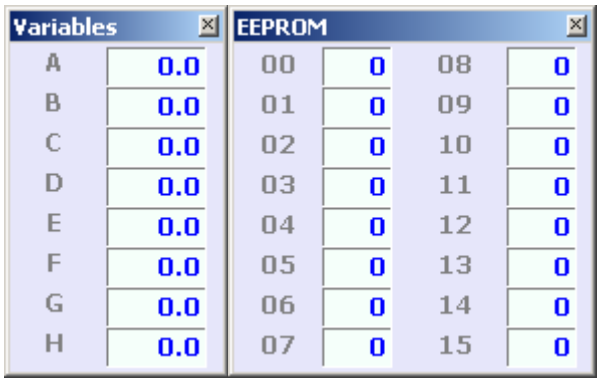


Fig 1.7. The Variables window and the EEPROM window.

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

Displaying and using BASIC

PIC-Logicator is able to convert any complete flowsheet into BASIC.

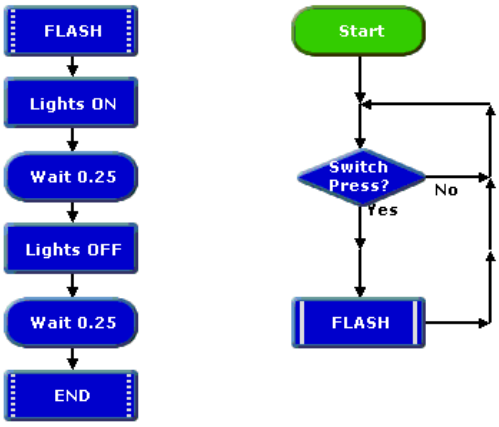
BASIC is a text based language that is used throughout the world to program everything from PIC microcontrollers to Personal Computers.

Why Convert?

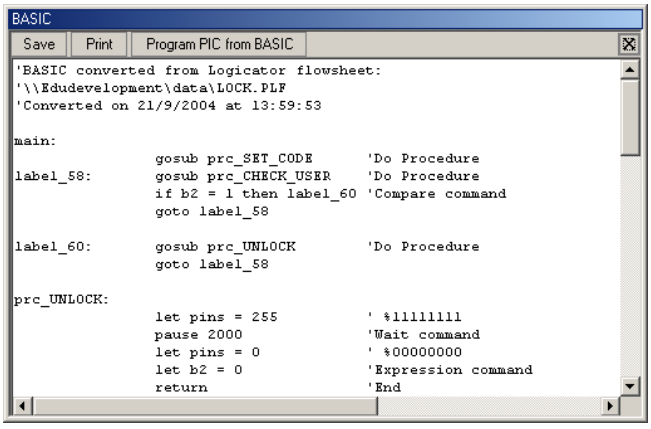
PIC-Logicator flowsheets are easy to understand and quick to build. BASIC programming languages offer more complexity to advanced level users and the ability to covert a flowsheet into BASIC offers a way of learning how BASIC programs are written.

Converting a flowsheet into BASIC

1. Design your flowsheet as normal and test the program using the flowsheet simulation tools provided in PIC-Logicator.



2. From the PIC menu, choose Convert flowsheet to BASIC.



3. The Flowsheet BASIC Conversion window is displayed containing the conversion of your flowsheet.

Notes:

Only commands that are in the flow of your program are converted.

Code in the Flowsheet BASIC Conversion window can be edited and then re-programmed into the selected type of PIC.

Converting the PIC-Logicator flowsheet into BASIC always overwrites any changes made to the code inside the BASIC Conversion window.

It is not possible to convert from BASIC to a flowsheet.

You can use the mouse to select text in the code window. Right click on the selection to copy the selection to the clipboard. The code can then be pasted into software such as Revolution's Programming Editor.

Using the BASIC command in PIC-Logicator you can add sections of BASIC code into a flowsheet. Whilst this is not simulated in the PIC-Logicator software, you can make use of BASIC code that you might have available. See the PIC-Logicator help for full information on the BASIC command.

For full information on the use of BASIC to program PIC chips see the PICAXE website at www.picaxe.co.uk.

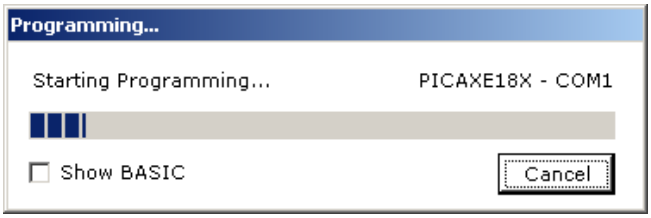
Downloading a flowsheet

PIC-Logicator can program either PICAXE chips direct from the computer (via a download cable) or a PIC chip via the PIC-Logicator programmer.

Do not attempt to program PICAXE chips through the PIC-Logicator programmer unit. Doing so *may* risk erasing the bootstrap code from the PICAXE chip.

Programming a PICAXE chip

1. Connect your PICAXE project to the serial port of the computer by the download cable.
2. Connect power to the PICAXE circuit board.
3. Note; your PICAXE chip, if already programmed may start running the program from its memory – this will not affect the programming process.
4. Click the Program PIC button on the toolbar  or PIC>Program PIC menu option.
5. The programming progress window will appear.
6. Programming times vary depending on the type of chip and amount of program code – the larger the flowsheet, the longer the programming time.
7. If successful, programming is complete when the progress window disappears.



Programming process window

Using the PIC-Logicator 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 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.

Outputs

Outputs command

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

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

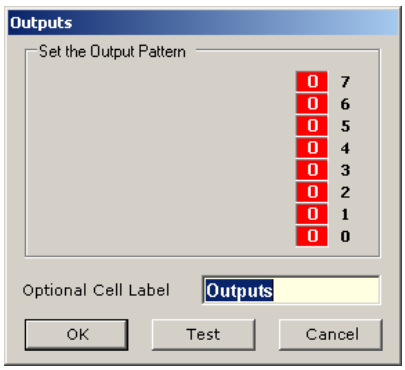


Fig 2.1. Outputs command Cell Details box

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.

1 This means: switch this output on.
0 This means: switch this output off.
- This means: ignore this output. Leave it in the state in which it was set by the previous Outputs command.

Wait command

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 65s. Min 0.001s) or a Variable.

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.

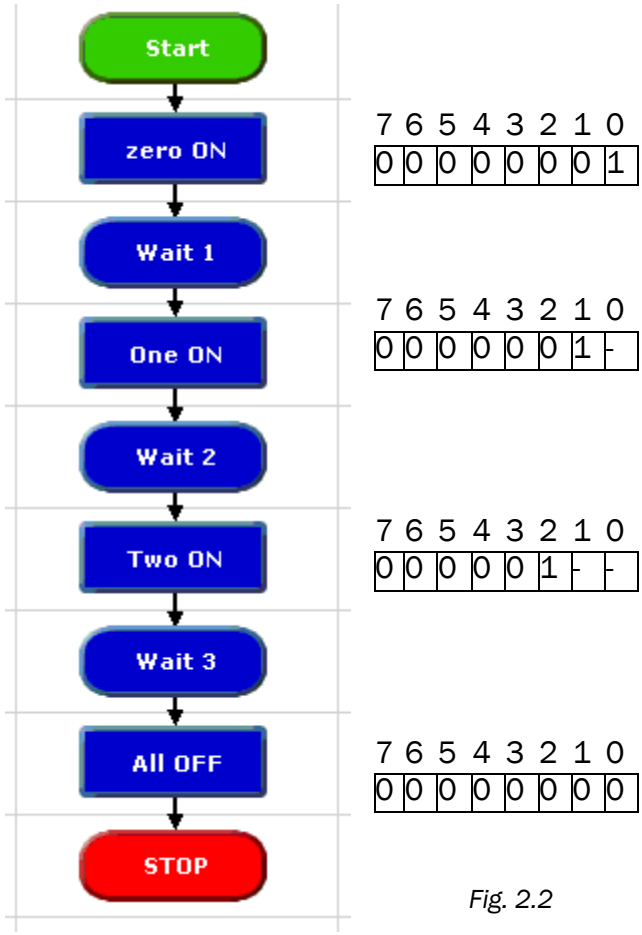


Fig. 2.2

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.

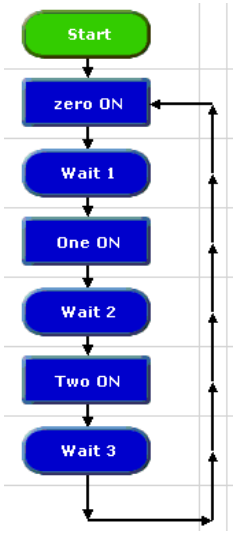


Fig 2.3.Repeating timed sequence.

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.

If you are familiar with the binary system then the Out command is a convenient way of switching combinations of outputs on or off.

Sound Command

Use a 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 simple tune.

There are two ways to view the cell details for the Sound command. Simple allows the setting of note, time and pin from drop down lists (Figure 2.4), whereas the Advanced option allows the selection of note via a 'keyboard' style layout (Figure 2.5).

The notes used in PIC-Logicator are true sampled wav files, but the notes played by the PIC may vary due to the limitations of the pulsing of the piezo sounder.

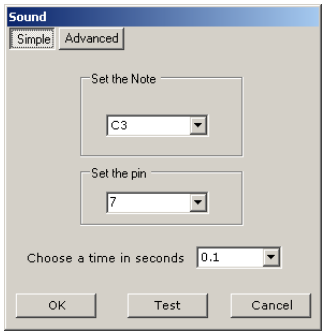


Fig 2.4. Simple Sound command cell details

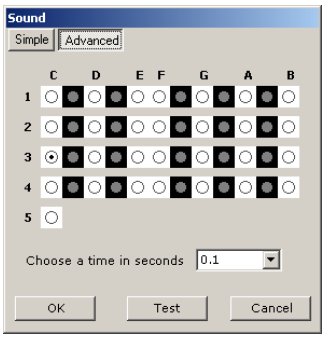


Fig 2.5. Advanced Sound command cell details

Motor command

The Motor command allows you to use pairs of 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 direction arrows indicate only that the directions will be different; not the actual direction in which motors in the project will turn.

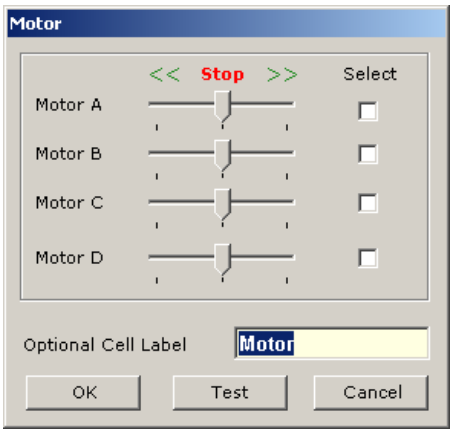


Fig. 2.8. Motor command Cell Details box.

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 55).

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 'on'.

So, to avoid inadvertently switching off an output device, un-check the select 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).

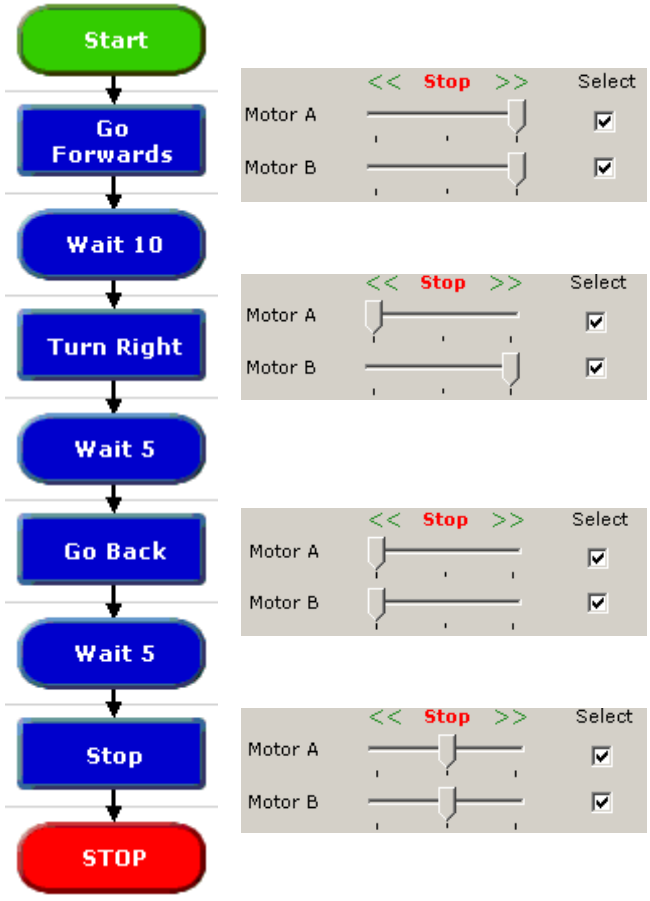


Fig 2.9. The Motor commands have been given labels to show what they do. The table beside each one shows how its Cell Details have been set.

Sleep command

This command puts the PIC microcontroller into low power mode for a specified number of seconds.

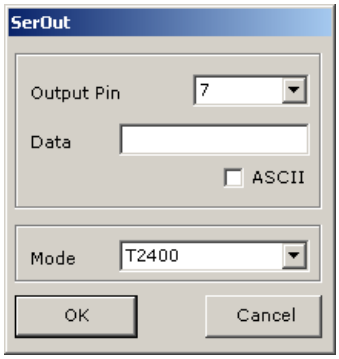
You can use this command to save battery power in your project. All output devices will be left in their current condition, but signals from input devices will not be responded to while the chip is in sleep mode.

Use the Cell Details box to set the number of seconds of sleep mode you require (this is in the form of number of multiples of 2.3 seconds). For example, a setting of 10 will sleep for 23 seconds.

Note that Sleep times are not as accurate as Wait times.

SerOut Command

This command allows you to output information from the PIC microcontroller to a device such as a serial printer, a serial LCD screen or another PIC which is connected to an output of a PIC microcontroller.



Use the first box to select the output pin on the PIC microcontroller to send the data through.

In the Data box either type in the ASCII text you wish to send or raw data. If sending raw data codes you must un-check the ASCII box.

ASCII codes are useful for sending commands to LCD screens e.g. clearing the display. Details of these control codes can normally be found with the instructions for the particular devices.

You can send a series of text characters e.g. “Hello” or a series of ASCII codes e.g. “254,1”. In the latter case, ASCII codes must be separated by a comma.

If you wish to send the value held in a variable, type in the variable name in square brackets e.g. “[B]”. Note you must use capital letters for the variable.

The last item to set is the serial mode. Set the mode to that specified by the device you are sending data to.

Example

The flowsheet shown in fig 2.11 will display the word “Hello” on an LCD screen connected to output pin 7 of a PIC microcontroller.

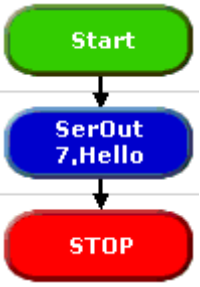


Fig 2.11. A sequence to display the word Hello.

Servo Command

Servos, as commonly found in radio control toys, are a very accurate motor/gearbox assembly that can be repeatedly moved to the same position due to their internal position sensor. Generally servos require a pulse of 0.75 to 2.25ms every 20ms, and this pulse must be constantly repeated every 20ms. Once the pulse is lost the servo will loose its position.

The Servo command starts a pin pulsing high for length of time pulse (x0.01 ms) every 20ms. This command is different to all other commands in that the pulsing mode continues until another servo command or outputs command. Outputs commands stop the pulsing immediately. Servo commands adjust the pulse length to the new pulse value, hence moving the servo.

The cell details for the servo command (fig 2.12) has two settings; the output pin that the servo motor is connected to and the pulse time.

The pulse time can be a value held in a Variable. Note that the value for the pulse time MUST be in the range 75 to 225. The servo motor may malfunction if the pulse is outside of this range.

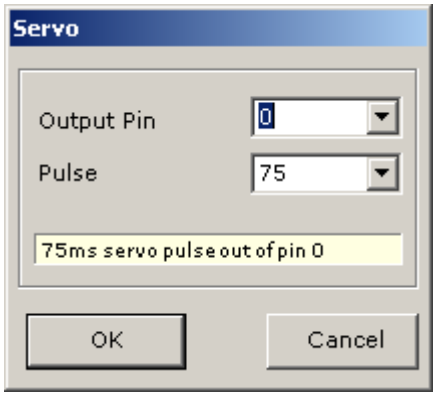


fig 2.12 Servo command cell details.

Example

The flowsheet in fig 2.13 will move a servo motor attached to output 0 from one extent of its travel to the other, repeating continually.

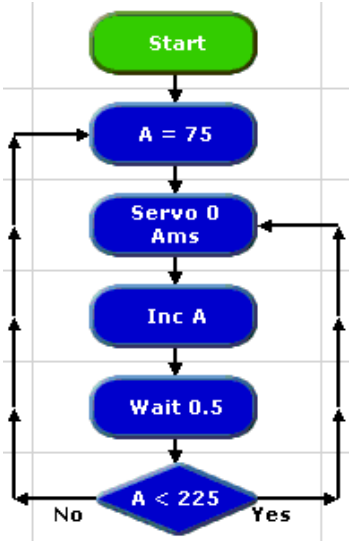


fig 2.13. Using the Servo command.

Always use a separate 6V (e.g. 4x AA cells) power supply for the servo, as they generate a lot of electrical noise. See section 2 of this manual for more information on connecting servo motors to PIC microcontrollers.

Note that the processing time required for processing the servo commands every 20ms causes the other commands to be slightly extended i.e. a pause command will take slightly longer than expected. The servo pulses are also temporarily disabled during timing sensitive SerIn and SerOut, commands.

PulseOut Command

The PulseOut command generates a pulse through the chosen output. If the output is initially off, the pulse will be on, and vice versa.

There are two items to set in the cell details box for the PulseOut command (fig 2.14); the output pin to send the pulse through, and the length of time that the pulse should exist for.

The time is in 10µs intervals, but for easier reading, the text area in the command converts this to milliseconds as the time is entered into the command. PulseOut times must be in the range 1 – 65535.

Note that PIC-Logicator cannot simulate the action of the PulseOut command.

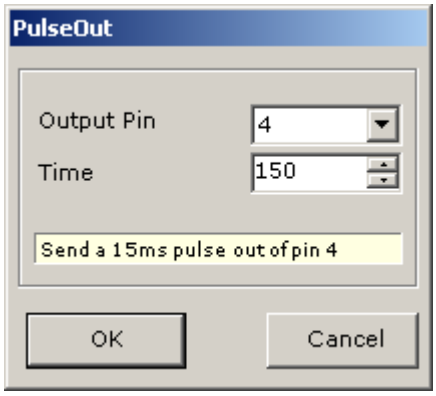


fig 2.14. PulseOut cell details box.

Example

The flowsheet in fig 2.15 sends a pulse of 15ms out of output pin 4 every half second.

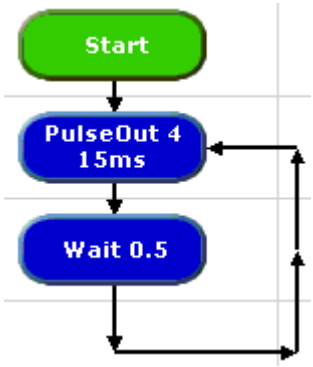


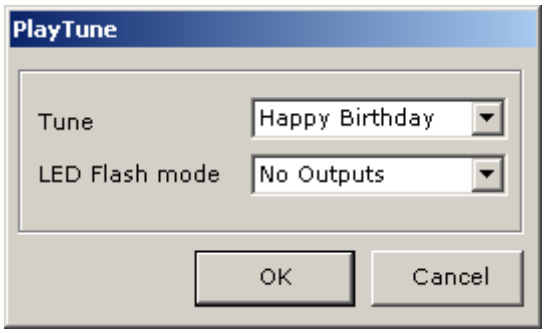
fig 2.15. Using the PulseOut command.

PlayTune command

The PICAXE-08M can play musical tones. The PICAXE-08M is supplied with 4 pre-programmed internal tunes, which can be output via the PlayTune command.

As these tunes are included within the PICAXE-08M bootstrap code, they use very little program memory.

The cell details require that the number of the tune is set and if you wish the outputs to flash in time to the tune.



The Tunes are:
- 0 - Happy Birthday
- 1 - Jingle Bells
- 2 - Silent Night
- 3 - Rudolf the Red Nosed Reindeer

The Flash modes are:
- 0 - No outputs
- 1 - Output 0 flashes on and off
- 2 - Output 4 flashes on and off
- 3 - Output 0 and 4 flash alternately

The following example will play Jingle Bells while flashing output 4.

PIC-Logicator cannot accurately simulate the flashing actions of the PlayTune command.

It is possible within the Programming Editor software available from Revolution Education, to program your own tune into a PICAXE08M.

Play User Tune Command

Working in a similar way to the PlayTune command, the Play User Tune allows special musical tunes to be played using the PICAXE08M.

The difference with the Play User Tune command is that it converts RTTTL mobile phone ringtone files to PICAXE tunes and plays them with or without flashing outputs.

RTTTL ringtone files are freely available on the internet (there is a very wide range of tunes available) and these can be downloaded as small text files. The files contain the notes and timings that make up the tune. PIC-Logicator converts these ringtones to a PICAXE tune upon download.

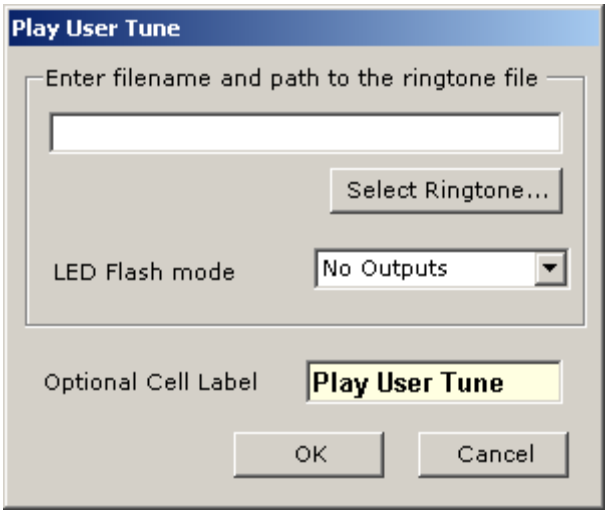


figure 2.16, Play User Tune cell details box.

Once you have downloaded your ringtone file (ensure it is an RTTTL format), save it to disk and open the cell details box for the Play User Tune command (see fig. 2.16).

Click the 'Select Ringtone...' button to browse the computer to find the file.

Select the LED flash mode that you require using the drop down box. The chosen outputs switch on/off in time to the tune. The Flash Mode can switch outputs 0 and 4. Ensure that you have configured the I/O pin 4 as an output using the Select PIC dialog in order to see all of the available options.

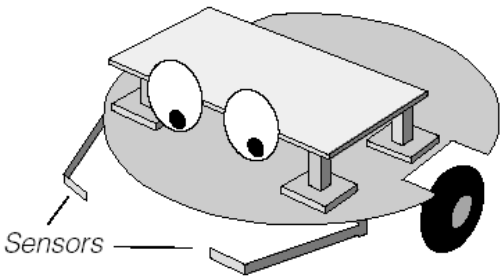
Note that unlike the PlayTune Command, the User Tune requires much more memory in the chip as all of the notes have to be specially programmed into the chip. If wanting to play your tune a number of time, use the Play User Tune command in a Procedure to save memory.

Inputs

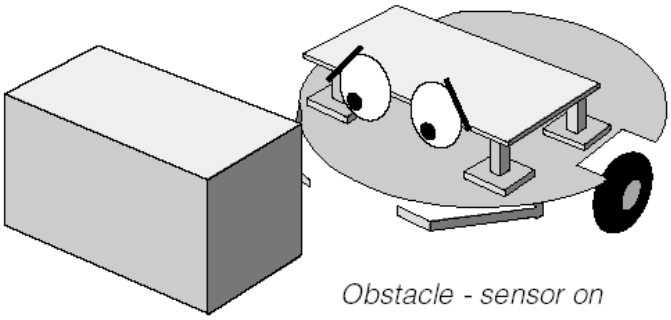
Input devices such as switches and sensors send information from the outside world into the control system. Output devices are switched on or off in response to the information provided by input devices.

Example one

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



No obstacles - sensor off



Obstacle - sensor on

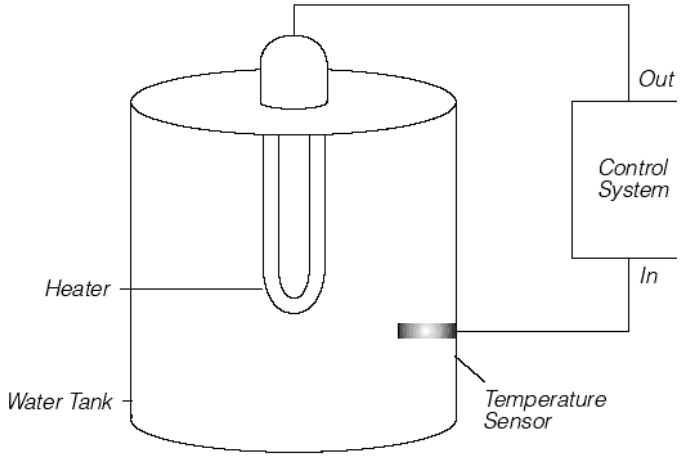
The information that the switch has been pressed can be used in the system to switch off the motors driving the buggy, and start a sequence of movements to move around the obstacle.

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.

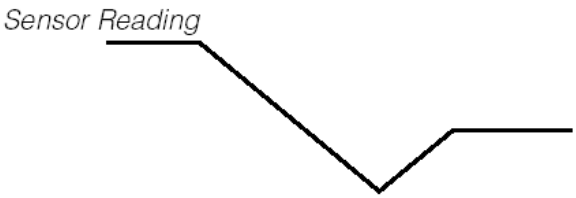
Example two

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

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 Yes or No direction depending on the result of the decision test. See fig 3.1.

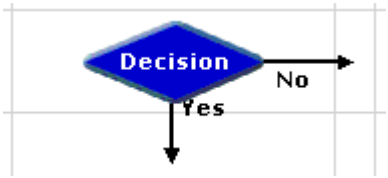


Fig 3.1. This Decision command is testing the state of a microswitch. If the switch is pressed, flow will go in the Yes route; if it is not pressed, flow will go in the No route.

The Cell Details box of the Decision command is shown in fig 3.2. The Input Pattern area shows the number of digital inputs available for use on the PIC microcontroller you have selected. Any unavailable inputs are shown without a number label and cannot be clicked upon.

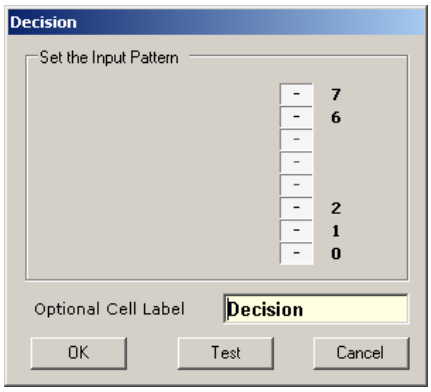


Fig 3.2. Decision Cell Details box for a PICAXE18X chip (five inputs).

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?
0 This means is this sensor OFF?
- This means ignore this sensor.

Drawing routes from a Decision command

The first line that you draw from a Decision command is the "Yes" direction, and the second line is the "No" direction.

Tip; you can swap the "Yes" and "No" routes by right clicking on the Decision and choosing "Swap Yes/No".

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 Yes 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.

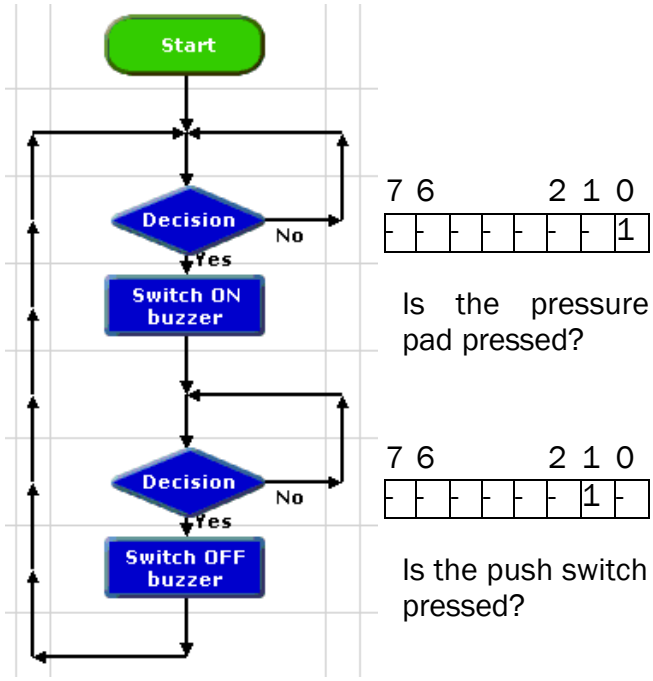


Fig 3.3. Security system.

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.

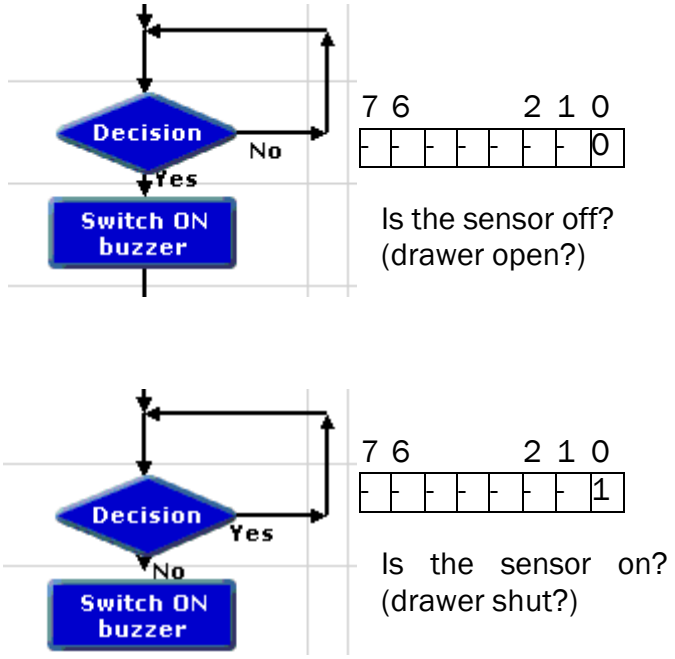


Fig 3.4. Notice that the direction of flow depends on how the command is set.

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.

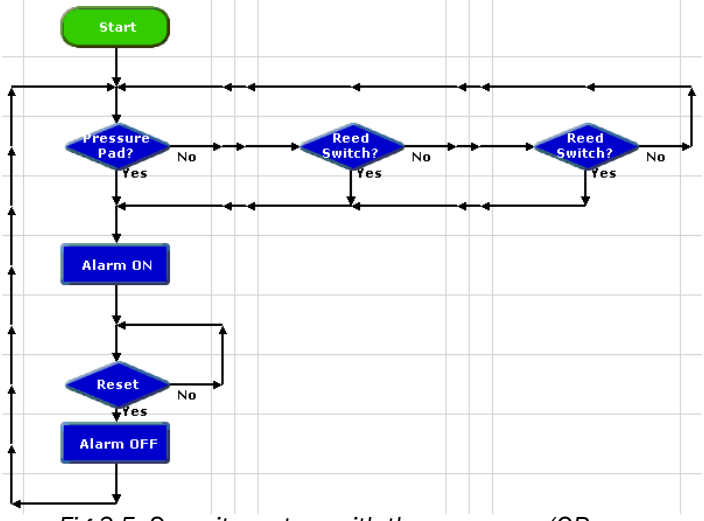


Fig 3.5. Security system with three sensors (OR function).

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 Yes 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.

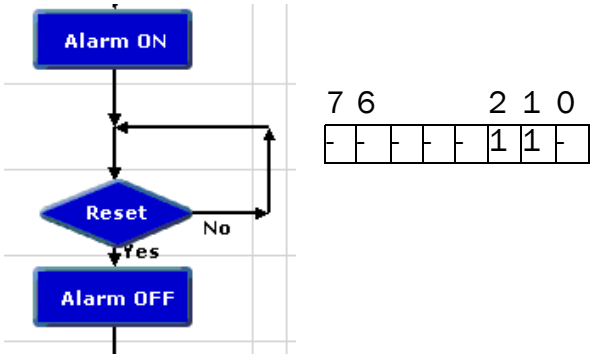


Fig 3.6. Decision command set to check if two switches are pressed at the same time (AND function).

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.

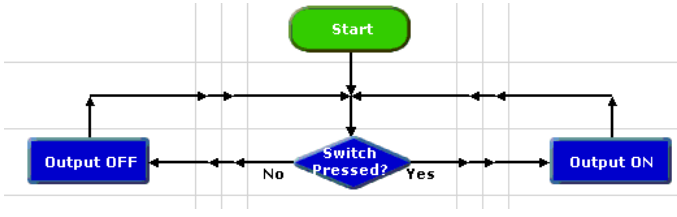


Fig 3.7 "Normally open" switch effect.

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 Yes and No on the Decision command as shown in fig 3.8.

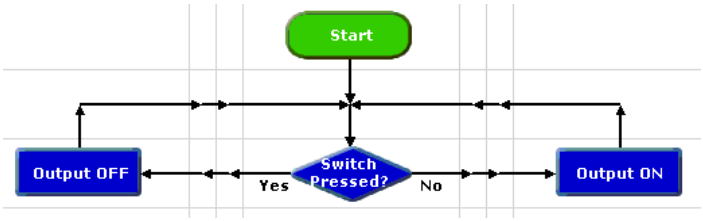


Fig 3.8 "Normally closed" switch effect.

Example four

A mono-stable 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.

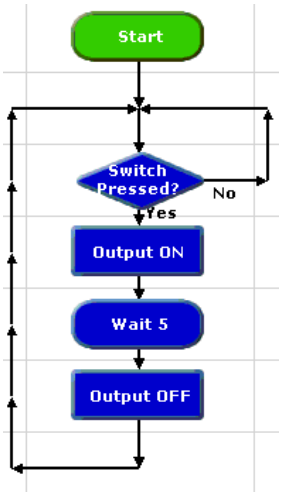


Fig 3.9 "Mono-stable" function.

Example five

A bi-stable 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.

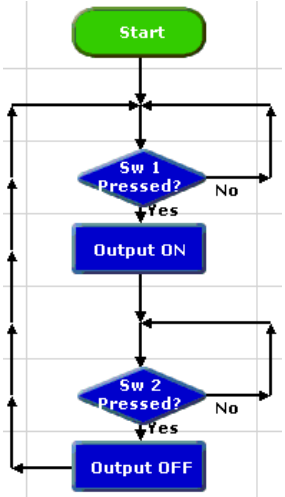


Fig 3.10 "Bi-stable" function using two switches for set and reset.

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 un-pressed 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.

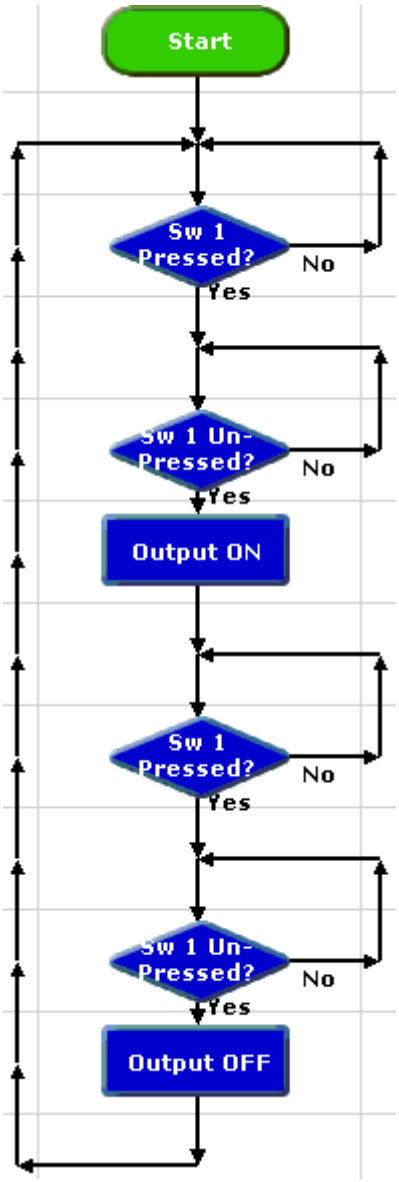


Fig 3.11 "Bi-stable" function using one switch for both set and reset

Interrupt (PICAXE only)

An Interrupt instantly captures the flow of control whenever a preset digital input condition occurs to trigger it e.g. when a switch is pressed.

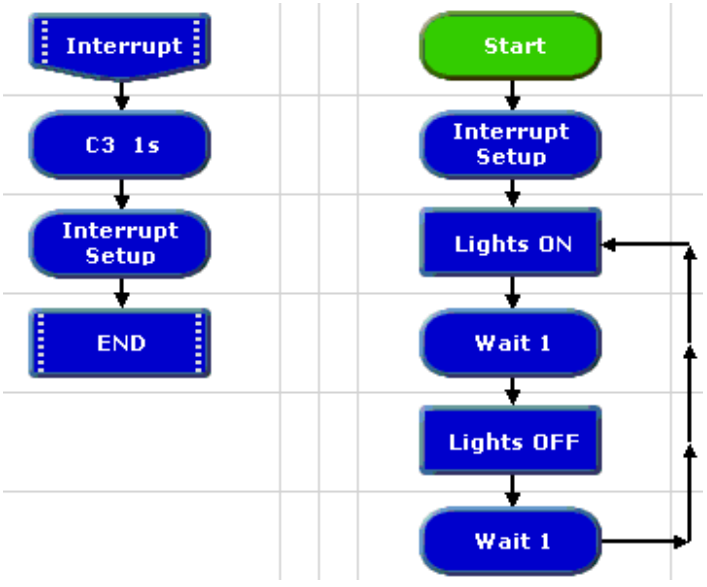
When the interrupt is triggered flow jumps immediately to the Interrupt command and then carries out any commands which follow until it reaches an End command. It then returns to the point which it was at when the Interrupt occurred.

In order to use an Interrupt, the PIC must be told to look for the input condition. This is done through the Interrupt Setup command. There are two options in the command – Enable or Disable.

To prevent the Interrupt re-triggering itself, the Interrupt is automatically disabled once it is triggered. To re-enable it another Interrupt Setup command is required.

Example

A PIC microcontroller running a continuous loop flashing lights needs to be able to react to a button press and play a warning sound.



The Interrupt is used to capture the flow and play a sound. The interrupt is then enabled once again before returning to the point at which it left the main flow.

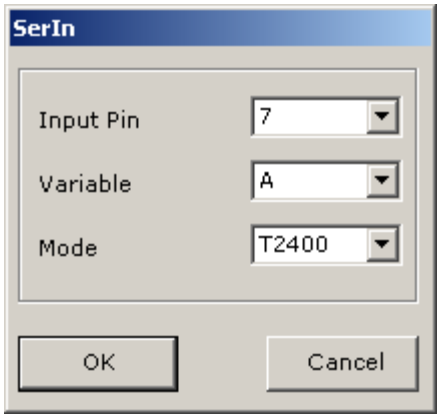
Note that the Interrupt MUST have an associated End command and will not be triggered again until this End command has been reached. There is no limit to the number of commands between the Interrupt and the End.

Only one Interrupt can be used per flowsheet.

SerIn Command

The SerIn command is a PICAXE only command used to receive serial data into an input pin of the microcontroller. It cannot be used with the serial download input pin, which is reserved for program downloads only.

The cell details box for the SerIn command has three boxes to set.



SerIn command cell details box.

The input pin is the input on the PICAXE that the data is to be received through. The Variable option is a variable location that the data is stored into once it is received.

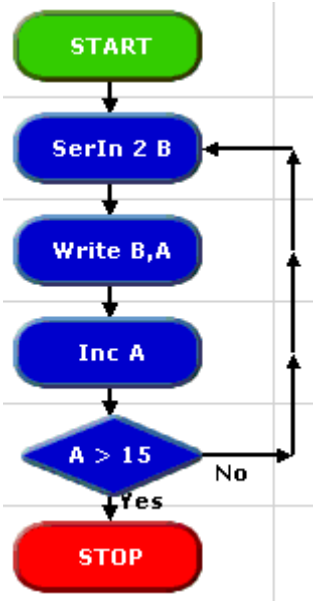
Lastly, the mode option specifies the baud rate and polarity of the signal. When using simple resistor interface, use N (inverted) signals. When using a MAX232 type interface use T (true) signals. The protocol is fixed at N,8,1 (no parity, 8 data bits, 1 stop bit).

For best results do not use a baud rate higher that 2400 on 4Mhz chips.

The SerIn command forces the PICAXE chip to wait until serial data is received through the chosen input. This data is stored in the chosen variable.

Example

Serial data is being received from another PIC chip and needs to be stored in the EEPROM.



Using the SerIn command to receive serial data.

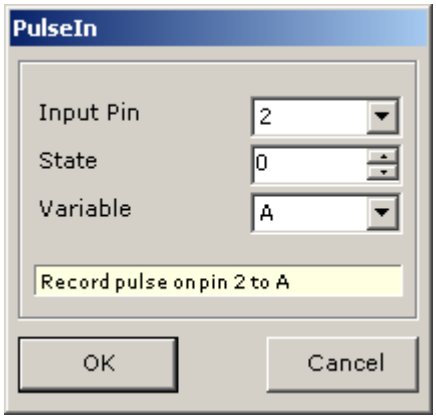
In the flowsheet shown above, the serial data is read into Variable A through input pin2. The Write command is used to store the value in Variable A in the EEPROM. This process is repeated 16 times to fill all the available EEPROM memory locations.

PulseIn Command

The PICAXE only command, PulseIn measures the length of a pulse through an input pin. If no pulse occurs within the timeout period, the result will be 0.

If State = 1 then a low to high transition starts the timing, if state = 0 a high to low transition starts the timing.

There are three items to set in the PulseIn command; the input pin, the State and the Variable to store the result in. The result is measured in multiples of 10µs and is in the range 1 – 255.



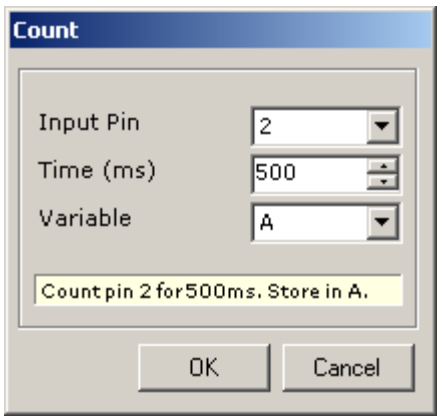
Cell details box for the PulseIn Command

Use the Count command to count the number of pulses with a specified time period.

Because the PulseIn Command works so quickly this command cannot be simulated in the PIC-Logicator software.

Count Command

The PICAXE only command, Count, is available on the PICAXE08M, 18X and 28X. The Count command checks the state of the input pin and counts the number of low to high transitions within the time 'period'. Up to 255 transitions can be counted.



The cell details box for the Count command.

Take care with mechanical switches, which usually cause multiple 'hits' for each switch push as the metal contacts 'bounce' upon closure.

Analogue Inputs

If you want to use analogue sensors in a project, you must use a PIC microcontroller that has analogue inputs (see page 58). All PICAXE chips have at least one analogue input.

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 58, 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.

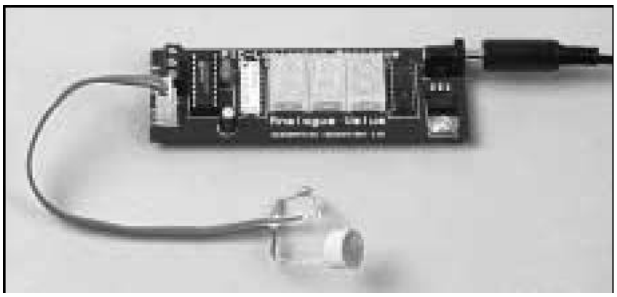


Fig 3.15 PIC-Logicator Analogue Calibration Board with light sensor connected.

Compare command

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 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 "Yes" or "No" direction depending on the result of the comparison. The Cell Details box of the Compare command is shown in fig 3.16.

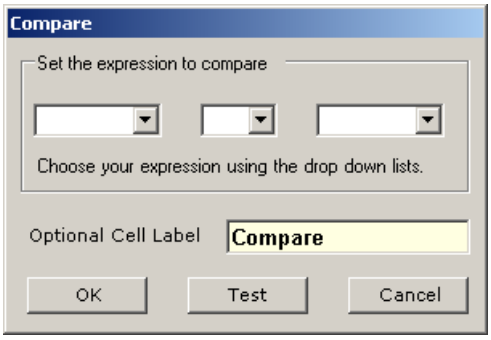


Fig 3.16. Cell Details box of the Compare command.

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.
3. 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.

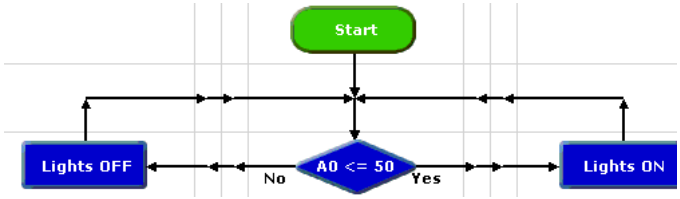


Fig 3.17 System to switch on a lamp automatically in dark conditions.

The Compare command is checking the reading from the light sensor. If the reading is less than or equal to 50, flow will go in the Yes 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.

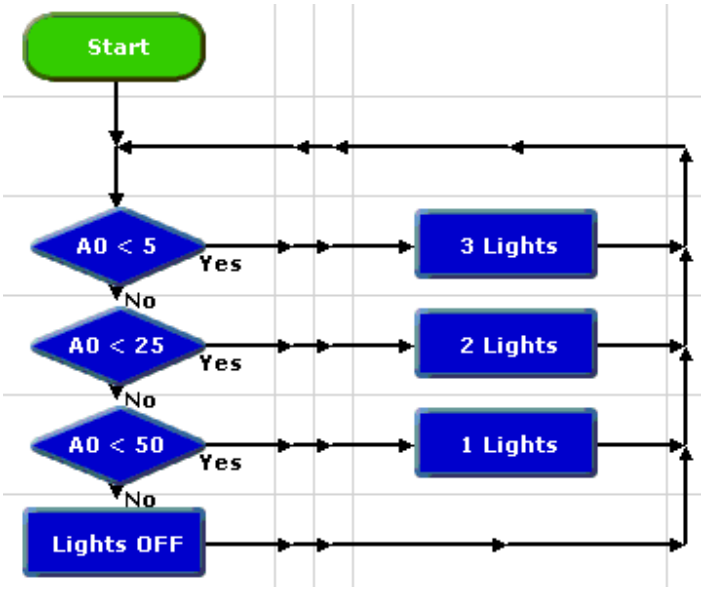


Fig 3.18. System to switch on three lamps in response to changing light levels.

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.

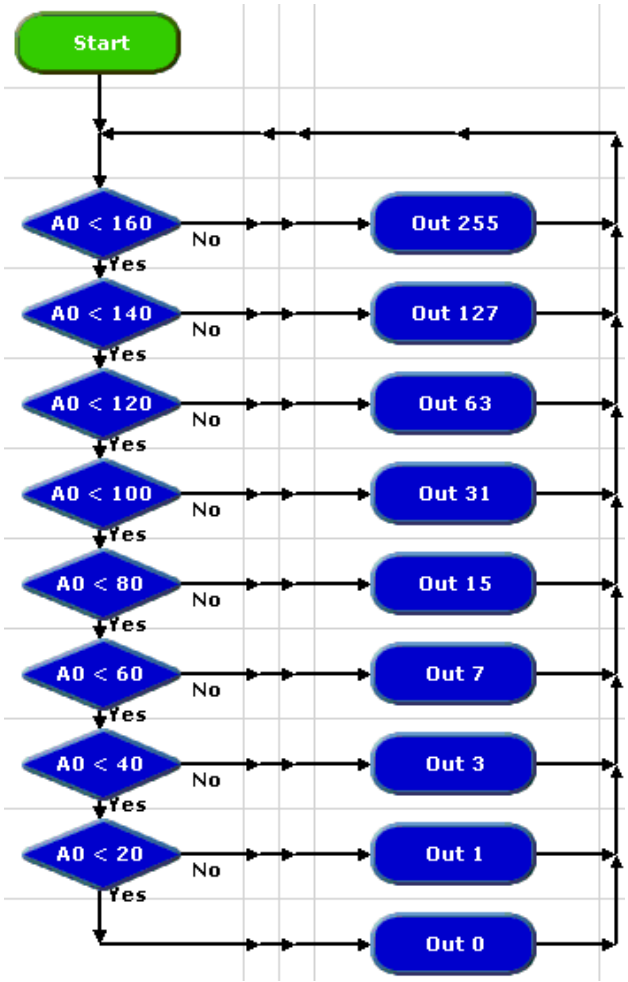


Fig 3.19. Light meter system.

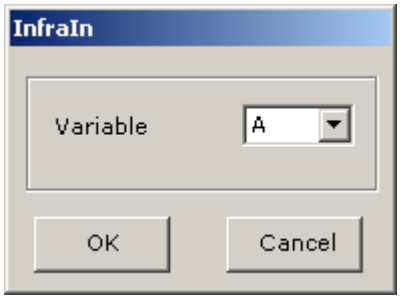
Using Infrared control

When using PICAXE chips, commands are available to support Infrared communication between PICs and TV style remote controls.

InfraIn

To receive information from an Infrared source, the InfraIn command is used. The command will wait for a new infrared signal from a infrared TV style transmitter. It can also be used to receive an InfraOut signal from a separate PICAXE08M chip.

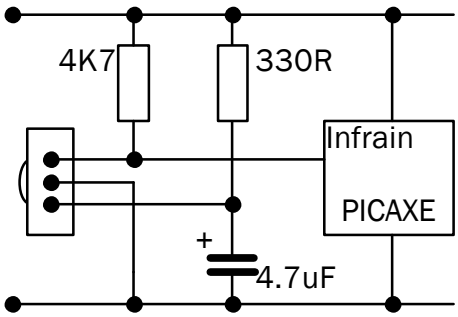
All processing stops until the new command is received. The value of the command received is placed in the chosen Variable.



The cell details are simple; only a Variable must be set.

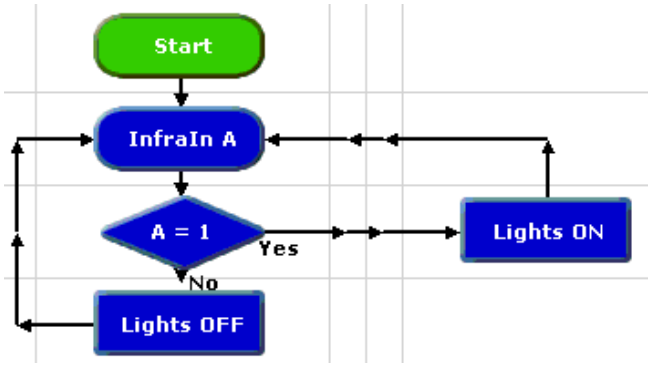
The infra-red input is input 0 on all 18 and 28 pin PICAXE chips that support this command. When using the PICAXE08M, input 3 must be used.

The basic circuit required for InfraIn is as follows. The device on the left side of the circuit is an IR receiver LED, part code LED020.



Example

To receive a signal from a TV Infrared remote control and switch on lights if key 1 is pressed, requires the following flowsheet.

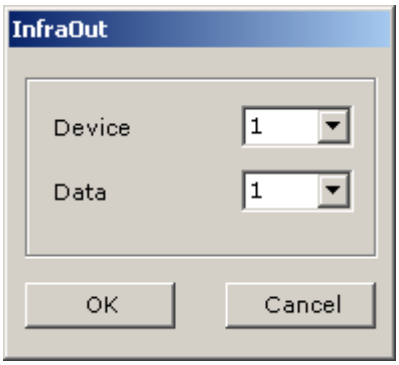


The InfraIn command waits until a signal is received, and saves this as a number in Variable A.

The Compare determines is this is '1' and the Yes route switches on the lights.

InfraOut

This command (only for the PICAXE08M) is used to transmit the infrared data to a Sony™ protocol device (can also be used to transmit data to another PICAXE that is using the InfraIn command).
Data is transmitted via an infra-red LED (connected on output 0) using the SIRC (Sony Infra Red Control) protocol.
When using this command to transmit data to another PICAXE the Device ID used must be value 1 (TV).



The InfraOut command can be used to transmit any of the valid TV commands (0-127). Note that the Sony protocol only uses 7 bits for data, and so data of value 128 to 255 is not valid.

Procedures

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 “procedures”. Note that Procedures were previously known as “Macros”.

How to build a procedure

Use a Procedure command to begin the procedure. 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.

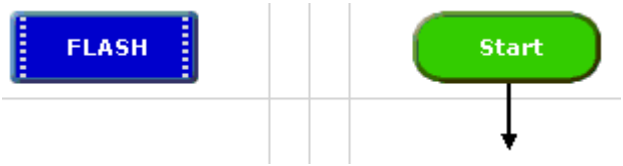


Fig 4.1. Placing the Procedure command.

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

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

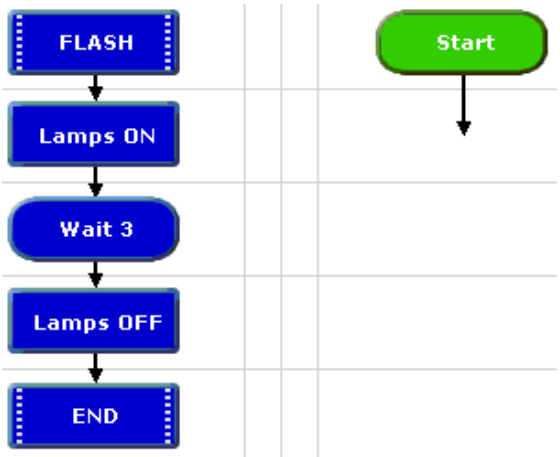


Fig 4.2. This procedure, called FLASH will switch on selected lamps for 3 seconds and then switch them off.

When you have created a procedure, you can test run it. Click on the Procedure command to select it, and click System>Run

How to use a procedure

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

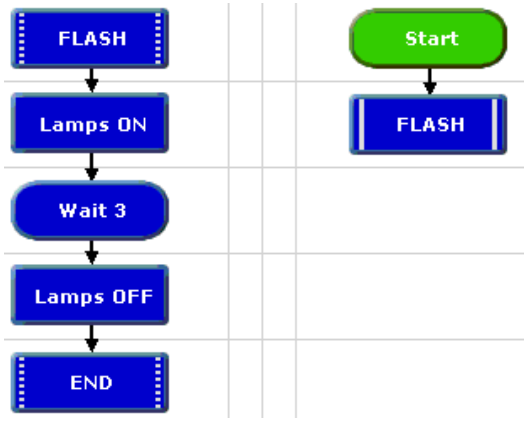


Fig 4.3. The Do Procedure command calls the procedure into use.

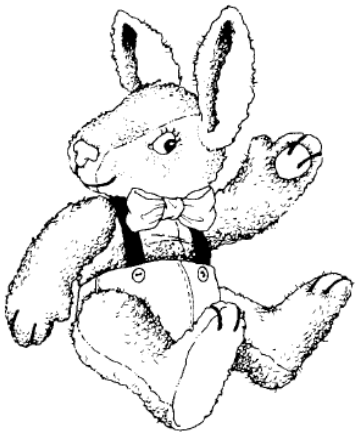
Drag a Do Procedure command onto the flowsheet. Place it at the point where you want the procedure to be called into use. Double click on the command to open its Cell Details box. Type in the name of the procedure or select it from the drop-down list. Click OK.

Note that all the procedures that have been built in a flowsheet are automatically listed in the drop-down box. When flow reaches a Do Procedure command, it jumps to the Procedure command with the same name. When the flow of control reaches an End command, the flow jumps back to the Do Procedure command that called the procedure. To test run the whole flowsheet, click on the START command to highlight it, and click System>Run

In the cell details box for the Do Procedure command it is also possible to set the number of times top run the Procedure. This will simply repeat the Do Procedure for the set number and then continue as normal.

Example one

A PIC microcontroller is used 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 procedure which can be tested and edited separately from the main routine.



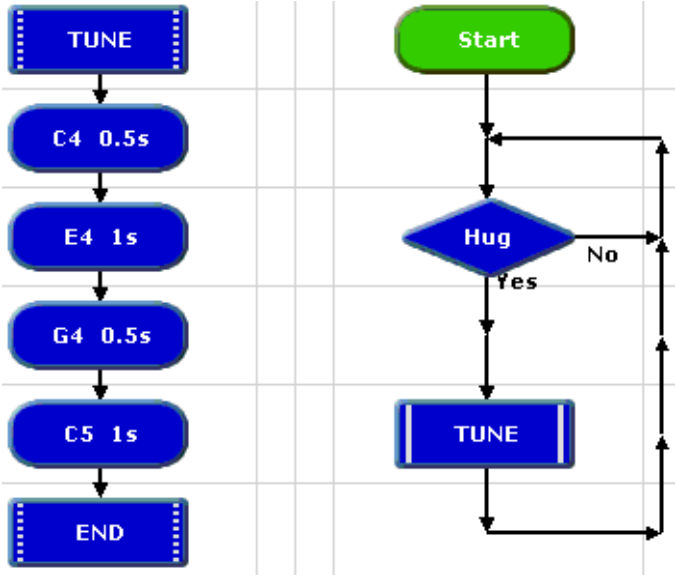
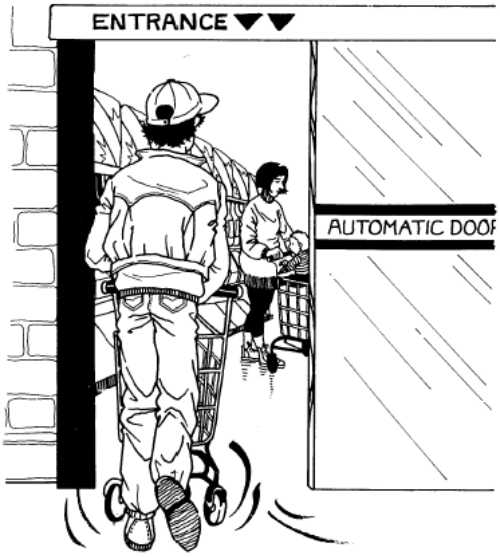


Fig 4.4. Using a Procedure to play a tune after an input condition is met.

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 micro-switches.

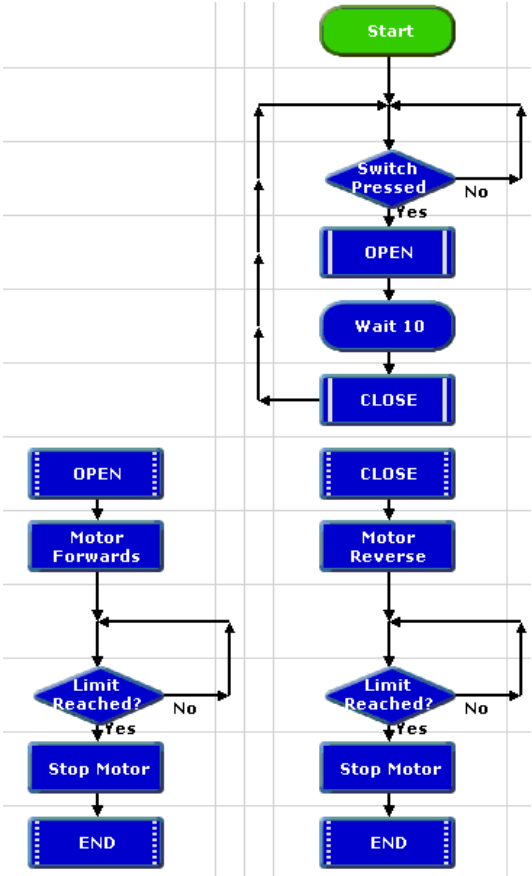


Fig 4.5. Sliding door control system using procedures.

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 procedure.

To begin with, the row is made “live” by switching on the output to which it is connected. Then a 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 procedure. When all three digits have been entered correctly, the solenoid is switched to unlock the door.

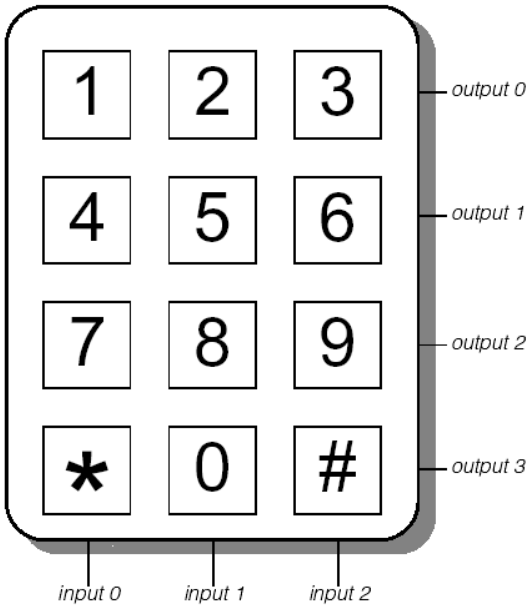


Fig 4.6. Keypad connections.

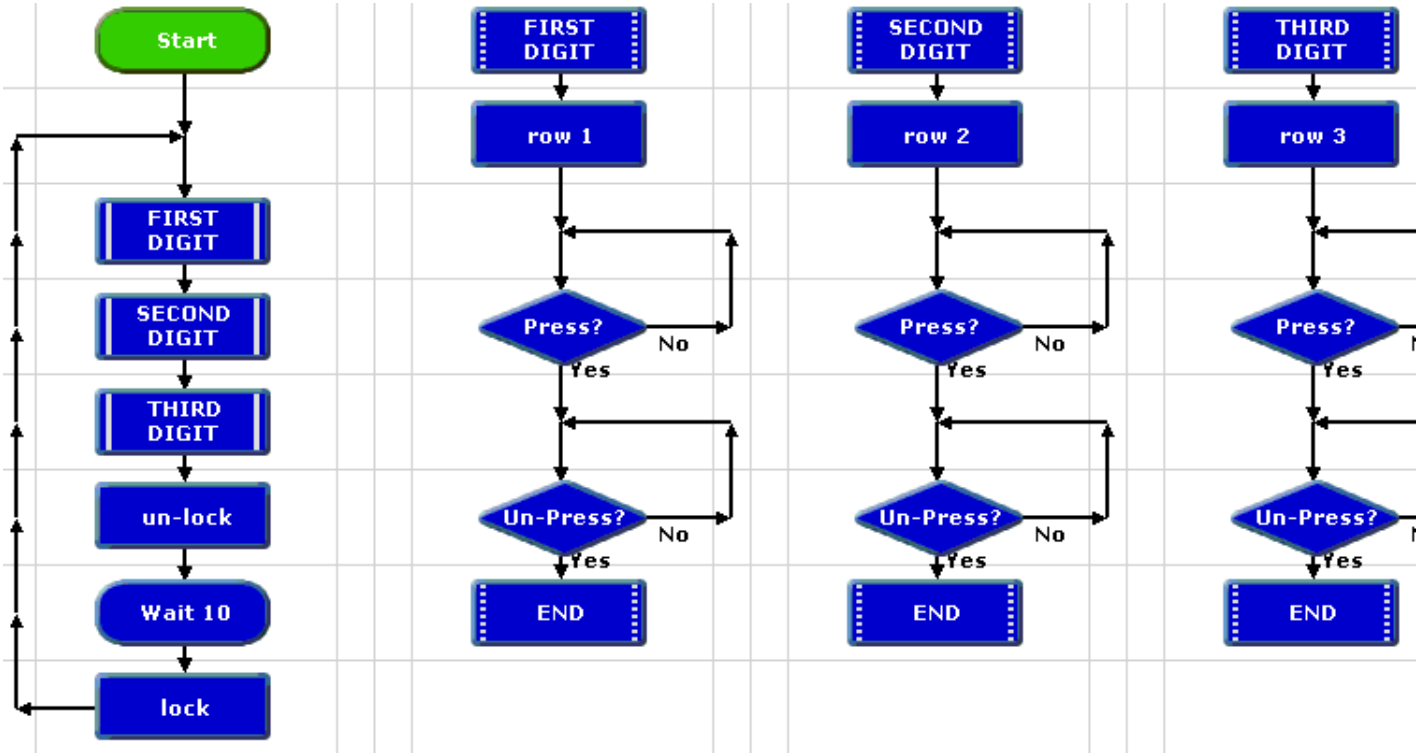


Fig 4.7. Flowsheet to scan the keypad.

Designing systems with procedures

Using procedures, 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 procedure. The following sequence shows how it can be used to develop a control system for a buggy which is fitted with micro-switches 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 Procedure commands as shown in fig 4.8.
2. Then each part of the system is built as a separate procedure as shown in fig 4.9. Each procedure can be test run independently.

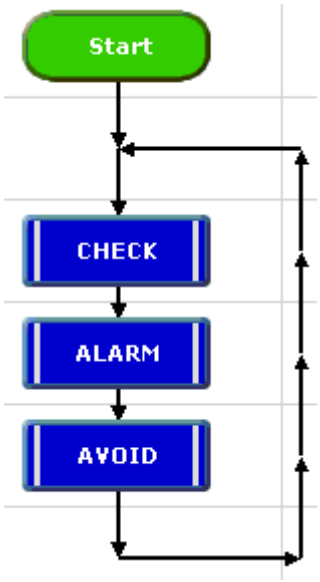


Fig 4.8. Main routine.

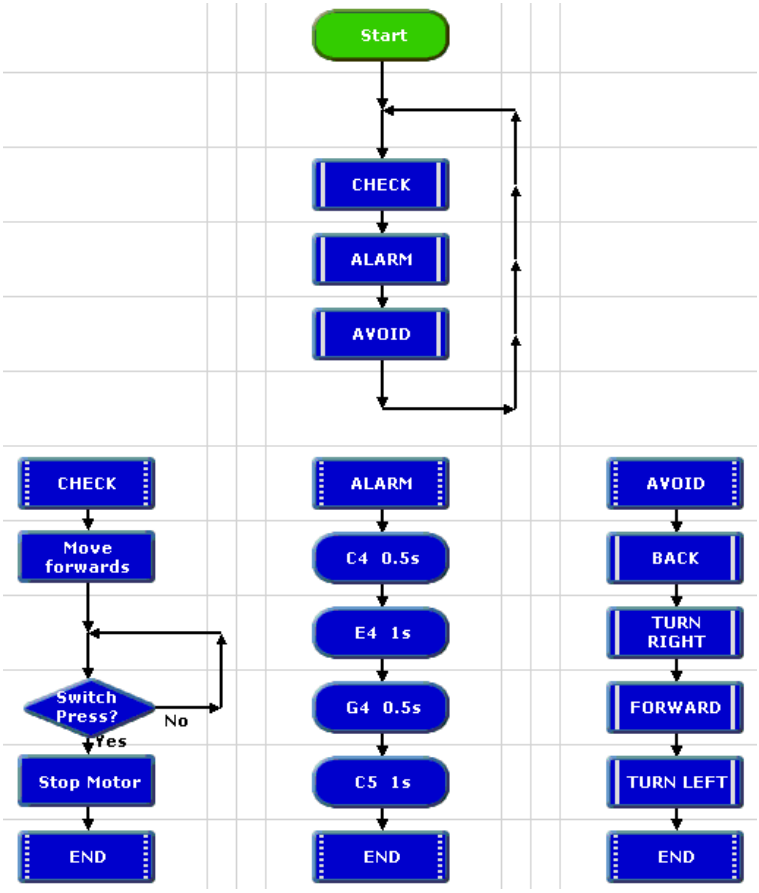
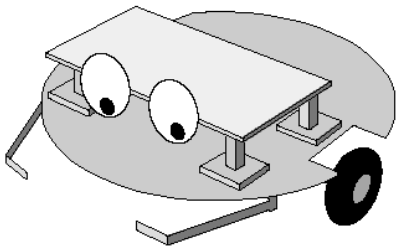


Fig 4.9. Notice that the AVOID procedure uses the top-down approach, so the flowsheet is incomplete at this stage.

3. The AVOID procedure 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 Procedure command. Then the details required for the buggy to make each movement can be dealt with separately as shown in fig 4.10.



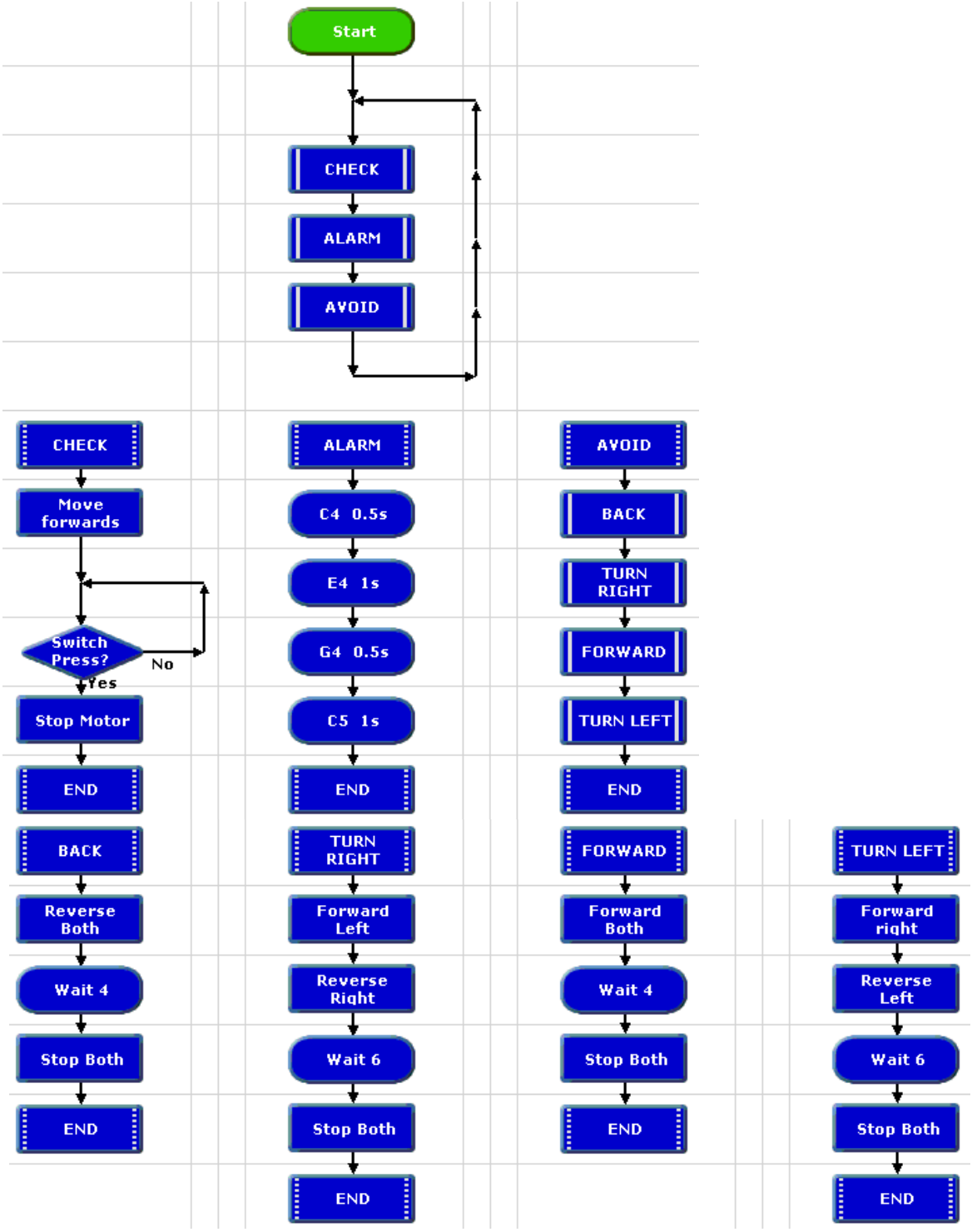
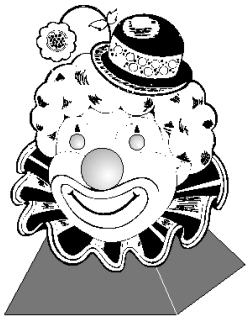


Fig 4.10. This flowsheet illustrates the way in which procedures may be called from within other procedure definitions.

The “bottom-up” approach.



This approach develops each part of the system separately as a procedure, 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 procedure is built and tested for each one of the three elements, as shown in fig 4.11.

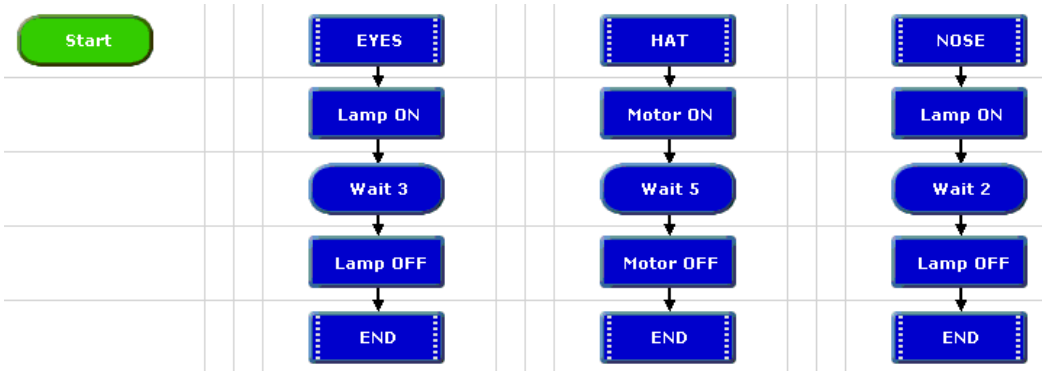


Fig 4.11. In this approach, the procedures are created first.

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

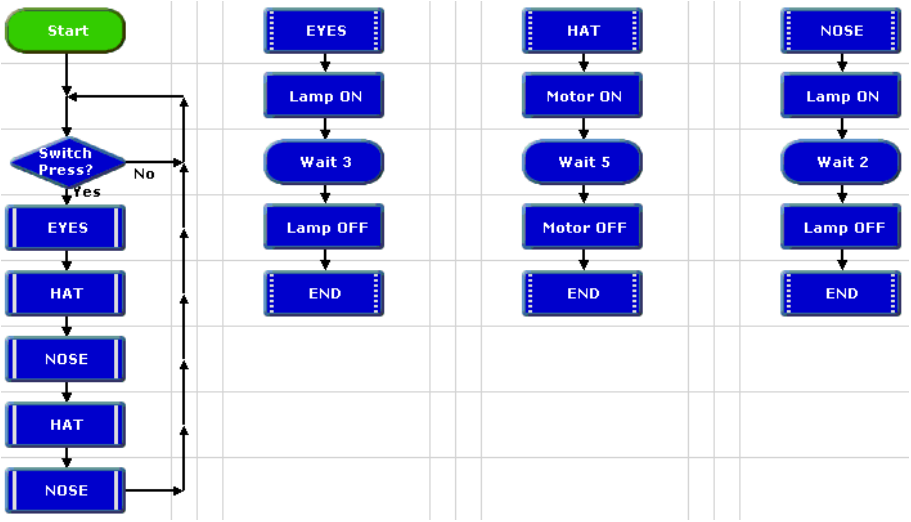


Fig 4.12. The complete system.

This flowsheet shows some of the advantages of using this approach. Once a procedure 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 Procedure commands can be moved around, deleted or copied to change the sequence as required. Procedures can be cut, copied and pasted between flowsheets. Remember that copied commands will retain their existing cell details.

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. This section explains how they can be used for a variety of mainly counting and timing purposes.

Counting

The Inc command

Each time flow passes through an Inc command, 1 is added to the value of the 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 procedure is done, and one is added to the value of variable A.

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

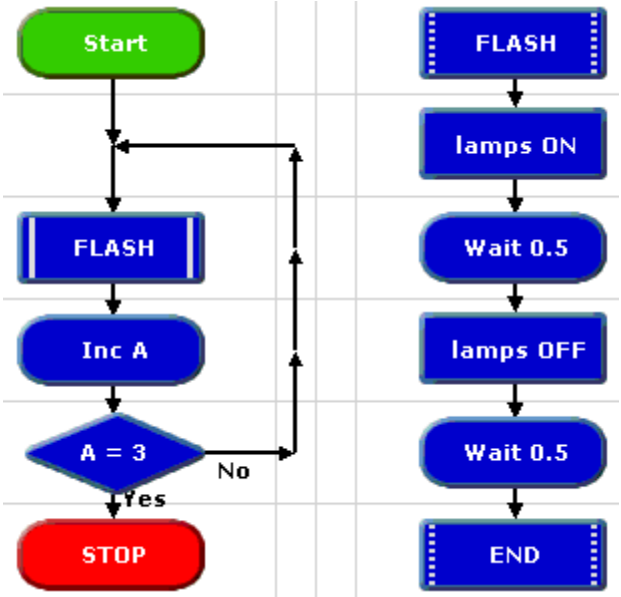


Fig 5.1. Repeating a sequence three times.

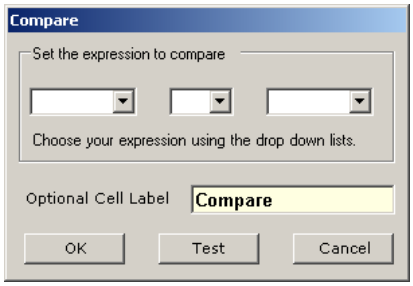


Fig 5.2. Cell Details box of the Compare command.

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.
3. 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 dropdown 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 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.

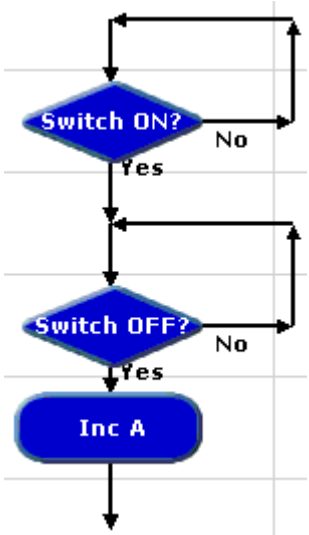


Fig 5.3. Ensuring a clean signal from a digital sensor.

You may well find that once it is downloaded into the chip, the flowsheet runs so quickly that even using the two 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.

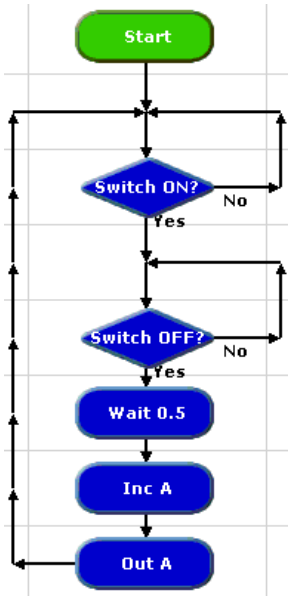


Fig 5.4. Flowsheet for making and displaying a count.

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.

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.

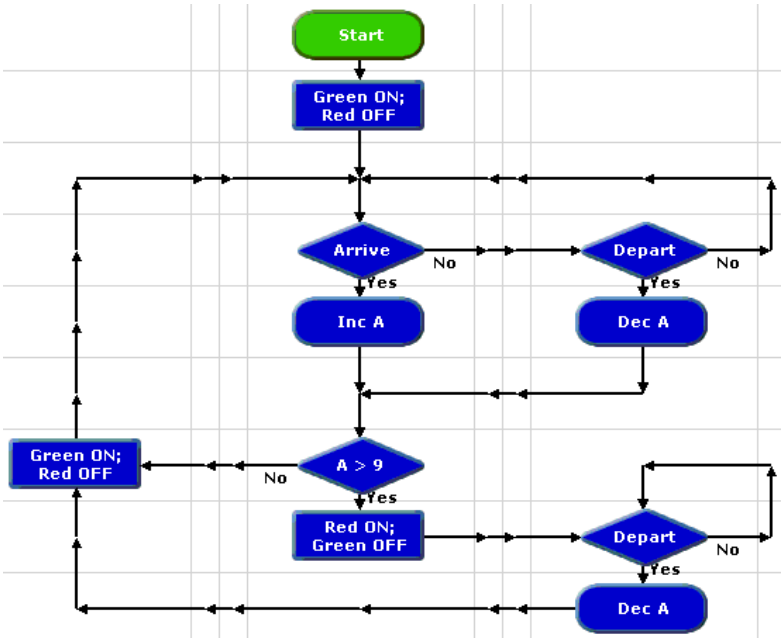


Fig 5.5. Car park counting system

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 procedure 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 procedure would use a series of SerOut commands as shown in fig. 5.7.

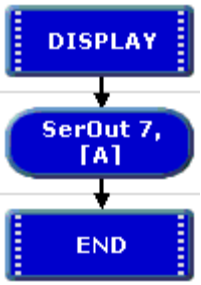
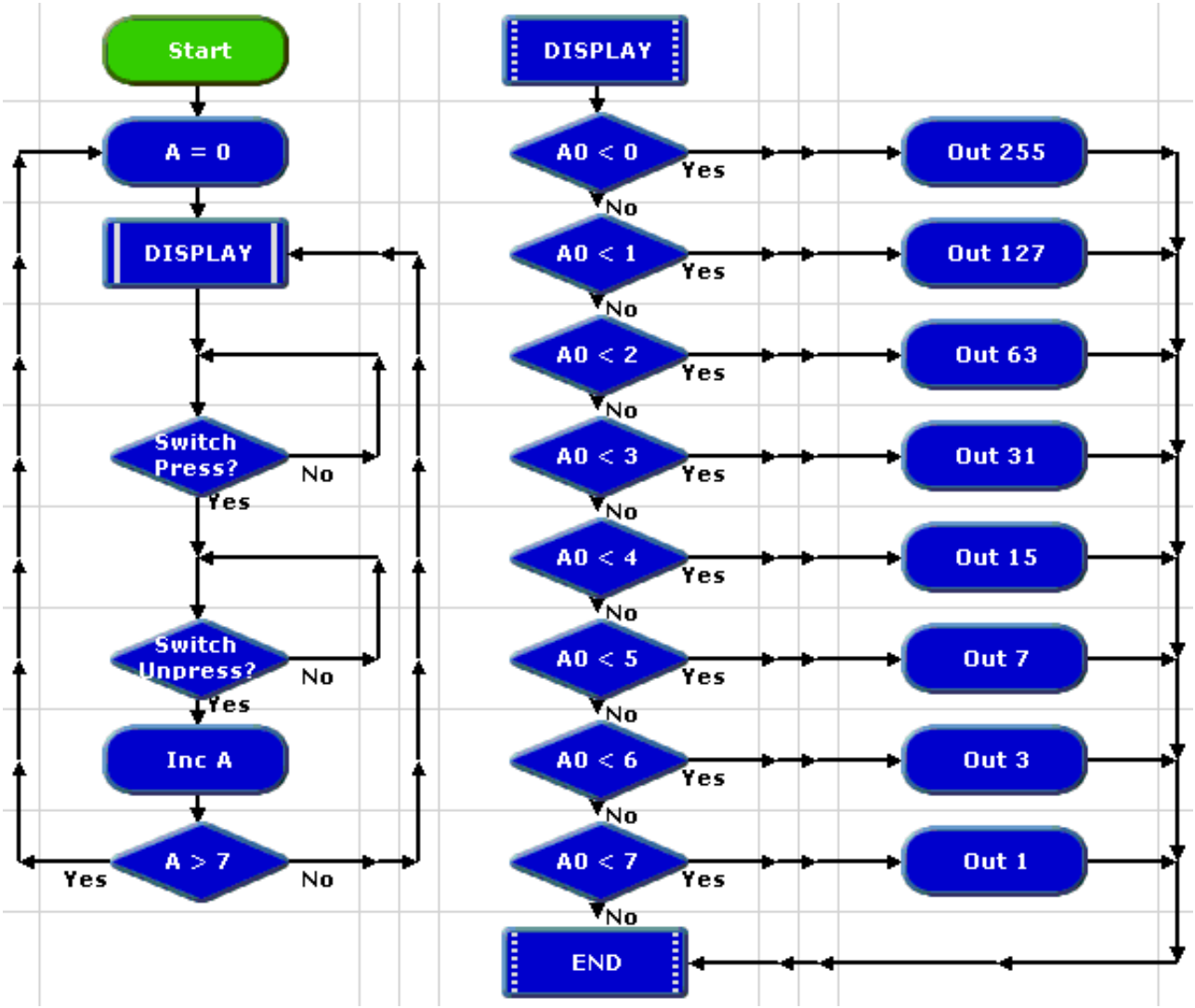


Fig 5.7. Part of an equivalent system that uses an LCD screen to display numbers.

Fig 5.6. A "Now Serving...." display system.



Timing

To repeat a sequence for a period of time, the Inc command can be used to count the elapsed time. The flowsheet shown in fig 5.8 shows how it can be used to repeat a sequence for 10 seconds.

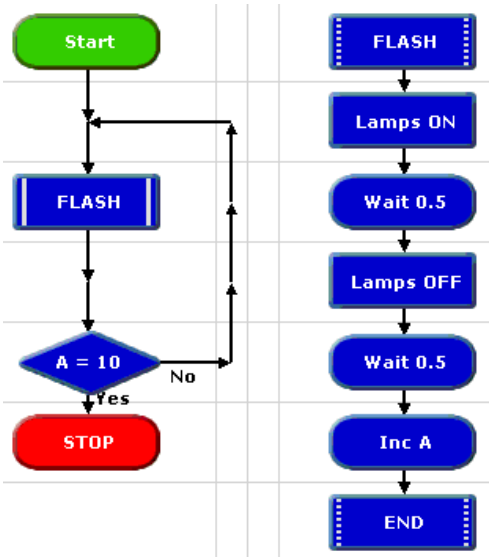


Fig 5.8.Repeating a sequence for 10 seconds.

A Compare is used to check the value of Variable A. When this value reaches 10, flow will go in the Yes direction and stop the flowsheet. Since we know that the FLASH Procedure will take 1 second to complete, repeating this for 10 times will take 10 seconds.

Setting the value of a variable

The Expression command

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 example shows how it can be used.

Example one

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 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.

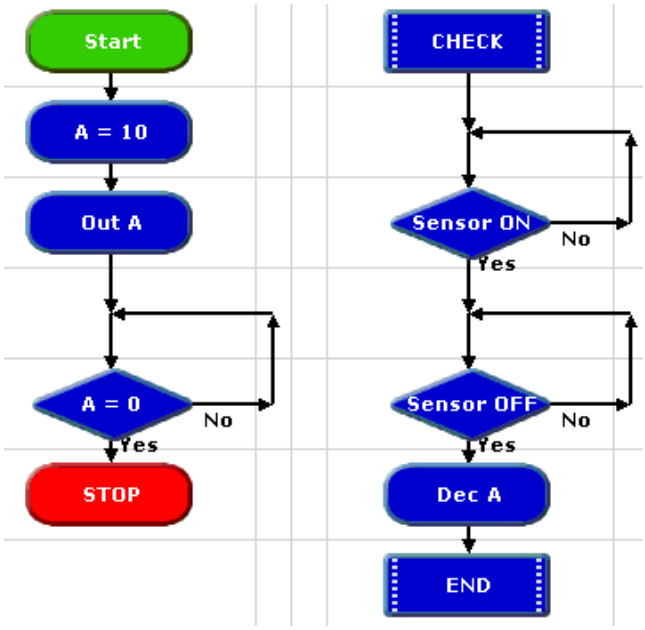


Fig 5.10. Counting Down

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.

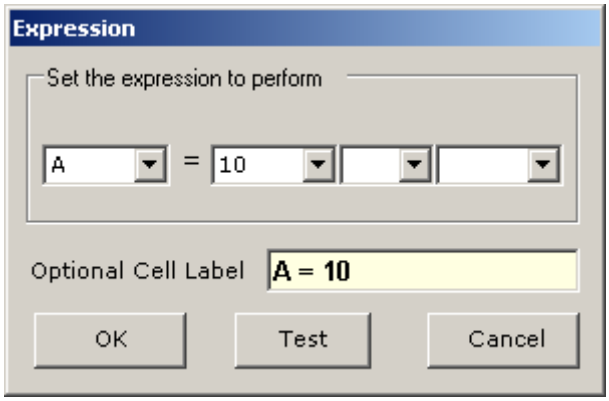


Fig 5.11. Expression command Cell Details box Setting the value of a variable

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. NOTE: the third box in the Expression Cell Details box contains a range of mathematical operators.

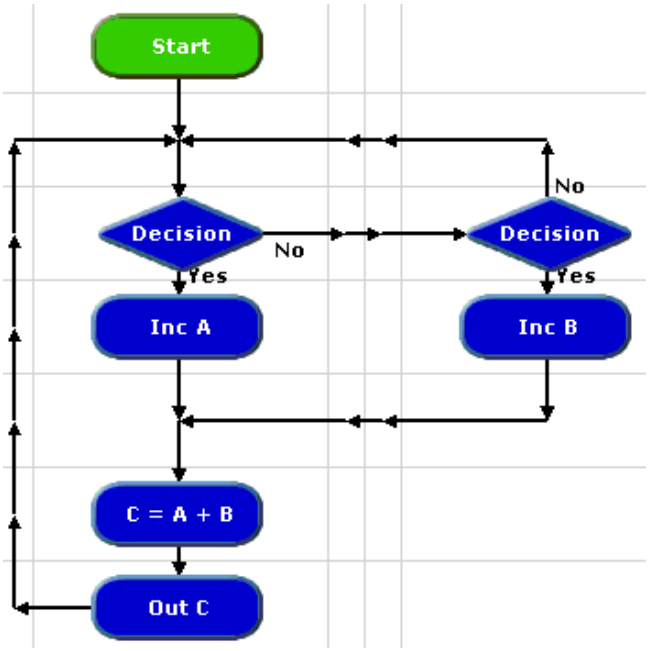


Fig 5.12. Displaying a combined count.

IN and RND

The IN command sets the value of a specified variable to the current binary value of the input port.

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 Yes direction and a solenoid-operated lock is opened. If any other combination of switches is pressed, flow goes in the No direction.

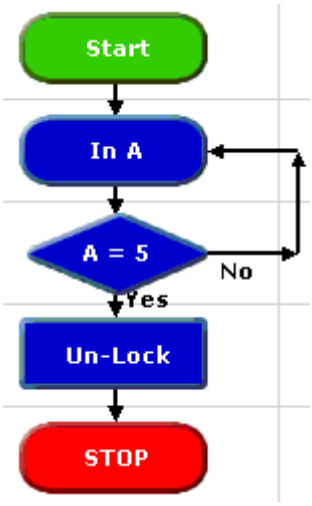


Fig. 5.13.Simple security system that responds to pressing two switches.

The RND expression

Within the expression command a Variable can be given a random value between 0 and 255. In the example shown in fig 5.14, 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.

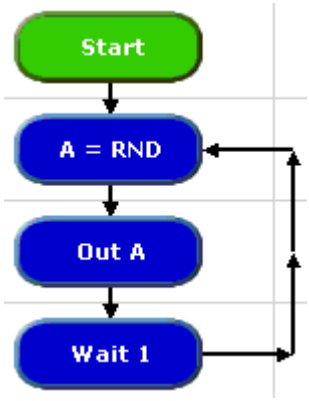


Fig 5.14.Using the RND command to create a random display of lights.

Note that as with all microcontrollers and computers, the generation of random numbers is based on a set sequence.

READ and 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.

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.

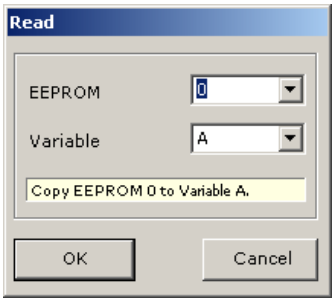


Fig 5.17. READ command Cell Details box.

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.

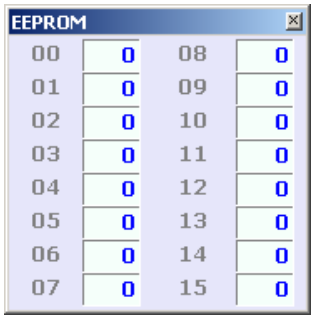


Fig 5.18 EEPROM window.

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

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

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.

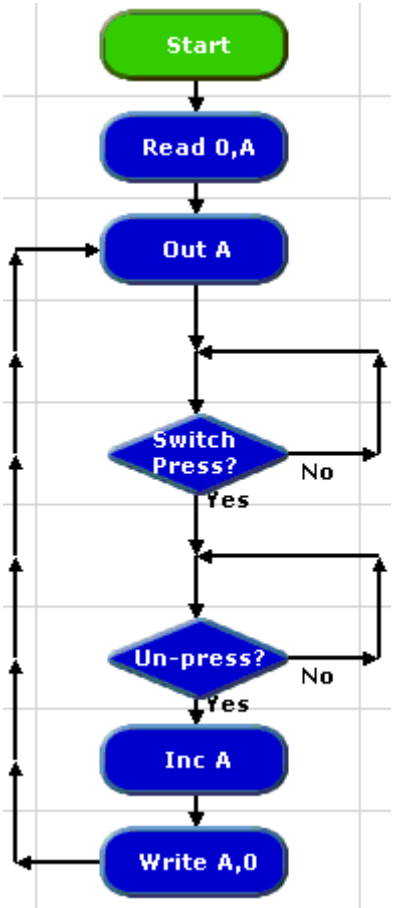
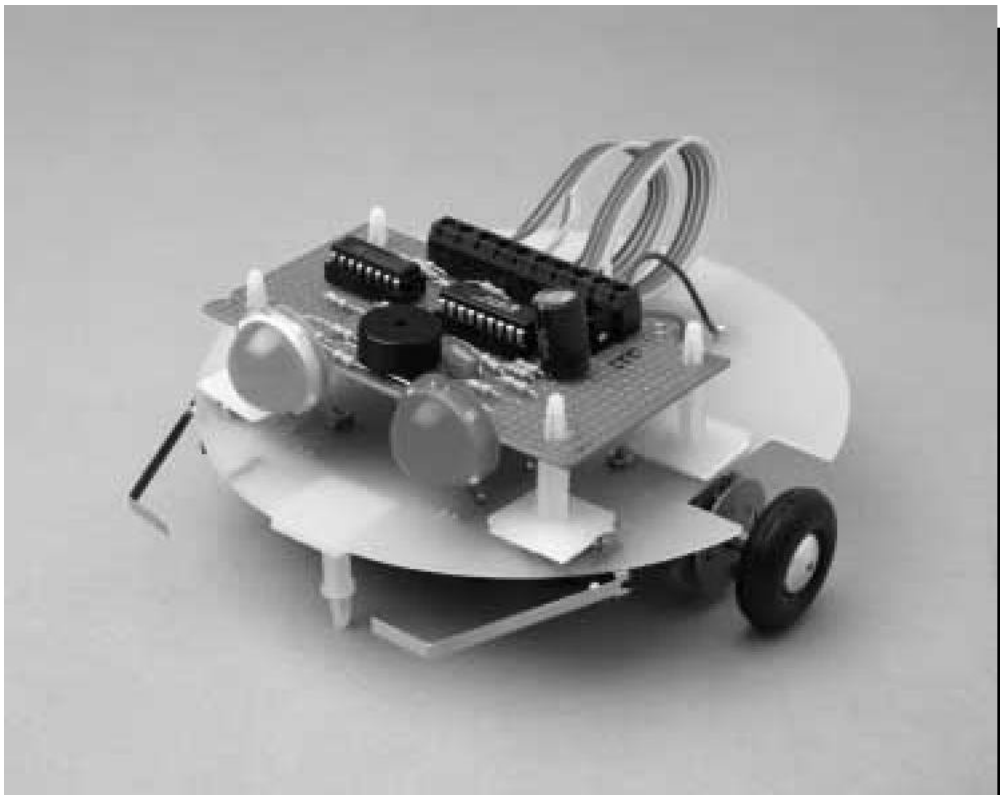


Fig 5.16. Using READ and WRITE commands to store a count.



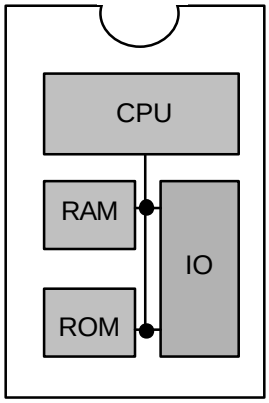
Section Two:

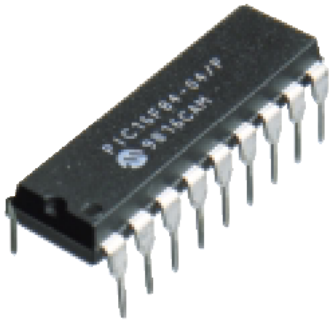
Connecting to the PIC Microcontroller

PIC Microcontrollers

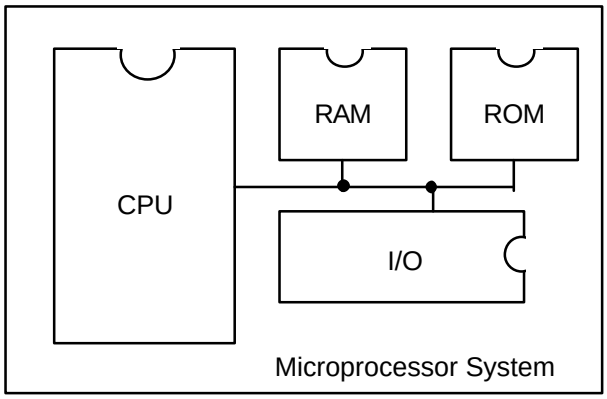
Note: definitions of terms used in connection with PICs are in the "Definitions" section on page 45.

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



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.

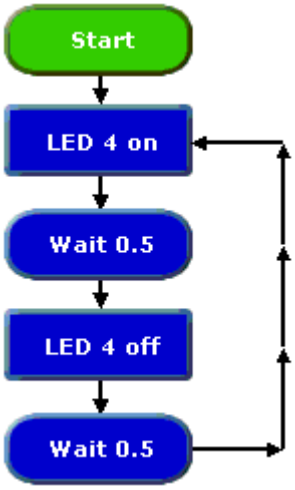
Low level languages (for example, assembler code) require a much deeper level of understanding of how the chips actually work. Their commands are complex and sometimes obscurely titled. The lower the level of the language, the nearer it is to the machine code and the harder it is to understand. On the 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 43 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.

PIC-Logicator Example (.plf)



Machine Code Example (.HEX)

```
:10000000031E080086015F308C000D3084008001E3
:10001000840A8C0B07282C282120840007398406A9
:1000200003108 40C840C840C041A841704128A01B3
:10003000820701340234043408341034203440344C
:10004000803407398A018207303431343234333412
:10005000343435343634373404304120013 0A30091
:10006000F430482004303D200130A300F430482013
:100070002C283A20073055203A280C20FF3A8005DA
:10008000452 80C2080044428FF3A84178005632803
:10009000A2005D2064002308220403190800F73041
:1000A000FF3E031D50285F204A2883160739F83881
:1000B00081006300FF3081006328A309A209A20A1E
:0C00C0000319A30A080083126400080062
:084000007F007F007F007F00BC
:04400E00F53F010079
:00000001FF
```

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@
```

BASIC Example (.BAS)

```
loop: High4
Pause 500
low 4
Pause 500
Goto loop
End
```

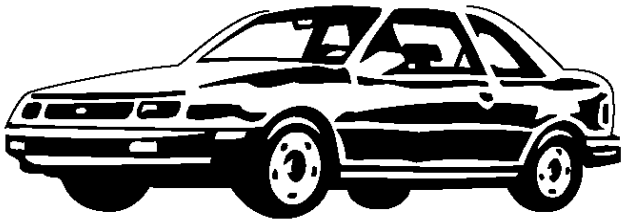
Different types of PIC

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.

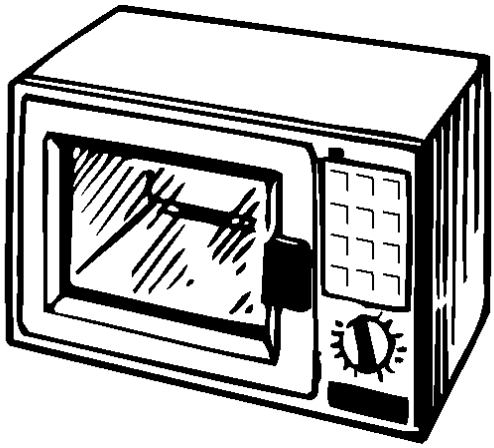
PICs also vary in terms of how they are programmed and erased. Some PICs are erasable by UV, some electrically and some not at all. UV light erasable chips are not widely used because they need a special UV light source to erase them and because of the health and safety problems associated with using 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”. PIC is a trademark of Arizona Microchip inc., the manufacturers of all PIC chips.

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. Most PIC microcontrollers used with PIC-Logicator have got the inputs and outputs preset to simplify programming, but because of the limited number of I/O pins available on 8 pin devices it is possible to customise which pins are assigned as inputs or outputs.

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. Most 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 and all of the PICAXE range 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.

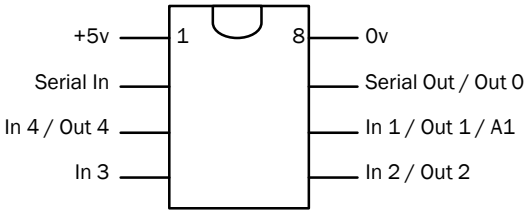
In PICAXE chips, certain input pins can be used as either analogue or digital inputs.

PIC Microcontroller Pin-out diagrams

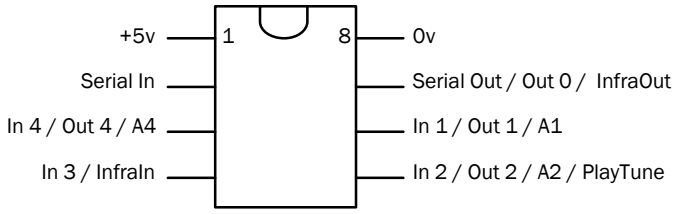
NOTE:
When you select a chip from the Options>PIC Type menu in PIC-Logicator software, the software automatically configures itself to display only the input, output and motor options available with that chip.

8 pin PIC microcontrollers

PICAXE08

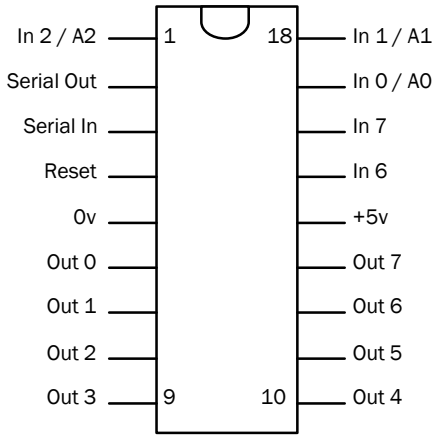


PICAXE08M

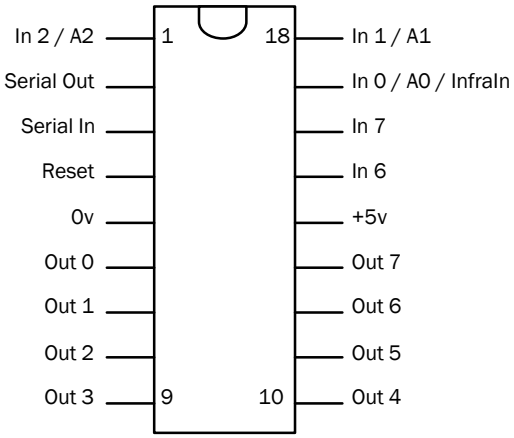


18 pin PIC microcontrollers

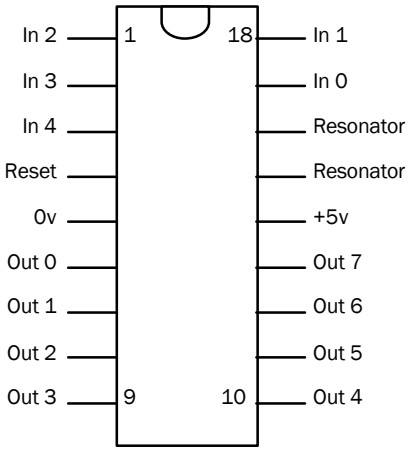
PICAXE18



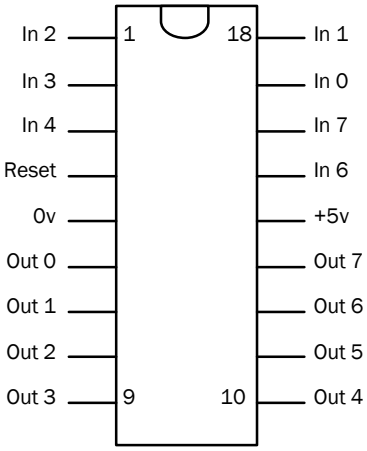
PICAXE18A, PICAXE18X



16F84, 16F84A

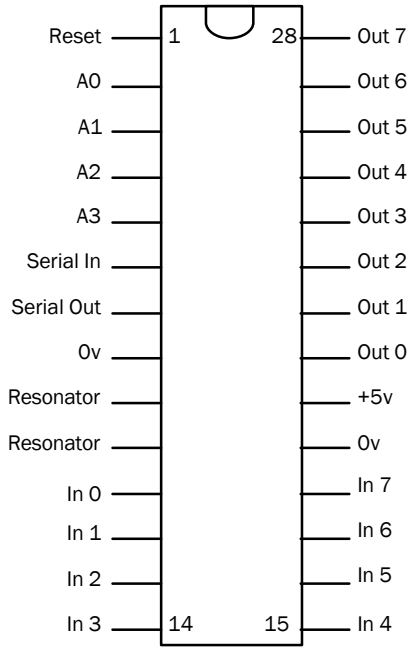


16F627, 16F628

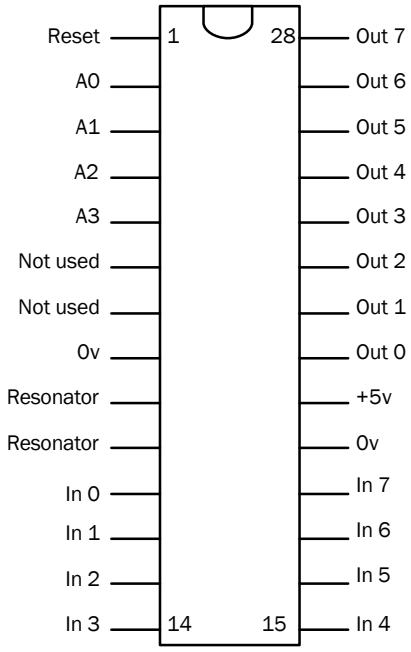


Note: the analogue inputs of the 16F627 and 16F628 are not available in PIC-Logicator version 2.

PICAXE28A, PICAXE28X



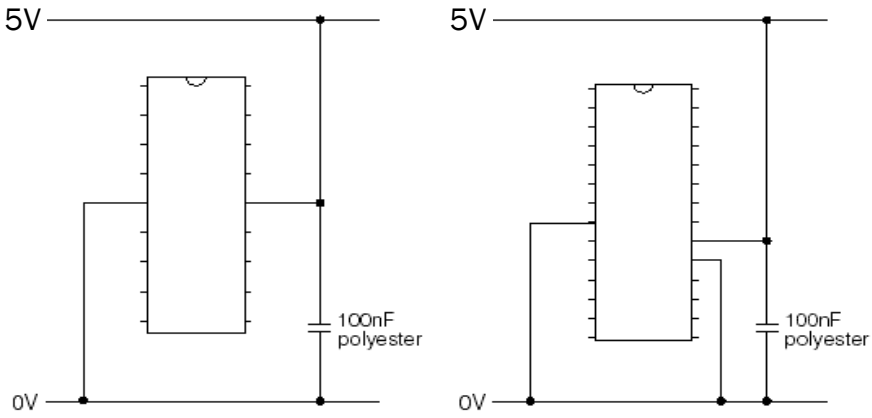
16F872, 16F873, 16F876



Basic Connections

Power

The PIC microcontroller requires a 4.5 - 5V DC supply. This can be provided by 3 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.

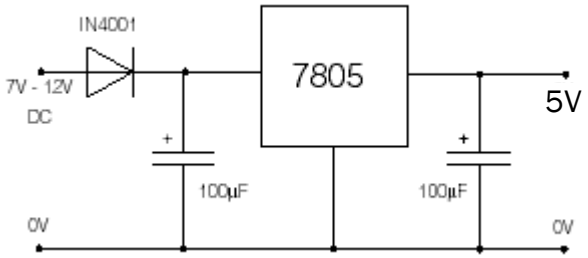
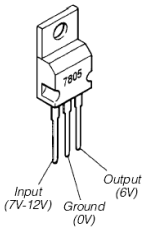


Fig 3.1 Using a 7805 voltage regulator



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.

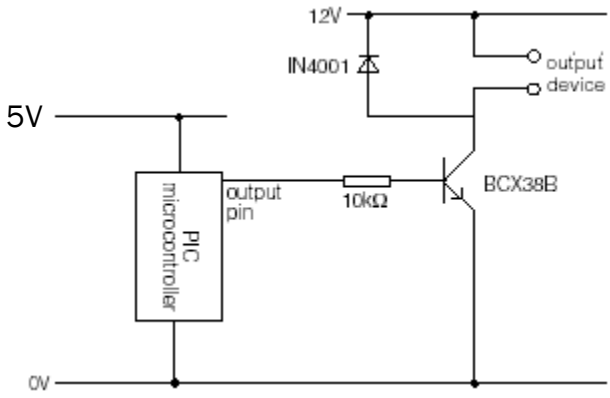


Fig 3.2 Using separate power supplies to power the process and output sections of a project.

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). Most modern PIC microcontrollers have an internal resonator. Others need to have a 4MHz ceramic resonator connected as shown in fig. 3.3.

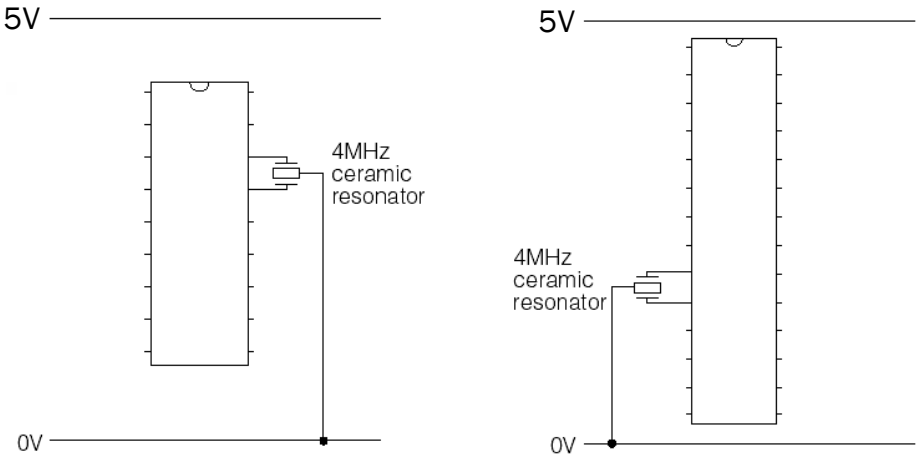


Figure 3.3 Ceramic Resonator

Reset

This pin must also be connected to the 5V 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.

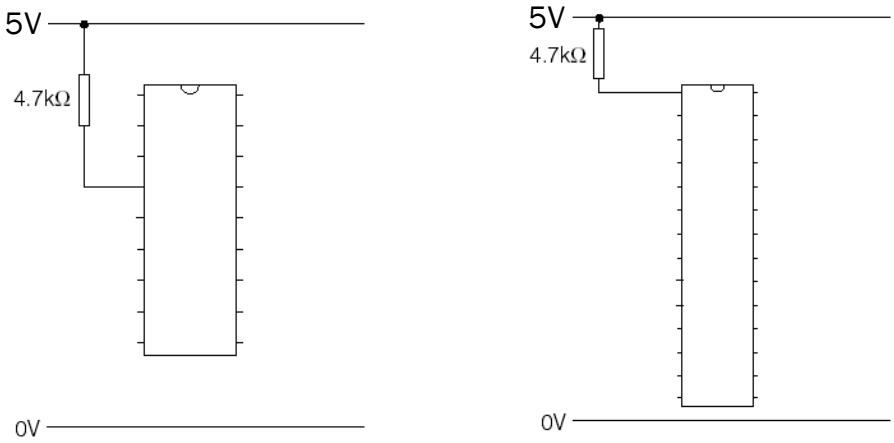


Fig. 3.4. Connecting the reset pin

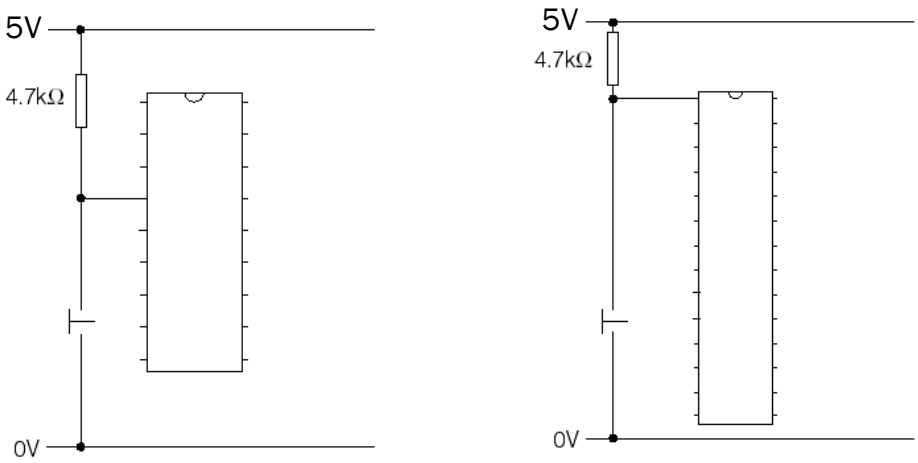
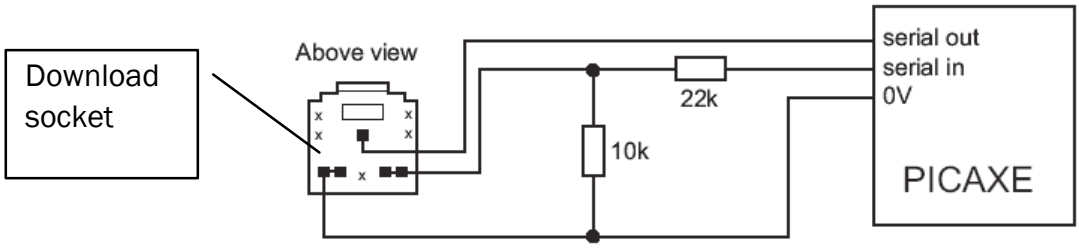


Fig. 3.5. Connecting the reset pin to include a reset switch

PICAXE Serial Download Circuit

The serial download circuit is identical for all PICAXE chips. It consists of 3 wires from the PICAXE chip to the serial port of the computer. One wire sends data from the computer to the serial input of the PICAXE, one wire transmits data from the serial output of the PICAXE to the computer, and the third wire provides a common ground.



Note that the two resistors are not a potential divider. The 22k resistor works with the internal microcontroller diodes to clamp the serial voltage to the PICAXE supply voltage and to limit the download current to an acceptable level. The 10k resistor stops the serial input 'floating' whilst the download cable is not connected. This is essential for reliable operation.

The two download resistors must be included on every PICAXE circuit (i.e. not built into the cable). The serial input must never be left unconnected. If it is left unconnected the serial input will 'float' high or low and will cause unreliable operation, as the PICAXE chip will receive spurious floating signals which it may interpret as a new download attempt.

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 48). 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 51) 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.

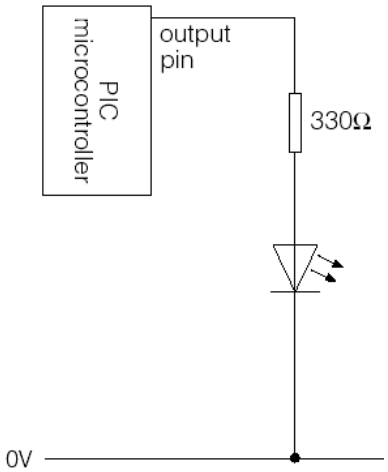


Fig 4.1 Connection of an LED

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.

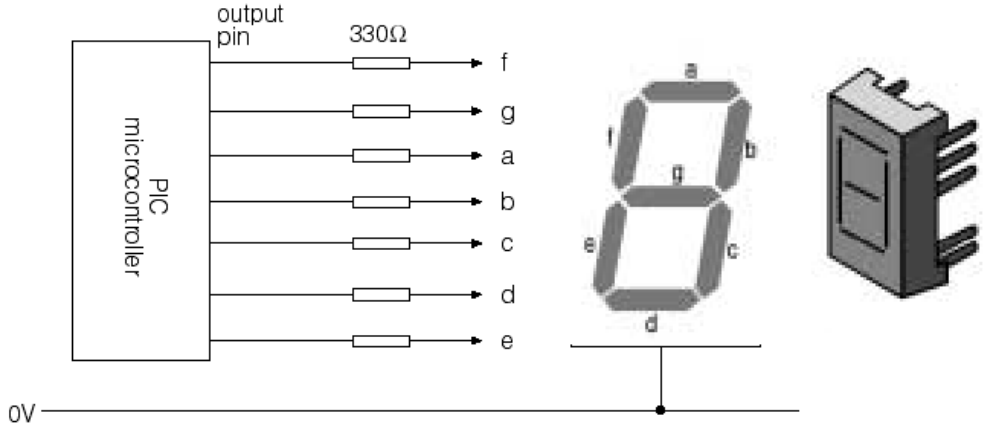


Fig 4.2 Connection of a seven segment display. Each LED bar has an identifying letter. Consult the pin-out diagram of the seven segment display that you are using, to identify the appropriate pins. This may be supplied with the component or it may be in the supplier's catalogue

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.

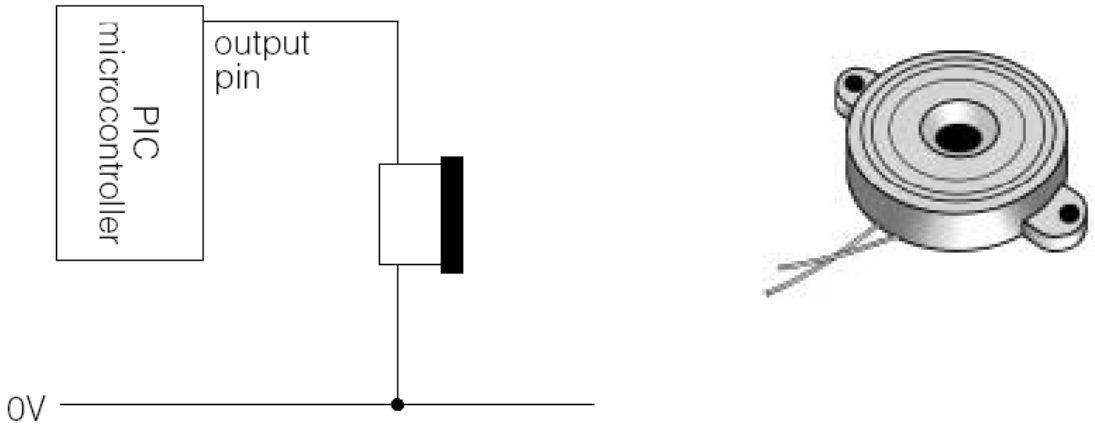


Fig 4.3 Connection of a piezo sounder

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.

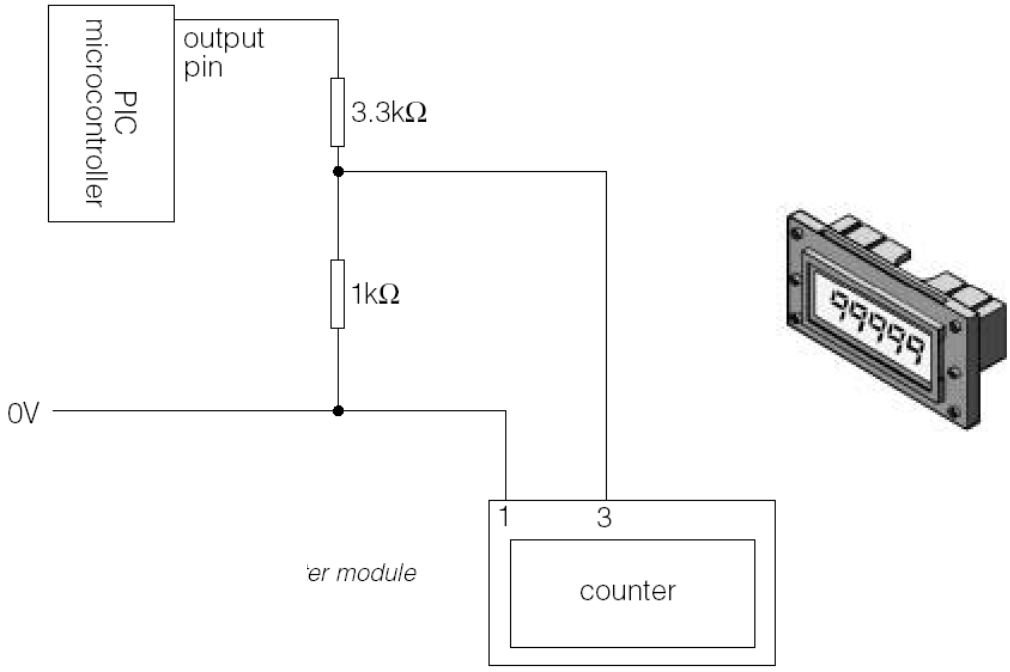


Fig 4.4 Connection of a counter module

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 54.

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.

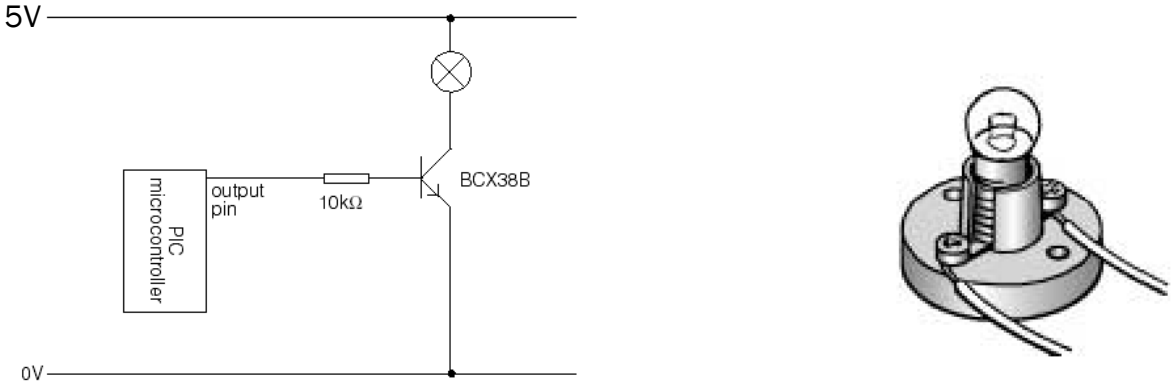


Fig 4.5 Connection of a signal lamp

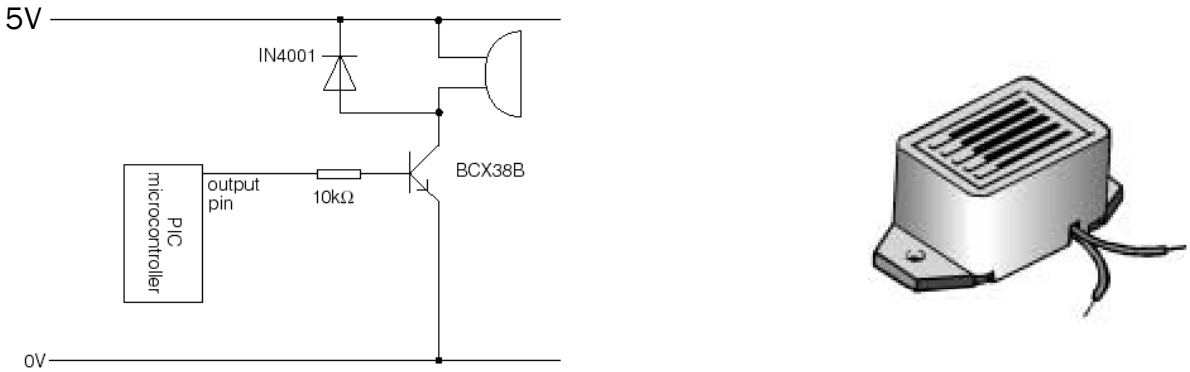


Fig 4.6 Connection of a buzzer

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.

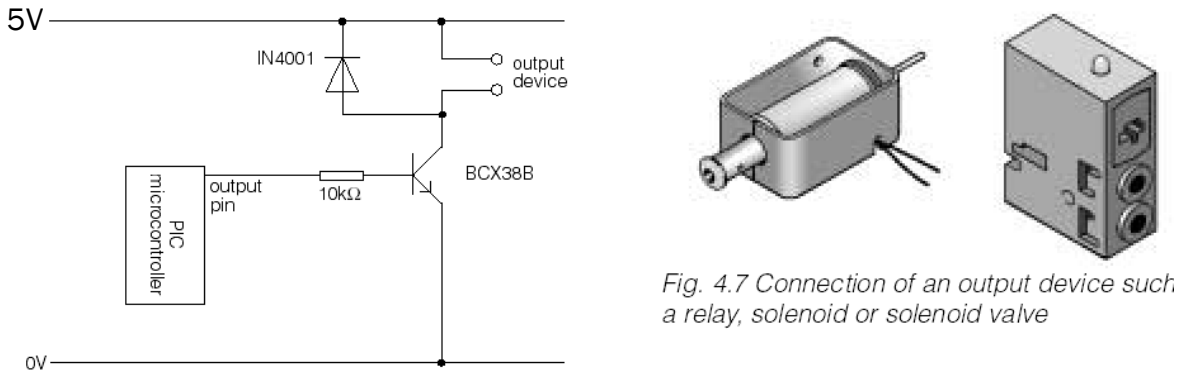


Fig. 4.7 Connection of an output device such a relay, solenoid or solenoid valve

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. A ULN2803A can also be used.

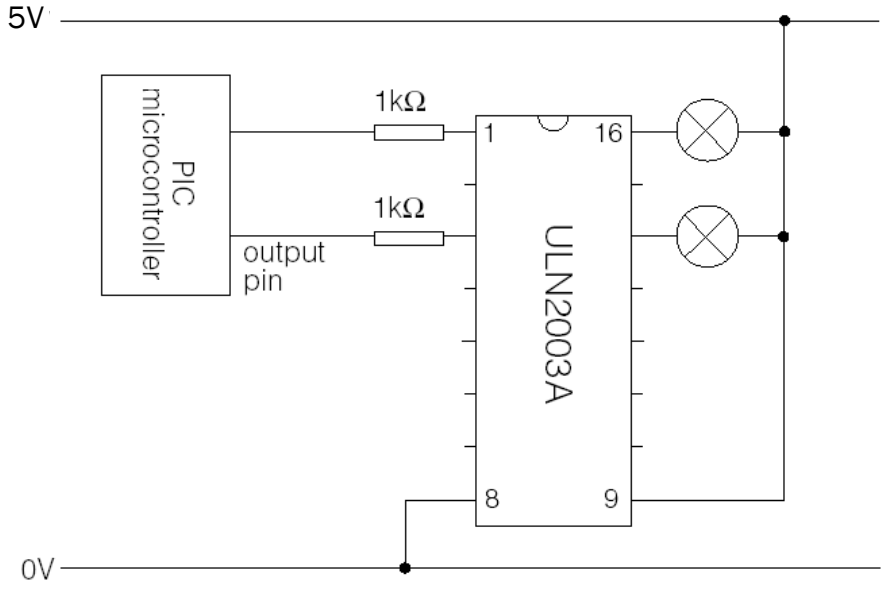


Fig. 4.8 Connection of a number of output devices using a ULN2003A Darlington driver IC

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.

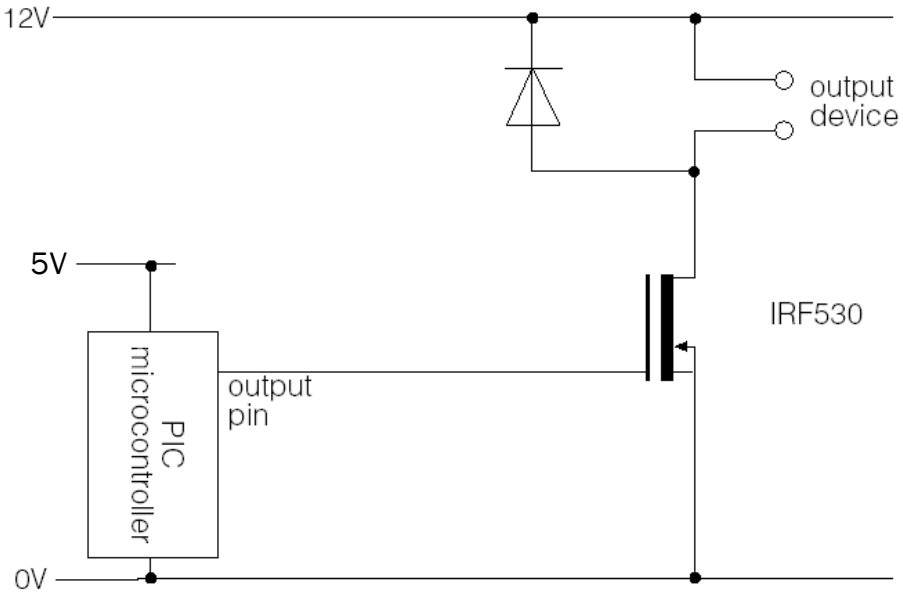


Fig. 4.9 Standard MOSFET circuit

C1. Low-voltage DC motors

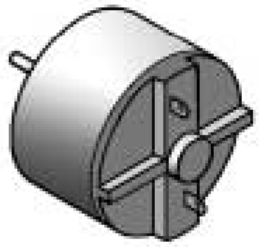
The higher quality low voltage DC motors often called “solar motors” are recommended.

The PIC-Logicator system simplifies the control of motors by using the Motor command (see page 15).

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.



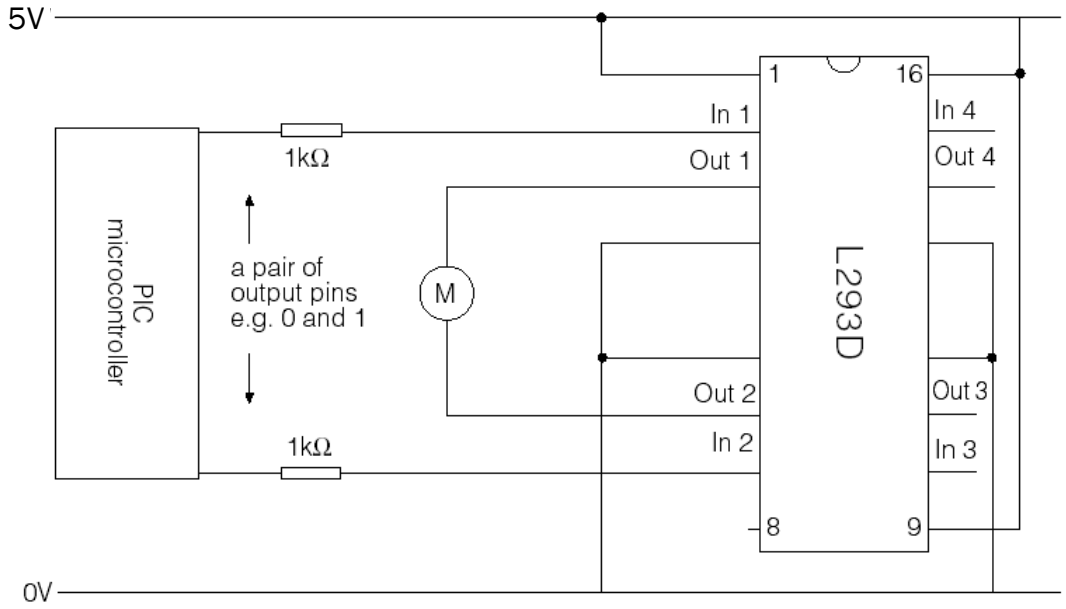


Fig 4.10 Using the L293D motor driver with a PIC microcontroller

NOTE: Pin 8 of the L293D should be connected to 5V 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. Use a short Wait time between each Output command to dictate the speed of the motor. By using Wait [A] and setting variable A to values, you can actively control the speed of the motor.

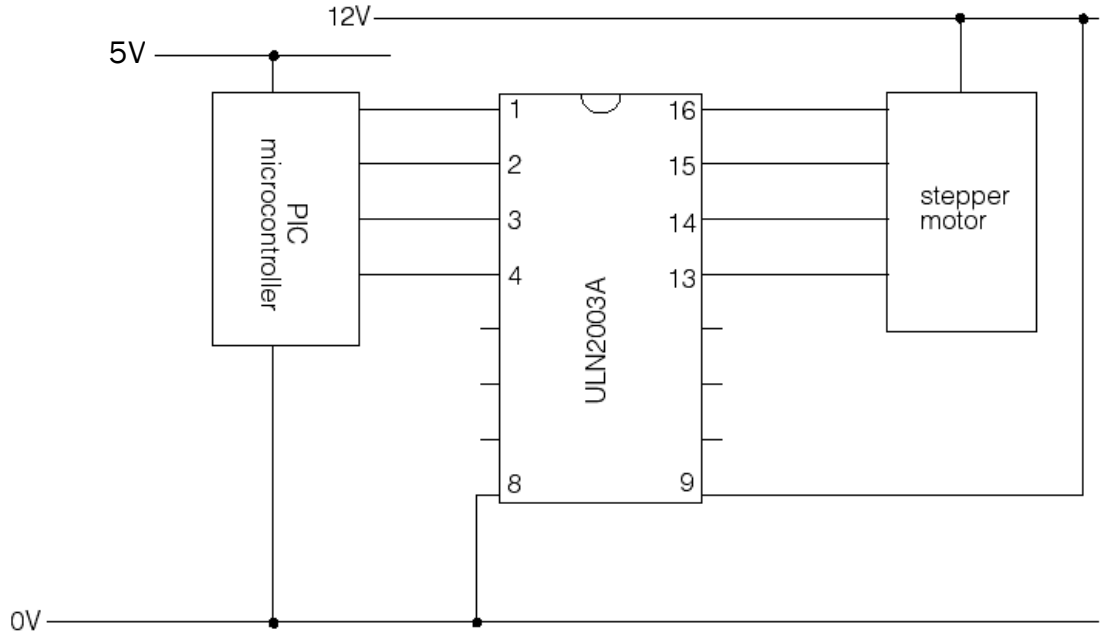
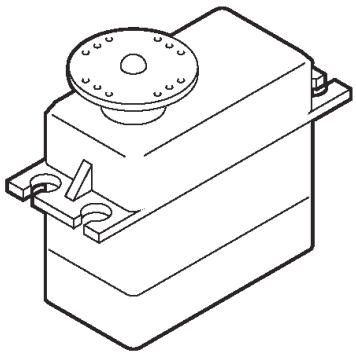


Fig 4.12 A ULN2003A Darlington Driver can be used to drive the stepper motor.

C3. Servo Motors



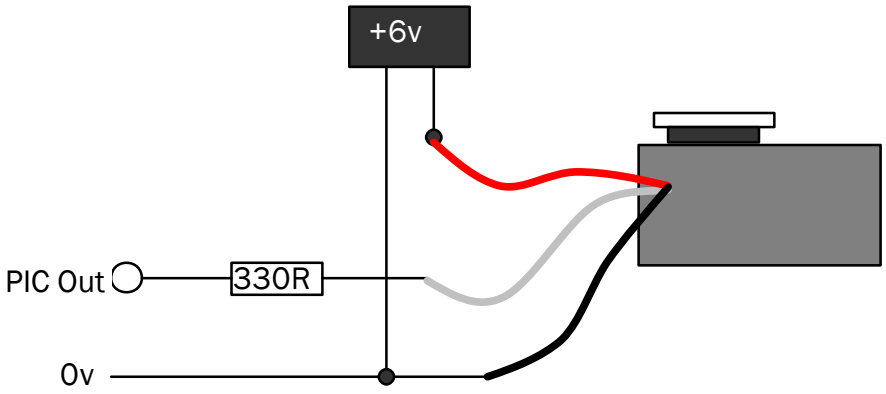
A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio controlled cars, puppets, and robots.

The servo motor has some control circuits and a potentiometer that is connected to the output shaft. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, its somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control.

Servo motors can create a high level of electrical noise which can disrupt the PIC microcontroller's processing. It is important to use a separate power supply for the servo motor, to that supplying the PIC, but ensure that the 0v lines of both power supplies are connected together.

Servos normally have three wire connection. The red and black should be connected to the power supply (red +6v) and the white connected to the PIC output as in the following diagram.



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 48). 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

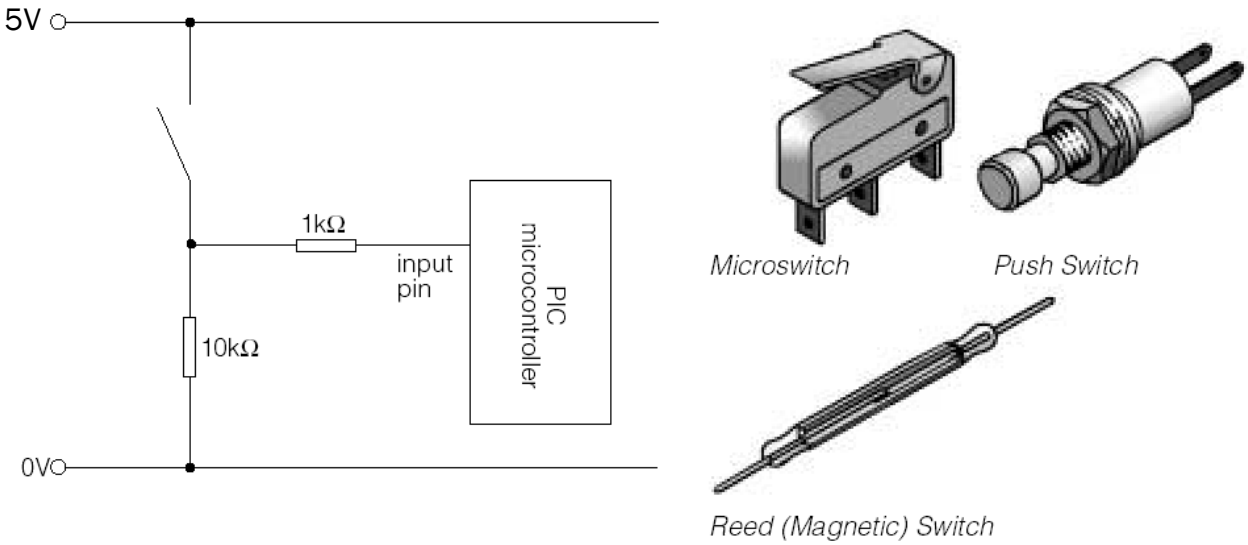


Fig 5.1 Connection of a switch

B. Analogue Sensors

B1. Potentiometer

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

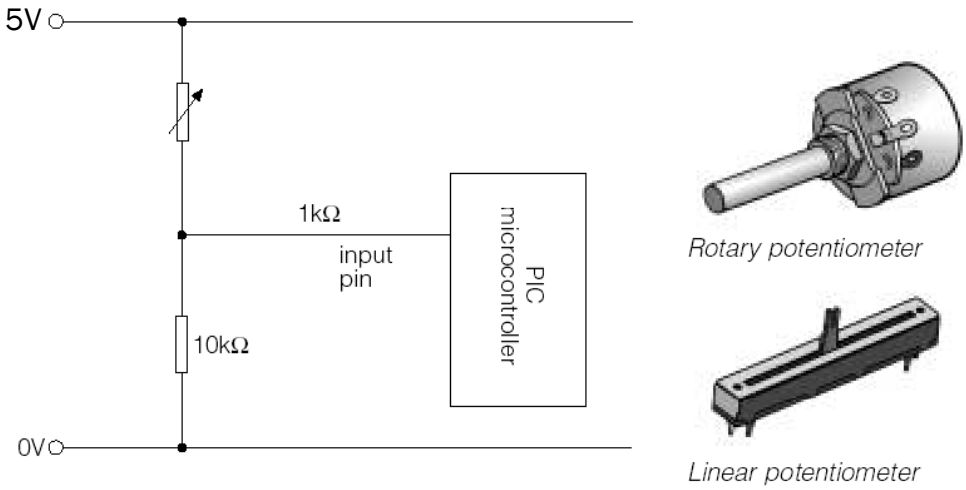


Fig 5.2 Connection of a potentiometer.

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.

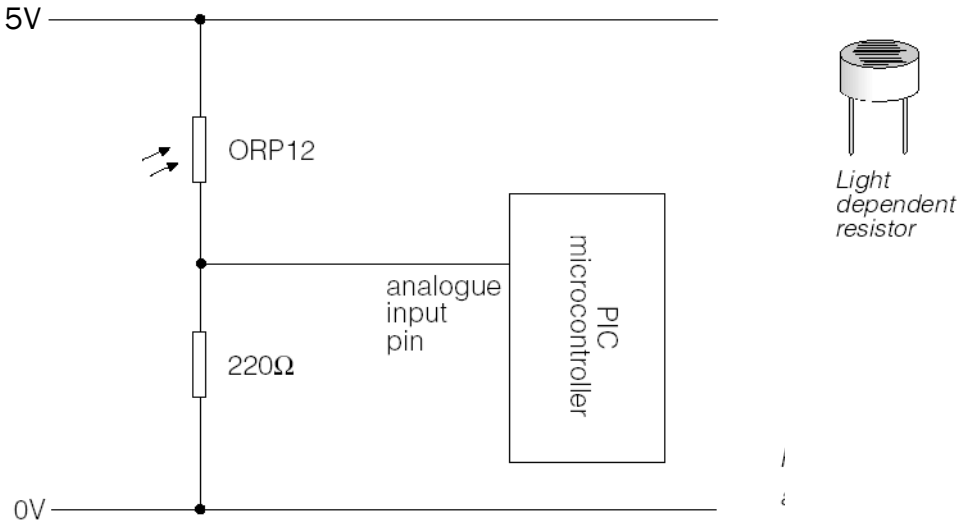


Fig 5.3 Connection of a light sensor

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.

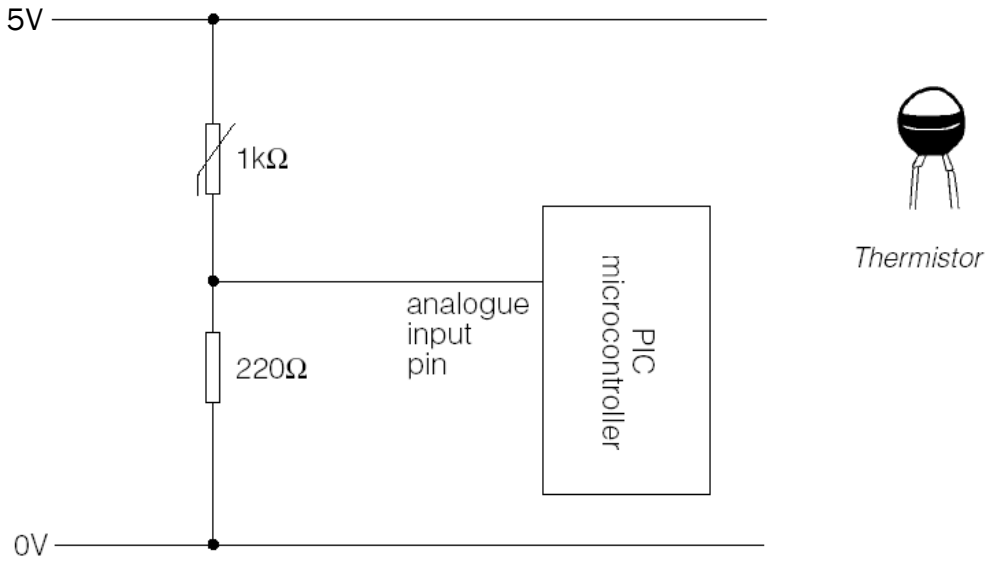


Fig 5.4 Connection of a temperature sensor

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 26) which is a valuable development tool because it displays the actual reading that is being obtained from the sensor.