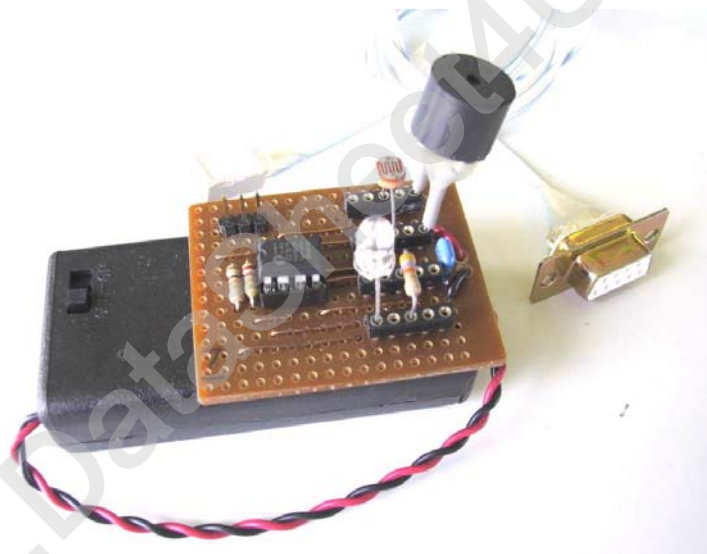


# THE PICAXE08

FOR BEGINNERS



A basic introduction to the PICAXE 08

Written by Adrian Steel

The PICAXE is a very cheap and very versatile programmable chip, which, with a little knowledge and imagination, can perform all sorts of useful and wonderful tasks.

This course is aimed at providing you with enough knowledge so that you feel comfortable with the chip and its programming language.

There are a lot of programming words to learn, and in this course we will be looking at the most useful ones.

The chip uses a computer and a simple programming language to program it with. Most versions of Windows will be OK, including XP. Your computer **MUST** have a useable COM port to attach the programming lead to. For those of you without COM ports, BELKIN make an excellent USB to COMPORT adapter. There are others but a lot of them will not work with the PICAXE. The Belkin one works fine.

The programming lead will need to be made up by you. Instructions on how to do this are found later on in this worksheet.

### ***THE SOFTWARE.***

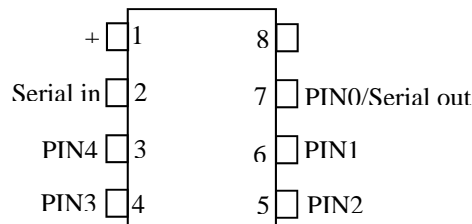
The software that is used by the PICAXE is a free download from the website ([www.rev-ed.co.uk](http://www.rev-ed.co.uk))

When it runs you will be presented with an OPTIONS screen. Here you will need to tell the programmer which type of PICAXE you are using. In this case choose PICAXE-08 (it might already be selected)

Next you will have to tell the program which COM port your lead is plugged into. Click the left hand button on SERIAL PORT tab. Any available COM ports will be shown. Those that **CANNOT** be used are greyed out. Select the right COM port.

### ***THE CHIP***

The chip is an 8 pin device. Of these 8 pins 5 are 'control' pins. To add to the confusion the programmable pins have been numbered differently to their positions on the chip (see diagram below)



NOTE PIN0 must always be an OUTPUT and PIN3 must always be an INPUT.

The other pins 1,2,4, can be either INPUT or OUTPUT.  
3—4.5 Volts

The chip is powered up using CHIP PIN1 to connect to positive and CHIP PIN8 to connect to negative. 2 resistors connect to CHIP PIN2 which is the input from the programming lead.

### ***BUILDING THE PROGRAMMER***

Look at your PROGRAMMER OVERLAY to see how the chip is wired up.

Place the overlay sheet onto the breadboard.

Build the programmer onto the breadboard, by placing the components through the overlay into the board

Once you have built it, test it works by doing the following.

Connect the battery pack

Connect the programming lead ( make sure that pin 3 on the lead plugs into the same row as CHIP-PIN8 )

Press F3 on the computer keyboard.— you should get a DOWNLOAD window. If not check your connections. Make sure the battery pack is switched on.

### ***FLASHING AN LED***

In order to flash an LED we need to provide power to the LED for a period of time and then remove the power.

We can do this using the PICAXE output pins.

The command HIGH sets a pin to positive. The command LOW sets a pin to negative.

Place a red LED into the breadboard so that the positive leg is in PIN1 and the negative leg is on the ground line

Write the following program . NOTE do not type the text after the \*\*\*\*\* this is to tell you what each line does

### ***LED FLASHER***

This program will flash an LED on and off at half second intervals.

Start:	*****	A label used for program flow make sure it has a colon (:)
at the end of the line)		
High 1	*****	Sets PIN1 to high puts a positive on PIN1 ( turns LED on)
Pause 500	*****	The program pauses here in milliseconds Therefore 500 is
half a second		
Low 1	*****	Sets PIN1 to low ( turns LED off)
Pause 500	*****	Another half second delay
Goto start	*****	Send the program back round to the first line

Once you have type it in. download the program to the chip by pressing the F5 key . The DOWNLOAD strip should appear and the SUCCESS alert should show once it has been successfully downloaded. The led should now flash.

### ***PULSOUT***

At first look you might think that this command does the same job as the previous program. In some respects it does, however it goes about it in a entirelyly different way. The syntax for the command is this....

pulsout pin, time

Where pin is the pin to which your device is attached i.e. and LED and time is the length of the pulse in microseconds.

Click on the FILE dropdown menu and select NEW

Type in the following program

Start:	*****	Program label
pulsout 1,50000	*****	Send a pulse of half a second to pin 1
pause 500	*****	Will pause for half a second
goto start	*****	loop program back round to the start

Press F5 to download the program to the chip. Can you see that the LED flashes exactly the same as before. If you wish to check, open up the previous program window by minimising the window you have just types into and opening up the first program window. Press f5 to download the program and see the LED flash exactly as it did before.

So far then apart from saving a few lines of code, PULSOUT seems to be doing exactly the same job as out first program.

Now try this..

Add a second LED by placing in on the breadboard so that the positive leg is in PIN2 and the negative leg is on the ground line .

Open a new program window and type in the following

Start:	****	program label
High 1	****	Sets PIN1 to high puts a positive on PIN1 ( turns LED on)
High 2	****	Sets PIN2 to high puts a positive on PIN1 ( turns LED on)
Pause 500	****	The program pauses here in milliseconds
Low 1	****	Sets PIN1 to low ( turns LED off)
Low 2	****	Sets PIN1 to low ( turns LED off)
Pause 500	****	Another half second delay
Goto start	****	Send the program back round to the first line

You can now see that BOTH LED's flash at the same time. We are switching them both on at the same time, even though they are on separate lines of code, and we are switching them off at the same time.

Now do this

Open up a new programming window

Type in the following

```
start:  
pulsout 1,50000  
Pulsout 2,50000  
pause 500  
goto start
```

Download the program to your chip.

Watch the LED's flash. Can you see the difference between the two programs now. In the first where we use the HIGH command and LOW command both LED's light at the same time and go off at the same time. When we use PULSOUT each LED gets a pulse for the predetermined amount of time. As such they alternate.

In most instances if you want to flash LED's then use the HIGH LOW command options.

### ***CHALLENGE NUMBER 1***

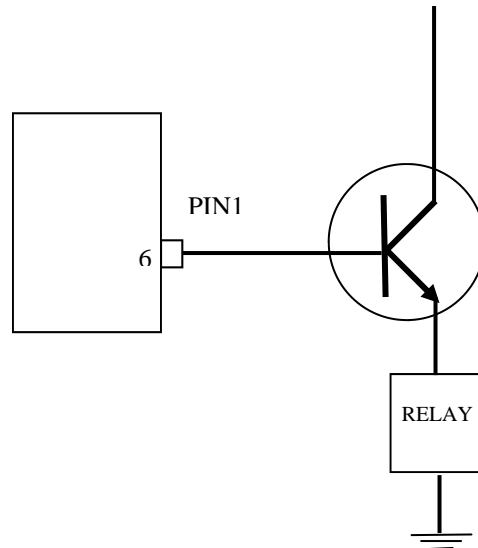
By using the HIGH and LOW commands, get the LED's to light up alternately.

Write your code in the box below.

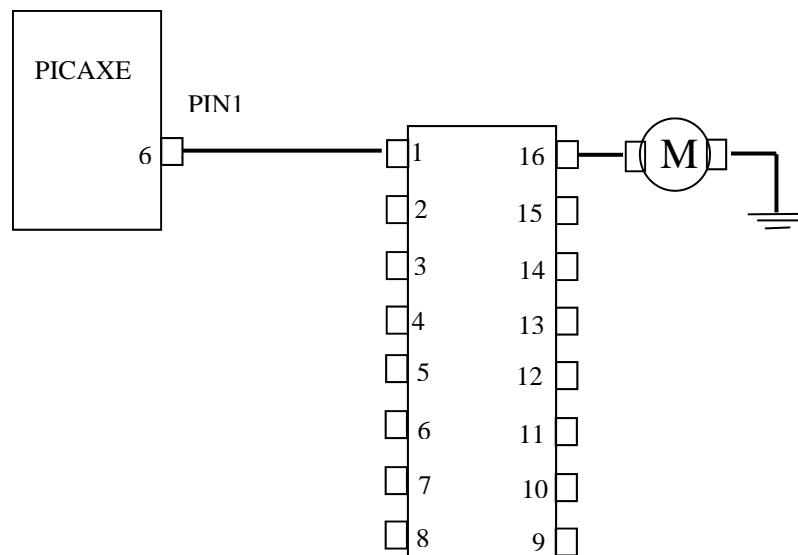
## ***MORE OUTPUTS.***

Of course the output pins will drive more than just LED's. Motors, solenoids, relays etc.. All can be driven by the chip. However, the higher current devices such as motors and relays need a transistor driver to pass the higher current.

The diagram below shows how a relay could be connected to the PICAXE



A transistor driver chip ULN2003 is a better option. It is a 16 pin IC which has 6 high current transistors built into it. It can be used for most high current devices. See the diagram below



## ***INPUTS***

An input to a pin can either be a logic 1 (positive) or a logic 0 (negative) depending on what it is your trying to do.

The chip can detect a button being pressed, or an LDR receiving light or shadow, a thermistor getting hot or cold etc.

We are going to hook up a push button and get an led to flash when the button is pressed.

Fit the pushbutton between PIN3 and positive.

Fit a 10K resistor between PIN3 and ground.

We need both of these attached to PIN3 as the resistor will give us a negative ( logic 0) and the pushbutton will give us a positive ( logic 1)

Open up a new editor window and type the following

```
start:
if pin3=1 then ledon
if pin3=0 then ledoff
goto start
```

```
ledon:
high 2
wait 5
low 2
goto start
```

```
ledoff:
low 2
goto start
```

Download the program (F5) and test it. If you push the button the led should light up stay lit for 5 seconds and then go out. If you do not push the button then the LED should remain off

Let us look at what is going on

if pin3=1 then ledon this is testing to see if pin 3 has a positive on it. It will get one if the button is pushed as the button is connected to the positive. If the button IS pressed the program will jump to the label ledon and run the program from that point.

if pin3=0 then ledoff this is testing to see if pin 3 has a negative on it. It will get one if the button is NOT pushed as the button is connected to the negative via the resistor. If the button is NOT pressed the program will jump to the

***CHALLENGE NUMBER 2***

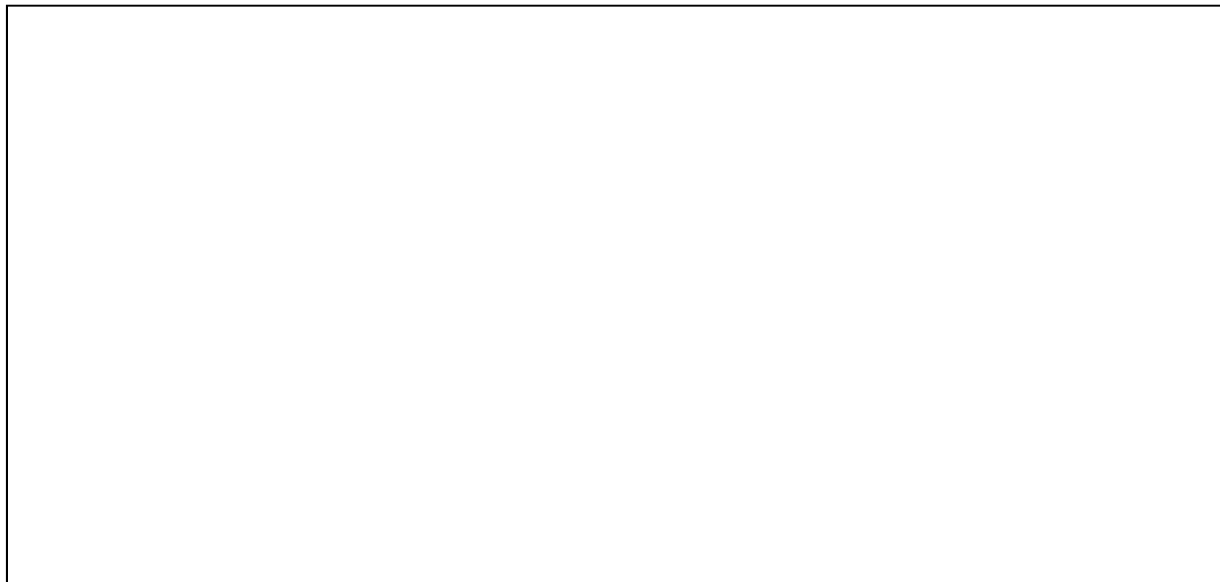
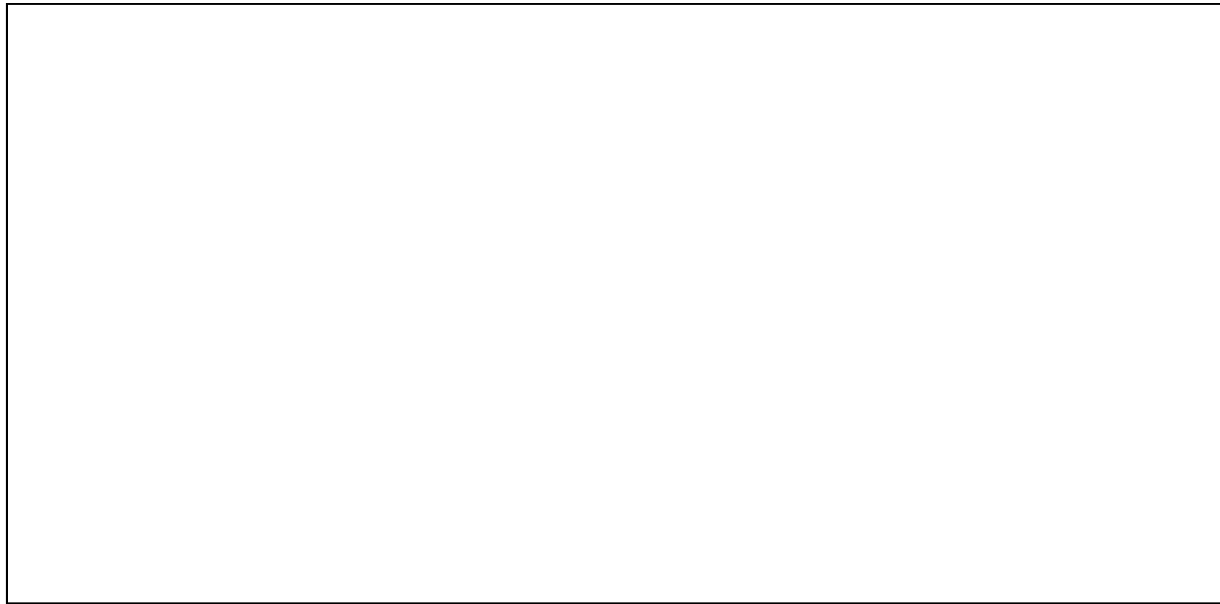
Write a program that will flash an LED if a button is pushed.

Use the box below to write your program and use the programs you have written already to help you.

***CHALLENGE NUMBER 3***

Write a program that will flash an LED if a button is pushed, and light another steady, if the button is NOT pushed

Use the box below to write your program





## *SOUND.*

The PICAXE is capable of driving a speaker direct. All you need to do is hook a speaker up to an output pin and use the correct command to get a noise out.

Syntax        sound pin, (note, duration)

                  i.e. sound 1,(57,1)

This will play the sound whose value is 57 for a duration of 1.

NOTE this can be in the range of 0 to 255, with 0 being no sound to 255 being a hiss. Notes 1-127 are ascending notes and notes 128-255 are white noise.

DURATION can be in the range of 1-255. Now I have not had much luck so far finding out what the duration actually is. But after some experimentation have discovered that 50 is about one second. Experiment yourself with the DURATION to get a feel for it.

You can play more than one note in the same statement.

Sound,(note,duration,note,duration,note,duration.....)

For instance the following line will play the notes A , B, C in turn ,each note lasting for 1 second

Sound,(49,50,54,50,57,50)

### ***LIST OF NOTES***

**A=49    A#=51    B=54    C=57    C#=61    D=65    D#=71    E=78    F=88    G=100**

### ***PLAYING NOTES***

Firstly hook up the piezo speaker to the chip. The negative if marked goes to the ground line and the positive if marked goes to PIN2.

Open up anew editor window and type in the following

sound 2, (57,50)

This will play the note C for about 1 second

Try experimenting with the DURATION. Then try experimenting with the NOTE.

Now try playing more than one not in the same statement. Also try experimenting with differing durations for the notes in the same statement. Remember if you want a rest ( period) in your tune then set the note to 0

Finally experiment with sounds above 128, white noise

## ***FOR NEXT LOOP***

The best way to experiment with the SOUND command is to use the FOR-NEXT loop. For those of you who are not familiar with this command I will explain it here.

The FOR syntax is   for variable = n1 to n2

What this means is we are going to run a counter which will start at n1 which is a number you will choose, usually 1, and then end at a higher number n2, again which you will choose.

In order to keep track of this counter we need to store it in a variable .

So the command might look like this

```
For b0=1 to 5
```

What this means is we are going to carry out a sequence 5 times and keep track of the count using the variable b0.

In order to loop the program round after each count we use the NEXT command.

So a program might look like this

```
For b0=1 to 5  
Print "hello"  
Next
```

Lets look at how that works.

Firstly when the program runs it sets b0 to 1. Then it goes on to the next line and prints HELLO Then it goes to the NEXT statement where it checks to see if the number stored in the variable b0 equals the higher number you set which in this case is 5. If it doesn't equal the higher number, the program loops back to the FOR statement. Now as it runs through again it sets the variable b0 to 2. Again it prints HELLO and again it goes to next to do a check. It will repeat this process until the number stored in the variable b0 DOES equal the higher number set. So the above program will run through 5 times, each time printing HELLO, so you end up with 5 HELLO's.

NOTE. Unlike the normal computer BASIC programming language, You cannot actually print to the screen using the PICAXE programming language. You can however print to a LCD screen hooked up to the PICAXE. But that is a more advanced skill.

Now because the number count is a variable, we can use it to our advantage.

Instead of using the note number in our SOUND command we can use a variable instead. The same for DURATION. We can use a variable for the duration of our note.

## ***MORE SOUND***

By using a FOR-NEXT loop we can quickly and easily program some cool sound effects

There are 127 notes that can be played. So in order for you to play them all you would need to write the SOUND command 127 times. However with a FOR NEXT loop we can do it in 3 lines

Consider the program below

```
For b0=1 to 127
Sound 2,(b0,10)
next
```

What is happening is that the first time the program runs it sets the variable b0 to 1. In the next line we are using pin2 to play the sound. The value of that sound is stored in the variable b0, which in this case is 1. and finally we are playing that sound for a duration of 10.

Now because the program count has not yet achieved the higher number ( 127) the NEXT part send it back to the beginning again.

When the program runs through a second time the SOUND value will be 2. so that value of note will be played.

And so on and so on until the count reaches the higher number of 127, where the program stops.

What this means is we are playing a steadily increasing note for a duration of 10.

Try it for yourself. Open a anew program window and type in the program above.

## ***ADVANCED FOR NEXT LOOP***

We can extend the capabilities of the for next loop by altering the count.

The following command will count in steps of 10

```
For b0=1 to 127 step 10
```

So the count will go 1 , 11, 21, 31 etc.

The following command will count DOWN in steps of 1

```
For b0=127 to 1 step -1
```

The following command will count down in steps of 10

```
For b0=127 to 1 step -10
```

Try them yourself.

## ***SOME GOOD SOUNDS***

The following are a few good sounds to try.

### ***AMERICAN POLICE SIREN***

flash:

for b2=100 to 125

sound 2,(b2,1)

next

for b2=125 to 100 step -1

sound 2,(b2,1)

next

goto flash

### ***BRITISH POLICE CAR SIREN***

start:

Sound 2,(100,50)

pause 100

sound 2,(85,50)

pause 100

goto start

### ***TWINKLE TWINKLE***

Sound

2,(57,50,57,50,80,50,80,50,85,50,85,50,80,100,74,50,74,50,71,50,71,50,64,50,64,50,57,100)

### ***RISING TONE***

for b0=1 to 127

sound 2,(b0,5)

Next

### ***FALLING TONE***

for b0=127 to 1 step -1

sound 2,(b0,5)

Next

Experiment with the above values to get some other effects.

There are a lot of other commands that the PICAXE uses. I suggest you visit the REV-ED site on the internet ([www.rev-ed.co.uk](http://www.rev-ed.co.uk)) and download all the PDF files regarding the PICAXE 08.

The other commands are more specialist and refer to specific inputs and functions.

Below is a list of these commands to whet your appetite

HIGH	Sets a pin high
LOW	Sets a pin low
TOGGLE	Toggle the state of an output pin
OUTPUT	Sets a pin as an output
INPUT	Sets a pin as an input
REVERSE	Reverse the output/input state of a pin
PULSOUT	Sends a pulse out to a pin
IF ... THEN	Conditional program loop
PUSLIN	Measures the length of an input to a pin
SOUND	Sends a noise out to a pin
PWM	Provides Pulse width Modulation ( good for controlling the speed of a motor)
READADC	Read an analogue channel into a variable
FOR..NEXT	Conditional program loop
BRANCH	Jump to an address specified
GOTO	Jump to a another place in the program
GOSUB	Temporarily jump to another place in the program then return
RETURN	The return command for GOSUB
(LET)	Perform variable mathematics
LOOKUP	Look up data and store in a variable
LOOKDOWN	Find targets match and store in a variable
RANDOM	Generate a random number
SEROUT	Output serial data from out put pin
SERIN	Input serial data to an input pin
EEPROM	Store data in EEPROM before downloading a BASIC program
READ	Read data EEPROM into a variable
WRITE	Write data into EEPROM location
NAP	Enter low power mode for a short period ( up to 2.3 secs)
SLEEP	Enter low power mode for period up to 65535 secs
END	power down until reset, an indefinite sleep
PAUSE	Wait for up to 65535 milli secs in milli seconds
WAIT	Wait for up to 65 seconds in seconds
DEBUG	Outputs variable data to PC via programming lead

Have fun with this chip. Remember the only limitation is your imagination. And If your not sure if it will work, then try it you will be surprised what effects you can get by experimentation with the programming language

ADRIAN STEEL  
Director Of Electrotechnology  
Rangitoto College  
Email [steela@rangitoto.school.nz](mailto:steela@rangitoto.school.nz)

# CONNECTION DIAGRAMS

