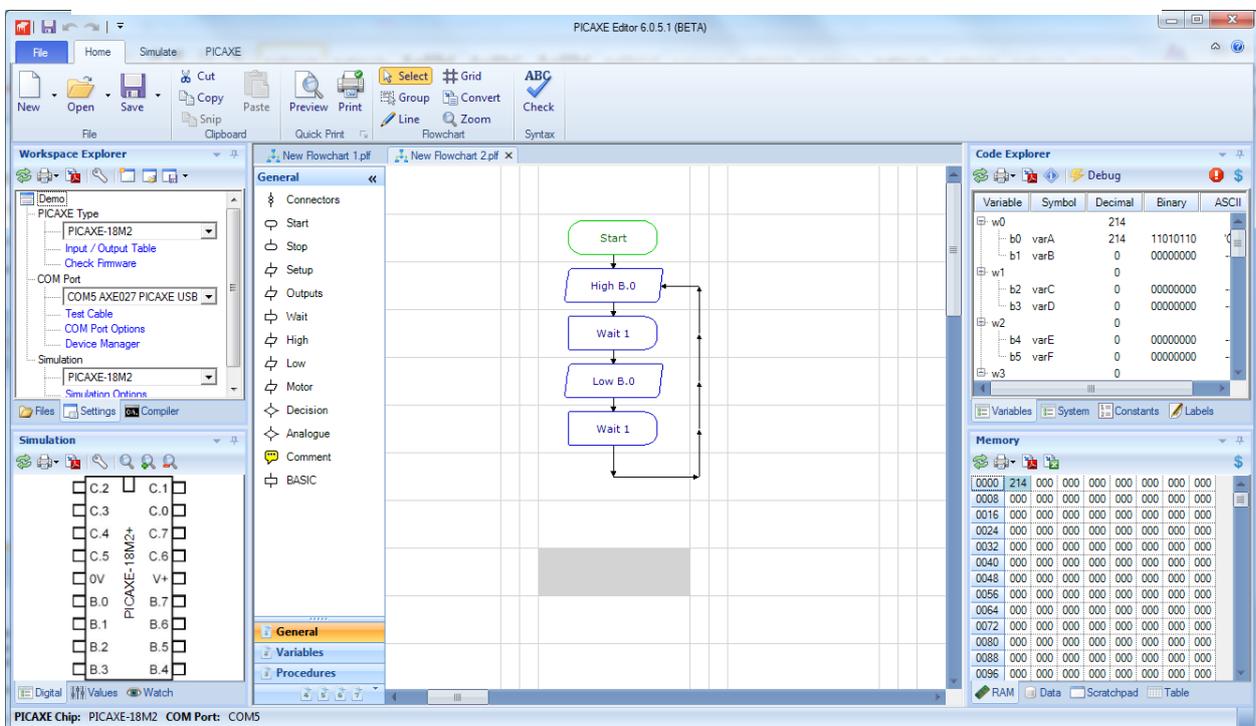


# A guide to using flowcharts within 'PICAXE Editor 6' to simulate and program a PICAXE microcontroller



© Copyright Revolution Education Ltd and New Media Learning 1999-2014.

Copyright is waived in the following circumstances: small number of copies may be made for use in the purchaser's school/college for use alongside PICAXE hardware.

These copies may not be sold or made available outside the purchaser's school.

## Overview

PICAXE Editor 6 provides a graphical flowchart environment for designing, testing, editing and downloading control sequences for PICAXE microcontrollers.

*PICAXE Editor 6 now incorporates, and hence replaces, the previous 'Logicator for PICs' product. To select the Logicator colour scheme right click over the flowchart and select the Colour Scheme menu.*

The wide range of PICAXE commands allows the user to control output devices, such as motors and LEDs that are connected to the PICAXE microcontroller. We 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 PICAXE microcontroller.

This section of the manual explains how the most common commands are used, giving examples of the common 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 flowchart

### 2. Outputs

This section shows: how to switch output devices and motors connected to outputs of a PICAXE microcontroller, using *Outputs*, *Motor*, *Sound* and *Play* 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 PICAXE microcontroller.

### 3. Inputs

This section shows: how to check the state of digital sensors connected to a PICAXE 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 PICAXE 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 *Random* 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 PICAXE microcontroller's EEPROM memory.

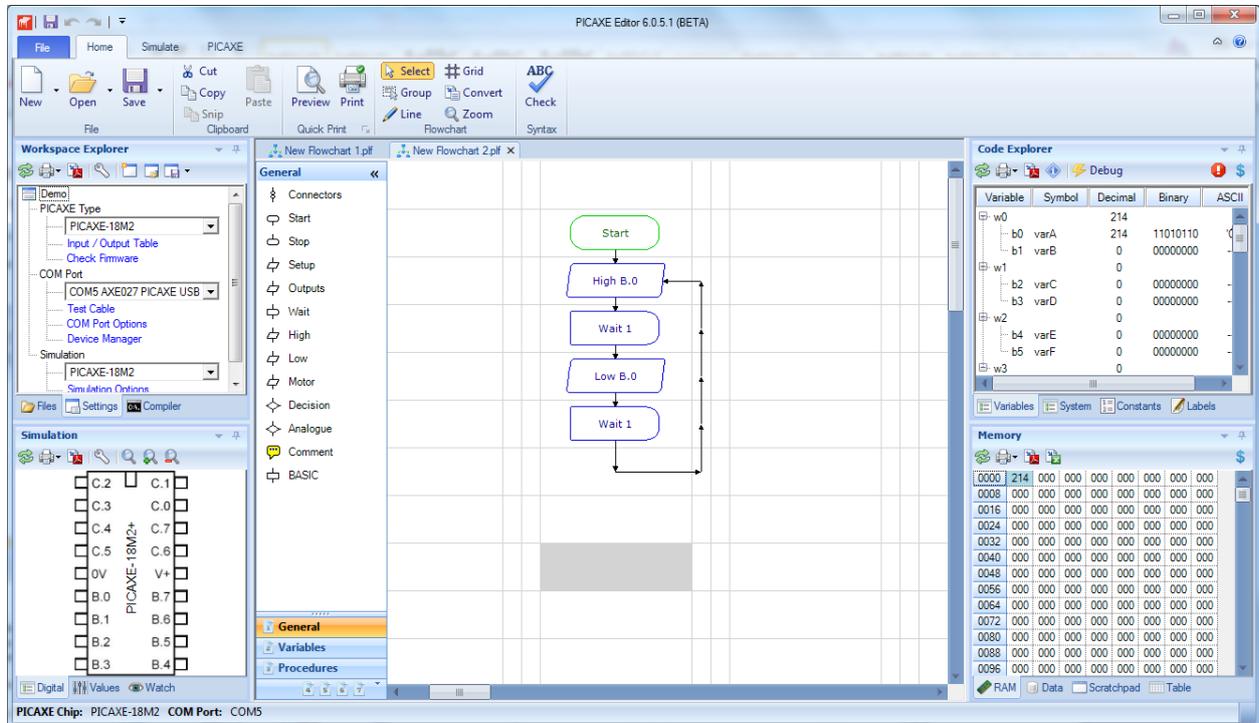
## Quick Start

If you are unfamiliar with the flowchart 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*. Use File>Open Samples and Section 1 ("How to build, edit and test run a flowchart") as a reference to help learn how flowcharts operate.

# Section 1. How to build, edit and test run a flowchart

To create a new flowchart click File>New Flowchart or use the toolbar 'New Flowchart' button.

You will then see the following screen:



Flowchart - This is the central area where your flowchart is drawn.

Toolbox - This is the collection of available commands to drag onto the flowchart

Ribbon/Toolbar –

This is the collection of shortcut buttons at the top of the screen to save/paste etc.

Workspace Explorer -

This is where the PICAXE microcontroller type, COM port etc. are selected

Simulation Panel –

This displays the animated simulation when the program is run 'on-screen'

Code Explorer / Memory Panel –

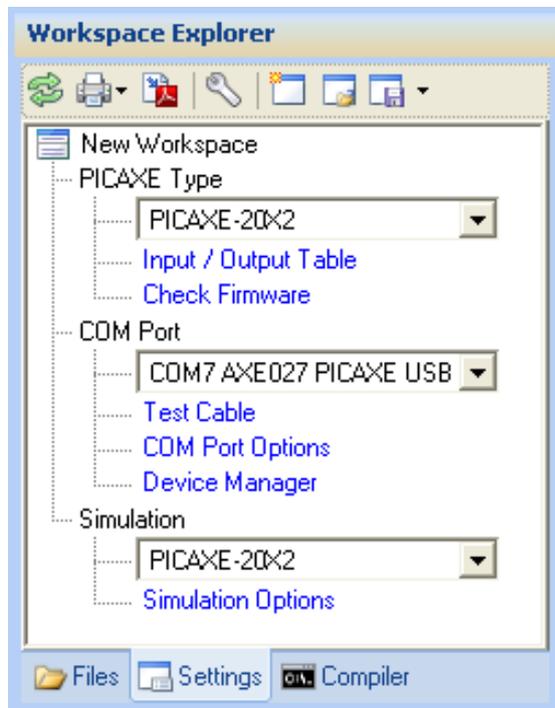
These display the values of the variables when a simulation is in progress

Statusbar -

This is at the bottom of the screen and displays the simulation speed slider etc.

## Selecting the correct PICAXE type

Before the flowchart is drawn the correct PICAXE microcontroller chip type, download cable COM port and simulation image should be selected from within the Settings tab on the Workspace Explorer.



*Note that the input/output pin configuration (e.g. for a PICAXE-08M2 chip) is setup within the Start command on the flowchart, not within the Workspace Explorer.*

Note that if you have the wrong PICAXE chip selected the available input/output pins displayed in the flowchart command cell dialogs will not be accurate. If the PICAXE type you are using is not currently shown in the drop down list use the File>Options>Compilers menu to display the chip type you wish to use.

*NOTE: This section deals only with drawing the flowchart. Details of how to use the various commands are given later.*

## Adding a command cell

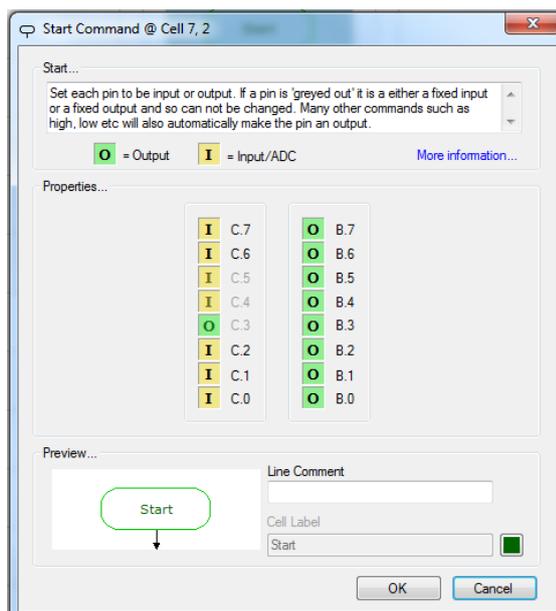
Drag the required command from the toolbox and place it on an unoccupied cell. Most commands have their own Cell Details dialog box which allows you to enter the command details. Double click on the command to open its Cell Details dialog box, and set the details of the command as required. When you have set the necessary details, click OK to close the dialog box.

## Start and Stop commands

A Start command marks the point where the flowchart starts running. When the PICAXE microcontroller is reset or powered up, the flowchart starts at the first Start command. Every flowchart must have at least one Start command. A flowchart will stop running whenever a Stop command is reached.

For PICAXE-M2 parts you can have up to 8 Start cells on each flowchart.

The first Start command is also used to set which of the PICAXE microcontroller pins you wish to use as inputs and which pins you wish to use as outputs.



## 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". Each cell is given a default label automatically, but you can edit this on most commands if desired.

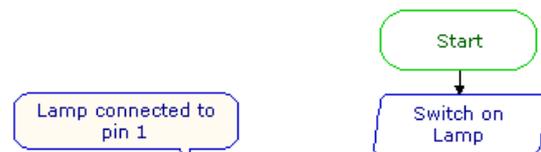
You can also add a line comment above the cell.

The cell label and line comment do not affect the operation of a command; they are only a label for 'humans' to read.

## Comment commands

Comment commands allow you to add longer explanatory notes to a flowchart. The number of characters actually appearing in a Comment cell on the flowchart will depend on factors such as the Zoom setting and screen setting.

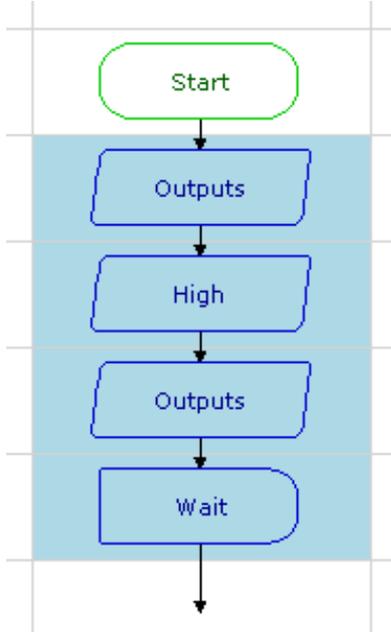
Comments have no effect on the operation of a flowchart.



*Explanatory information can be added to the flowchart 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. Alternately use the Group Select Tool from the toolbar.



*A block of commands in the selection frame*

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

## 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 flowchart or into a different flowchart.

Alternatively, you can copy commands or blocks of commands within a flowchart 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.

## Inserting/deleting rows/columns

Right click over the flowchart and select Column / Row and then Insert or Delete as required from the context menu.

Note also that if you drop a command over an existing command the whole flowchart will be automatically 'shuffled down' so that the new command can be inserted in a newly added row.

## Line Routes

Lines can be drawn through the middle of a cell, or in either one of the two rails between cells. Lines must be drawn in the direction that you want flow to take when the flowchart runs.

## Drawing Lines



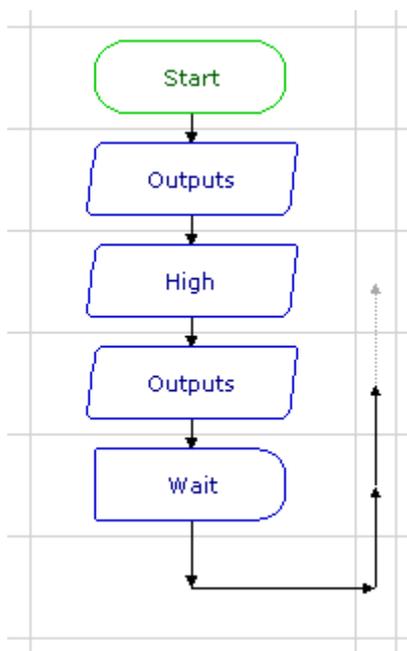
Click on the Line Drawing icon on the toolbar to select drawing mode.

The mouse cursor changes to a pen icon.

Click at a command cell where the line should start. Now drag the pointer along the straight route you wish to draw. A dotted line will show where the route will be drawn. At a corner release the mouse button to complete the line. Then click and drag again to continue drawing a new line.

Right click to end drawing mode.

Drawing mode can also be deselected by clicking the arrow icon on the toolbar.



*Routes can be drawn through cells or between rails.*

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

## Tip

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

## Deleting Lines

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.

## Grid

The grid can be hidden or displayed via the Grid toolbar icon. Lines can be drawn in either the large command cells or the small line cells.

## Connectors

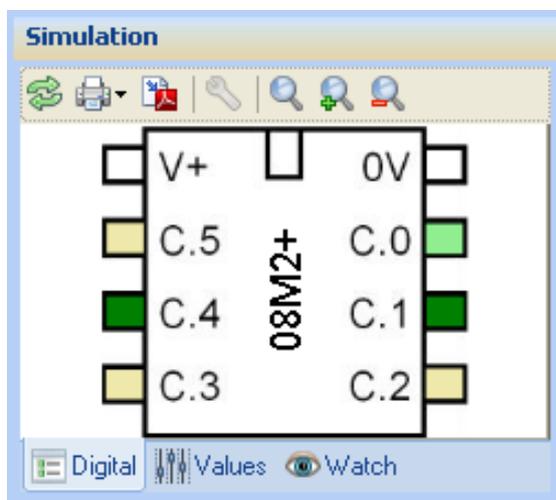
Lines cannot cross. Therefore to connect a line to a completely different position in the flowchart use a connector pair. Draw the line into the first connector and then out of the second connector (which is dragged to a different flowchart position).

## How to test run a flowchart

Before you download a flowchart to a PICAXE microcontroller, it is useful to be able to check that it works as you intend it to. Flowcharting mode has a number of features that allow you to test run the flowchart in the software.

### 1. The Digital Panel

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



### 2. Simulating digital inputs

To change the state of an input simply click on the input in the Digital Simulation Panel.

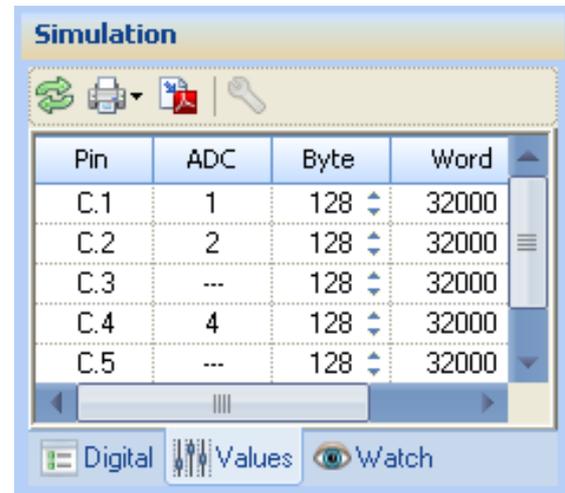
The function keys on the computer keyboard may also be used to simulate portC inputs while a flowchart is running.



<Shift> + Function keys <F2> to <F9> will simulate digital sensors connected to inputs C.0 to C.7 on a PICAXE microcontroller. Key F2 simulates input 0; key F9 simulates input 7.

### 3. Simulating analogue inputs

The Values Panel allows you to simulate the changing reading from analogue sensors while a flowchart is running. Identify the sensor which you wish to simulate, and change the value in the byte column to vary the simulated reading from 0 to 255.



### 4. Run and Stop



To test run a flowchart click the Run button on the toolbar or press <Ctrl>+<F5>



To stop a flowchart running, either click the Stop icon or simply click anywhere in the flowchart.

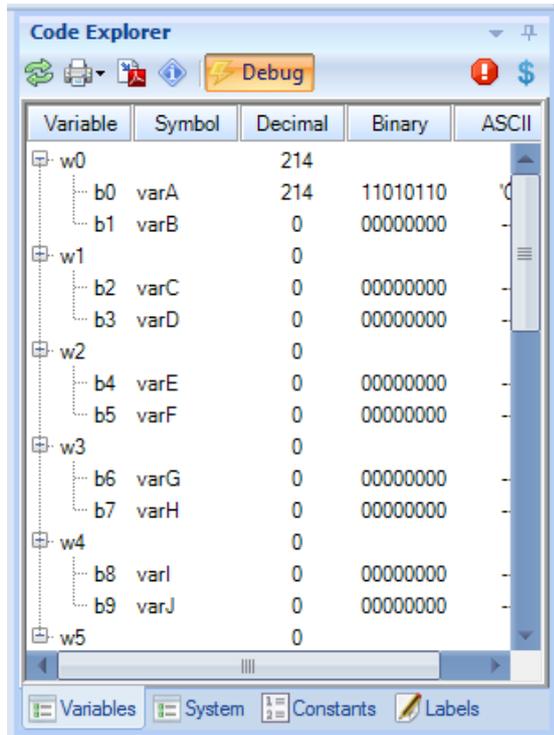
As the flowchart runs, the flow of control is highlighted so that you can follow it. If you want to slow down the speed at which flow is highlighted is controlled by the slider in the status bar in the bottom right corner of the screen

### 5. Breakpoints

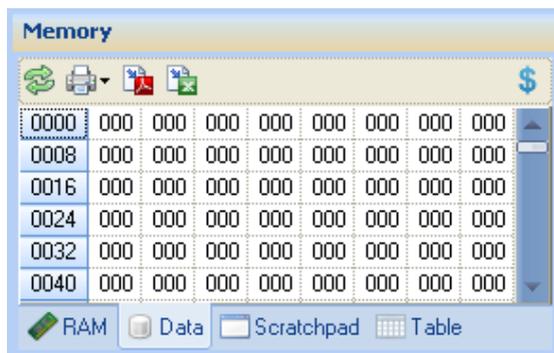
Right click to add a breakpoint flag to any cell. When the simulation reaches this point the flowchart will then pause.

## 6. Variables and EEPROM display

If your flowchart uses variables, it is useful to display the Code Explorer and memory windows when you test run it. The changing values of any of the variables that are used in the flowchart will be displayed as the flowchart runs.



To manually change a value pause the simulation and then double click on the appropriate value in the memory panel.



## Downloading a flowchart into a PICAXE chip

1. Connect your PICAXE project to the computer by the AXE027 USB download cable.
2. Connect power to the PICAXE circuit board, normally for 3x AA batteries (4.5V).
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 button on the PICAXE toolbar or press <F5>.
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 flowchart, the longer the programming time.
7. If successful, programming is complete when the progress bar disappears.

If you are having difficulty programming try the hard reset procedure as described in part 1 of the PICAXE manual.

## Displaying and using BASIC

PICAXE Editor is also able to convert any complete flowchart into BASIC.

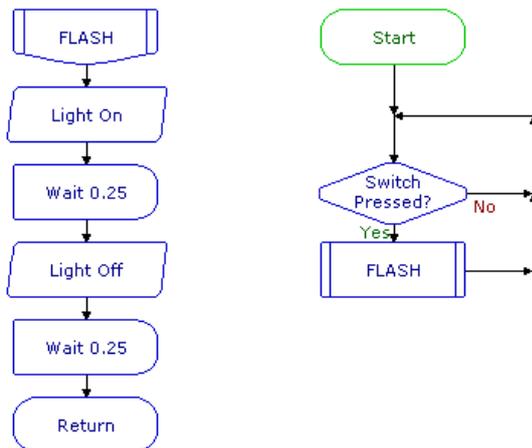
BASIC is a text based language that is used throughout the world to program everything from PICAXE microcontrollers to personal computers.

### Why Convert?

Flowcharts are easy to understand and quick to build. BASIC programming languages offer more complexity to advanced level users and the ability to convert a flowchart into BASIC offers a way of learning how BASIC programs are written.

### Converting a flowchart into BASIC

1. Design your flowchart as normal and test the program using the flowchart simulation tools.
2. From the PICAXE toolbar tab, choose the 'Convert to BASIC' button.
- 3 The BASIC text window is then displayed containing the conversion of your flowchart.



```
1 'BASIC converted from flowchart:
2
3 [Symbols]
40
41 main:
42
43 Cell_7_4:
44     if b1 <> b1 then
45
46         goto Cell_7_5
47     end if
48     goto Cell_7_4
49
50 Cell_7_5:
51     gosub prc_FLASH
52     goto Cell_7_4
53
54 prc_FLASH:
55     high A.0
56     pause 250
57     low A.0
58     pause 250
59     return
```

### Notes:

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

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

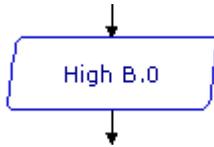
It is not possible to convert from BASIC backwards to a flowchart.

Using the BASIC command you can add sections of BASIC code into a flowchart.

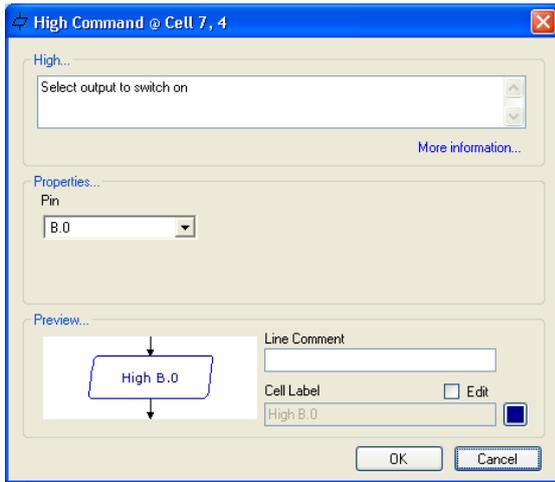
For full information on the use of BASIC to program PICAXE chips see the PICAXE website at [www.picaxe.com](http://www.picaxe.com)

## Section 2. Outputs

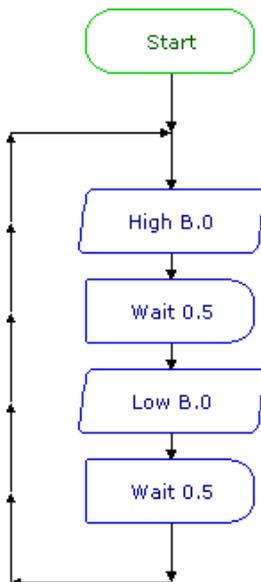
### High / Low commands



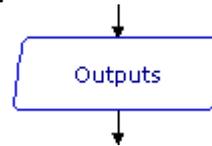
The high and low commands are used to switch a single output on or off.



High command Cell Details box

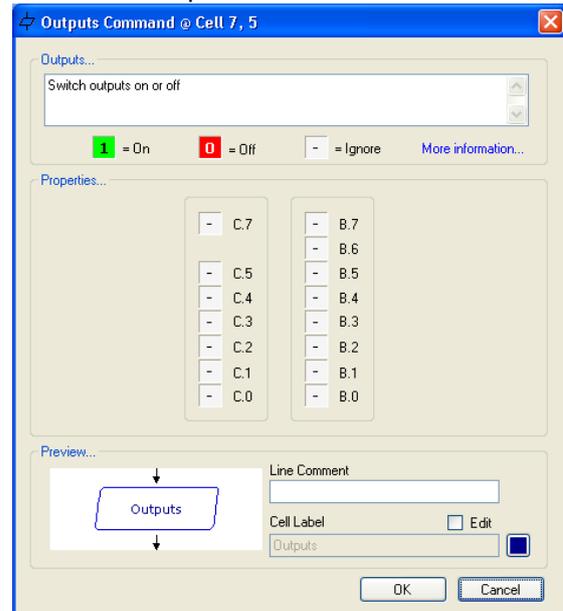


### Outputs command



We can use an Outputs command to switch on or off multiple outputs at the same time.

The Cell Details box (below) shows the number of outputs available for use.

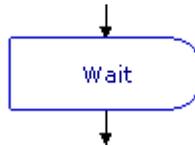


Outputs command Cell Details box

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

-  This means: switch this output on.
-  This means: switch this output off.
-  This means: ignore this output  
i.e. leave it in the state in which it was set by the previous commands.

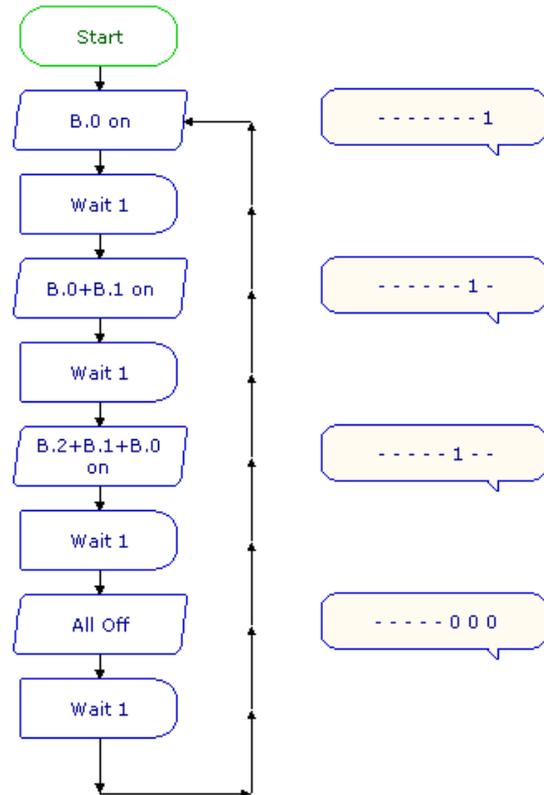
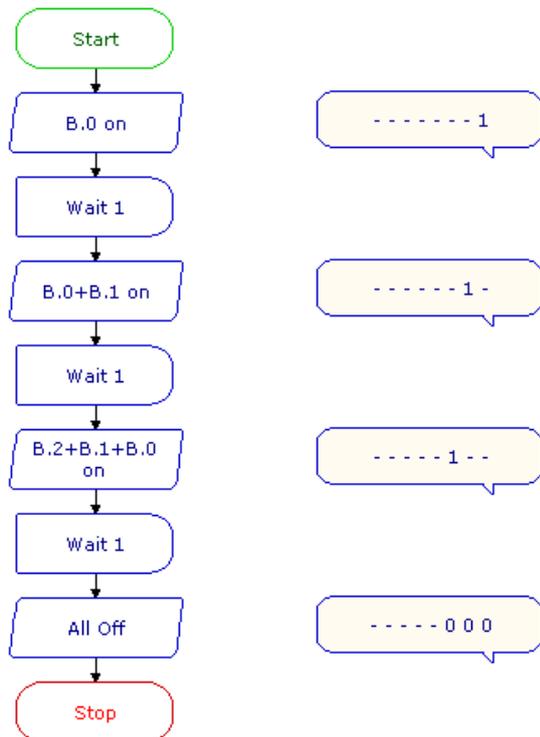
## Wait command



A Wait command makes a running flowchart 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.

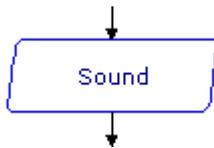
### Example

A PICAXE microcontroller has 3 LEDs connected to outputs 0, 1 and 2. The flowchart shown top right will switch them progressively 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.

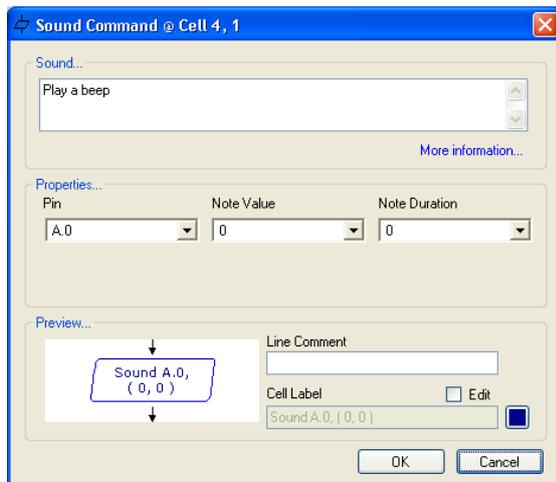


The flowchart shown above will continue to repeat the sequence until power to the chip is switched off. Notice that another Wait command has been added to the repeating sequence. The PICAXE microcontroller operates so quickly that, without Wait commands, the LEDs would switch on and off so quickly that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

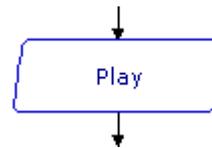
## Sound Command



Use a Sound command to send a pulsed signal to a piezo sounder connected to an output of a PICAXE microcontroller. You can use a sequence of sound commands to play a simple tune.



## Play command



Most PICAXE chips have 4 pre-programmed internal tunes, which can be output via the Play command.

As these tunes are often included within the PICAXE 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.



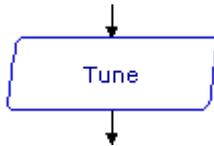
The Tunes are:

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

The following example will play Happy Birthday while flashing output 4.



## Tune Command



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

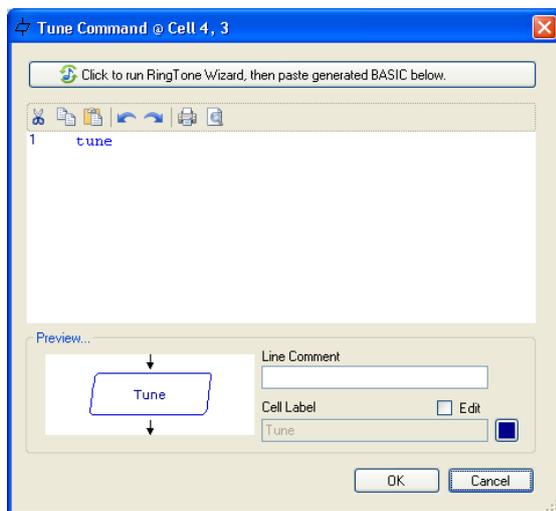
The difference with Tune command is that it converts RTTTL mobile phone ringtone files to PICAXE tunes and plays them with or without flashing outputs.

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. The Tune Wizard converts these ringtones to a PICAXE tune command upon download.

configured the I/O pin 4 as an output using the Select PICAXE dialog in order to see all of the available options.

In order to use the ringtone in a simulation you must click the 'Generate .wav' button. Finally, choose the OK button.

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



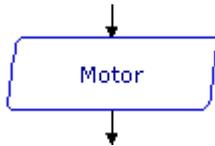
*Play User Tune dialogue box*

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.

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

Select the output to flash 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

## Motor command

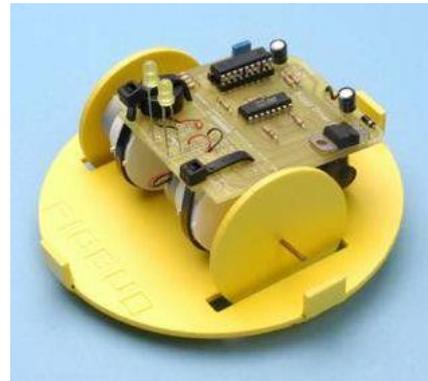


The Motor command allows you to use pairs of outputs on a PICAXE microcontroller to switch a motor forward, reverse or off.

Use its Cell Details box to set the motor or motors to drive forward or reverse; or to stop.

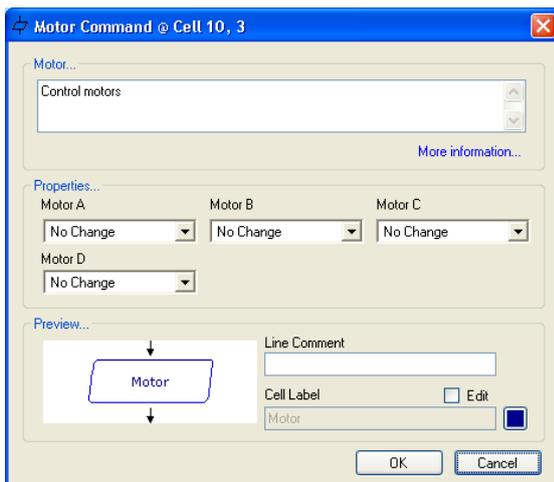
Remember that the direction in which a motor turns depends on which way current flows through it, and therefore on the way it is connected to power. For this reason, the direction arrows indicate only that the directions will be different; not the actual direction in which motors in the project will turn.

disable them, and set unused outputs in a Outputs command to their 'ignore' state



### Example

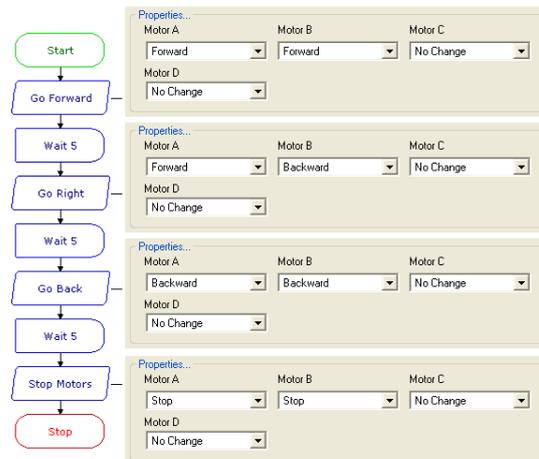
A steerable buggy is usually driven by two motors, one powering each driving wheel with a free-running jockey wheel to keep it stable. The flowchart below shows how a sequence of Motor commands can be used to drive a buggy which has one motor connected to outputs 0 and 1 (motor A) and the other motor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



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

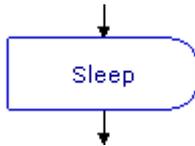
NOTE: Outputs and Motor commands both use the same output lines to switch the outputs of a PICAXE microcontroller. The default state of both commands is such that they will automatically switch off any outputs that are not set 'on'.

So, to avoid inadvertently switching off an output device, un-check the select boxes of unused motors in a Motor command to



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



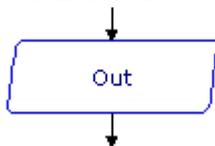
This command puts the PICAXE microcontroller into low power mode for a specified number of seconds.

This command can be used 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.

The Cell Details box is used to set the number of seconds of sleep mode required (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.

## Out command



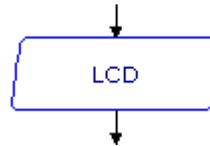
When flow passes through an Out command, the output portB is set to the binary value of the number entered in the command.

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

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

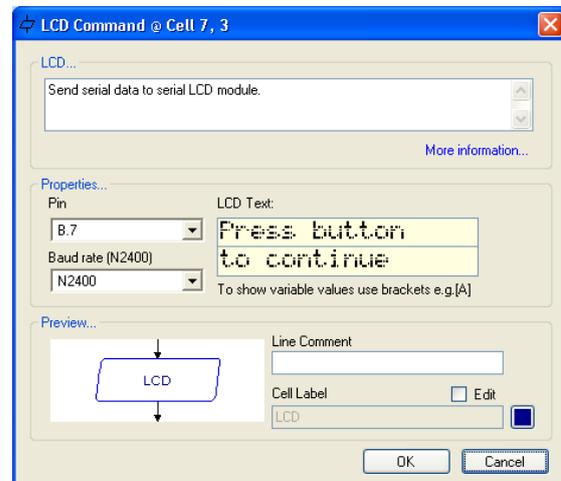
In the table above the 'bits' can be switched on by sending the selection value of the bit., e.g. 'Out 4', which will turn on an LED at B.2.

## LCD



This command can be used to display a message on an LCD screen attached to a PICAXE-driven circuit board.

In the cell details window a message can be written over two lines if required and an output pin is assigned.



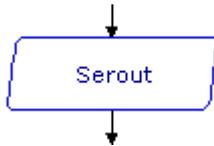
This command will be simulated if the flowchart is run from Simulate > Run.

A small LCD screen window will pop up during the run to display the LCD message (make sure the simulation pin for the LCD is correctly set under File>Options>Simulation)

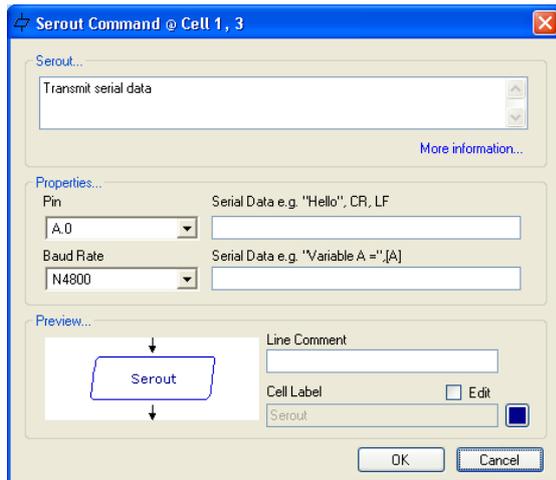


Simulated LCD screen

## Serout Command



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



The first box is used to select the output pin on the PICAXE 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 the ASCII box must be unchecked.

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".

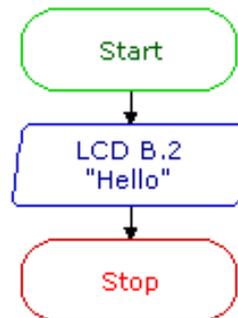
In the latter case, ASCII codes must be separated by a comma.

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.

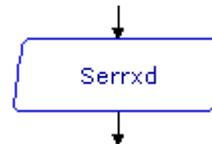
## Example

The flowchart shown below will display the word "Hello" on an LCD screen connected to output pin B.2 of a PICAXE microcontroller.



A sequence to display the word 'Hello'

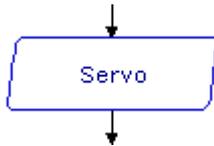
## Sertxd Command



The sertxd command is similar to the serout command, but acts via the serial output pin rather than a general output pin. This allows data to be sent back to the computer via the programming cable. This can be useful whilst debugging.

*See the PICAXE Manual for more information*

## 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 lose its position.

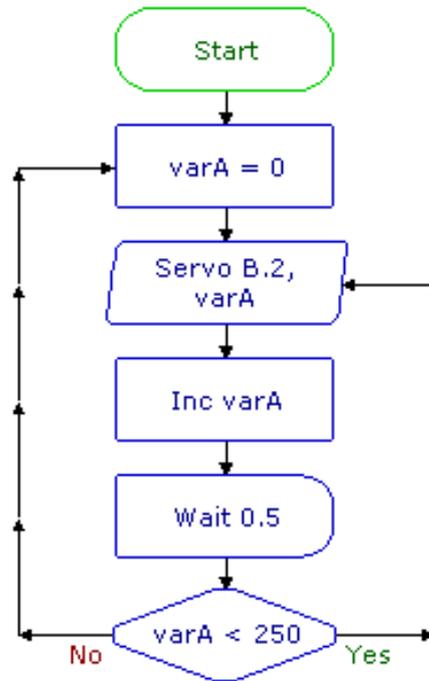
The Servo command starts a pin pulsing high for length of time pulse (x0.01 ms) every 20ms.

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.

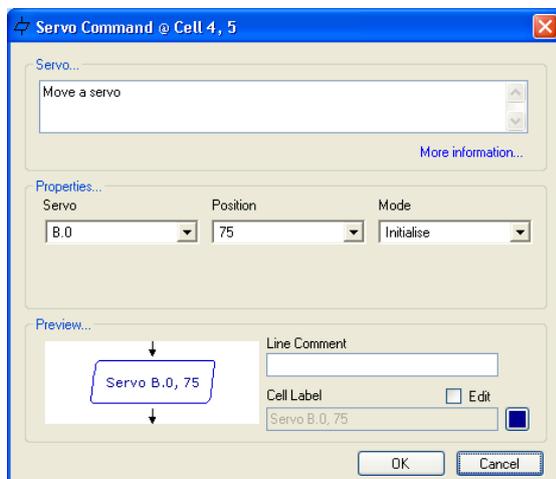
The cell details for the servo command have two settings; the output pin that the servo motor is connected to and the pulse time. The pulse time can be a value held in a Variable. Note that the value for the pulse time MUST be in the range 75 to 225. The servo motor may malfunction if the pulse is outside of this range.

## Example

The flowchart below will move a servo motor attached to output B.2 from one extent of its travel to the other, repeating continually.

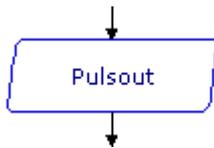


Using the Servo command



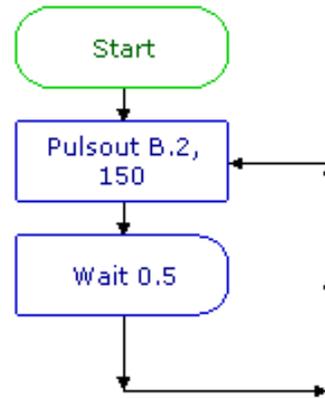
Servo command cell details

## Pulsout Command

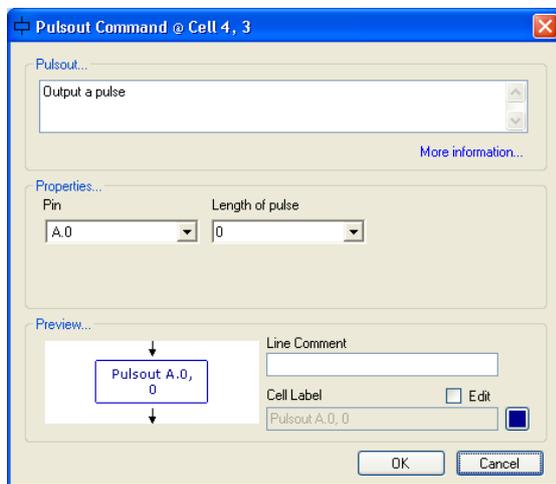


The PulseOut command generates a pulse through the chosen output. If the output is initially off, the pulse will be on, and vice versa.

There are three items to set in the cell details box for the PulseOut command below; the output pin to send the pulse through, and the length of time that the pulse should operate for. The final option is the multiple for the value entered (10us, 1ms or 10ms).



*Using the PulseOut command*



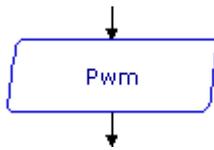
The pulse time is in multiples of the multiplier selected (the greatest possible legal value is  $65535 \times 10\mu\text{s} = 655350 \mu\text{s} = 655 \text{ ms}$ )

Note that PICAXE Editor cannot simulate the action of the PulseOut command.

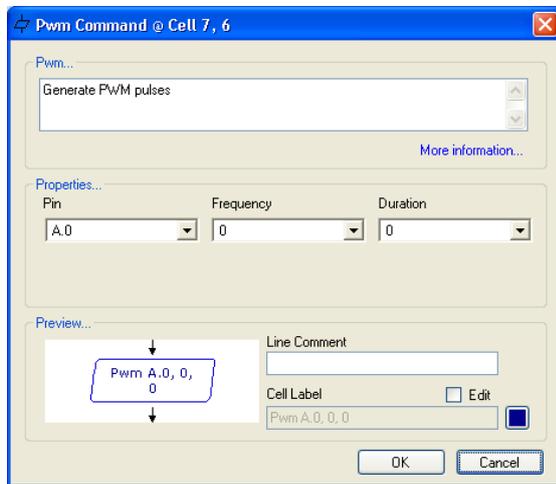
### **Example**

The flowchart below sends a pulse of 1500us (1.5ms) out of output pin B.2 every half second.

## PWM Command



The PWM command is used to provide 'bursts' of PWM output to generate a pseudo analogue output on the PICAXE-08/08M (pins 1, 2, 4). This is achieved with a resistor connected to a capacitor connected to ground; the resistor-capacitor junction being the analogue output. PWM should be executed periodically to update/refresh the analogue voltage.



*PWM dialogue window*

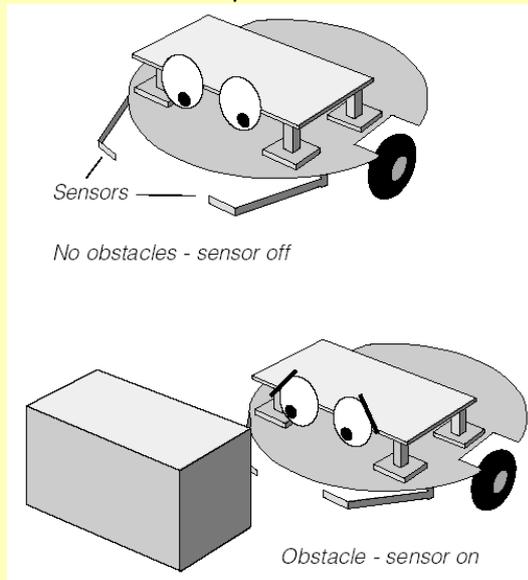
The parameters are: the Output pin used, the analogue level 0-255 (Duty) and the number of 5ms cycles that specifies the duration.

## Section 3. Inputs

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

### Example

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



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.

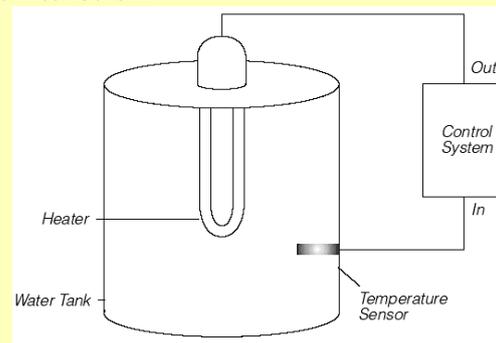
A microswitch is a digital sensor. It has only two states - "on" (or "closed") and "off" (or "open").

These states are often labelled by the digits 1 and 0, which is why the sensors are called digital sensors.

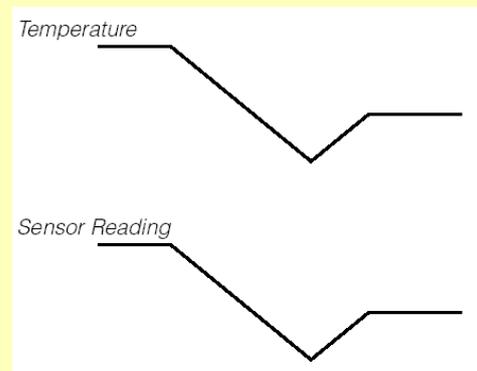
### Example

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.

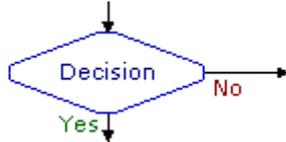


## Section 3A. Digital Inputs

### Decision command

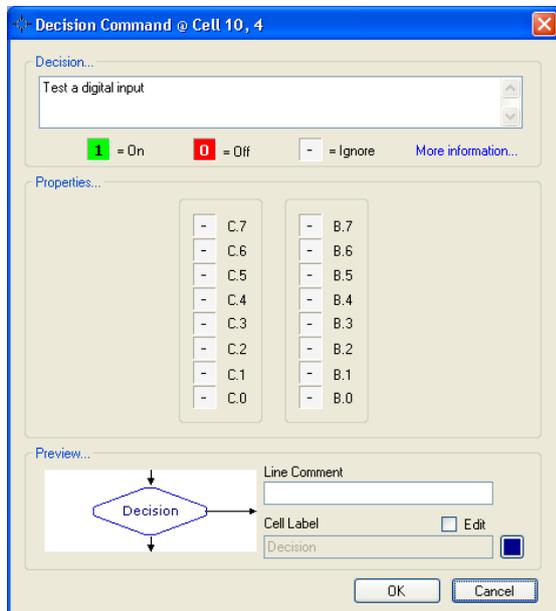
Use this command to test the state of a digital sensor connected to a digital input of a PICAXE microcontroller.

When flow reaches a Decision cell, it continues in either the Yes or No direction depending on the result of the decision test.



*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 below. The Input Pattern area shows the number of digital inputs available for use on the PICAXE microcontroller you have selected. Any unavailable inputs are shown without a number label and cannot be clicked upon.



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

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

### Drawing routes from a Decision command

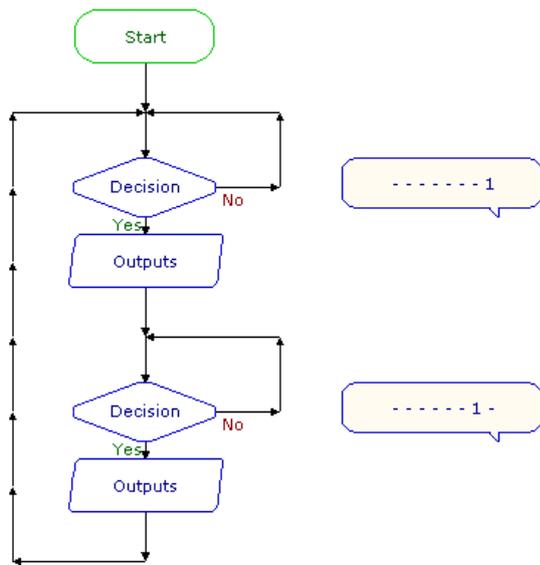
The first line that you draw from a Decision command is the “Yes” direction, and the second line is the “No” direction.

Tip; you can swap the “Yes” and “No” routes by right clicking on the command and choosing “Swap Yes/No” .

**Example**

A PICAXE 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.

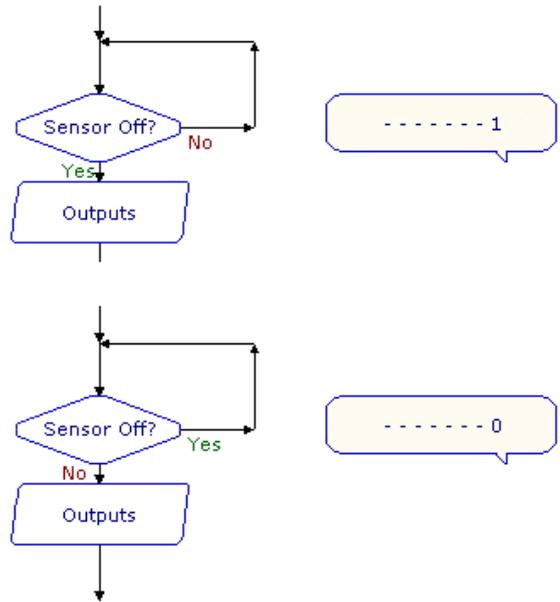
Below is a flowchart for the control system, showing how the two Decision commands are set. When the chip is powered, the pressure pad is tested. If it is not pressed, flow will go in the N route and will continue to go round this loop until the pad is pressed. When the pad is pressed, flow will go in 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.



Security system

A similar flowchart 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).

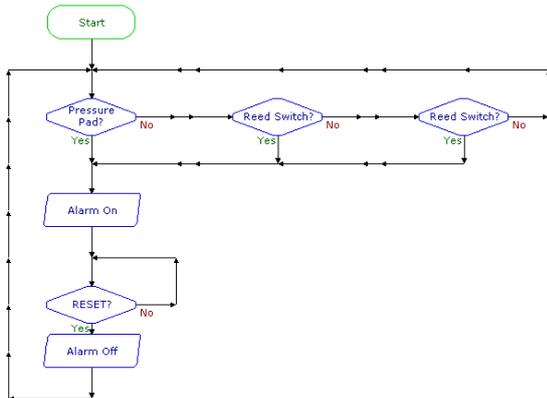
The flowcharts below shows two different ways of using a Decision command to test the micro-switch in this system.



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

**Example**

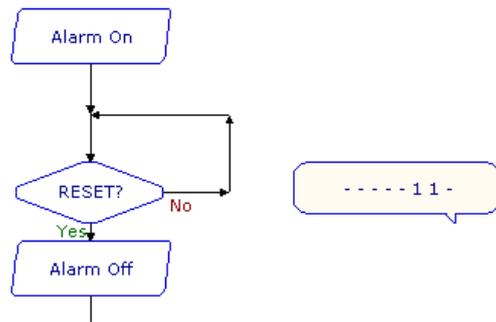
Home security systems often have a number of sensors in different parts of the house. If any one of them is activated, the alarm is switched on. The flowchart below shows a security system which has three sensors and a reset switch.



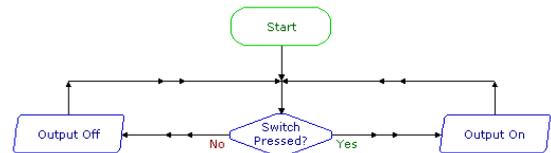
Security system with three sensors (OR function).

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 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. The flowchart below shows how you can set a Decision command to test two switches in this way.



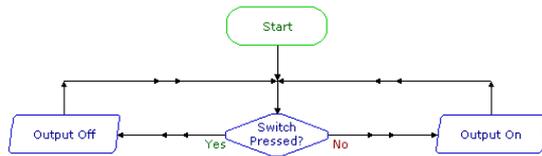
Decision command set to check if two switches are pressed at the same time (AND function).

**Example**

In the flowchart shown below, 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.

This is the equivalent of a simple electrical circuit containing a normally open push switch and an output device.

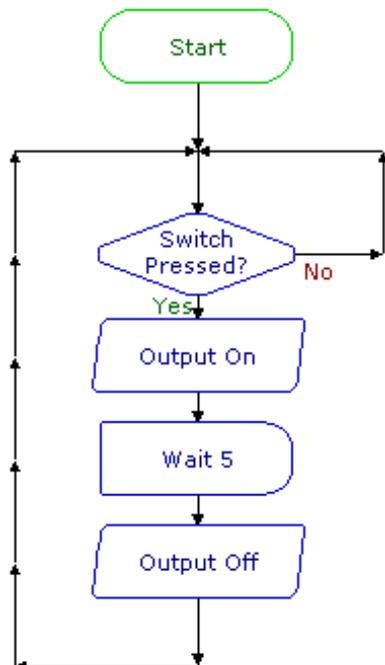
The difference is that you can change the way the system works in software, by simply changing over the Yes and No on the Decision command :



*"Normally closed" switch effect.*

**Example**

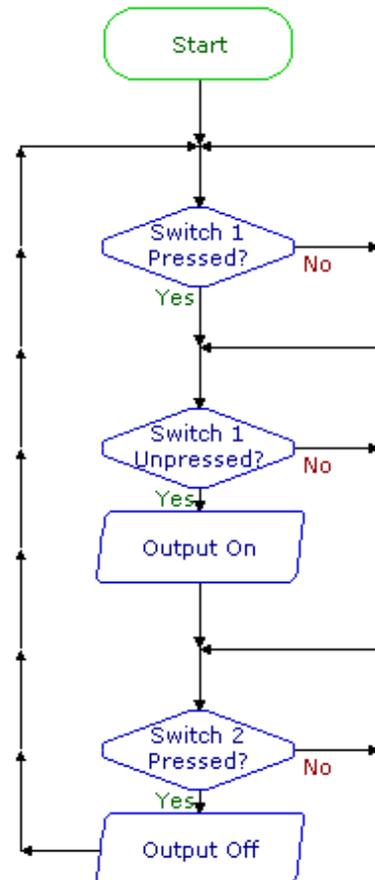
A mono-stable device has only one stable state. It changes state when it is triggered by an input, and stays in that state for a certain time. It then goes back to its original state.



*mono-stable" function.*

**Example**

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. The flowchart below shows how this function can be produced in PICAXE Editor.



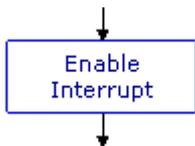
## Interrupt



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 a Return command. It then returns to the point which it was at when the Interrupt occurred.

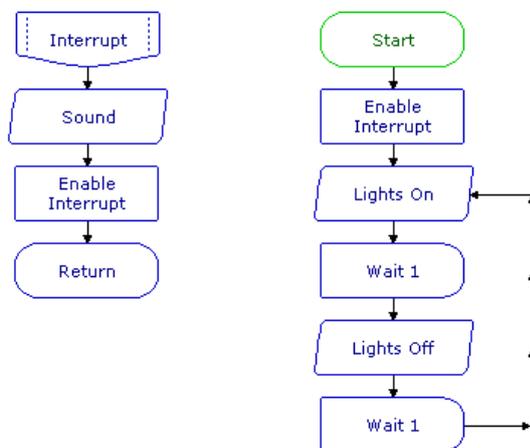
In order to use an Interrupt, the PICAXE must be told to look for the input condition. This is done through the Interrupt Setup command. There are two options in the command – Enable or Disable.



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

### Example

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

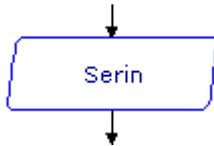


The Interrupt is used to capture the flow and play a sound. The interrupt is then enabled once again before returning to the point at which it left the main flow.

Note that the Interrupt MUST have an associated Return command and will not be triggered again until this Return command has been reached. There is no limit to the number of commands between the Interrupt and the Return. It is a common technique to add a 'Enable Interrupt' command just before the Return command, so that when the Interrupt sub procedure returns the interrupt is re-enabled.

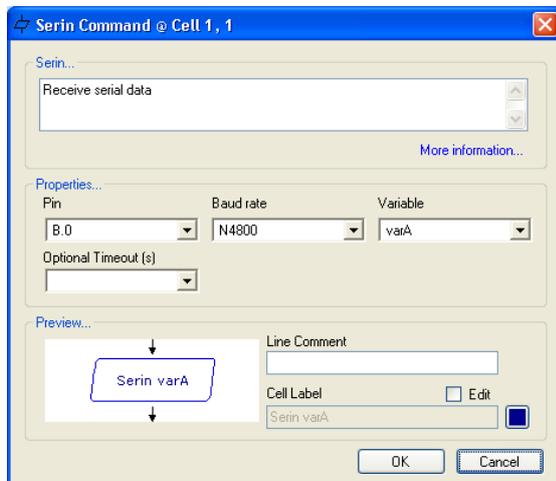
Only one Interrupt can be used per flowchart.

## SerIn Command



The SerIn command is 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.

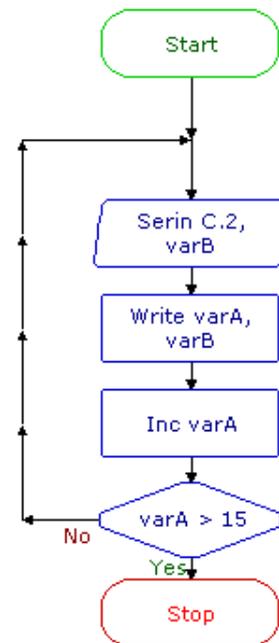
Lastly, the mode option specifies the baud rate and polarity of the signal. When using simple resistor interface, use N (inverted) signals. When using a MAX232 type interface use T (true) signals. The protocol is fixed at N,8,1 (no parity, 8 data bits, 1 stop bit). For best results do not use a baud rate higher than 4800 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

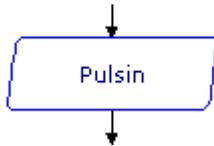
Serial data is being received from another PICAXE chip and needs to be stored in the EEPROM.

In the flowchart shown below, 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



*Using the SerIn command to receive serial data*

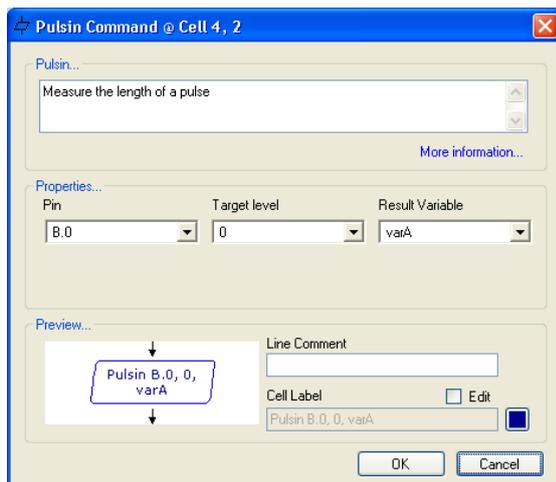
## Pulsin Command



The PulsIn command 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 Pulseln command; the input pin, the State and the Variable to store the result in. The result is measured in multiples of the selected range 10us/1ms/10ms and is a value of 1 – 255.

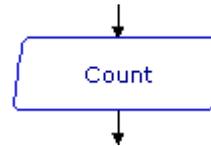


*The cell details box for the Pulseln command*

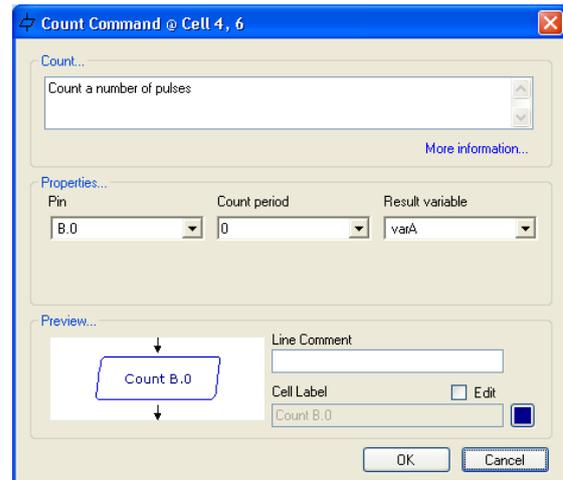
Use the Count command to count the number of pulses with a specified time period.

Because the Pulsin Command works so quickly this command cannot be simulated in the PICAXE Editor software via mouse clicks. Instead it takes the value from the simulation panel.

## Count Command



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.

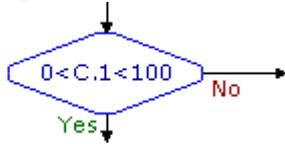


*The cell details box for the Count command*

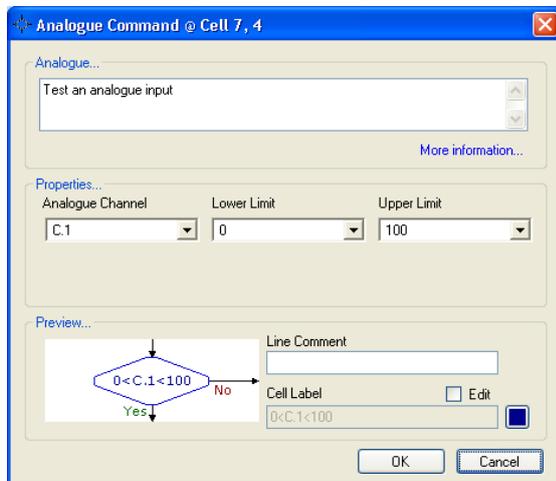
Take care with mechanical switches, which usually cause multiple 'hits' for each switch push as the metal contacts 'bounce' upon closure.

## Section 3B. Analogue Inputs

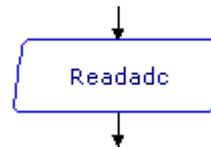
### Analogue command



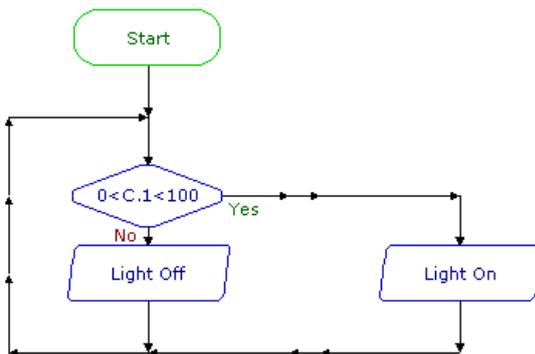
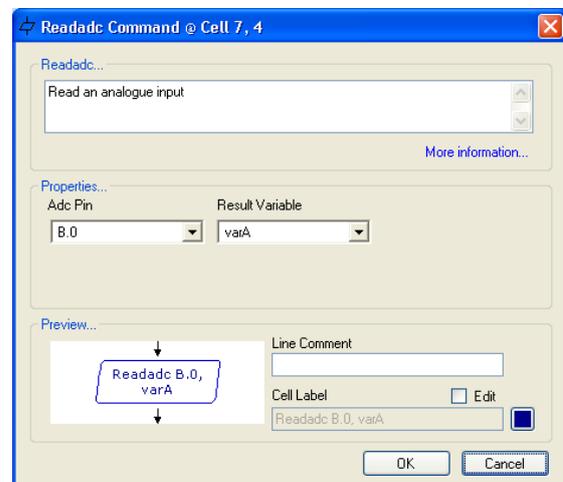
This command is used to read an analogue value and decide if it is between two preset values (called the 'threshold values'). If the value is between the two values the 'yes' route is followed, if not the 'No' route is followed.



### ReadADC



This command is used to read an analogue value from an analogue channel and assign the value to a variable. It is equivalent to using an Expression to set a variable equivalent to an analogue channel, as in the expression: `varA = A1`. The value can then be tested using a Compare command.

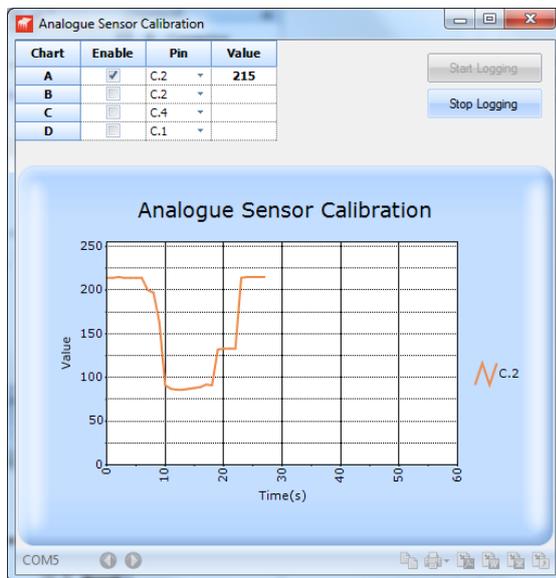


## Calibrating a sensor

To use an analogue sensor it is necessary to know the threshold value, the value at which you want the system to be activated.

Normally this value is found by experimentation – e.g. the threshold value for a light sensor would be the value which is half way between the ‘bright light’ sensor value and the ‘dark’ sensor value.

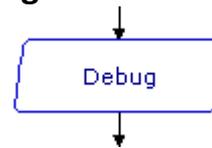
The PICAXE Editor includes a very useful calibration wizard that allows you to see these sensor values by transmitting them in real time via the programming cable to the computer screen.



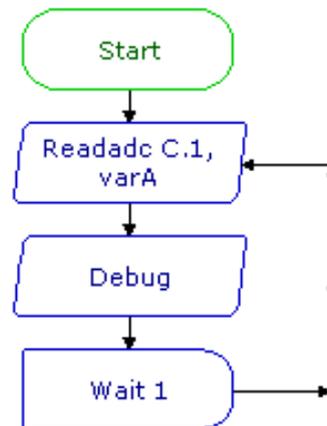
To use the calibration wizard select the appropriate input pins in the table (up to 4 sensors can be tested at the same time) and then click ‘Start Logging’. The sensor values will be updated on screen every second.

The threshold value is then normally calculated as the mid position between the ‘average high’ and ‘average low’ values, so in the graph above a threshold value of 150 would be ideal. Therefore an analogue command would be setup as  $0 < C.2 < 150$

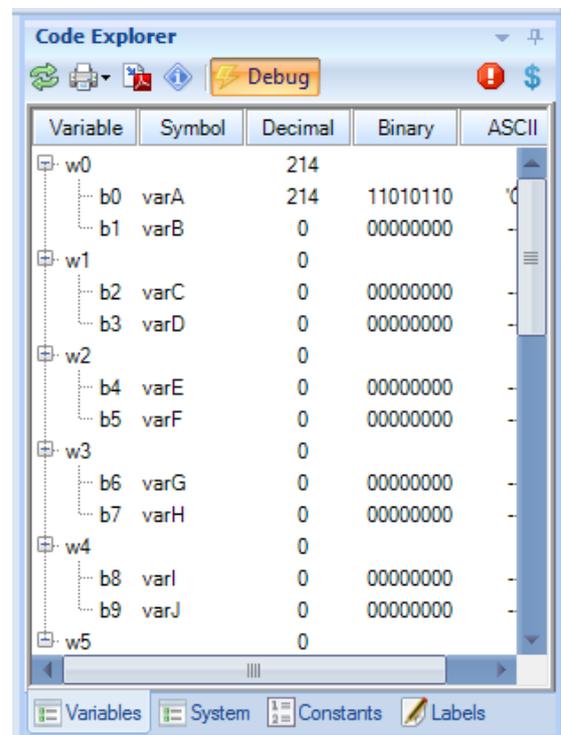
## Debug



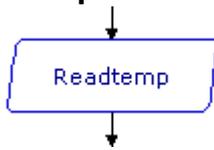
To read analogue values ‘live’ from a PICAXE chip we can also use the Debug command in a loop like in the flowchart below.



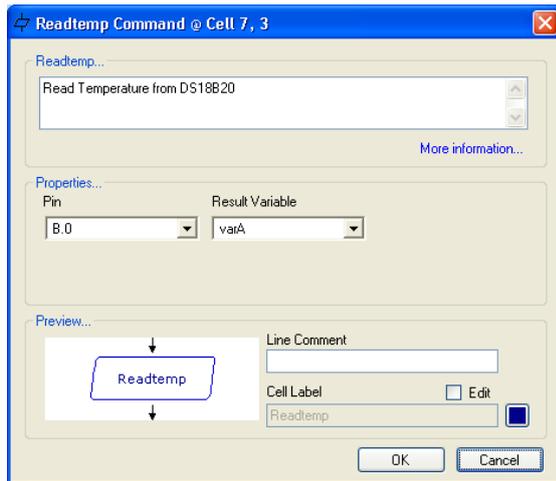
We then click the Debug button on the Code Explorer. The value of varA will update very second.



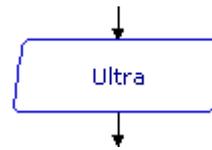
## ReadTemp



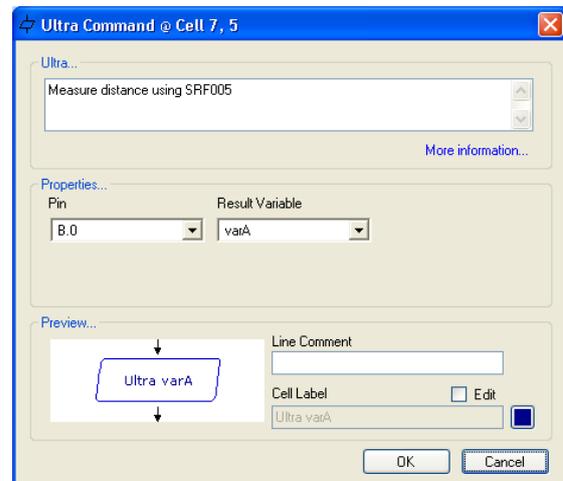
This command is used to read the temperature value (in degrees Celsius) from a D18B20 temperature sensor.



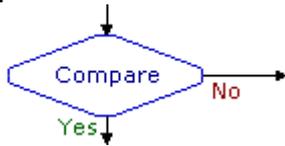
## Ultra



The Ultra command is used to detect an object using the SRF005 ultrasonic sensor. When the output and input pins area assigned to the sensor position the command returns the distance to an object (cm) and assigns this value to a variable.



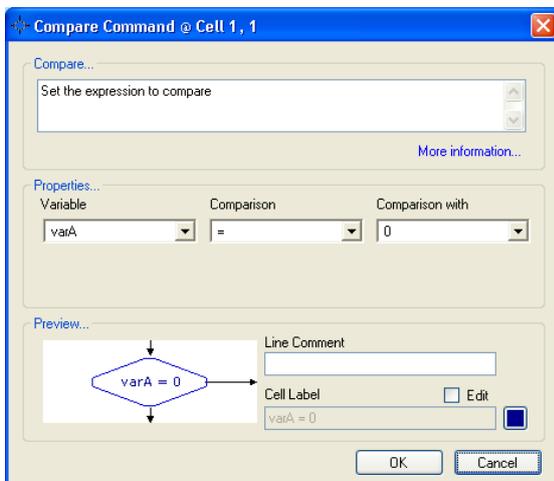
## Compare command



This command can be used to check the reading from an analogue sensor connected to an analogue input of a PICAXE 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 below.



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.

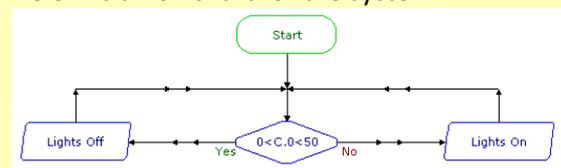
2. Use boxes two and three to complete the comparison. The drop-down list in box two contains a list of operators such as “greater than” (>), “less than” (<), and “equals” (=). Select the one that you require. NOTE: It is

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.

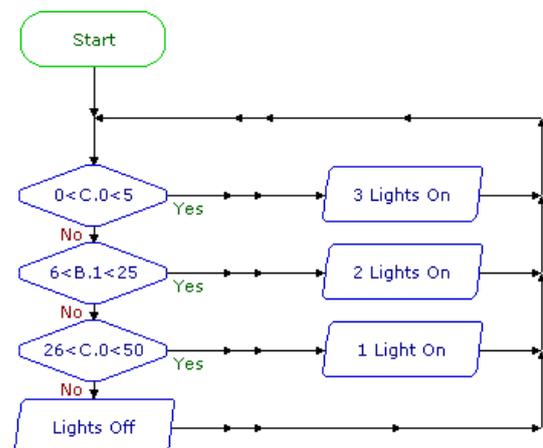
### Example one

A PICAXE 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. Below is a flowchart for the system.



System to switch on a lamp automatically in dark conditions.

The Compare command checks the reading from the light sensor. If the reading is less than or equal to 50, flow will go to the Yes route and switch on the lamp; if the reading is greater than 50, flow will go in the No route and switch off the lamp. The system could be extended as shown below. This system controls three separate lamps, which it switches on one by one as darkness falls.

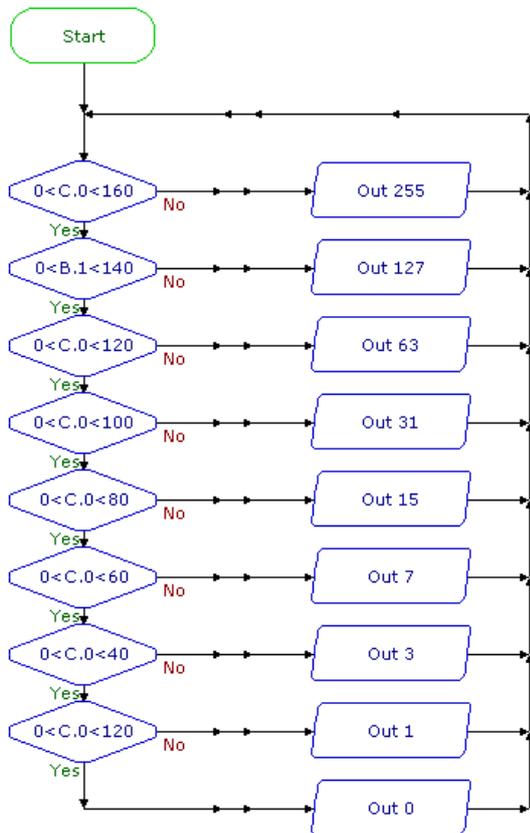


System to switch on three lamps in response to changing light levels.

### Example

A PICAXE 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.

Below is the flowchart for the system. Notice the use of the Out command to switch on combinations of outputs.

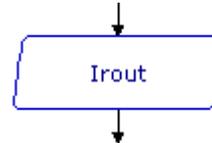


Light meter system

## Using Infrared control

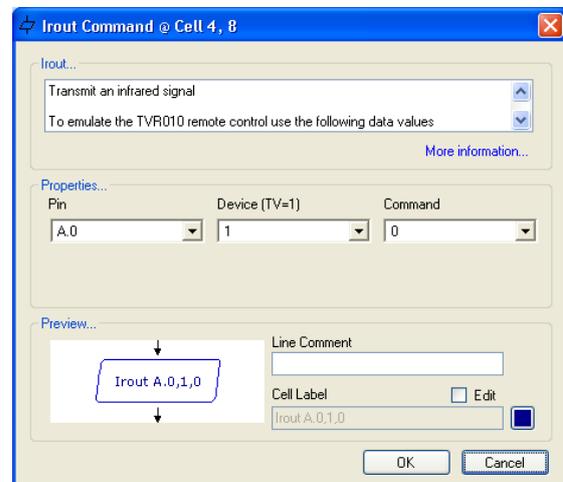
When using PICAXE chips, commands are available to support Infrared communication between PICAXEs and TV style remote controls.

### IROUT



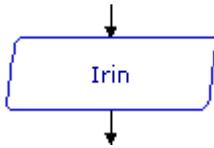
This command is used to transmit the infrared data to a Sony™ protocol device. It can also be used to transmit data to another PICAXE circuit that is using the irin command. Data is transmitted via an infra-red LED (connected on output 0) using the SIRC (Sony Infra Red Control) protocol.

When using this command to transmit data to another PICAXE chip the Device ID used must be value 1 (TV).



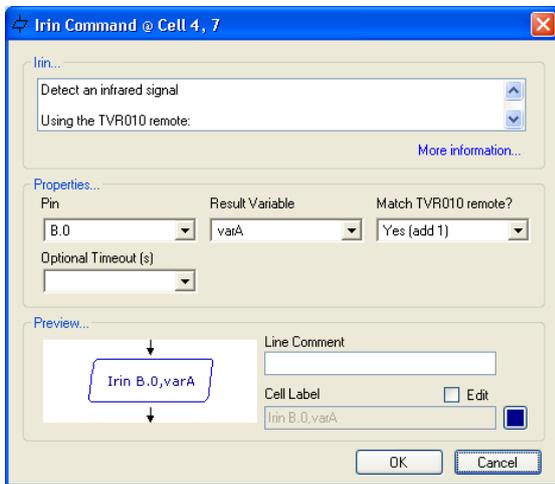
The irout command can be used to transmit any of the valid TV commands (0-127). Note that the Sony protocol only uses 7 bits for data, and so data of value 128 to 255 is not valid.

## Irin



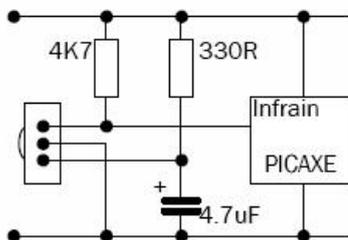
To receive information from an Infrared source, the Infrain command is used. The command will wait for a new infrared signal from an infrared TV style transmitter. It can also be used to receive an InfraOut signal from a separate PICAXE chip.

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



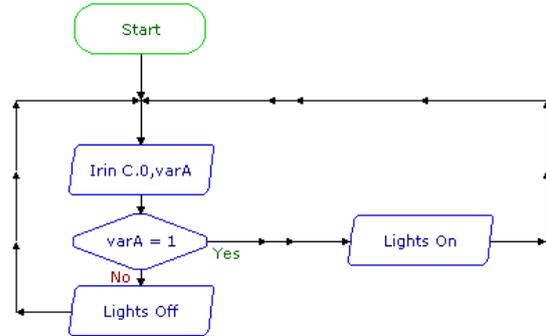
The cell details are simple; only a Variable must be set.

The 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

In the following flowchart a signal is received from a TV Infrared remote control. Lights are switched on if key 1 is pressed.



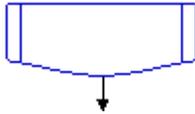
The irin command waits until a signal is received, and saves this as a number in Variable A.

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

## Section 4. Procedures

PICAXE Editor software provides a clear, step-by-step method of building a complex control system, by creating a number of linked subsystems called “sub procedures”.

### How to build a sub procedure



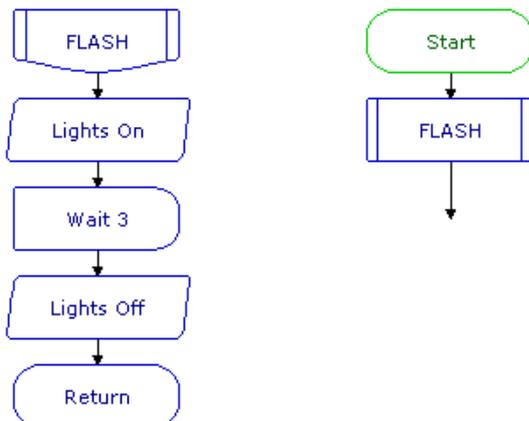
Use a Procedure command to begin the procedure. Drag the command onto the flowchart and place it separately from the START command as shown below. Double click on the command to open its Cell Details box. Type in any appropriate name, and click OK. The software automatically puts the name into capitals.



Placing the Procedure command

Use other commands as normal to create the procedure.

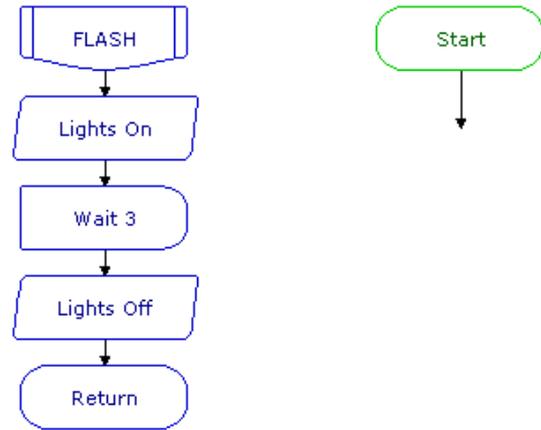
Place a RETURN command at the end of the procedure as shown in the flowchart below.



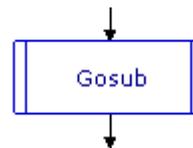
This procedure, called FLASH will switch on selected lamps for 3 seconds and then switch them off

### How to use a procedure

Once you have built a procedure, you can call it into use whenever you like in the flowchart by using the Gosub command, as shown below.



The GOSUB command calls the procedure into use.



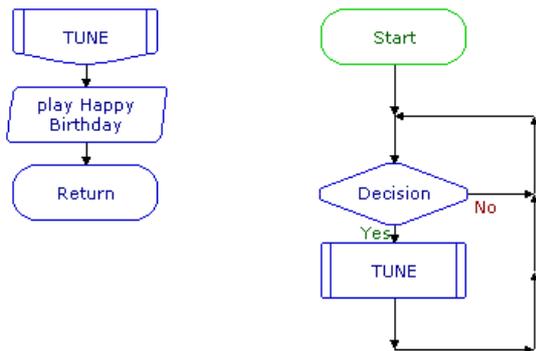
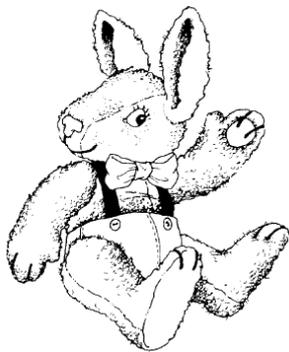
Drag a gosub command onto the flowchart. Place it at the point where you want the procedure to be called into use. Double click on the command to open its Cell Details box. Type in the name of the procedure or select it from the drop-down list. Click OK.

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

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

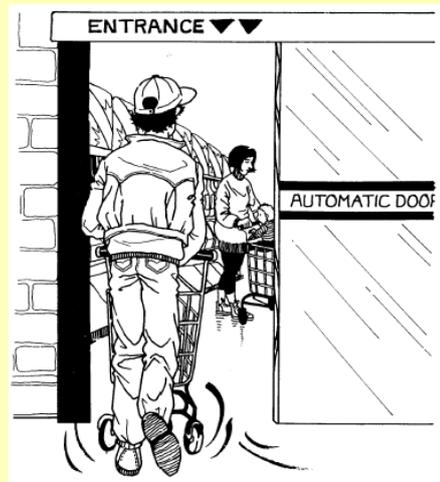
**Example**

A PICAXE 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 flowchart for the system is shown below. The tune is created as a procedure which can be tested and edited separately from the main routine.

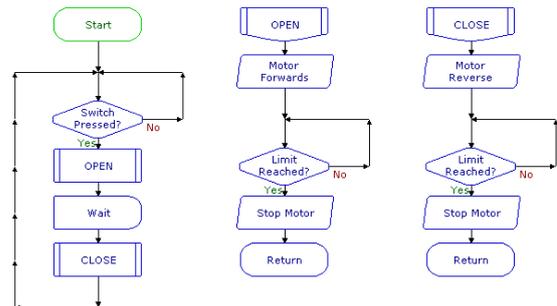


Using a Procedure to play a tune after an input condition is met

**Example**



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



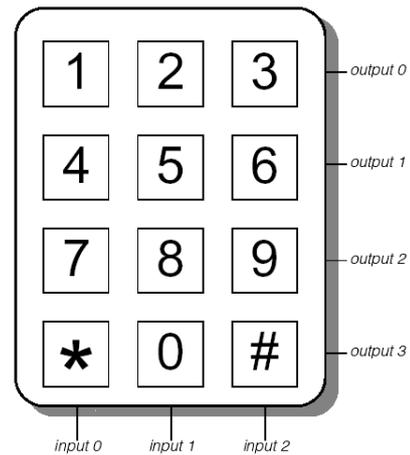
Sliding door control system using procedures

### Example

A keypad is a useful input device. This example shows how the PICAXE Editor 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 PICAXE microcontroller using inputs and outputs as shown right.

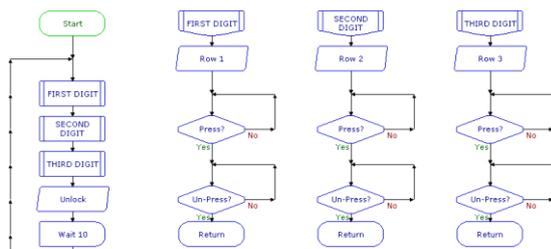
The flowchart below shows how the scanning is done.



Keypad connections

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.



A Flowchart to scan the keypad

When all three digits have been entered correctly, the solenoid is switched to unlock the door.

## Designing systems with procedures

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

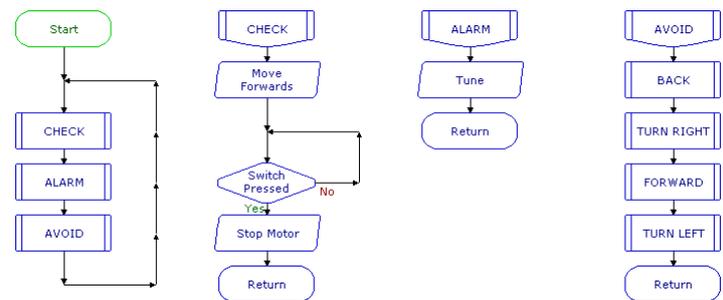
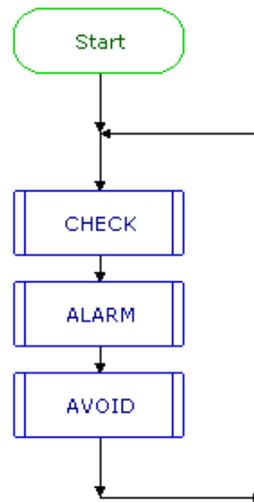
### Example

#### 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 Gosub commands as shown right.

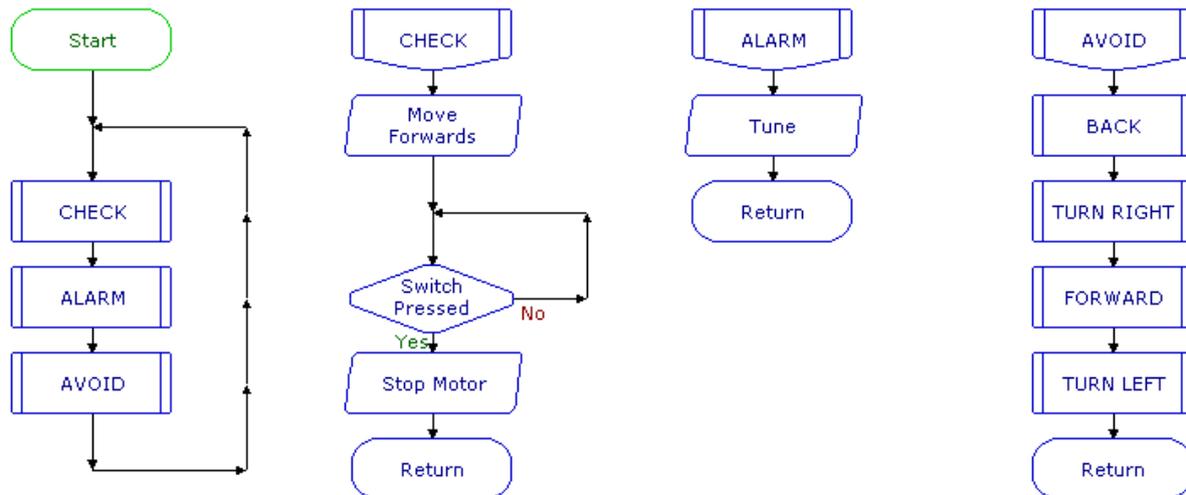
2. Then each part of the system is built as a separate procedure as shown below. Each procedure can be test run independently.



Main Routine

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

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

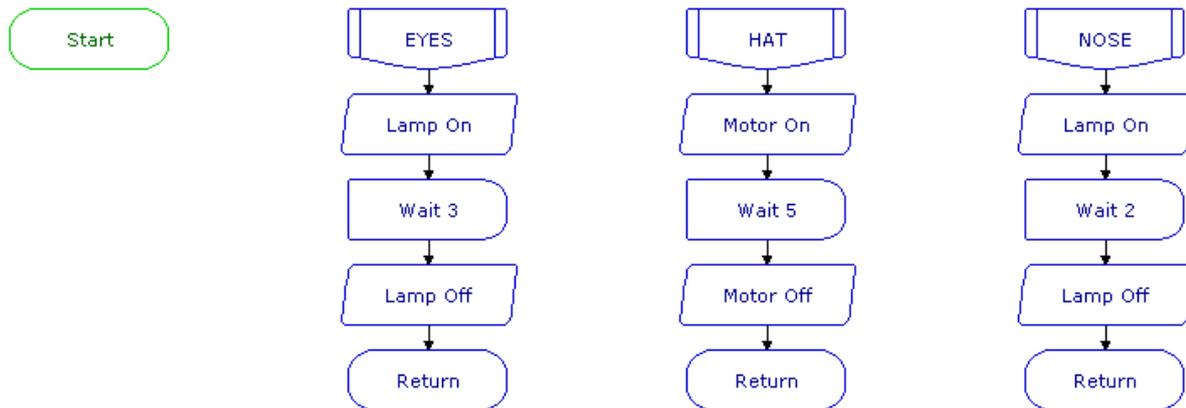


**Example**

**The 'bottom-up' approach**

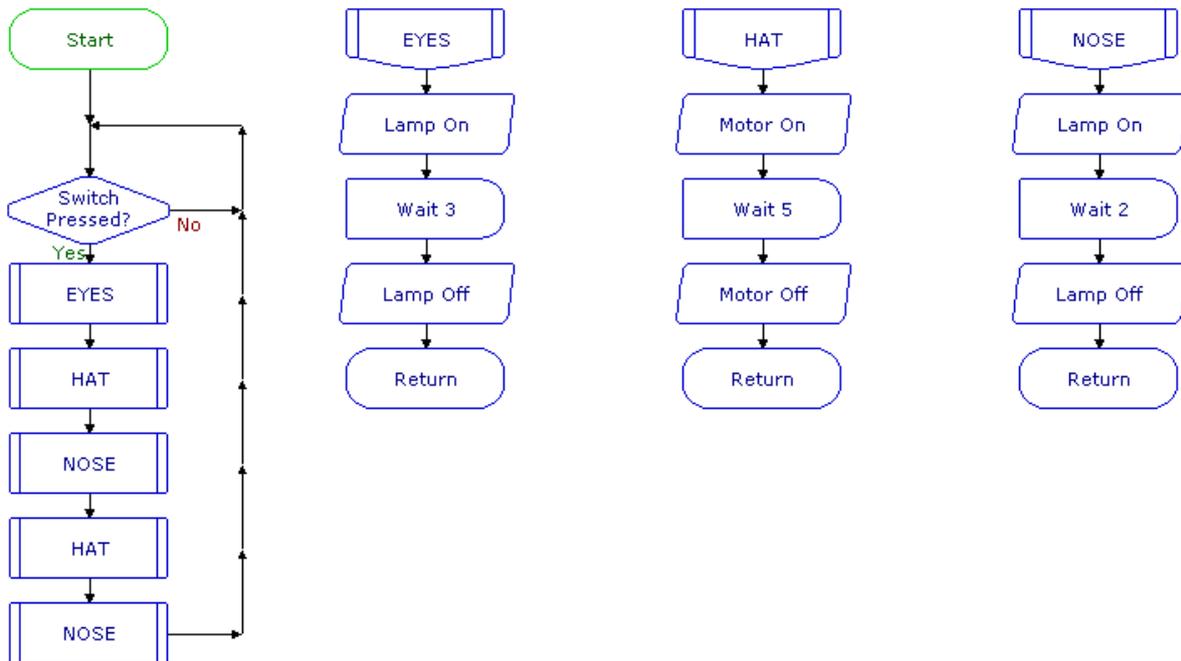
This approach develops each part of the system separately as a procedure, and then writes the main routine to link them. The following sequence shows how it can be used to develop a control system for an animated clown's head on which the eyes and nose light up and the hat rotates.

1. A separate procedure is built and tested for each one of the three elements, as shown below:



*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.



*The complete system*

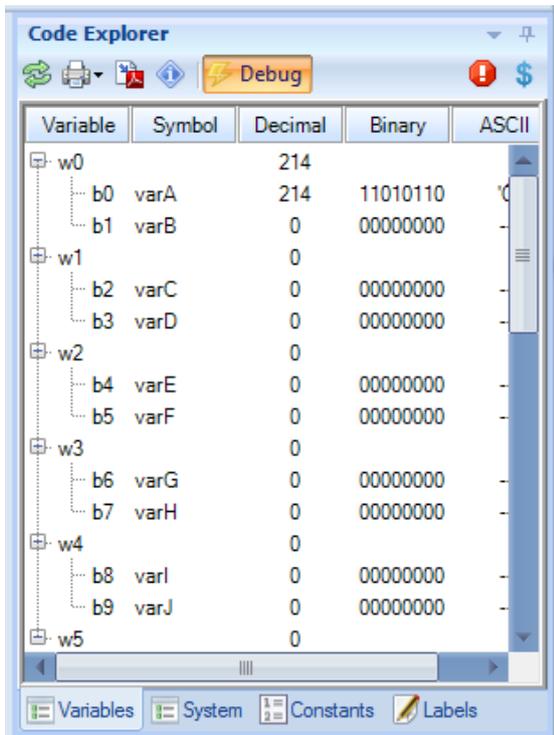
This flowchart shows some of the advantages of using this approach. Once a procedure has been created, it can be called into use as many times as you like within the flowchart. Editing the sequence is easy.

The gosub commands can be moved around, deleted or copied to change the sequence as required.

## Section 5. Variables

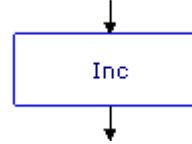
In PICAXE Editor a variable is a 'number container' that can hold a given value between 0-255. The variables are called varA to varZ. This section explains how they can be used for a variety of mainly counting and timing purposes.

The current value of a variable can be seen during a simulation in the Code Explorer panel.



## Counting

### The Inc command

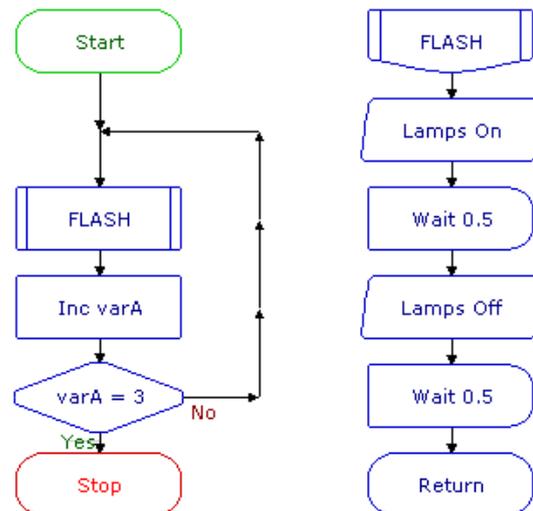


Each time flow passes through an Inc command, 1 is added to the value of the selected variable (Inc is short for increment). This is the same as using an Expression command to make  $\text{varA} = \text{varA} + 1$ .

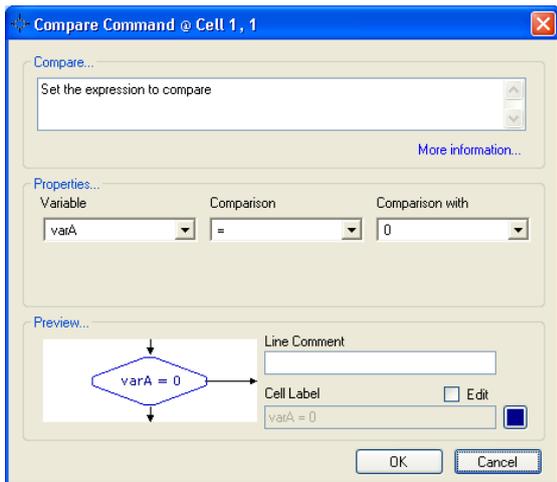
When you open the Cell Details box, simply select which variable you want to use, and click OK.

The flowchart shown below shows how it can be used to repeat a sequence three times. Each time that flow goes round the loop, the FLASH procedure is undertaken, and 1 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 flowchart.



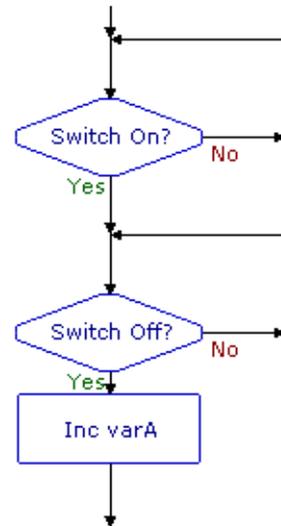
*Repeating a sequence three times*



Cell Details box of the Compare command

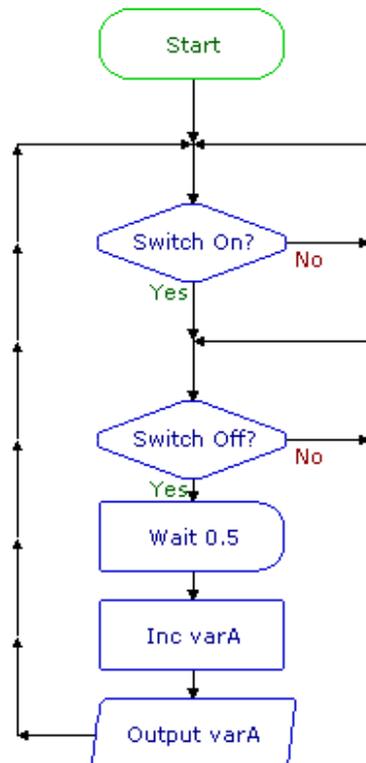
1. Use box one to select the variable that you want the command to check.
2. Use boxes two and three to complete the comparison. The drop-down list in box two contains a list of operators such as “greater than” (>), “less than” (<), and “equals” (=). Select the one that you require.
3. Use box three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



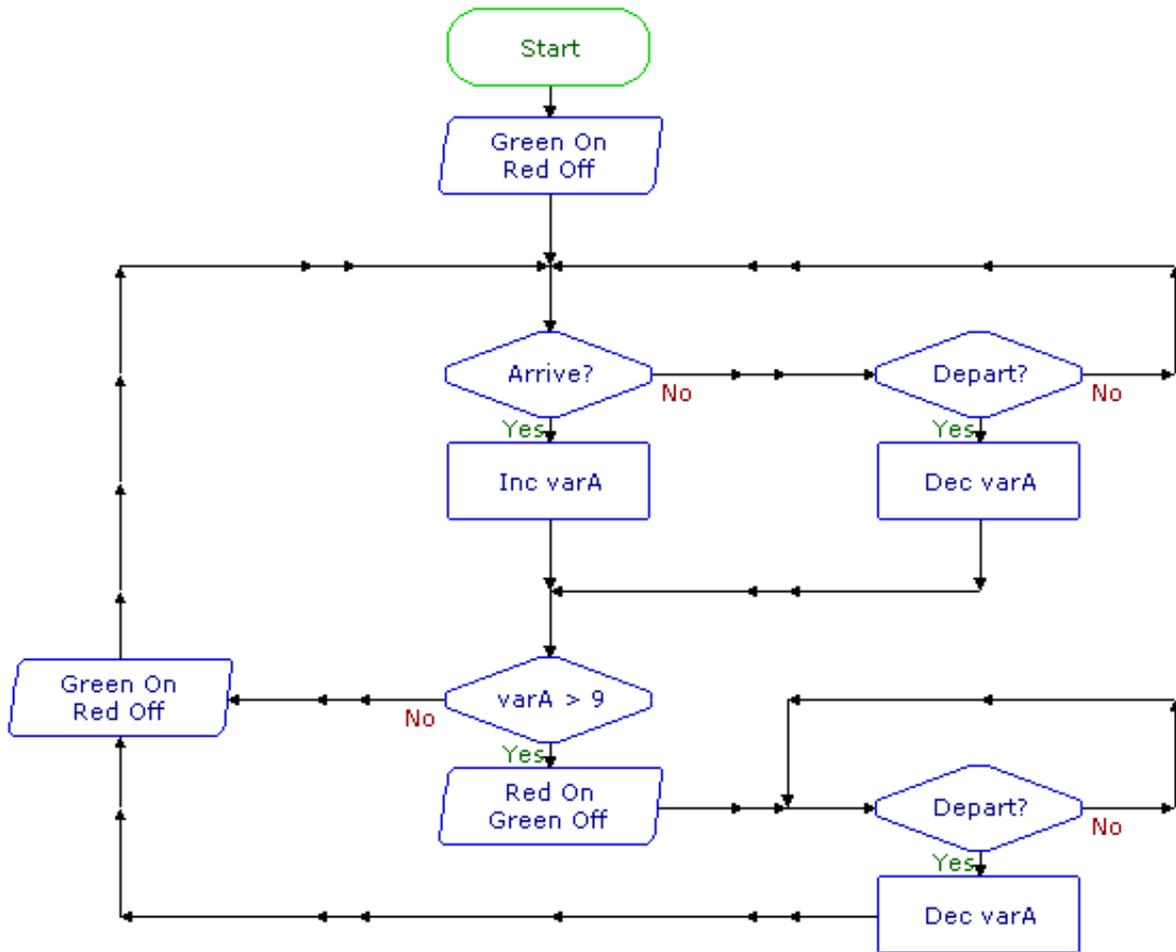
Ensuring a clean signal from a digital sensor

You may well find that once it is downloaded into the chip, the flowchart 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 the flowchart on the right. This flowchart 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 PICAXE microcontroller.



**Example**

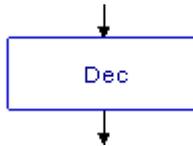
A PICAXE microcontroller is used to control a system for counting cars entering and leaving a car park using two digital sensors.



*Flowchart for making and displaying a count.*



## The DEC command



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.

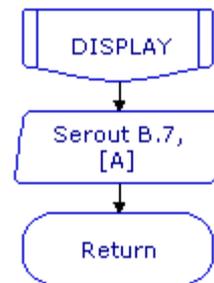
### **Example**

A seven-segment display is a useful output device for displaying counting and timing. The flowchart below 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.

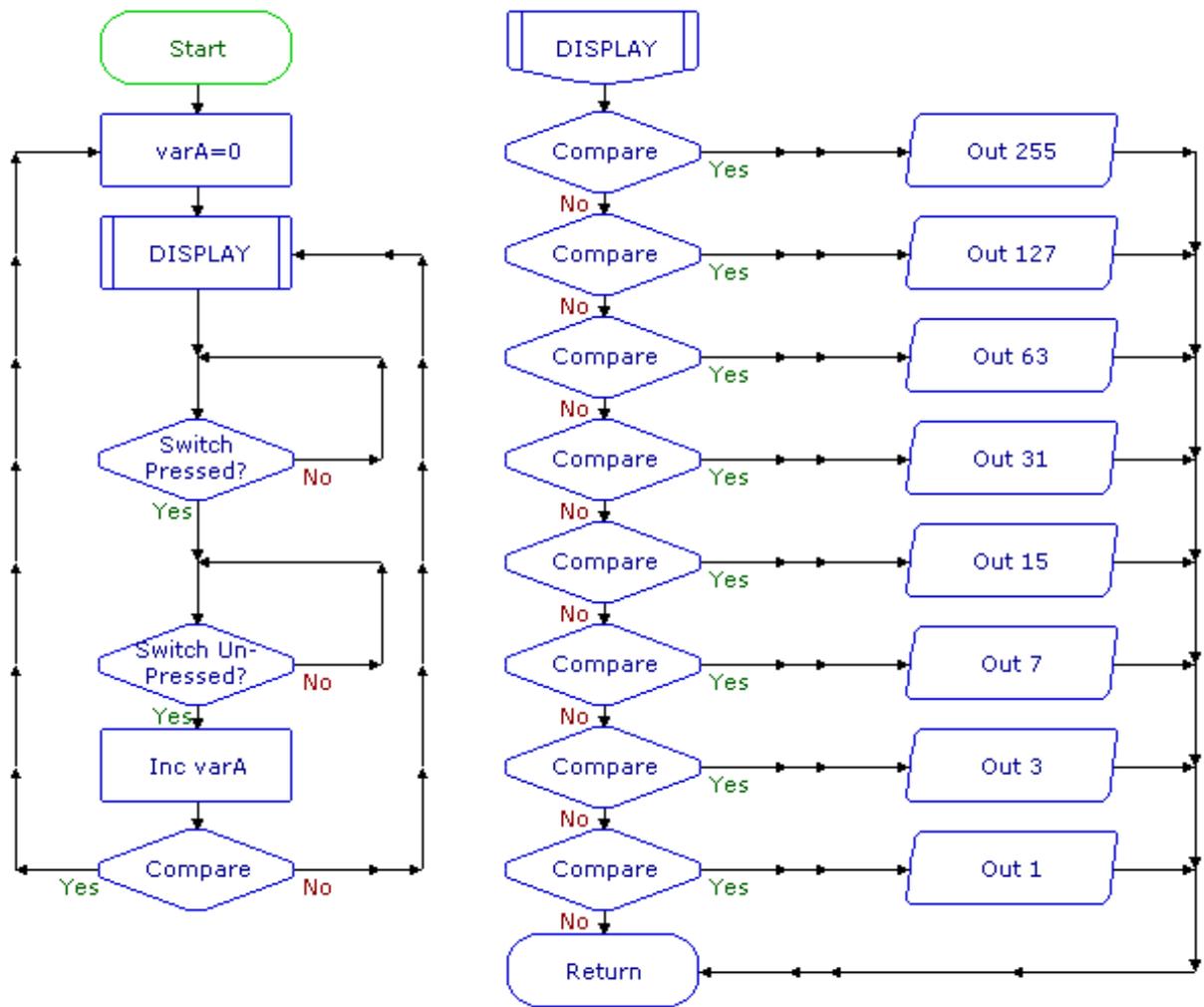
The main routine uses an Inc command to increment (add one to) the value of the variable A each time the assistant presses the switch. The DISPLAY procedure makes an efficient way of translating the current value of A into an Outputs command which is set to switch on the appropriate number of outputs

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 below:



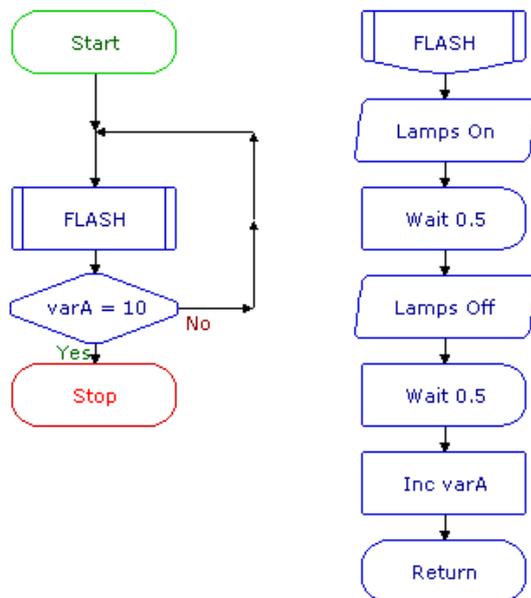
*Part of an equivalent system that uses an LCD screen to display numbers.*



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 flowchart shown below shows how it can be used to repeat a sequence for 10 seconds.

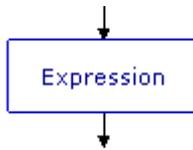


*Repeating a sequence for 10 seconds*

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 flowchart. 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

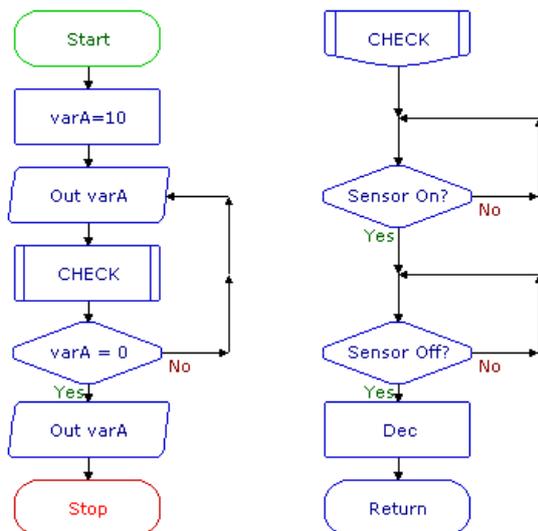


The Expression command is used to give a value to a variable as a flowchart runs. The variable is given its value as flow passes through the command. The following example shows how it can be used.

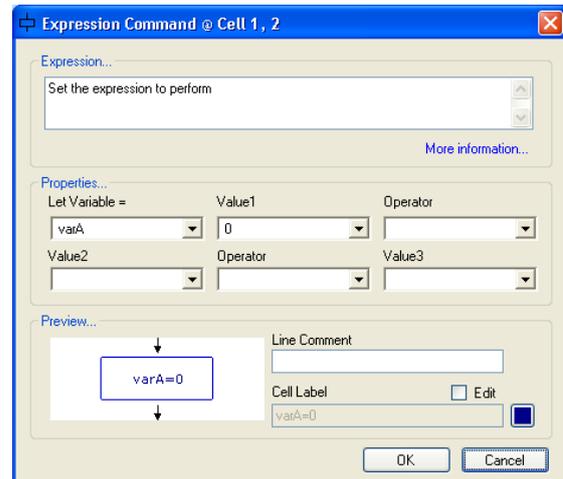
#### Example

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 next flowchart 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 PICAXE.



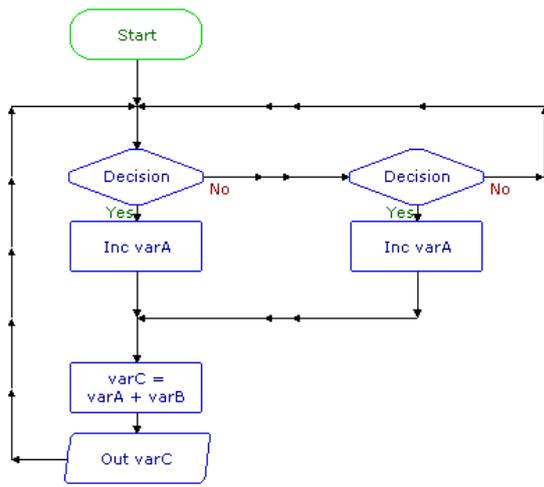
The Dec command counts down, so an Expression command is used to set the value of variable A to 10 at the start of the countdown when the container is full. The Expression command Cell Details box is shown below. Use the first two boxes to enter the expression `varA = 10`.



Expression command : Setting the value of a variable

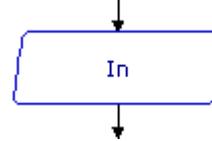
## Mathematical expressions

A value can also be given to a variable in the form of a mathematical expression as shown in the flowchart below. 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 39 expression  $C=A + B$ . NOTE: the third box in the Expression Cell Details box contains a range of mathematical operators.



Displaying a combined count

## The IN command

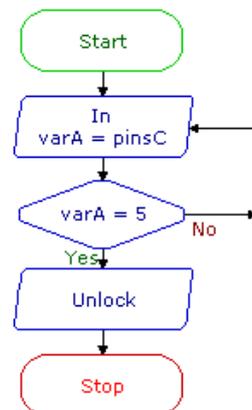


The IN command sets the value of a specified variable to the current binary value of the portC input port.

For example, if switches connected to inputs 0 and 1 are pressed, then the value of the variable will be 3. The next flowchart shows how this can be used to make a simple security system.

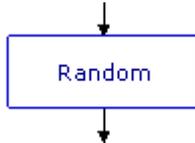


When switches connected to inputs 0 and 2 are pressed at the same time the binary value of the input port equals 5 (4+1), flow from the Decision command goes in the Yes direction and a solenoid-operated lock is opened. If any other combination of switches is pressed, flow goes in the No direction.

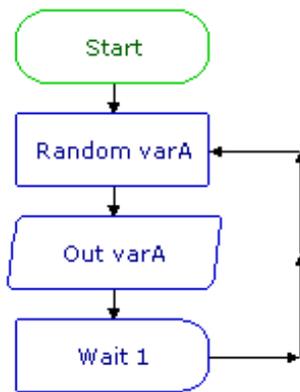


Security system that responds to pressing two switches

### The Random command



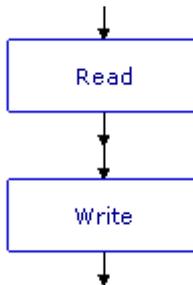
Using the random command a Variable can be given a random value between 0 and 255. In the example shown below, a set of display lights for a small Christmas tree are connected to 8 outputs of a PICAXE microcontroller. Every second the display will change at random.



*Using RND 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.

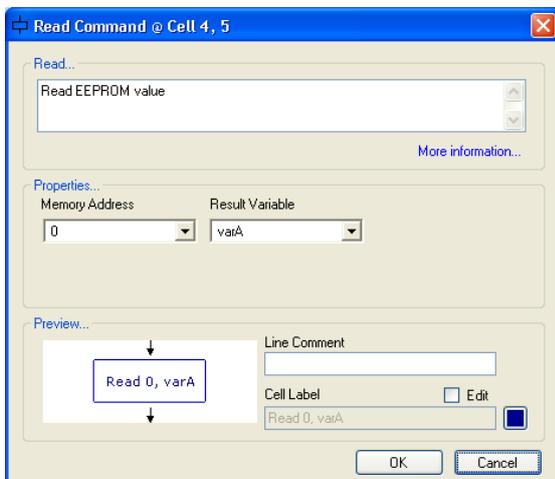
## Read and Write



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

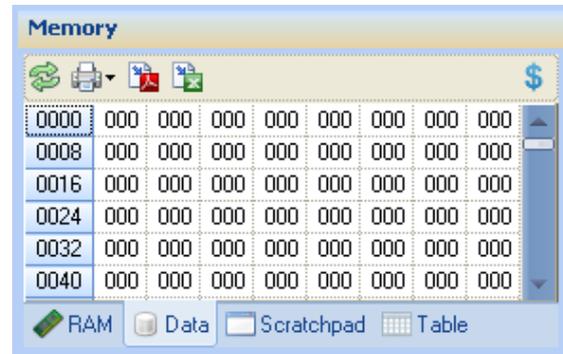
If you want to retain variable values when the PICAXE microcontroller is powered up or reset, you can use the WRITE command to store values in the chip's data EEPROM memory. The READ command is used to retrieve the values from the chip's memory. The flowchart below right shows an example of how the commands 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 (below) to enter the variable and the address from which the value is to be read.



READ command Cell Details box

The PICAXE microcontroller's data EEPROM memory has 255 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 flowchart.



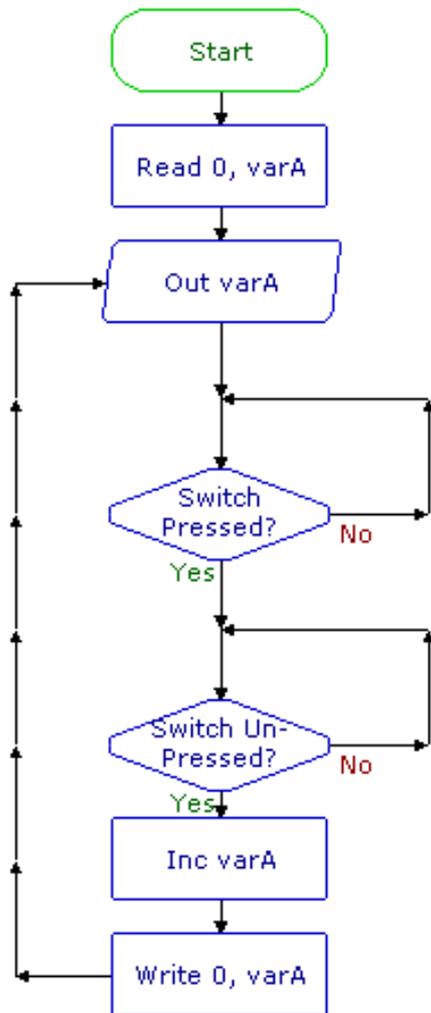
Data EEPROM window

The OUT A command in the flowchart displays the current value of A using 8 LEDs connected to outputs of the PICAXE 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 PICAXE microcontroller is powered down, the value of A is stored in the chip's memory.

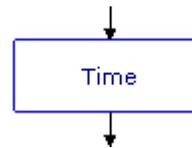
When the PICAXE microcontroller is powered up, the first thing that happens is that the READ 0,varA command retrieves the value of A which has been stored in address 0. The EEPROM window gives an accurate simulation of the way these commands work when the flowchart is downloaded.



Using READ and WRITE commands to store a count

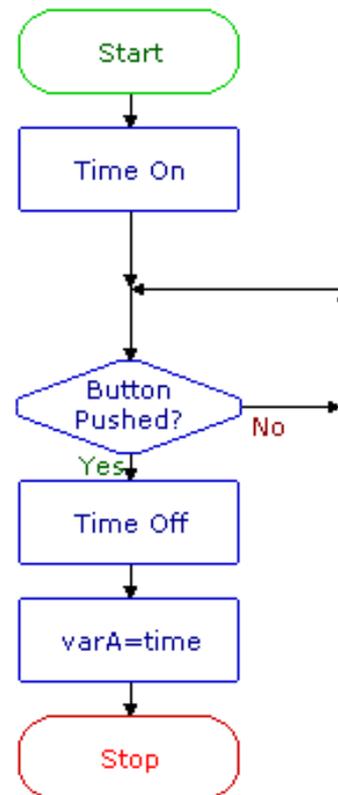
This flowchart shows how the Read and Write commands are used to store the number of times a switch is pressed.

## Time

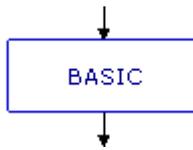


The PICAXE M2 chips have an internal clock module. The Time On command starts the clock cycle. The Time Off command will stop the clock. The elapsed time will be measured in seconds by the variable 'Time'.

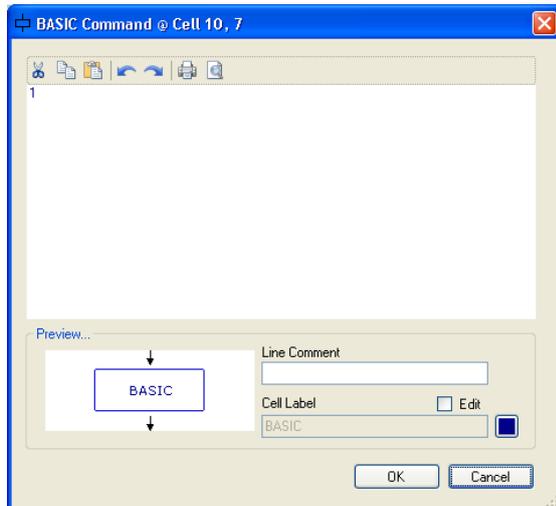
The flowchart below will measure the elapsed time until a button is pressed. The Time can be viewed in the Time panel and in this case its value is stored in the variable A.



## BASIC

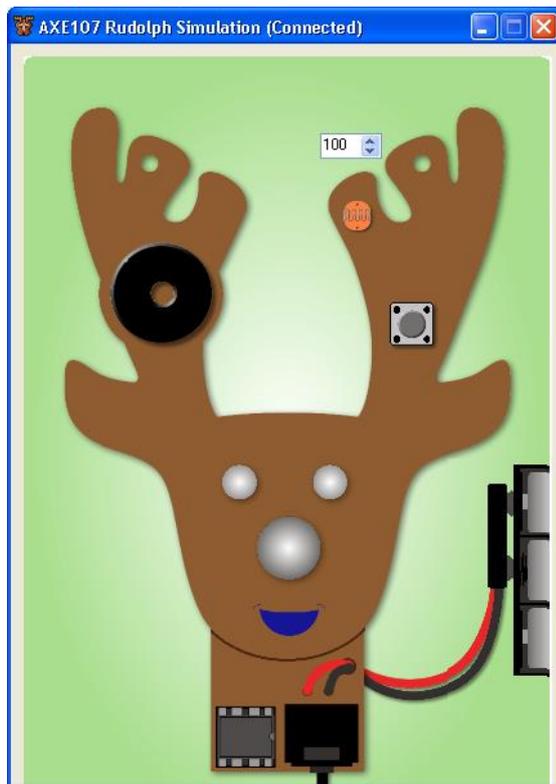


This command is used as an extension to a flowchart. Any valid PICAXE BASIC code can be typed into the command cell window. When program flow arrives at this command the BASIC code within the command will be processed as if it were a procedure.



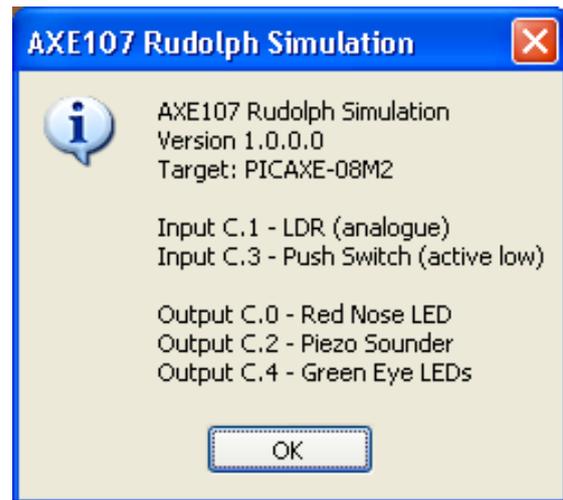
## Section 6. Simulations

PICAXE Editor also supports the on-screen simulation of PICAXE project kits. To start a Simulation select the Simulate>Connect to Software Simulation menu.



*The Rudolph Software Simulation*

A helper screen can be opened from the right click Help menu. This tells us which PICAXE chip the circuit is simulating and how the inputs and outputs are connected.



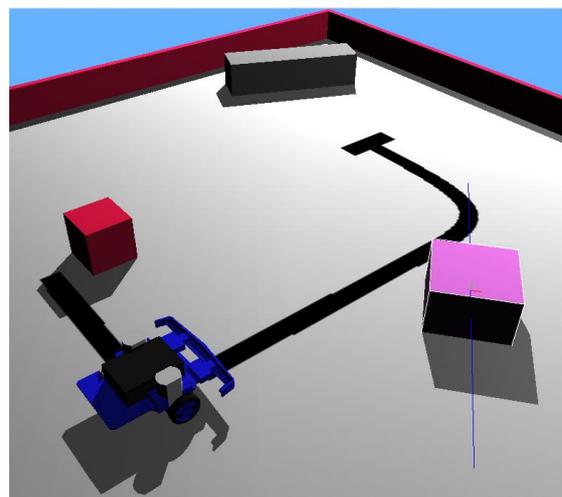
*The Rudolph helper window*

The simulation outputs will animate as on a real-life board and the inputs can be clicked to generate a simulated input.

With this simulation the student can program and test the PICAXE kit as though it was a real system.

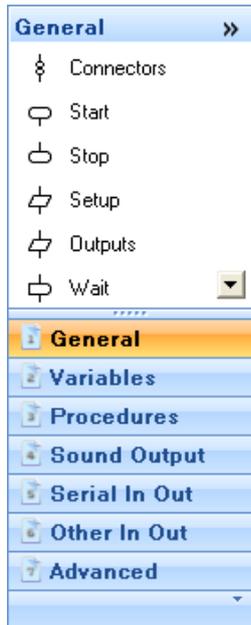
This simulation is also available as a real circuit board, part AXE107K..

PICAXE Editor also supports connections to third party circuit and robot simulator software such as 'Webots'. When the simulation is run within PICAXE Editor the simulation on the other product will be exactly synchronized.



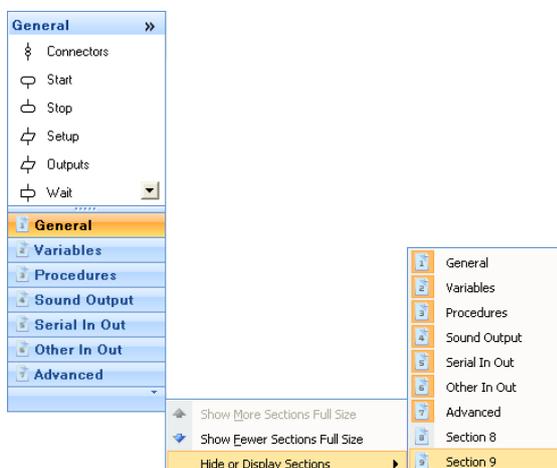
## Appendix A – Toolbox

The default flowchart toolbox contains different flowchart command items grouped into various sections. The default toolbox sections match the previous Logicator product sections. However they may be changed.

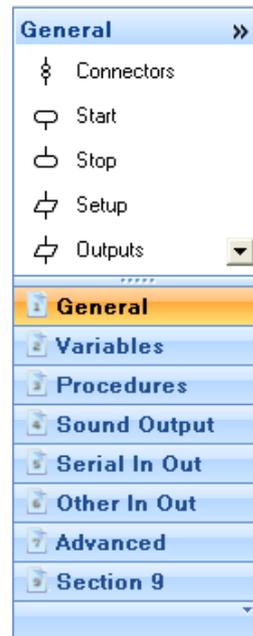


You can also build a completely new toolbox (to display more or less sections) or add or remove sections to an existing toolbox as desired.

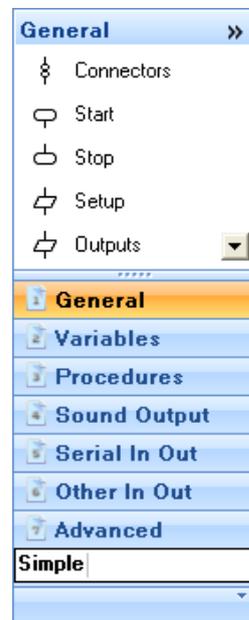
To add an extra section to the toolbox; click on the down arrow at the right of the bottom of the toolbox. Move the mouse over Hide or Display Sections and a list of the 9 available sections will be shown.



Using the mouse select an unused section such as "Section 8" or "Section 9" and it will be added at the bottom of the toolbox list.

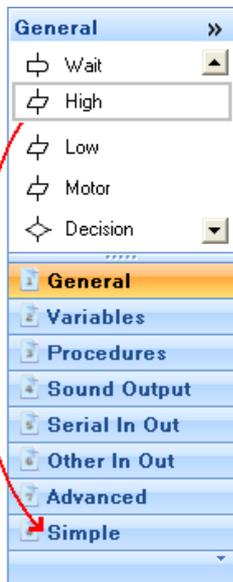


Right-click over the toolbox section added and you can then edit that section title.

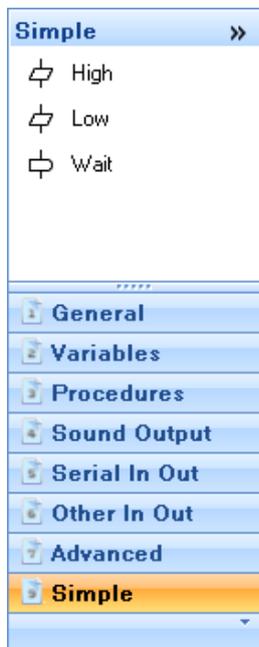


To add commands to this section right click within the main section of the toolbox and then select the 'Add New Item' menu.

To rearrange the order of the commands simply click and drag them. You can also drag commands from one section to another.



Once complete you will have a toolbox section with the desired commands within it.



Finally right click over the toolbox and select 'Save Toolbox As' so that it may be used again in the future.

PICAXE Editor will automatically remember which toolbox was used last and use that again next time a new flowchart is created.

## Appendix B – Commands

It is also possible to create completely new specialised flowchart commands (e.g. for a particular sensor or component) and add them to the toolbox. This process requires a knowledge and understanding of PICAXE BASIC (in order to create the BASIC for the flowchart conversion process).

All command items are saved as .xml text files, so the easiest way to create a new command is generally to find an existing similar command and then duplicate and edit it's parameters as required.

For further detail on this process please contact PICAXE technical support with your requirements.

## Appendix C – Upgrading

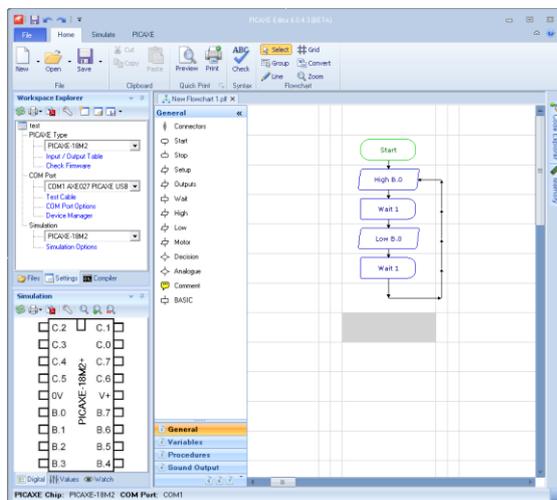
### What are the main differences between PE6 and 'Locator for PICAXE v3' (LFP)?

#### Free Software

PE6 is free. There is no charge for use – free for schools/colleges, free for commercial companies and free for individuals at home. PE6 supports both BASIC code and flowcharts. Naturally all LFP flowchart files open in PE6.

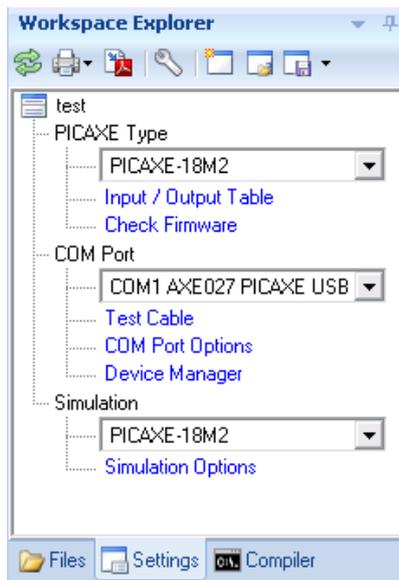
#### Modern Ribbon Interface

PE6 can use the new modern ribbon interface or the traditional toolbar interface as desired.



#### PICAXE Selection

It is now easier to select PICAXE type, COM port etc. via the Workspace Explorer panel.



#### Floating/Docking Panels

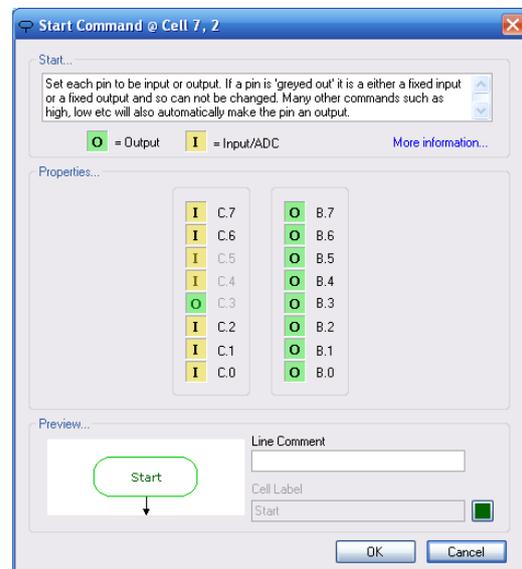
All panels (e.g. Simulation Panel) can be setup as docking or floating as required. They no longer overlay other applications.

#### Easier Line Drawing

Lines are now drawn by click and drag, which students find a more intuitive way of drawing. Blocks of commands can now also be selected via the new group select tool.

#### Pin Configuration

All current PICAXE chips (all sizes) can now be configured – so you can choose which pins are outputs and which pins are inputs. This is achieved via double clicking on the first Start command in the flowchart.



So you are no longer limited to a maximum of 8 inputs or 8 outputs, although you can naturally choose the same configuration as PFL for existing project boards.

#### Up to 8 Parallel Tasks

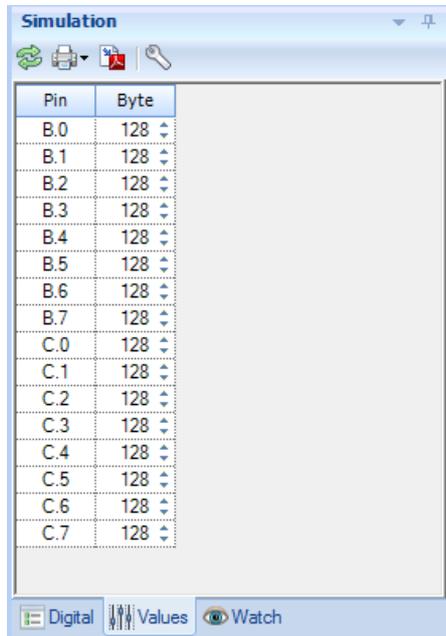
You can use up to 8 parallel tasks (starts) on chips that support them.

#### More Variables (RAM) and EEPROM

PE6 supports more variables and they are now labelled varA, varB etc. for clarity. Advanced users may also use word variables if they choose.

### More Analogue inputs

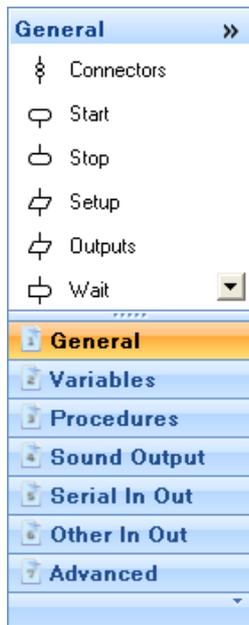
LfP was limited to 4 analogue inputs. PE6 supports all ADC that are available in the chip.



Pin	Byte
B.0	128
B.1	128
B.2	128
B.3	128
B.4	128
B.5	128
B.6	128
B.7	128
C.0	128
C.1	128
C.2	128
C.3	128
C.4	128
C.5	128
C.6	128
C.7	128

### Toolbox

The toolbox has been improved to allow end user configuration (e.g. to hide/show commands and sections as desired). The icons are also now smaller to enable more commands to be displayed at the same time.



### Connectors

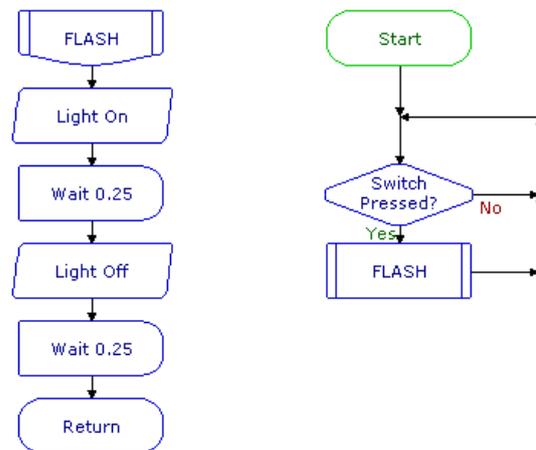
New flowchart connectors allow lines to be joint at different locations in the flowchart.

### Multiple Flowchart Support

Multiple flowcharts can now be opened at the same time. This makes it much easier to cut and paste sections of commands between flowcharts.

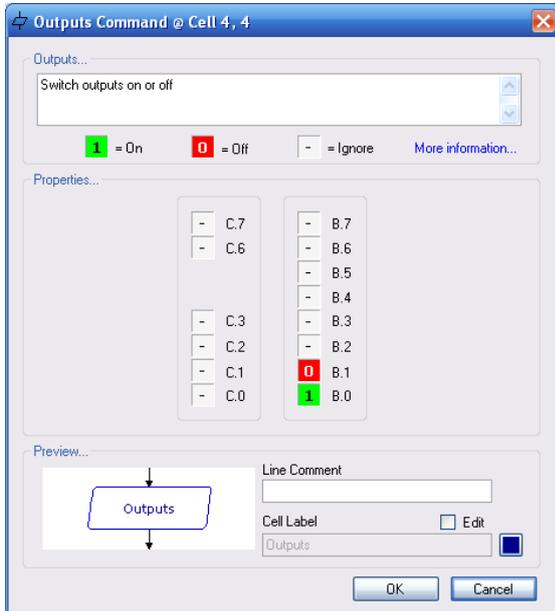
### Modern Colour Scheme

The software now defaults to a modern colour scheme, however the old Logicator colour scheme is still available if desired (right click over the flowchart and select Colour Scheme).



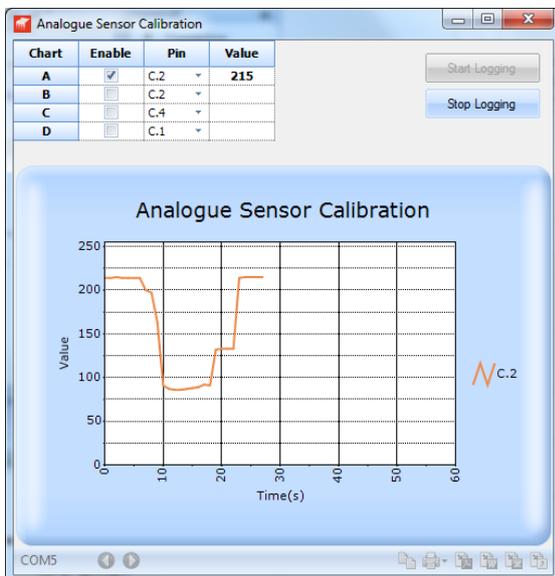
## Command Edit Dialog

The command edit dialogs have all been modernised and now include a preview of the command.



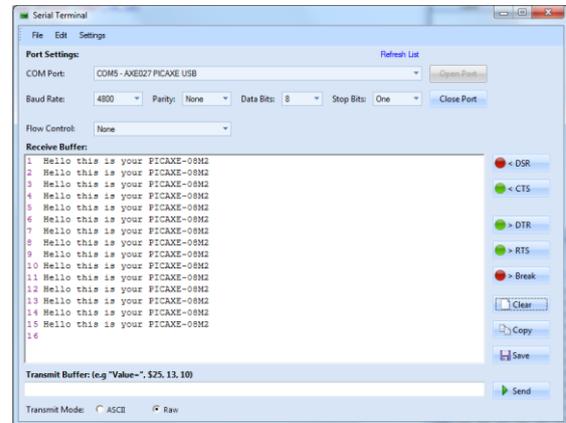
## Analogue Sensor Calibration Wizard

Calibration of analogue sensors is now much simpler, with a detailed graph of the values in the real time experiment.



## Serial Terminal

An advanced Serial Terminal, for use with the serin/serout commands, is now available.



## Trouble Shooting Wizards

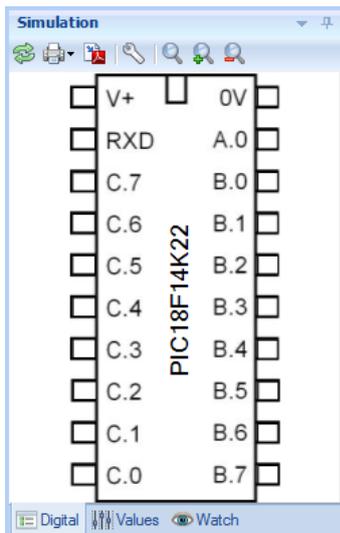
Many additional wizards, e.g. testing an AXE027 download cable, are now included.



### Improved Simulation

All commands, including generic BASIC cells, are now fully simulated. New custom commands can also be added by the end user.

Many PICAXE project kit simulations can now also be used within the simulation panel (as well as the default chip shape).

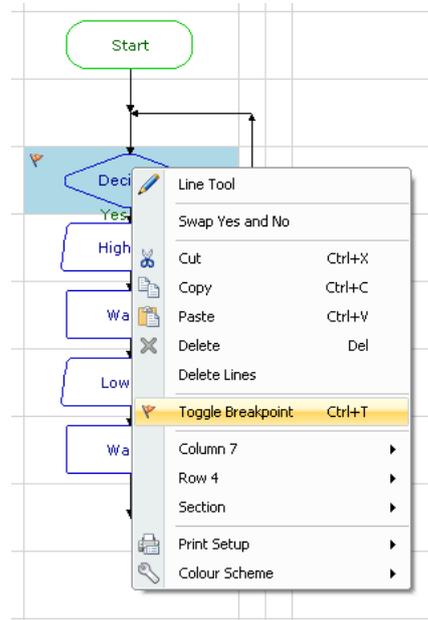


Many of the internal panels e.g. LCD are now also much more realistic.



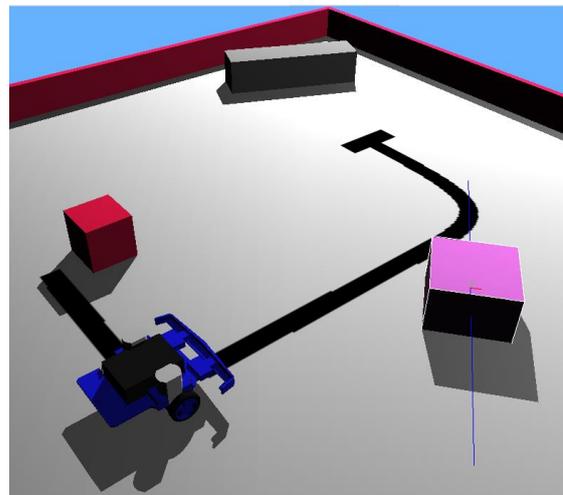
### Breakpoints

Breakpoints can now be added to any cell, so that the flowchart simulation can be automatically paused at that point.

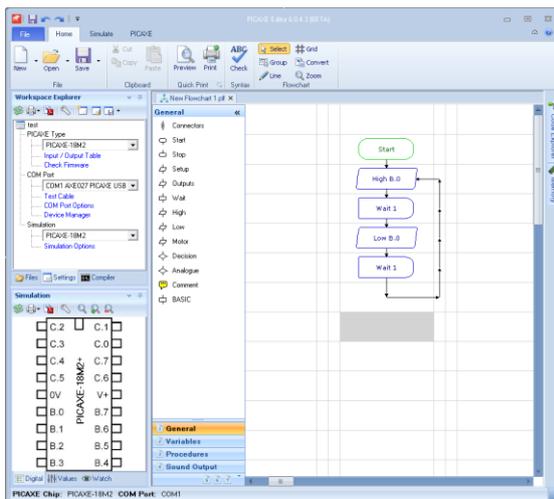


### 3rd Party Simulators

Simulation links to other 3rd party circuit and 3D model simulators is now also supported e.g. this is a 3D simulation of the BOT120 PICAXE-20X2 microbot in 'Webots'. The robot moves as the flowchart simulated and the two software applications are fully synchronised.



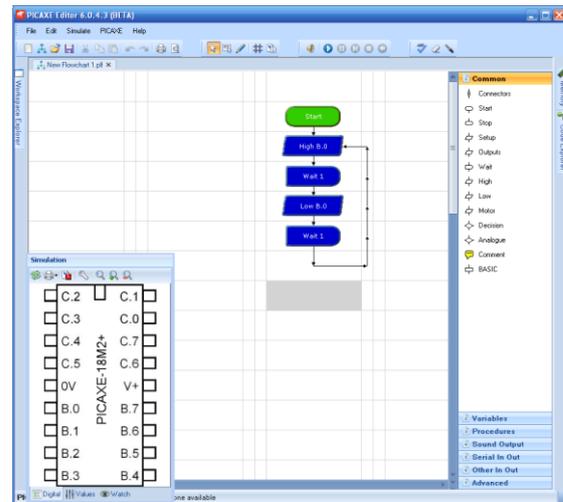
## How do I make PE6 look like Logicator?



We expect most people to enjoy using PE6 with the default settings as above (e.g. with the new ribbon interface and modern colour scheme). However if you wish to make PE6 look more like LfP:

- 1) Select 'Legacy Toolbar' mode in File>Options
- 2) Hide the Workspace Explorer Panel and drag out the Simulation Panel so that it is now floating (it can also be resized as you choose).
- 3) Right click over the flowchart and select the Logicator colour scheme
- 4) Right click over the toolbox and toggle the toolbox docking side.

## PE6 (modified layout) view:



## Logicator for PICAXE v3 view:

