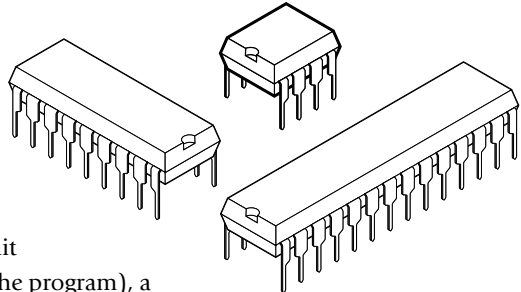


ELECTRONIC DICE

What is a microcontroller?

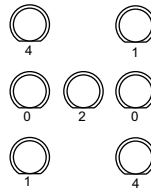
A microcontroller is often described as a 'computer-on-a-chip'. It can be used as an 'electronic brain' to control a product, toy or machine.



The microcontroller is an integrated circuit ("chip") that contains memory (to store the program), a processor (to process and carry out the program) and input/output pins (to connect switches, sensors and output devices like motors).

Microcontrollers are purchased 'blank' and then programmed with a specific control program. This program is written on a computer and then 'downloaded' into the microcontroller chip. Once programmed the microcontroller is built into a product to make the product more intelligent and easier to use.

Example use of a microcontroller.



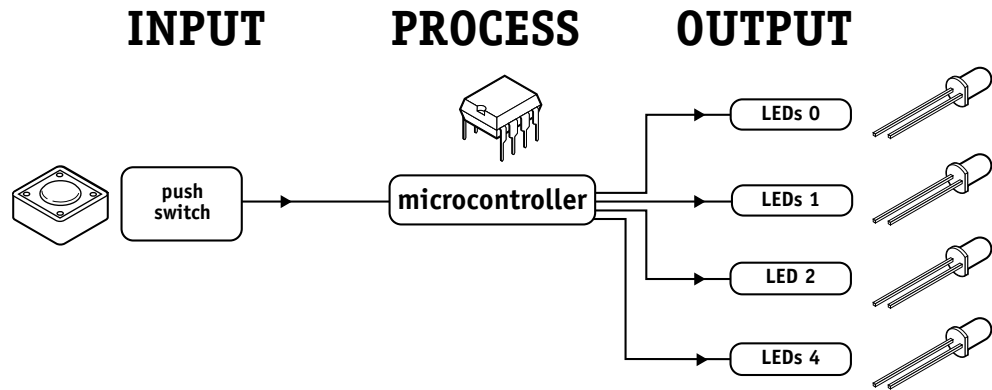
The picture above shows an electronic dice that can be made to generate a random number. It can be used like a normal dice in games.

The dice works by switching Light Emitting Diodes (LEDs) on and off in the pattern of dots found on a traditional dice. The microcontroller is the 'brain' of the dice. Microcontrollers are powerful electronic components that have a memory and can be programmed to switch things on and off in any sequence. The microcontroller in the dice can switch the LEDs on and off to show numbers between one and six.

As the 'diagonal' pairs on LEDs in the dice always light at the same time, they can be switched on and off by the same microcontroller output. Therefore 4 outputs (3 pairs and the central 'dot') are required.

BLOCK DIAGRAMS

The electronic system for an electronic dice can be drawn as a 'block diagram'.



The push switch is an electronic device that can detect movement and is known as an 'input'. The microcontroller then 'decides' how to behave and may then switch the output LEDs on in different patterns.

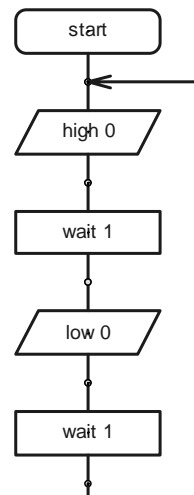
WHAT IS THE PICAXE SYSTEM?

The microcontrollers used in devices such as electronic games can be difficult to program, as they generally use a complicated programming language called 'assembler code', which can be quite difficult to learn.

The PICAXE system makes the microcontrollers much easier to program. The control sequence can be drawn (and simulated) on the computer as a flowchart, or written in a simpler programming language called BASIC. This makes it much easier to use the microcontroller as the complicated 'assembler code' does not need to be learnt.

A sample BASIC program and flowchart are shown here. In this case both programs do the same thing - flash a light (connected to output 0) on and off every second.

```
main:
    high 0
    wait 1
    low 0
    wait 1
    goto main
```



BUILDING YOUR OWN ELECTRONIC DICE

Design Brief

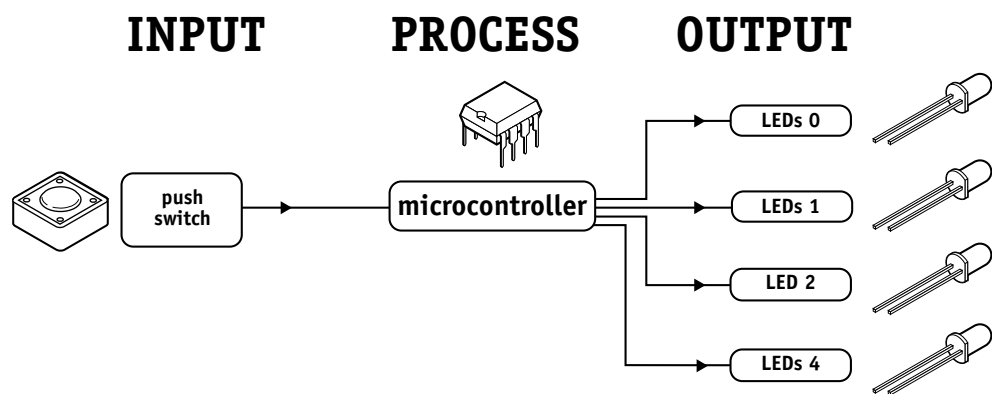
Design and make an electronic dice. The dice must show the numbers 1-6 randomly by switching LEDs on and off.

Design Specification Points

- 1) The design will use a PICAXE-08 microcontroller as its controller.
- 2) The design will include LEDs in a dice pattern.

Block Diagram

The block diagram for your safety light may look like this:



Personalising Your Electronic Dice

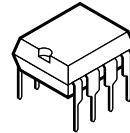
There are several ways to personalise your dice.
Here are some things to think about:

- 1) Are you going to build your dice on the circuit board or mount it in a case? If you use a case you will need to attach the LEDs and switch using wires.
- 2) What colour and size of LEDs are you going to use? The most common LEDs are red, but many other sizes and colours are available (e.g. blue).
- 3) How will you activate the dice? Normally a push switch is used, but you could use many other types of sensor, for example, you could have a light sensor (LDR) that can detect changes in light level when you put your hand over it.

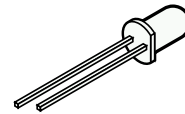
ELECTRONIC COMPONENTS

The main electronic components you may need for your electronic dice are shown here. The next few pages describe each of these components in more detail, and also provide some programming ideas that may be useful when you are later programming your dice light patterns.

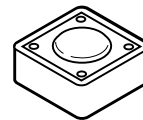
PICAXE-08 microcontroller



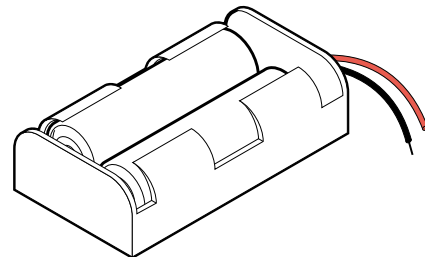
light emitting diode (LED)



push switch

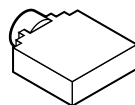


3 x AA battery box



and you will also need

picaxe download socket



resistors

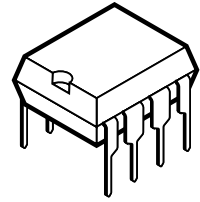


SECTION 2 - ELECTRONIC COMPONENTS

MICROCONTROLLERS

What is a microcontroller?

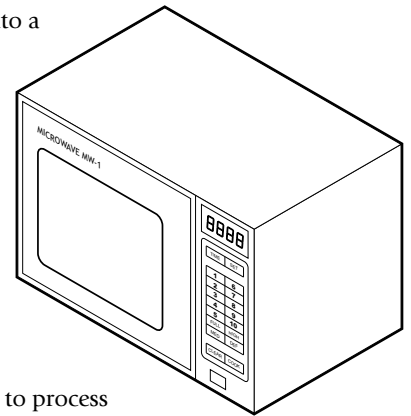
A microcontroller is often described as a 'computer-on-a-chip'. It is an integrated circuit that contains memory, processing units, and input/output circuitry in a single unit.



Microcontrollers are purchased 'blank' and then programmed with a specific control program. Once programmed the microcontroller is built into a product to make the product more intelligent and easier to use.

Where are microcontrollers used?

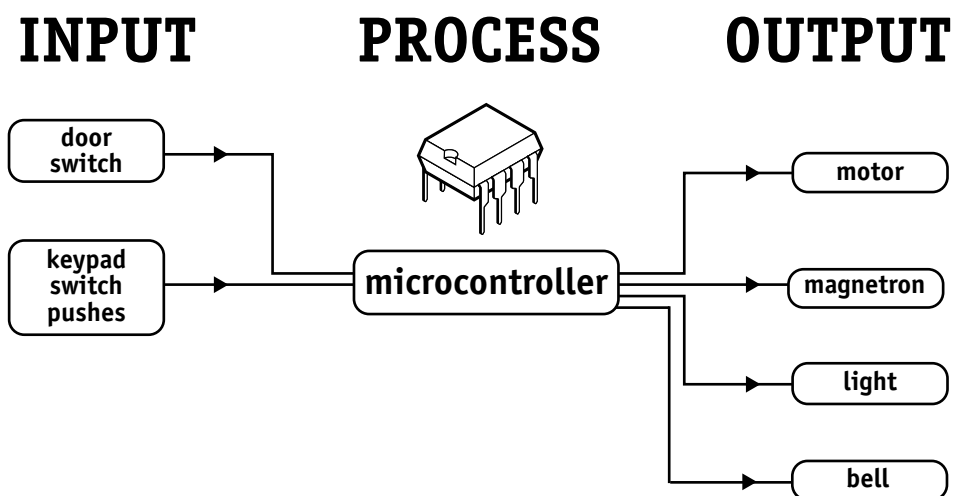
Applications that use microcontrollers include household appliances, alarm systems, medical equipment, vehicle subsystems, and electronic instrumentation. Some modern cars contain over thirty microcontrollers - used in a range of subsystems from engine management to remote locking!



As an example, a microwave oven may use a single microcontroller to process information from the keypad, display user information on the seven segment display, and control the output devices (turntable motor, light, bell and magnetron).

How are microcontrollers used?

Microcontrollers are used as the 'brain' in electronic circuits. These electronic circuits are often drawn visually as a 'block diagram'. For instance a simplified block diagram for the microwave above could be drawn like this:



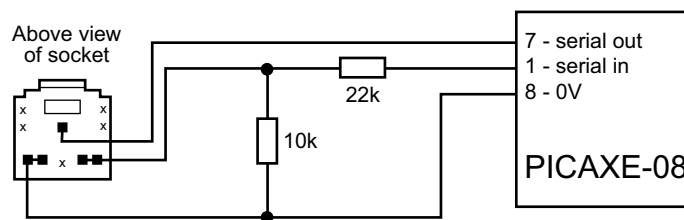
The program for the microcontroller is developed (and tested) on the computer and then downloaded into the microcontroller. Once the program is in the microcontroller it starts to 'run' and carries out the instructions.

How are programs written?

Programs are drawn as flowcharts or typed as 'BASIC' listings. This is explained in the programming section (section 3) later in this booklet.

How is the program transferred to the microcontroller?

The PICAXE-08 microcontroller is programmed by connecting a cable from the serial port at the back of the computer to a socket on the printed circuit board (PCB) beside the microcontroller. This socket (which looks like a headphone socket as found on a portable CD player) connects to two legs of the microcontroller and to 0V from the battery. This allows the computer and the microcontroller to 'talk' to allow a new program to be downloaded into the microcontroller's memory.



The socket and interfacing circuit is included on every PCB designed to be used with the PICAXE-08 microcontroller. This enables the PICAXE microcontroller to be re-programmed without removing the chip from the PCB - simply connect the cable whenever you want to download a new program!

The circuit diagrams of PICAXE circuits often do not include the components above to make it easier to understand the input/output connections. However the two resistors and the socket are always built onto every PICAXE project board!

Output 0

With the PICAXE-08 system leg 7 has two functions - when a program is being run the leg is known as output 0 and can control outputs like LEDs and motors.

When a program is being downloaded the same leg acts as the 'serial out' pin, 'talking' to the computer. Therefore if you also have an output such as an LED connected to the leg, you will find that the LED will flicker on and off as the program download takes place.

Note:

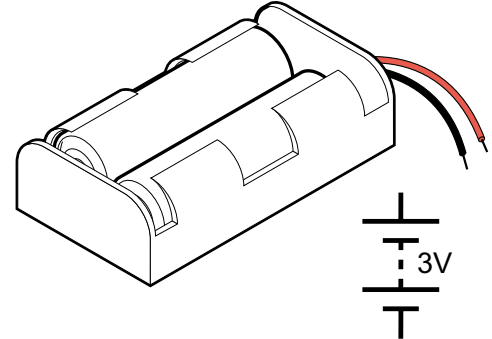
Most modern computers have two serial ports, normally labelled COM1 and COM2. The Programming Editor software used to create the programs must be configured for the correct serial port - select **View>Options>Serial Port** to select the correct serial port for your machine.

If you are using a new laptop computer it may only have the newer 'USB' type connector. In this case you must buy a USB to serial adapter to use the PICAXE system. These are available from most high street computer stores or online from www.tech-supplies.co.uk (part USB010).

BATTERIES

What is a battery?

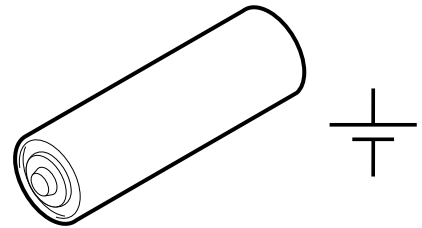
A battery is a self-contained source of electronic energy. It is a portable power supply. Batteries contain chemicals that store energy. When connected into a circuit this chemical energy is converted to electrical energy that can then power the circuit.



Which battery size should I use?

Batteries come in all sorts of types and sizes. Most battery packs are made up of a number of 'cells', and each cell provides about 1.5V. Therefore 4 cells will generate a 6V battery and 3 cells a 4.5V battery.

As a general rule, the larger the battery the longer it will last (as it contains more chemicals and so will be able to convert more energy). A higher voltage battery does not last longer than a lower voltage battery. Therefore a 6V battery pack made up of 4 AA cells will last much longer than a 9V PP3 battery, as it contains a larger total amount of chemical energy as it is physically larger. Therefore items that require more power to work (e.g. a CD walkman which contains a motor and laser to read the CD's) will always use AA cells rather than PP3 batteries.



Microcontrollers generally require 3 to 6V to work, and so it is better to use a battery pack made up of two, three or four AAA or AA size cells. Never use a 9V PP3 battery as the 9V supply will damage the microcontroller.

Which battery type should I use?

Different batteries are made of different chemicals. Zinc-carbon batteries are the cheapest, and are quite suitable for many microcontroller circuits. Alkaline batteries are more expensive, but will last much longer when driving devices like motors that require larger currents. Lithium batteries are much more expensive but have a long life, and so are commonly used in computer circuits to provide a clock backup.

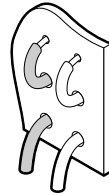
Rechargeable batteries can be recharged when they 'run-down'. They are generally made up of nickel and cadmium (Ni-cad) or nickel metal hydroxide (NiMH) chemicals.

Safety!

Never 'short circuit' any battery. Alkaline and rechargeable batteries can provide a very large current, and can get so hot that they will actually melt the battery box if you short circuit them! Always make sure you connect the battery around the correct way (red positive (V+) and black negative (0V or ground)). The microcontroller chip will get hot and be damaged if the battery is connected the wrong way around.

Using battery snaps.

Battery packs are often connected to electronic printed circuit boards by battery snaps or wires. Always ensure you get the red and black wires the correct way around. It is also useful to thread the wires through holes on the board before soldering it in place - this provides a much stronger joint that is less likely to snap off.



Never accidentally connect a 9V PP3 battery to the battery snap - this will damage the microcontroller, which only works between 3 and 6V.

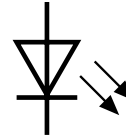
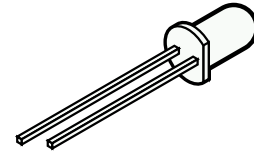
Soldering to battery boxes.

Some small battery boxes require wires to be soldered to metal contacts on the battery box. In this case you must be very careful not to overheat the metal contact. If the contacts get very hot they will melt the plastic and fall off. A good way of stopping this happening is to ask a friend to hold the metal contact with a pair of small pliers. The pliers will act as a 'heat-sink' and help stop the plastic melting.

LIGHT EMITTING DIODE (LED)

What is an LED?

A Light Emitting Diode (LED) is an electronic component that gives out light when current passes through it. An LED is a special type of diode. A diode is a component that only allows current to flow in one direction. Therefore when using a diode, it must always be connected the correct way around.



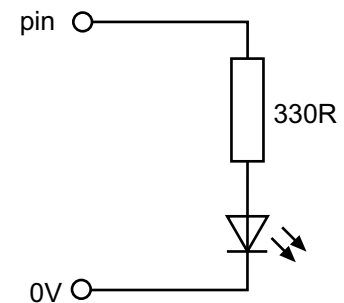
The positive (anode) leg of an LED is longer than the negative (cathode) leg (shown by the bar on the symbol). The negative leg also has a flat edge on the plastic casing of the LED.

What are LEDs used for?

LEDs are mainly used as indicator lights. Red and green LEDs are commonly used on electronic appliances like televisions to show if they are switched on or in 'standby' mode. LEDs are available in many different colours, including red, yellow, green and blue. Special 'ultrabright' LEDs are used in safety warning devices such as the 'flashing lights' used on bicycles. Infra-red LEDs produce infra-red light that cannot be seen by the human eye but can be used in devices such as video remote-controls.

Using LEDs.

LEDs only require a small amount of current to work, which makes them much more efficient than bulbs (this means, for instance, that if powered by batteries the LEDs will light for a much longer time than a bulb would). If too much current is passed through an LED it will be damaged, and so LEDs are normally used together with a 'series' resistor that protects the LED from too much current.



The value of the resistor required depends on the battery voltage used. For a 4.5V battery pack a 330R resistor can be used, and for a 3V battery pack a 120R resistor is appropriate.

Connecting the LED to a microcontroller.

Because the LED only requires a small amount of current to operate, it can be directly connected between the microcontroller output pin and 0V (with the series protection resistor). Two LEDs can be driven from the same pin if you use a resistor for each LED.

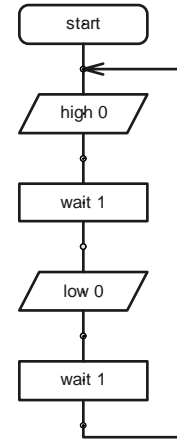
Testing the LED connection.

After connecting the LED it can be tested by a simple program like this:

```
main:
    high 0
    wait 1
    low 0
    wait 1
    goto main
```

This program would switch the LED (connected to output pin 0) on and off every second. If the LED does not work check:

- 1) the LED is connected the correct way around
- 2) the correct resistor is used
- 3) the correct output pin number is being used in the program
- 4) all the solder joints are good



This program flashes the LED connected to output pin 0 on and off 15 times using a BASIC programming technique called a for...next loop (this technique cannot be used with flowcharts). The number of times the code has been repeated is stored in the memory of the PICAXE chip using a 'variable' called b1 (the PICAXE contains 14 variables labelled b0 to b13). A variable is a 'number storage position' inside the microcontroller than the microcontroller can use to store numbers as the program is carried out.

```
main:  for b1 = 1 to 15      ' start a for...next loop
        high 0             ' switch pin 0 high
        pause 500          ' wait for 0.5 second
        low 0              ' switch pin 0 low
        pause 500          ' wait for 0.5 second
    next b1                 ' end of for...next loop

    end                     ' end program
```

Switching more than one LED at once.

Sometimes it is useful to switch more than one LED on or off at the same time. This saves time when lots of high and low commands would have to be used together.

Pin	4	2	1	0
Value	16	4	2	1

The command that does this is called let pins =

After the equals sign a number is used. Each output pin is given a value, and the number used in the program is the sum of these values.

Therefore this program switches all of the outputs on, and then all off, and then one on at a time, in sequence.

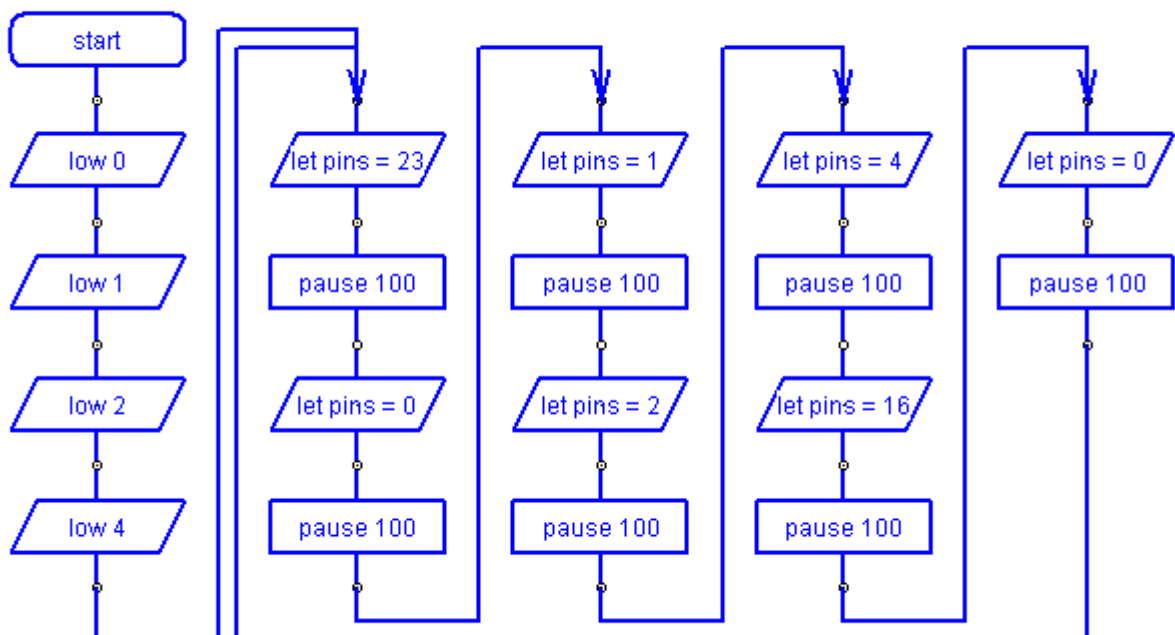
```

start: low 0          ' switch all outputs off
      low 1
      low 2
      low 4

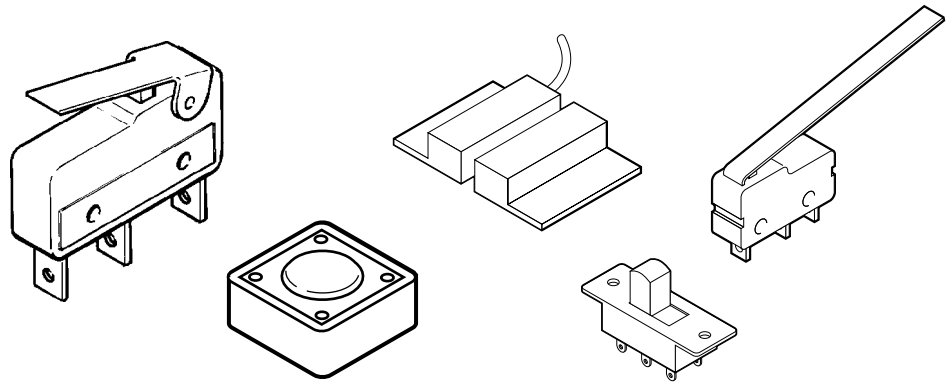
main:
      let pins = 23   ' switch all on (16+4+2+1)
      pause 100       ' wait for 0.1 second
      let pins = 0    ' switch all off
      pause 100       ' wait for 0.1 second
      let pins = 1    ' switch pin 0 on, others off
      pause 100       ' wait for 0.1 second
      let pins = 2    ' switch pin 1 on, others off
      pause 100       ' wait for 0.1 second
      let pins = 4    ' switch pin 2 on, others off
      pause 100       ' wait for 0.1 second
      let pins = 16   ' switch pin 4 on, others off
      pause 100       ' wait for 0.1 second
      let pins = 0    ' switch all off
      pause 100       ' wait for 0.1 second
      goto main       ' loop back to start

```

IMPORTANT! The let pins command only works after the pins have been set as outputs. To do this you must use a 'low' command for each pin at the start of the program.

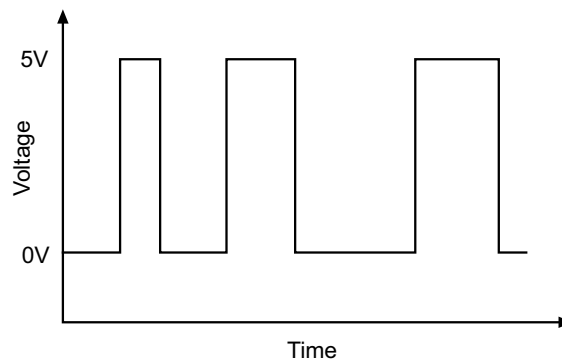


DIGITAL SENSORS (SWITCHES)



What are switches?

A digital sensor is a simple 'switch' type sensor that can only be 'on' or 'off'. If a graph is drawn of the on-off signals as the switch is pushed it will look like this:



Switches are electronic components that detect movement. There are a large number of different types of switches e.g:

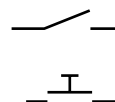
- push switches that detect a momentary 'push'
- micro-switches with long levers that detect small movements
- tilt-switches that detect jolting
- reed-switches that detect a magnet being moved

What are switches used for?

Push switches are commonly used on device like keypads. Micro-switches are used in burglar alarms to detect if the cover is removed from the alarm box. Reed switches are used to detect doors and windows being opened and tilt switches are often used to detect movement in devices such as toys, hair-dryers and tool-box alarms.

Switch Symbols.

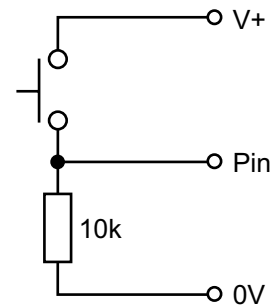
The symbols for a slide switch and a push switch are shown here.



Using switches

A switch is used with a resistor as shown in the diagram. The value of the resistor is not that important, but a 10k resistor is often used. When the switch is 'open' the 10k resistor connects the microcontroller input pin down to 0V, which gives an off (logic level 0) 0V signal to the microcontroller input pin.

When the switch is activated, the input pin is connected to the positive battery supply (V+). This provides an on (logic level 1) signal to the microcontroller.



Testing the switch

After connecting the switch it can be tested by a simple program like this. This program will switch an output on and off according to if the switch is pushed or not.

```

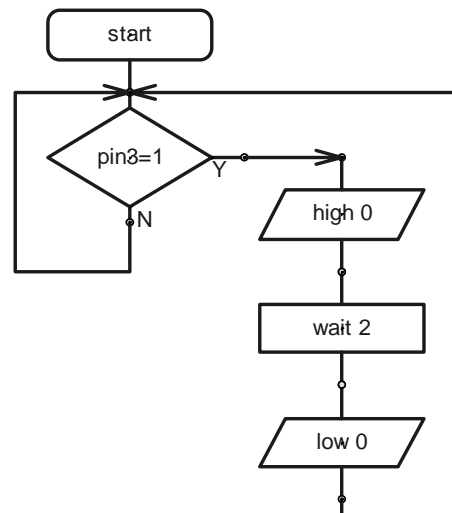
main:                                     ' make a label called 'main'
    if input3 is on then flash           ' jump if the input is on
    goto main                           ' else loop back around

flash:                                   ' make a label called 'flash'
    high 0                               ' switch output 0 on
    wait 2                               ' wait 2 seconds
    low 0                                ' switch output 0 off
    goto main                            ' jump back to start
  
```

In this program the first three lines make up a continuous loop. If the input is off the program just loops around time and time again.

If the switch is then pushed the program jumps to the label called 'flash'. The program then flashes output 0 on for two seconds before returning to the main loop.

Note carefully the spelling in the **if...then** line – **input3** is all one word (without a space). You can use the word **pin3** or **input3** to mean the same thing. Note also that only the label is placed after the command **then** – no other words apart from a label are allowed at this point.



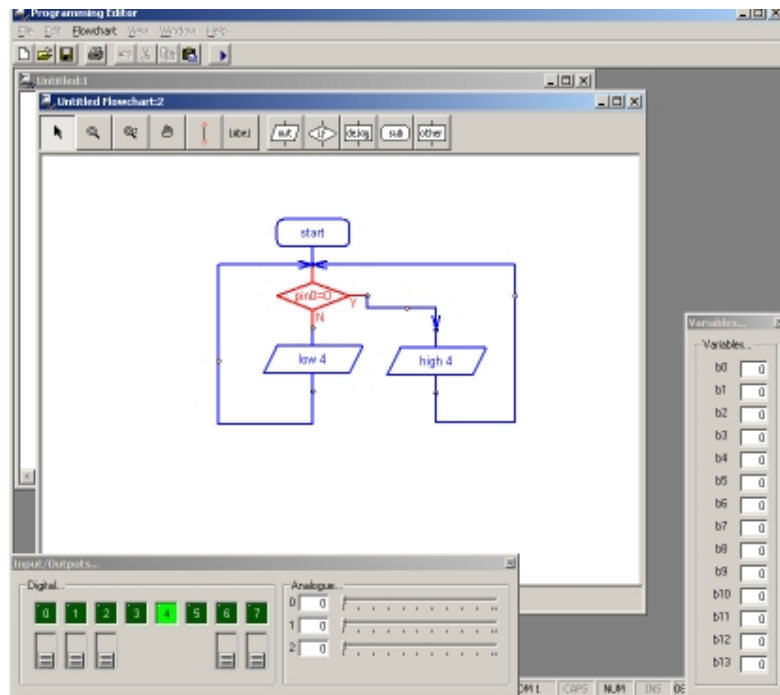
SECTION 3

PROGRAMMING - DRAWING FLOWCHARTS

Flowcharts are a useful tool that allow programs to be drawn graphically to make them easier to understand. The Programming Editor software includes a flowchart editor that allows flowcharts to be drawn on screen. These flowcharts can then be converted to BASIC listings for download into the PICAXE. The flowcharts can also be printed or exported as graphics files for inclusion within project portfolios.

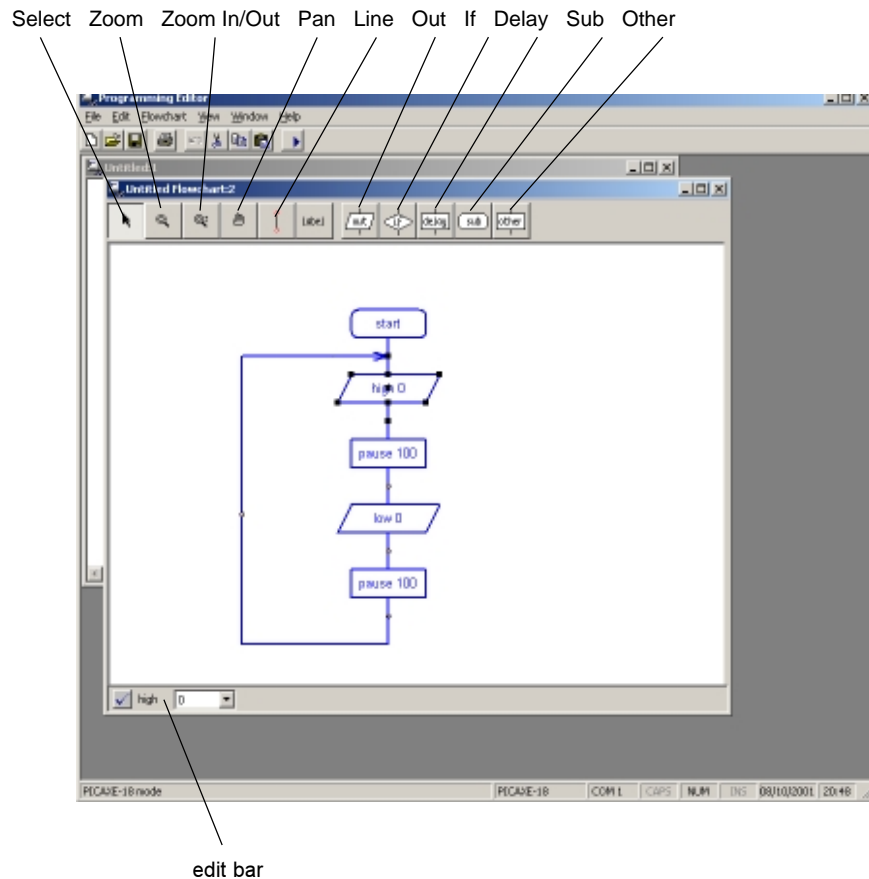
Detailed instructions for drawing/downloading a flowchart:

1. Connect the PICAXE cable to the computer serial port. Note which port it is connected to (normally labelled COM1 or COM2).
2. Start the Programming Editor software.
3. Select View>Options to select the Options screen (this may automatically appear).
4. Click on the 'Mode' tab and select PICAXE-08
5. Click on the 'Serial Port' tab and select the serial port that the PICAXE cable is connected to. Click 'OK'
6. Start a new flowchart by clicking the File>New Flowchart menu.
7. Draw the flowchart by dragging the correct symbols onto the screen, and then using the mouse to draw arrows between the symbols.
8. Once the flowchart is complete it can be converted into a BASIC program by selecting Flowchart>Convert Flowchart to BASIC. The BASIC program can then be downloaded into the PICAXE by clicking the PICAXE>Run menu.
9. To print or save the flowchart, use the File menu options. To export the flowchart as a graphic file, use the File>Export menu. To publish the image in a Word document select file type EME. To publish the flowchart on an internet web page use the GIF file type.



Flowchart Screen

The Flowchart Editor allows flowcharts to be drawn and simulated on-screen. The flowchart can then be automatically converted into a BASIC program for downloading into the microcontroller.



Flowchart Screen

Select Tool

Use this to select and move shapes. When a single shape is selected it's BASIC code can be edited in the edit bar at the bottom of the window.

Zoom

Use to zoom in to an area of the graph. Right click to zoom out.

Zoom In/Out

To zoom in click and move the mouse up. To zoom out click and move the mouse down.

Pan

Use this tool to move around the flowchart.

Line Tool

Use this tool to draw lines between shapes. Corners can be added by clicking once. When the line is near to a shape it will 'snap' to the connection point.

Label Tool

Use this tool to add descriptive labels or titles to the flowchart.

Out / If / Delay / Sub / Other

Click on these buttons to move to the command sub-menu to select commands.

Drawing Flowcharts

To draw a flowchart click on one of the command menu buttons (out / if / delay / sub / other) on the toolbar to move to the appropriate command sub-menu. Select the appropriate command and then click on the screen where the shape is required. Do not try to locate the shape precisely at first – just drop it in the general area and then use the select tool to move the shape to the correct position.

Once the shape is in position click on it so that it is highlighted. The BASIC code for the shape will then appear in the edit bar at the bottom of the screen. Edit the code as required.

For further information about each command see the 'BASIC Commands' help file.

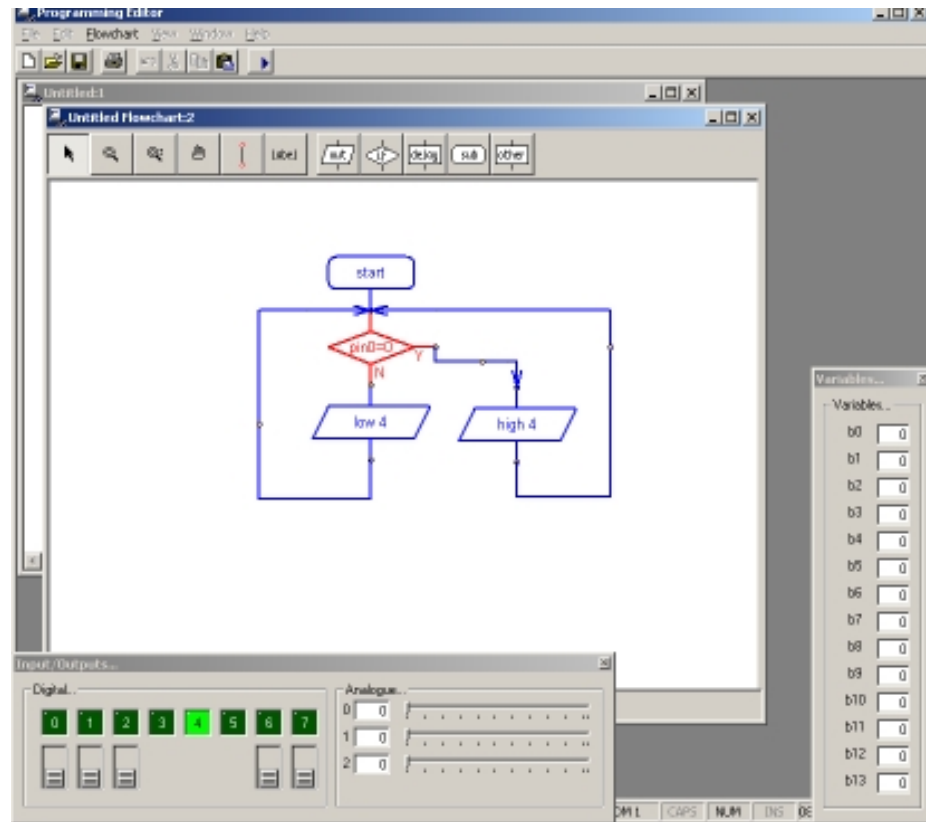
Joining Shapes

Shapes are joined by moving them close together until they 'snap' together. Alternately lines can be drawn between the shapes using the 'line tool' from the main toolbar. Note that it is only possible to join the bottom (side) of shapes to the top of other shapes (you cannot connect lines to lines). Only one line is allowed out of the bottom of each shape.

To enable neat diagrams, corners to the lines can be added by clicking with the mouse. When a line moves close to a connection point it will snap into position and then a click will finish the line.

Lines cannot be moved. If you try to move a line it will be deleted and a new line must be created.

On Screen Simulation



To simulate the flowchart, click 'Simulate' from the Flowchart menu. The program will then start to run on-screen.

As the program runs each cell is highlighted red as it is carried out. The 'Inputs/Outputs' and 'Variables' windows also appear when a simulation is being carried out. To adjust the input values click the on-screen switch (shown beneath the output LED) or slide the analogue input slider.

The time delay between shapes can be adjusted via the Flowchart options (View>Options>Flowchart menu).

Note that certain commands have no on-screen simulation equivalent feature. In this case the command is simply ignored as the flowchart runs.

Downloading Flowcharts

Flowcharts are not directly downloaded to the microcontroller. First the flowchart is converted into a BASIC program, which is then downloaded.

To convert a program select 'Convert' from the Flowchart menu. The BASIC program for downloading will then be created.

Shapes that are not connected to the 'start' or 'sub' shapes in the flowchart are ignored when the conversion takes place. The conversion will stop if an unconnected shape is found. Therefore always use a 'stop' shape or line to complete the flowchart before simulation or conversion.

Note that it is possible to quickly convert and then download a flowchart by pressing the shortcut key <F5> twice.

Using Symbols

Inputs, Outputs and Variables can all be renamed using the 'Symbol Table' from the Flowchart menu. When a symbol is renamed the new name appears in the drop-down menus on the edit bar. Note that you should not use commands (e.g. switch or sound) as a symbol as this will generate errors in your converted BASIC program.

Saving and Printing Flowcharts

Flowcharts can be saved, printed and exported as graphic files (for adding to word processor documents) via the File menu. Flowcharts can also be copied to the Windows clipboard (for pasting into other applications) via the Edit menu.

SECTION 4

PROGRAMMING - BASIC

Programming in BASIC is more powerful than using flowcharts. This is because BASIC contains more commands, eg. for...next loops, which cannot be used with the graphical flowchart methods. However you have to be more accurate in your 'typing' as no spelling mistakes are allowed!

The following program is a sample BASIC program which switches output 0 on and off every second. When you download the program an LED connected to output 0 would flash on and off every second..

```
main:
    high 0
    pause 1000
    low 0
    wait 1
    goto main
```

This program uses the **high** and **low** commands to control output pin 0, and uses the **pause** and **wait** commands to make a delay. Wait uses whole second units, whilst pause uses 1 millisecond (ms) units (1000 ms = 1 second). Therefore in this program both the delays are the same, just written in different ways.

The last **goto main** command makes the program 'jump' back to the label **main:** at the start of the program. This means the program loops forever. Note that the first time the label is used it must be followed by the colon (:) symbol. This tells the computer the word is a new label.

Detailed instructions:

1. Connect the PICAXE cable to the computer serial port. Note which port it is connected to (normally labelled COM1 or COM2).
1. Start the Programming Editor software.
2. Select View>Options to select the Options screen (this may automatically appear).
3. Click on the 'Mode' tab and select PICAXE-08
4. Click on the 'Serial Port' tab and select the serial port that the PICAXE cable is connected to. Click 'OK'
5. Type in the following program:

```
main:
    high 0
    pause 1000
    low 0
    wait 1
    goto main
```

(NB note the colon (:) directly after the label 'main' and the spaces between the commands and numbers)

6. Make sure the PICAXE circuit is connected to the serial cable, and that the batteries are connected.
7. Select PICAXE>Run. A download bar should appear as the program downloads. When the download is complete the program should start running automatically – the LED on output 0 should flash on and off every second.

Programming Editor Software Reminders:

Toolbar short-cuts:



To download/run a BASIC program:

1. Check the download cable is connected to the PICAXE and the computer's serial port
2. Check that the battery is connected to the PICAXE
3. Make sure the Programming Editor software is in the correct mode (look for 'PICAXE-08' in the statusbar at the bottom left of the screen).
4. Click **PICAXE>Run** (or the toolbar icon) (or press the shortcut key F5)

To save a program/flowchart:

1. Click **File - Save As...** (or the toolbar icon)
2. Type in a filename
3. Click <OK>

To open a saved program/flowchart:

1. Click **File - Open...** (or the toolbar icon)
2. Select the file type (BASIC or flowchart)
3. Select a filename from the list by clicking on it
4. Click <OK>

To start a new BASIC program:

1. Click **File - New**

To start a new flowchart:

1. Click **File - New Flowchart** (or the toolbar icon)

To on-screen simulate a flowchart:

1. Click **Flowchart - Simulate...** (or the toolbar icon)
2. Click on the flowchart to stop the simulation

To convert a flowchart to BASIC:

1. Click **Flowchart - Convert to BASIC...** (or press the shortcut key F5)

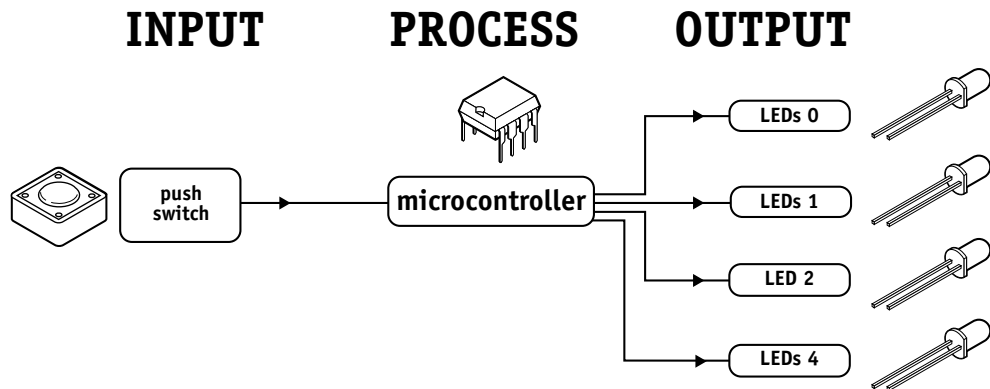
To print a program/flowchart:

1. Click **File - Print...** (or the toolbar icon)
2. If you want each program line printed in A BASIC program to have a number, make sure the 'Print Line Numbers' box is checked
3. Click <OK>

SECTION 5 - THE ELECTRONIC DICE PCB

The Electronic Dice project uses a PICAXE-08 microcontroller with LED outputs. The project also uses a switch to activate the LEDs.

The electronic block diagram is shown below.

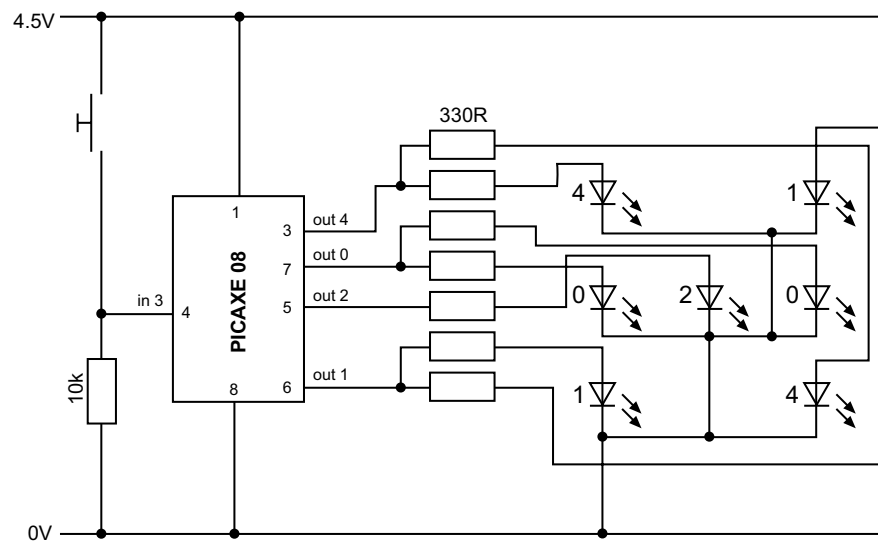


output - pin0 (leg 7) is connected to LEDs 0
 output - pin1 (leg 6) is connected to LEDs 1
 output - pin2 (leg 5) is connected to LED 2
 output - pin4 (leg 3) is connected to LEDs 4
 input - pin3 (leg 4) is connected to the switch

Remember not to confuse the chip 'leg' number with the input/output pin number!

Circuit Diagram

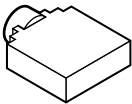
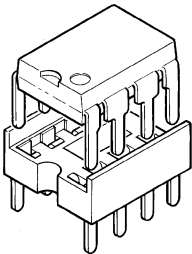
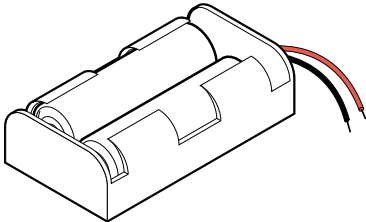
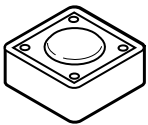
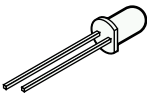
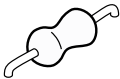
The circuit diagram for the electronic dice project is shown below:



BUILDING THE ELECTRONIC DICE PCB

What you will need:

- R1 to 7 330R resistor (orange orange brown gold)
- R8 22k resistor (red red orange gold)
- R9,10 10k resistor (brown black orange gold)
- LED1 to 7 5mm LEDs
- SW1 push switch
- IC1 8 pin IC socket
- IC1 PICAXE-08 microcontroller
- CT1 PICAXE download 3.5mm socket
- BT1 4.5V (3xAA) battery box
- PCB printed circuit board
- single core wire to connect LEDs and battery box (if required)



Tools:
soldering iron and solder
side cutters

Resistor colour codes

Black	0	0	Black x1	Silver $\pm 10\%$
Brown	1	1	Brown x10	Gold $\pm 5\%$
Red	2	2	Red x100	
Orange	3	3	Orange x1000	
Yellow	4	4	Yellow x10,000	
Green	5	5	Green x100,000	
Blue	6	6	Blue x1,000,000	
Violet	7	7		
Grey	8	8		
White	9	9		

Example shown:

blue, grey, brown, gold

= 680R $\pm 5\%$

Soldering the PCB.

The printed circuit board (PCB) is specially manufactured with a 'solder resist' layer to make it simpler to solder. This is the green 'lacquer' layer that covers the tracks so that the solder does not stick to these tracks. However for successful assembly the PCB must be carefully assembled and soldered.

When soldering always make sure the solder iron tip is hot and clean. To test if it is hot enough try to melt a piece of solder on the tip. The solder should melt almost instantly. Then clean off the melted solder by wiping the tip on a damp sponge.

Remember that solder will only 'stick' to hot surfaces. Therefore never melt the solder on the soldering iron tip and then try to 'drop' it onto the joint – this won't work as the joint will be cold and so the solder won't stick.

To successfully solder you must hold the soldering iron in one hand and the solder in the other. Therefore make sure the board is held on the table so it won't move (e.g. use a bulldog clip or get someone else to hold it for you).

Steps to soldering:

- 1) Clean the soldering iron tip on the damp sponge
- 2) Press the soldering iron tip against the pad on the PCB AND the leg of the component. Count to 3 to give the joint time to warm up.
- 3) Keep the soldering iron in position and touch the solder against the joint. Allow enough solder to melt to cover the joint.
- 4) Take the solder away first, then the soldering iron
- 5) Allow the solder to cool for about 5 seconds before trying to move the board.

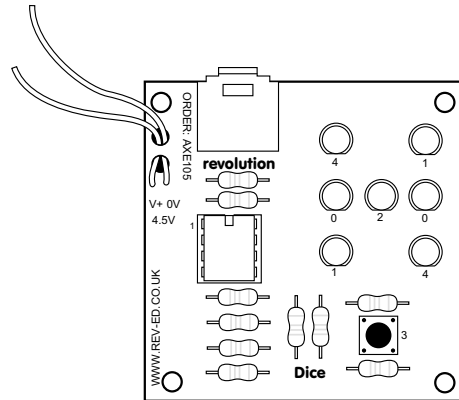
After each joint is made make sure it does not accidentally 'bridge' across to other joints. However be aware that some solder joints (e.g. on the two sides of the PICAXE download socket) have two wires very close together that are already connected by a track (line) on the PCB. In this case it does not matter if the solder joins together.

Tips!

- 1) Always start with the smallest components like the resistors. Then move onto larger components like the IC socket and then finish with the tall components like capacitors and transistors. Do not try to put all the components in position at once, only do two or three at a time.
- 2) Always make sure that the components lie flat on the board before they are soldered. When using components with long legs like resistors and LEDs, bend the legs so that the component is held firmly in position before soldering.
- 3) Make sure the PICAXE stereo download socket 'snaps' into position flat on the board before it is soldered.
- 4) Make sure that the components that only work one way around (LEDs, diodes, transistors and capacitors) are correctly aligned before soldering (see the marks on the PCB).
- 5) Piezo sounder wires are very thin. Make sure you do not overheat them or they may melt.
- 6) Always thread the battery snap wires down and up through the two thread holes before soldering. This helps make a much stronger joint which is less likely to snap off.

With the electronic dice the LEDs and switch can be soldered directly to the PCB or connected via wires. The switch wires should be connected to the two diagonal pads marked with a rectangle.

- 1) Place the 22k (red red orange gold) resistor and the two 10k (brown black orange gold) resistors in position. Bend the legs to hold the resistors in position and then solder.
- 2) Place the seven 330R (orange orange brown gold) resistors in position and solder. This may be easier if the resistors are positioned in two groups. Bend the legs to hold the resistors in position and then solder.
- 3) Push the PICAXE stereo download socket onto the PCB and **make sure it clicks into position** (so that it lies flat on the board). Solder the five metal square contacts (the five round plastic support post holes do not have to be soldered). Do not worry if the solder joins on the two metal contacts either side of the socket as they are supposed to be joined anyway.
- 4) Push the IC socket into position. Make sure the notch at one end points towards the top. Fold the legs over to hold the socket in position and then solder.
- 5) Solder the push switch in position. The switch only fits one way around.
- 6) Solder the LEDs into position. Make sure that the flat on one side of the LED aligns with the flat marked on the PCB
- 7) Thread the battery clip down through the large hole by the letters AXE. Thread it back up through the large hole by the letters 105 then solder the black wire into the hole marked 0v and the red wire into the hole marked V+
- 8) Carefully check the board to make sure there are no missed joints or accidental solder bridges.
- 9) Insert the microcontroller into the socket, ensuring pin1 faces the resistors.



Testing your circuit.**Step 1 – Check the solder joints.**

Check that the solder does not accidentally bridge between two pads. This is most likely to happen on the LEDs. On the stereo socket the two square pads close together on each side can be joined as they are already joined by a track on the board. However they must not be joined to the central round hole.

Step 2 - Check the components.

- 1) Check that the black battery clip wire is in the hole marked '0V' and the red battery clip wire is in the hole marked 'V+'
- 2) Check that the PICAXE-08 chip is in the socket correctly, with the dent (showing pin1) closest to the stereo socket.
- 3) Check that the flat edge of the LEDs is connected to the correct hole on the PCB.
- 4) Check that the stereo socket is correctly soldered, including the middle square pad which is often forgotten by mistake.

Step 3 - Connect the battery.

Check the 3 AA batteries are in the battery box correctly. Connect to the battery snap and then put your finger on the PICAXE chip. If it starts to get hot remove the battery box immediately as there is a problem – most likely that the chip or the battery wires are around the wrong way.

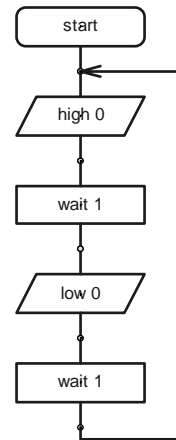
Step 4 – Download a program to test LED 0.

Connect the cable to the back of the computer and to the PICAXE socket on the PCB. Make sure the cable is pushed fully into the socket on the PCB.

Make sure the software is in the PICAXE-08 mode and the correct serial port is selected (see section 4 of this booklet for more information).

Type in and download the following program:
program like this:

```
main:
    high 0
    wait 1
    low 0
    wait 1
    goto main
```



The LEDs should flicker as the program downloads. After the download is complete the LEDs should flash on and off every second. If the LED does not flash check that it is around the correct way and that the 120R resistors are in the correct positions on the PCB.

If the program does not download check that the 22k, 10k, socket and IC socket are all soldered correctly. Use a multimeter to make sure you are getting 4.5V across the top legs (1 and 8) of the microcontroller. Check that the cable is pushed firmly into the socket and that the correct serial port is selected within the software.

Step 5 – Test LED 1.

Repeat the program in step 4, but use high 1 and low 1 instead of high 0 and low 0. This will test the other LEDs.

Step 6 – Test LED 2.

Repeat the program in step 4, but use high 2 and low 2 instead of high 0 and low 0. This will test the other LED.

Step 7 – Test LED 4.

Repeat the program in step 4, but use high 4 and low 4 instead of high 0 and low 0. This will test the other LEDs.

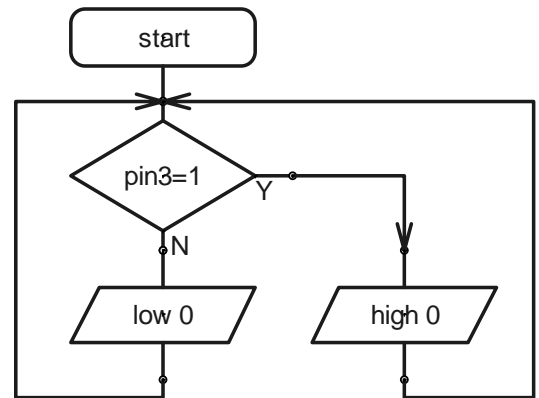
Step 8 – Test The switch.

```

main:
    if pin3 = 1 then LEDon
    low 0
    goto main

LEDon:
    high 0
    goto main

```



The LED should light as you press the switch. If they do not check that the switch and 10k resistor are correctly soldered.

If all these tests pass, you can be congratulated as you have correctly built and assembled your Electronic Dice! It is now time to develop and test your own program!

SECTION 6 - PROGRAM IDEAS.

Now that you have assembled and tested your Electronic Dice, it is time to develop your own program. This program can make different LED patterns appear on the Electronic Dice.

Included on the next pages are two example programs. These are designed to give you a starting point for your program. You may choose to modify them or to start a completely new program if you prefer.

Be creative!

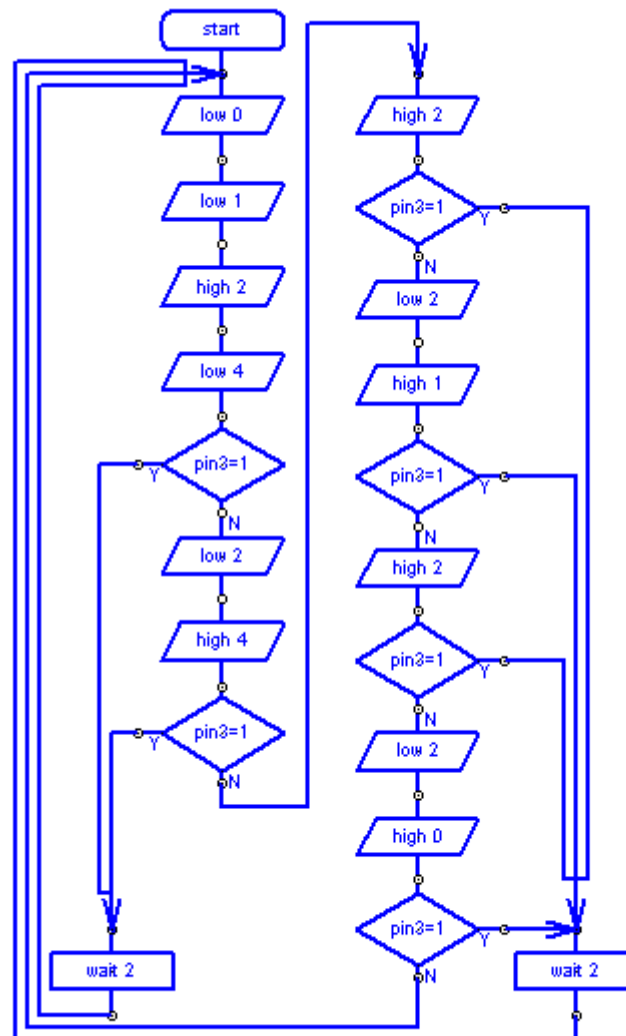
Program 1 Explanation

This program has a loop which counts through all the possible dice patterns so quickly they cannot be seen. When the button is pressed, the loop pauses for two seconds so the 'rolled' number can be seen..

Program 2 Explanation

This program uses the random command to generate a number. Because the random command generates a number between 0 and 255, this number has to be split into 'zones' consisting of a sixth of the highest number. For example, if the number generated is between 0 and 42, the microcontroller lights the LEDs for the number 1. If the number is between 43 and 84, the LEDs for the number 2 are lit etc. Note also the use of the 'pins' command, to keep the program shorter, and the 'low' commands at the start of the program to ensure the LED pins are correctly configured as outputs before the 'pins' command is used.

Program 1



Program 1

```

***** Dice *****
main:
    low 0          'switch other pins off
    low 1
    low 4
    high 2         'switch on the LEDs for 1

    if input3 is on then display

    low 2
    high 4         'switch on the LEDs for 2

    if input3 is on then display

    high 2         'switch on the LEDs for 3

    if input3 is on then display

    low 2
    high 1         'switch on the LEDs for 4

    if input3 is on then display

    high 2         'switch on the LEDs for 5

    if input3 is on then display

    low 2
    high 0         'switch on the LEDs for 6

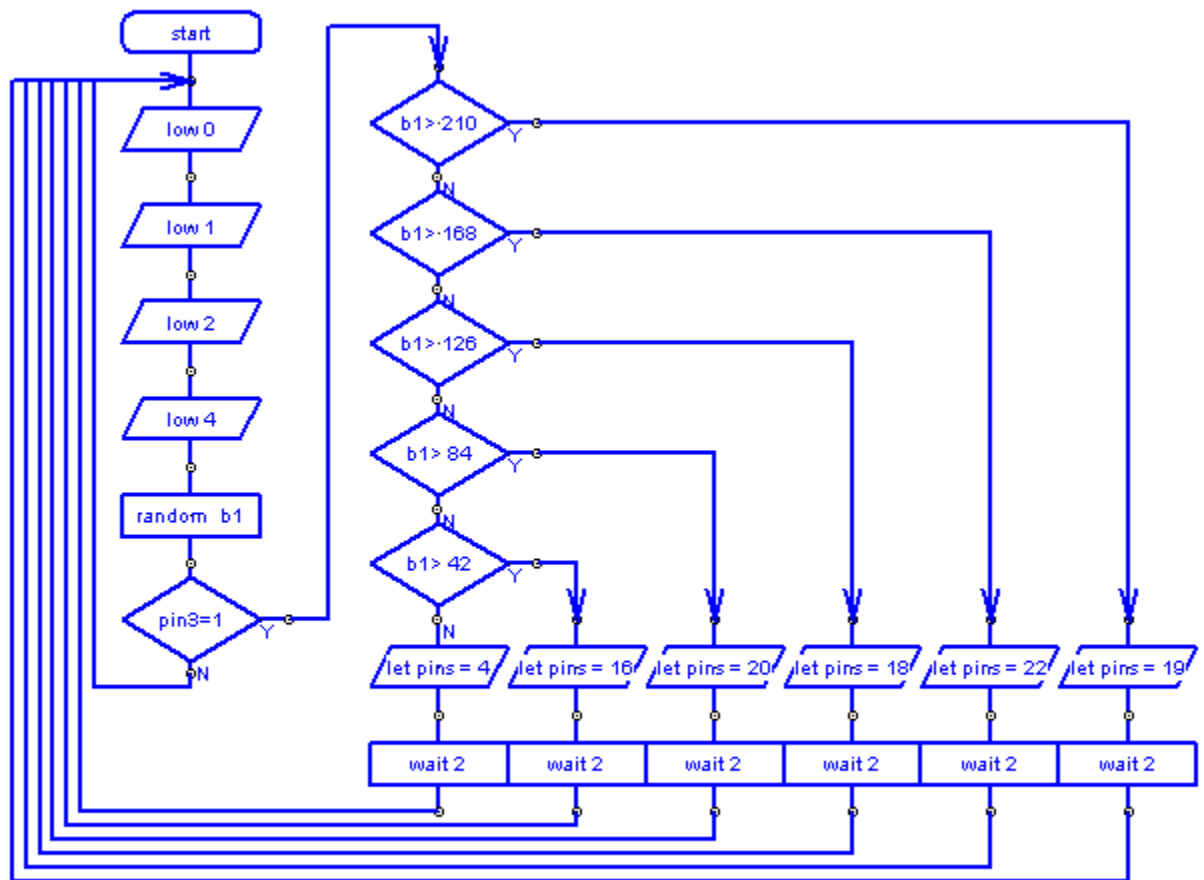
    if input3 is on then display

    goto main

display:
    wait 2         'wait 2 seconds
    goto main

```

Program 2



Program 2

```

' ***** Dice using random command *****
main:
    low 0          ' make LED pins outputs
    low 1
    low 2
    low 4
    if input3 is on then makenumber      ' test switch
    goto main

makenumber:
    if b1 > 210 then showsix          ' test random number and jump
    if b1 > 168 then showfive
    if b1 > 126 then showfour
    if b1 > 84  then showthree
    if b1 > 42  then showtwo

showone:
    let pins = 4    ' show LEDs for 1
    wait 2
    goto main

showtwo:
    let pins = 16   ' show LEDs for 2
    wait 2
    goto main

showthree:
    let pins = 20   ' show LEDs for 3
    wait 2
    goto main

showfour:
    let pins = 18   ' show LEDs for 4
    wait 2
    goto main

showfive:
    let pins = 22   ' show LEDs for 5
    wait 2
    goto main

showsix:
    let pins = 19   ' show LEDs for 6
    wait 2
    goto main

```

ACKNOWLEDGEMENT

This project development was funded by the
UK Offshore Oil and Gas Industry.
www.oilandgas.org.uk/education/

(c) Revolution Education Ltd 2002
www.rev-ed.co.uk

All rights reserved.

May be photocopied for non-commercial educational
use in classrooms in schools and colleges only.

PICAXE is a trademark of Revolution Education Ltd

