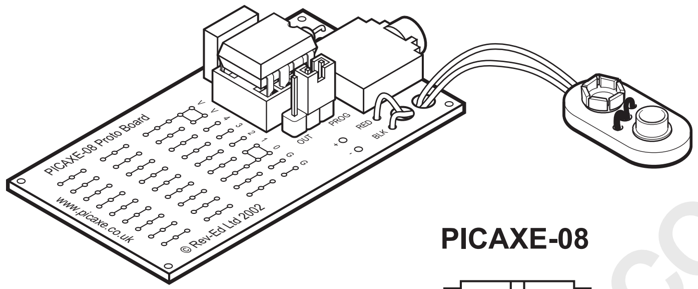
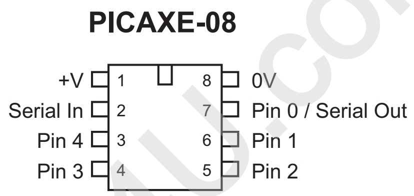
PICAXE-08 MICROCONTROLLER PROGRAMMING SYSTEM





The ‘PICAXE’ is an easy-to-program microcontroller system that exploits the unique characteristics of the new generation of low-cost ‘FLASH’ memory based microcontrollers. These microcontrollers can be programmed over and over again without the need for an expensive programmer.

The power of the PICAXE-08 system is its simplicity. No programmer, eraser or complicated electronic system is required - the microcontroller is programmed (with a simple ‘BASIC’ program of graphical flowchart) via a 3-wire connection to the computer’s serial port. The operational PICAXE circuit uses just 3 components and can be easily constructed on a prototyping breadboard, strip-board or PCB design.

The PICAXE-08 microcontroller provides 5 input/output pins.

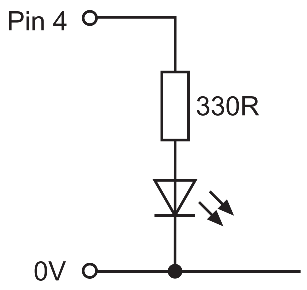
- *low-cost, simple to construct circuit*
- *5 input/output pins (1 input with analogue capabilities)*
- *rapid download via serial cable*
- *free, easy to use Programming Editor software*
- *simple to learn BASIC language*
- *can also be programmed via flowcharts*
- *free manuals and online support forum*

The comprehensive starter pack includes the following items:

- *PICAXE-08 prototype board*
- *download cable*
- *CDROM containing software and manuals*
- *PICAXE-08 microcontroller chip*

Downloading your first program

This first simple program can be used to test your system. It requires the connection of an LED (and 330R resistor) to output pin 4. If connecting the LED directly to a PICAXE chip on a home-made board or proto board, connect the LED between the output pin and 0V (G). (Make sure the LED is connected the correct way around!).



Note: The PICAXE microcontroller can be connected to a computer via the PICAXE-08 proto board OR via a simple self-made circuit board. Circuit connections are provided later in this booklet and in th e 'interfacing electronics' booklet.

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 Mode 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.
6. Click 'OK'
7. Type in the following program:

```
main:  high 4
       pause 1000
       low 4
       pause 1000
       goto main
```

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

8. Make sure the PICAXE circuit is connected to the serial cable, and that the batteries are connected. Make sure the LED and 330R resistor are connected to output 4. If using the proto board make sure the download header is in the 'PROG' position.
9. Select PICAXE menu>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 4 should flash on and off every second.

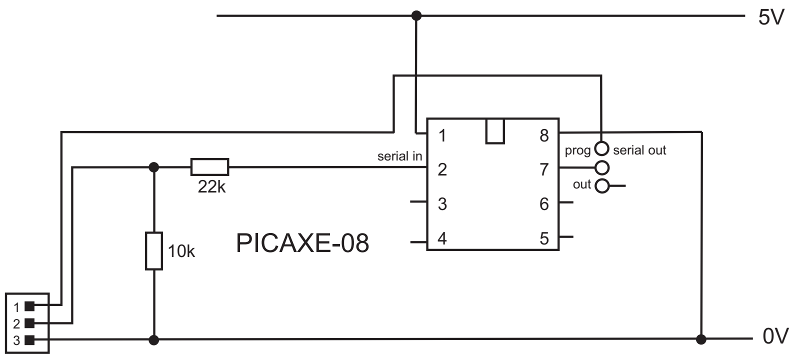
Trouble-shooting

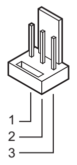
If an error message appears check the on-screen warning description. Common mistakes are:

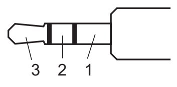
- *Not connecting the cable to the PICAXE circuit or computer serial port.*
- *Not placing the download jumper in the 'PROG' position.*
- *Not selecting the correct COM port setting within the Programming Editor software.*
- *Not connecting the battery, or trying to use a flat battery.*

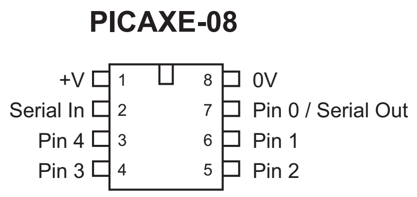
The PICAXE-08 Circuit

The basic PICAXE-08 circuit is shown below.









The PICAXE-08 microcontroller

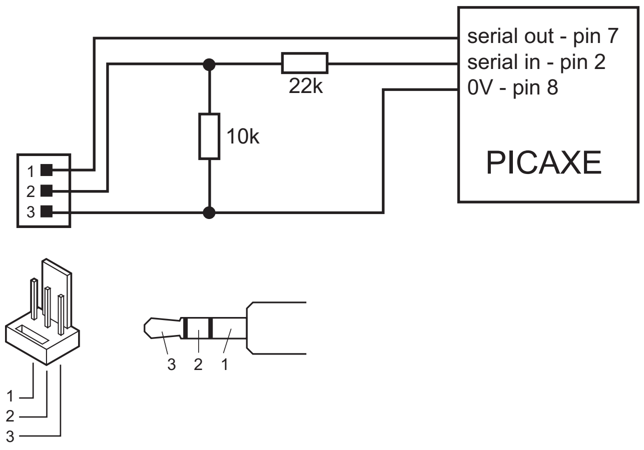
Please note that the PICAXE-08 microcontroller is not a blank microcontroller! The PICAXE microcontroller is pre-programmed with a bootstrap program that enables the direct cable download. Blank microcontrollers will not contain this bootstrap program and so cannot be used within the PICAXE system.

Also remember that the input/output pin numbers are not numbered the same as the physicalexternal leg numbers!

This booklet provides a brief introduction to the PICAXE-08 system. For more detailed information please see the 'PICAXE Tutorial', the 'BASIC Commands' and 'Electronic Interfacing' help files.

The PICAXE computer interface circuit

The PICAXE system uses a very simple interface to the computer serial port. Although this interface does not use true RS232 voltages, it is very low-cost and has proved to work reliably on almost all modern computers.

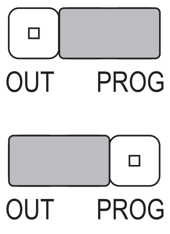


It is strongly recommended that this interfacing circuit is included on every PCB designed to be used with the PICAXE microcontroller. This enables the PICAXE microcontroller to be re-programmed without removing from the PCB.

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

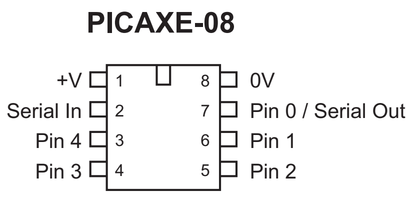
When using a computer with the older 25-pin serial port connector, use a 9-25 way mouse adapter to convert the 9-pin PICAXE cable. These adapters can be purchased from all good high street computer stores.

Jumper:
The PICAXE-08 proto and motor driver boards have a jumper to switch between the serial output (required during a download) and the normal output pin 0, as both functions use the same leg of the PICAXE-08 micrcontroller. Therefore it is essential the jumper is in the correct position when programming or running the program.



Inputs and Outputs

The PICAXE-08 microcontroller has 5 input/output pins. Unlike the larger PICAXE microcontroller (where the pins are pre-defined) the user can select whether some of the pins are used as input or as outputs.



Pin 0 must always be an output, and pin 3 must always be an input (this is due to the internal construction of the microcontroller). The other 3 pins can be selected to be inputs or outputs, and so the user can select any input/output combination between the limits of 1 input-4 outputs and 4 inputs-1 output.

In addition pin 1 also contains a low-resolution analogue to digital converter and so can be used as an analogue input pin if required.

Important - Don't Get Confused!

The input/output pin numbers are NOT the same as the external 'leg' numbers, as the input/output pin numbering follows the microcontrollers manufacturers port allocation. To avoid confusion this booklet always talks about 'legs' where referring to the external physical location of the input/output pin.

Leg	Description	Notes
1	Positive Supply, V	Use a 3V to 6V battery pack as the power supply
2	Serial In	Used for the program download
3	Pin 4	Input or output
4	Pin 3	Input only
5	Pin 2	Input or output
6	Pin 1	Input or output or analogue input
7	Pin 0 / Serial Out	Output only. Also used for program download
8	Ground, G	Connect to the power supply (0V)

Special Note - Output Pin 0

Pin 0 (leg 7) is used during the program download, but can also be used as a normal output once the download is complete. On the project boards a jumper link allows the microcontroller leg to either be connected to the download socket (PROG position) or to the output (OUT position). Remember to move the jumper into the correct position when testing your program!

If you are making your own pcb you can include a similar jumper link or small switch, or you may prefer to connect the microcontroller leg to both the output device and the program socket at the same time. In this case you must remember that your output device will rapidly switch on and off as the download takes place (not a problem with simple outputs like LEDs, but could cause problems with other devices such as motors).

Selecting Inputs or Outputs.

When the PICAXE-08 first powers up, all pins are configured as input pins (except pin0, which is always an output). There are three methods of setting the other pins to be outputs (if required)

Method 1 – use a command that requires the pin to be an output.
This is the simplest method, used by most educational users. As soon as a command that involves an output pin (such as high, low, toggle, serout or sound) is used, the PICAXE-08 microcontroller automatically converts the pin to an output (and leaves the pin as an output).

Therefore the simplest way to setup outputs is just to put a 'low' command at the start of the program for each output pin. This tells the microcontroller to make the pin an output, and to make sure the output is condition low (off).

Method 2 – use the input and output command.
The command 'output ?' (where ? is the pin number) can also be used to tell the pin to be an output at the start of a program. Likewise the 'input ?' command can be used to set the pin as an input, although this is not normally necessary as most of the pins are set as inputs by default. Note that the output command does not set the pin into a known high or low state, so it is often preferable to use the 'low' command instead.

The input and output commands have no effect on pin 0 (output) and pin 3 (input), which cannot be altered.

Method 3 – (advanced) use the let dirs = command
The 'let dirs = %000100111' command can be used to simultaneously set all the pins at the same time. This is quicker than using multiple input/output commands but requires an understanding of binary bits (explained in the Basic Commands manual). Placing a 0 for the pin number bit will make the corresponding pin an input, a 1 will make the pin an output. The value of bits 0,3,5,6,7 can be either 0s or 1s as they have no effect on the microcontroller and are simply ignored.

Selecting pin 1 to be an analogue input.

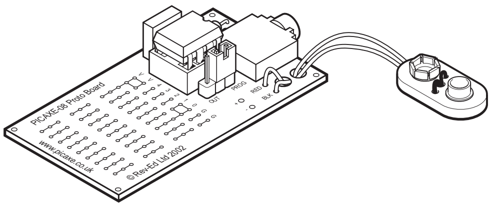
Use of the readadc command will automatically configure pin 1 to be an analogue input. Therefore use the command 'readadc 1,b2' whenever you wish to take an analogue reading (presuming use of variable b2 to store the analogue reading).

Building Your Circuit

The PICAXE-08 circuit can be quickly constructed on a prototyping breadboard, stripboard or by designing a pcb. However to enable rapid prototyping three kits are also available.

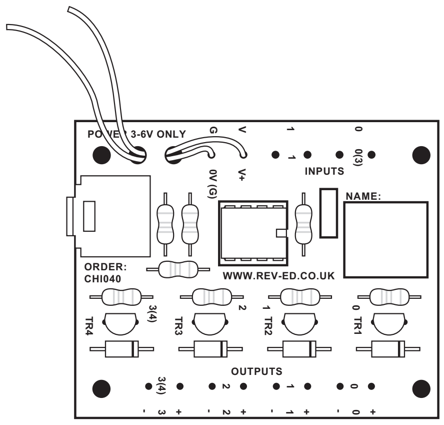
AXE021 PICAXE-08 Proto Board

The proto board is designed to allow the user to rapidly prototype projects using the PICAXE-08 microcontroller. The board provides the basic download circuit, with a small 'prototyping' area to solder in input and output components. Using this board the inputs and outputs can be configured in any combination.



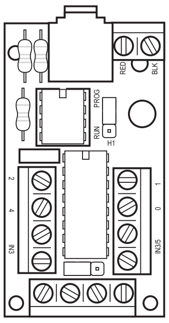
CHI040 Project Board

The project board provides the basic ciruit with up to four outputs and one output . The outputs are buffered by transistors. If desired the unused outputs can be used as inputs in addition to the standard single input.



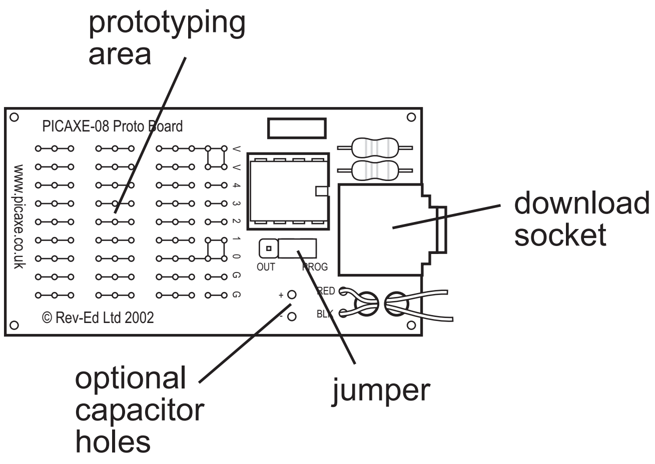
AXE022 8 pin motor driver board

The motor driver board is pre-configured to provide 4 power outputs and one input. The outputs are buffered by an L293D motor driver IC, which also enables the outputs to be used in 'pairs' to give forward-reverse-stop control of DC motors.



The PICAXE-08 Proto Board (AXE021)

The PICAXE-08 proto board is designed to allow the user to rapidly prototype projects using the PICAXE-08 microcontroller. The board provides the basic download circuit, with a small 'prototyping' area to solder in input and output components.

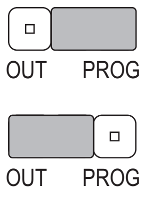


The prototyping area consists of pads which are pre-connected in sets of three pads (marked by lines on the top of the board). Each microcontroller input/output pin is connected to two holes at the top of the prototyping area, and multiple V+ and G (0V) power connection pads are also provided. This allows the user to rapidly develop simple interfacing circuits without the requirement for any other board.

Power should be connected to the board via a battery snap, which can be threaded through the two large holes to create a stronger connection point. The PICAXE-08 microcontroller will function at battery voltages between 3 and 6V (do not use a 9V PP3 battery).

When using electrically 'noisy' components (e.g. motors or buzzers) it is recommended a capacitor (e.g. 100uF 16V electrolytic) is soldered in the optional capacitor holes to help smooth the power supply.

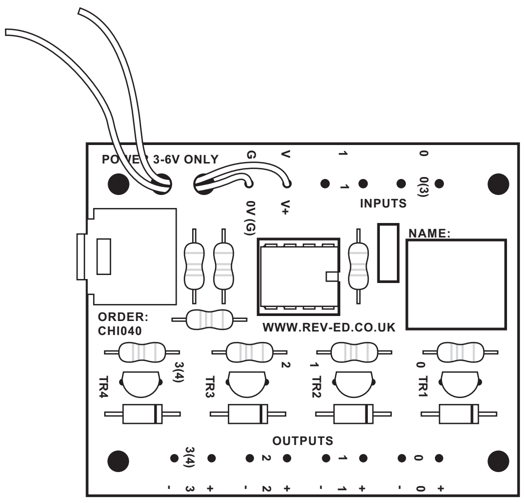
Note that the jumper must be used to connect the serial output (PROG position) during a download, or the output pin 0 (OUT position) during normal operation, as both functions share the same microcontroller leg.



The 8 pin Project Board (CHI040)

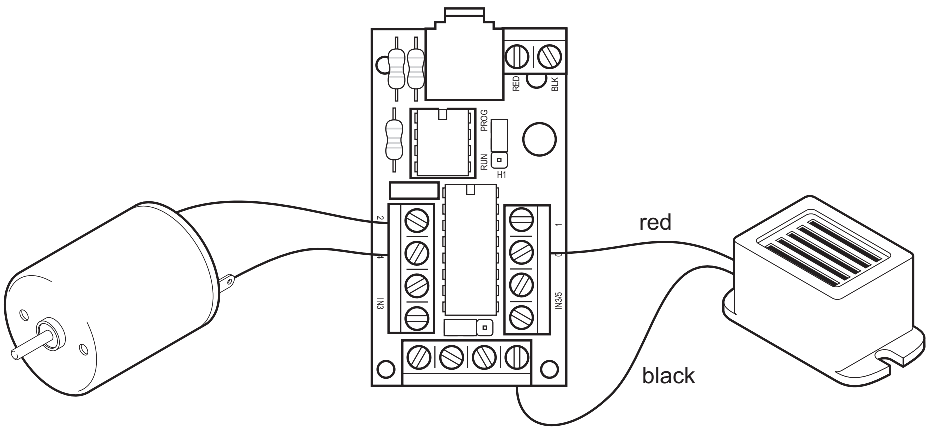
The 8 pin project board provides four outputs which are buffered by BC548B transistors. A single input is also provided. This board provides a convenient method of creating a project that has 4 outputs and 1 input.

The outputs are provided on pins 0, 1, 2 and 4. The input is provided on pin 3.
See the CHI040 datasheet for further information.



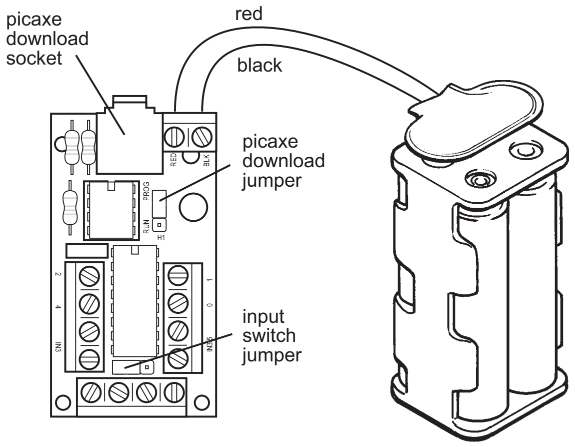
The 8 pin Motor Driver Board (AXE022)

The 8 pin motor driver board provides 4 outputs on pin numbers 0,1,2 and 4. Pin 3 is used as an input.



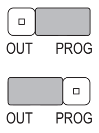
The outputs can be used individually to driver electronic devices such as buzzers and signal lamps. To connect outputs in this way the red wire is connected to the terminal block at the side of the board and the black wire is connected to 0V (G) at the bottom of the board.

The outputs can also be used in pairs to give forward-reverse stop control of motors, so the board will drive up to two motors. To connect motors in this way the two wires from the motors are connected to the pair of outputs (0 and 1 or 2 and 4) at the side of the board.



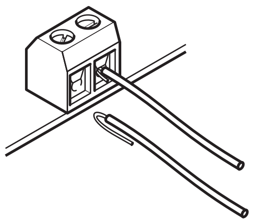
Power should be connected to the board via a battery snap, which can be threaded through the two large holes to create a stronger connection point. The PICAXE-08 microcontroller will function at battery voltages between 3 and 6V (do not use a 9V PP3 battery). When using electrically 'noisy' components (e.g. motors or buzzers) it is recommended a capacitor (e.g. 100uF 16V electrolytic) is soldered in the optional capacitor holes to help smooth the power supply.

Note that the jumper must be used to connect the serial output (PROG position) during a download, or the output pin 0 (OUT position) during normal operation, as both functions share the same microcontroller leg.



Connecting Wires to the Board

To make stronger connections it is suggested that when connecting wires the 'bare end' of the wire is folded back over the insulation so that both the insulation and the bare wire are tightened into the cage of the screw terminal. This makes a stonger joint than just connecting the bare wire alone.



Controlling Motors.

Motors are controlled by pairs of outputs as shown by the table below.

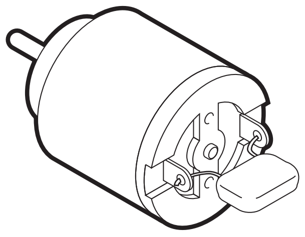
Note that there is a 1.5V voltage drop within the L293D driver chip and so, for instance, if a 6V supply is used the motor voltage will be 4.5V.

The chip is designed to run warm in use. This is normal.
The direction of rotation of the motors is defined as follows:

Pin 0	Pin 1	Motor A
off	off	off
on	off	forward
off	on	reverse
on	on	off

Pin 2	Pin 4	Motor B
off	off	off
on	off	forward
off	on	reverse
on	on	off

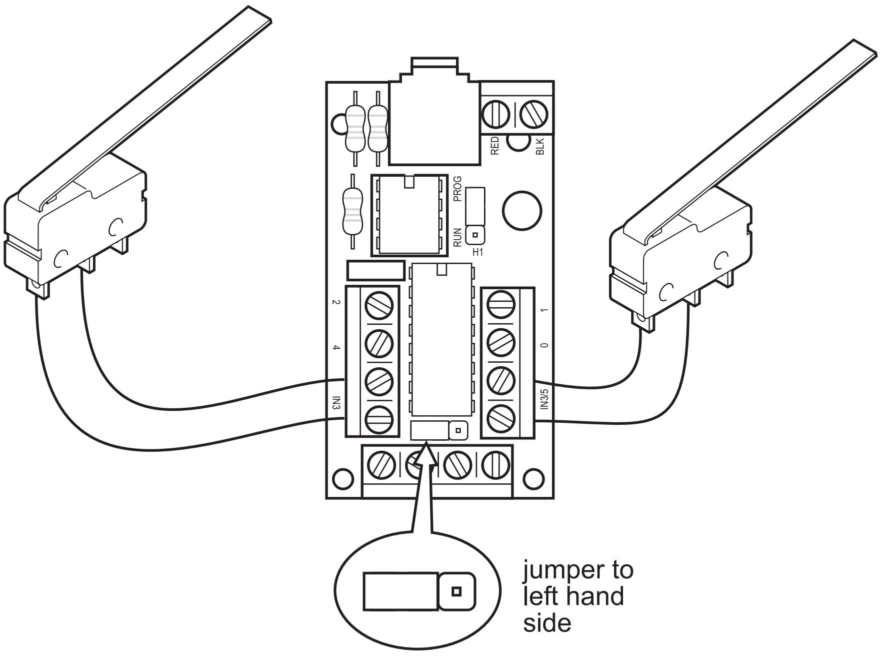
Noise Suppresion on motors:
Note that motors should be suppressed by soldering a 220nF polyester capacitor across the motor terminals to prevent electrical noise affecting the circuit.



Connecting the Input Switch

Only pin3 can be used as an input with the motor driver board, and the input header (H2) below the L293D chip must be in the left hand side position (pin5 cannot be used with the PICAXE-08 system as this is the serial input pin).

However two switches may be connected to the board, one either side as shown below. Note that when switches are connected like this they are connected in parallel, so either of the switches can be pressed to activate the input.

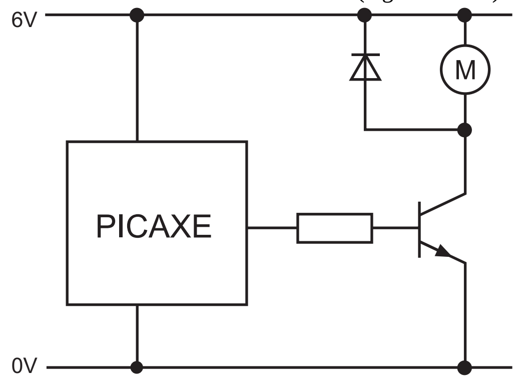


General Interfacing Circuits

(See the Interfacing Electronics help file for more information)

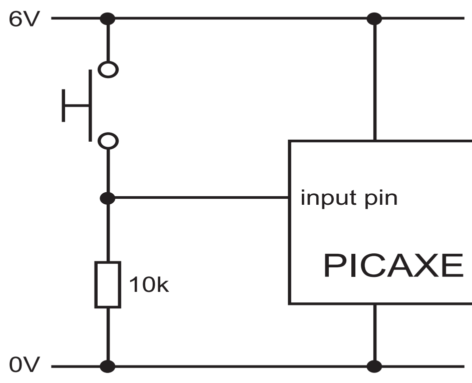
Digital Outputs

Digital outputs can be interfaced via a transistor (e.g. BC548B) as shown below.



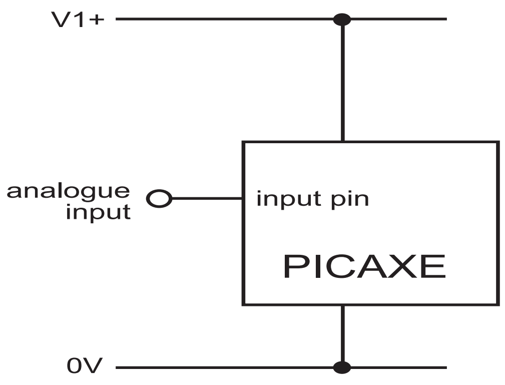
Digital Inputs

Digital inputs can be interfaced with a 10k pull down resistor as shown below.



Analogue Inputs

Only pin 1 can be used as an analogue input. The pin is automatically configured as a digital input until a 'readadc' command is used on that pin. At this point the pin is automatically reconfigured as a low-resolution analogue input.



BASIC Programming Basics

The following programs are included to provide a brief introduction to a few of the main programming techniques. All programs can be tested by connecting an LED (with a 330R resistor) to outputs 4 and 0, a switch to input 3, a piezo sounder to pin 2 and an LDR (with 1K resistor) to analogue channel 1. This arrangement is also used on the Cyberpet PCB (AXE101) which can also be used for each of these exercises.

For further details about each program see the Basic Commands, Electronics Interfacing and PICAXE Tutorial helpfiles within the Programming Editor software

Switching outputs on and off

The following program switches output 4 on and off every second. The program demonstrates how to use the high, low, wait, pause and goto commands.
When you download this program the LED on output pin 4 should flash on and off continuously.

```
main:                   ' make a label called 'main'
      high 4            ' switch output 4 on
      wait 1            ' wait 1 second
      low 4             ' switch output 4 off
      pause 1000        ' wait 1000ms (= 1 second)
      goto main         ' jump back to start
```

The first line simply adds a label called 'main' to the program. A label is used as a positional marker in the program. In this program the last line uses the command 'goto main' to jump back to the first line. This creates a loop that continues forever.

Note the 'high 4' command automatically sets pin4 to be an output pin.

A label can be any word (apart from keywords such as 'high'), but must begin with a letter. When the label is first defined it must end with a colon (:). The colon 'tells'the computer that the word is a new label.

It is usual to put some spaces (or a tab) at the start of each line, apart from where the line starts with a label. This makes the program easier to read and understand. Comments can also be added after an apostrophe (') symbol, to make the program operation easier to understand.

Note that the commands wait and pause both create time delays. However wait can only be used with whole seconds, pause can be used for shorter time delays (measured in milliseconds).

Detecting Inputs

The following program makes output pin 4 flash every time a switch on input pin 0 is pushed.

```
main:                           ' make a label called 'main'
      if pin3 = 1 then flash    ' jump if the input is on
      goto main                 ' else loop back around

flash:                                ' make a label called 'flash'
      high 4                    ' switch output 4 on
      pause 500                 ' wait 0.5 second
      low 4                     ' switch output 4 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 4 on for half a second before returning to the main loop.

Note carefully the spelling in the if…then line – pin3 is all one word (without a space). Note also that only the label is placed after the command then – no other commands apart from a label are allowed after then.

Using Symbols

Sometimes it can be hard to remember which pins are connected to which devices. The 'symbol' command can then be used at the start of a program to rename the inputs and outputs.

```
symbol red = 4          ' rename output4 'red'
symbol green = 0        ' rename output0 'green'

main:                   ' make a label called 'main'
      high red          ' red LED on
      wait 2            ' wait 2 seconds
      low red           ' red LED off
      high green        ' green LED on
      wait 1            ' wait 1 second
      low green         ' green LED off
      goto main         ' jump back to the start
```

For…Next Loops

It is often useful to repeat the same part of a program a number of times, for instance when flashing a light. In these cases a for…next loop can be used.

```
symbol red = 4                ' rename output4 'red'
symbol counter = b0           ' define a counter using variable b0

main:                         ' make a label called 'main'
      for counter = 1 to 25   ' start a for…next loop
         high red             ' red LED on
         pause 500                  ' wait 0.5 second
         low red              ' red LED off
        pause 500             ' wait 0.5 second
      next counter            ' next loop
      end                     ' stop the program
```

In this program all the code between the for and next lines is repeated 25 times. The number of times the code has been repeated is stored in a variable called 'counter', which in turn is a symbol for the variable 'b0'. There are 14 available variables, b0 to b13, which can be used in this way. A variable is a location in the memory where numbers can be stored.

Using Variables and making sounds

The variables are also commonly used to store 'numbers' as a program runs. This program makes lots of different sounds on the piezo sounder on pin 2.

```
main:                         ' make a label called 'main'
      let b0 = b0 + 1         ' add one to b0
      sound 2,(b0,50)         ' make a sound of pitch b0
      pause 10                ' wait 0.01 second
      goto main               ' jump back to the start
```

Note that b0 is a byte variable. This means it supports any value between 0 and 255. This means that the program above will eventually 'overflow' at the highest number i.e. …253-254-255-0-1-2… This is an important fact to remember when performing mathematics with byte variables. Note that the piezo sounder works best with pitch values between 50 and 150, values above 150 do not tend to give usable sounds.

For further details about the mathematical capabilities of the PICAXE microcontroller see the BASIC Commands help file.

Reading Analogue Input Channels

The value of the analogue input can be easily copied into a variable by use of the 'readadc' command. The variable value (e.g. light level) (value between 0 and 160) can then be tested. The following program switches on one LED if the value is greater than 150 and a different LED if the value is less than 100. If the value is between 100 and 150 both LEDS are switched off.

```
main:                           ' make a label called 'main'
      readadc 1,b0              ' read channel 1 into variable b0
      if b0 > 150 then red      ' if b0 > 150 then do red
      if b0 < 100 then green    ' if b0 < 100 then do green
      low 4                     ' else switch off 4
      low 0                     ' and switch off 0
      goto main                 ' jump back to the start

red:                            ' make a label
      high 4                    ' switch on 4
      low 0                     ' switch off 0
      goto main                 ' jump back to start

green:                          ' make a label
      high 0                    ' switch on 0
      low 4                     ' switch off 4
      goto main                 ' jump back to start
```

Note that the PICAXE-08 microcontroller has 1 analogue channel on pin 1.
When using analogue sensors it is often necessary to calculate the 'threshold' value necessary for the program (ie the values 100 and 150 in the program above). The debug command provides an easy way to see the 'real-time' value of a sensor, so that the threshold value can be calculated by experimentation.

```
main:                           ' make a label called 'main'
      readadc 1,b0              ' read channel 0 into variable b0
      debug b0                  ' transmit value to computer screen
      pause 100                 ' short delay
      goto main                 ' jump back to the start
```

After this program is run a 'debug' window showing the value of variable b0 will appear on the computer screen. As the sensor is experimented with the variable value will show the current sensor reading. Please note that due to the low-cost, low-resolution internal resonator of the PICAXE-08 microcontroller the debug command may not workcorrectly with all computer systems.

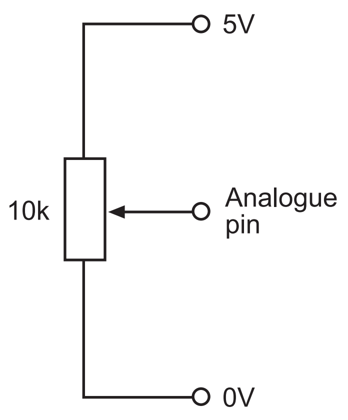
Using the Low-Resolution Analogue Input

A standard analogue input will provide 256 different analogue readings (0 to 255) over the full voltage range (e.g. 0 to 5V). A low-resolution analogue input will provide 16 readings over the lower two-thirds of the voltage range (e.g. 0 to 3.3V). No readings are available in the upper third of the voltage range.
To ensure consistency between standard and low-resolution analogue input readings, the low-resolution reading will 'jump' in 16 discrete steps between the nearest standard readings, according to the table below.

Standard Reading Range	Low Resolution Reading
0-10	0
11-20	11
21-31	21
32-42	32
43-52	43
53-63	53
64-74	64
75-84	75
85-95	85
96-106	96
107-116	107
117-127	117
128-138	128
139-148	139
149-159	149
160-170	160
Values greater than 170 (170-255)	160

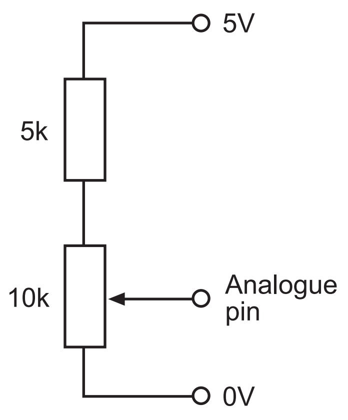
Advanced technical information on using the low resolution ADC

The low-resolution analogue pin 1 within the PICAXE-08 has an internal fixed resistor at the upper end of the voltage range. This results in an unavoidable 'dead-spot' between the standard values 160 and 255. Therefore if a simple potentiometer is used as the sensor (between 0 and 5V), this will result in the analogue value not changing above 160 for approximately the last third rotation of the potentiometer spindle.



For many projects, particularly when using LDR or thermistor sensors, this is of no consequence and the upper 'non-changing' area is simply ignored.

However, if desired, this area can be avoided by using an additional fixed resistor at the top of the potential divider circuit. This fixed resistor should have a value of 0.5 of the total resistance of the original potential divider circuit.



For example, when using a 10k potentiometer, a fixed 5k (5k1) resistor should also be included. In this case the highest analogue value read will still not exceed 160, but the changing values will be distributed over the full turning range of the potentiometer spindle (rather than two thirds of the turning range of the spindle).

Appendix 1 – Example PICAXE-08 Project

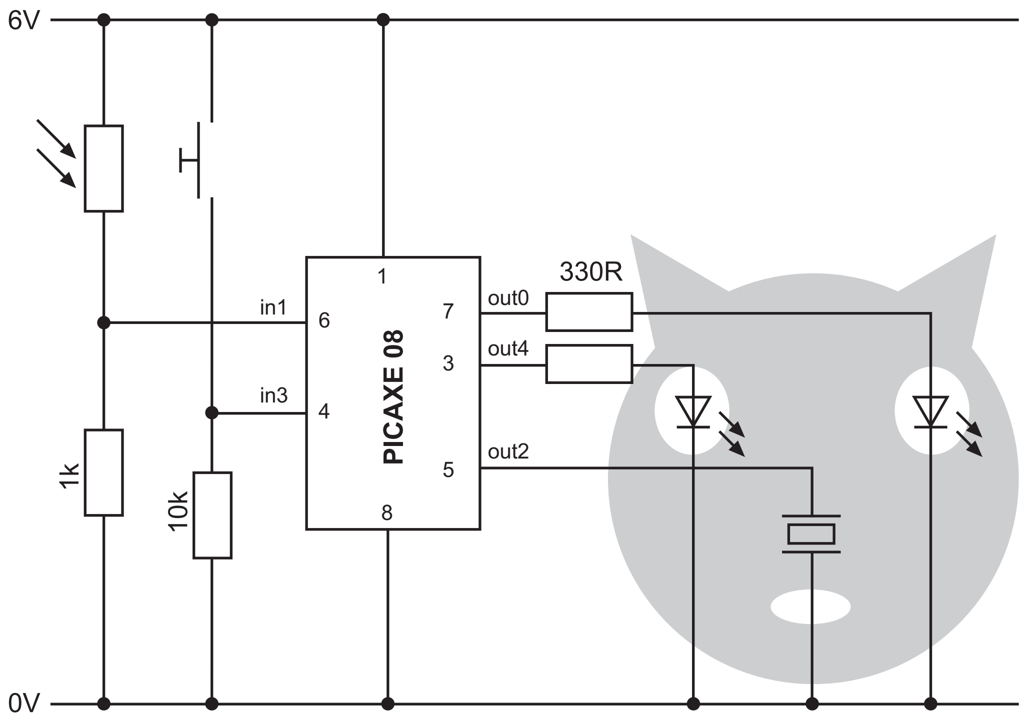
(Kit Available - part AXE101)

Design Brief
Design an electronic pet game.

Parts List

R1-2	- 10k resistor (brown black orange gold)
R3	- 22k resistor (red red orange gold)
R4-5	- 330R resistor (orange orange brown gold)
R6	- 1k resistor (brown black red gold)
LED1-2	- 5mm LEDs
PZ	- piezo sounder
LDR	- miniature light dependant resistor
CT1	- picaxe stereo socket
SW1	- miniature 6mm switch
PCB	- printed circuit board

Electronic Circuit



The circuit consists of two LED eyes that can light up, and a piezo sounder that generates sounds. By pressing the switch or altering the light level (measured by the LDR sensor) the actions of the pet can be altered.Note that for clarity this circuit does not show the serial download circuit (2 resistors and connector) which should also be included on the pcb (see page 4)

Microcontroller Input / Output Pins
Pin 0 (leg 7)– LED Eye (output)
Pin 1 (leg 6)– LDR light sensor (input)
Pin 2 (leg 5)– Piezo sounder (output)
Pin 3 (leg 4) – Push switch (input)
Pin 4 (leg 3)– LED Eye (output)

Program 1 Explanation
The program has a main loop which flashes the LED eyes on and off, and also checks the light sensor and the push switch. Whenthe push switch is pressed a sound is generated on the piezo sounder.
If the LDR light sensor is covered the pet will 'go to sleep' until the light level rises again.

Program 2 Explanation
This program is much more advanced. It has a main loop which fades the LED eyes on and off, and also checks the light sensor and the push switch. When the push switch is pressed a sound is generated on the piezo by use of the sound command. After three switch pushes (countedby a variable called pet), a tune is played.
If the LDR light sensor is covered the pet will 'go to sleep'.

The program makes use of a technique called pwm (pulse width modulation) to allow the eyes to 'fade' on and off , rather than just being full on or full off as would be achieved with the high or low commands. PWM works by switching the output on and off very quickly, quicker than the human eye can see. By varying the mark (on-time) to the space (off-time) the brightness of the LED can be altered.

-

```
' Cyper Pet Program 1
' For PICAXE-08

' LDR on pin 1
' Push switch on pin 4
' LEDs on pins 0 and 2
' Piezo sounder on pin 3

' define a variable to store the light value
symbol light = b1

' ***** main loop *****

' loop here flashing lights
' and checking switch and light sensor

main:

' LEDs full on and read light value
      high 4
      high 0
      readadc 1,light

' if light value low then go to bed
      if light < 80 then bed

' if switch pushed do sound
      if pin3 = 1 then purr

' do a delay
      pause 500

' LEDs full off and check sensor again
      low 4
      low 0
      readadc 1,light

' if light value low then go to bed
      if light < 80 then bed

' if switch pushed do sound
      if pin4 = 1 then purr

      pause 500
      goto main


' ***** make sound *****
purr:
      sound 2,(120,50,80,50,120,50)
      pause 200
      goto main


' ***** bed routine when in dark ****
' in dark so switch off LEDs and wait till
' light again

bed:
      low 0
      low 4
      readadc 1,light
      if light > 80 then main
      goto bed
```

```
' Cyper Pet Program 2
' For PICAXE-08

' LDR on pin 1
' Push switch on pin 4
' LEDs on pins 0 and 2
' Piezo sounder on pin 3

symbol bright = b0
symbol light = b1
symbol pet = b2
symbol mark = b3
symbol space = b4

' ***** start of program reset counters *****

' first reset the pet count to 0
' and the PWM off time to 15
main:
      let pet = 0
      let space = 15

' ***** main loop *****

' loop here flashing lights
' and checking switch and light sensor


loop:
' first make LED brighter in 8 steps
      for mark = 0 to 8
         gosub flash
       next mark

' LEDs full on and read light value
      high 4
      high 0
      readadc 1,light

' if light value low then goto bed
      if light < 80 then bed

' if switch pushed add petting value
      if pin3 = 1 then addpet

' do a delay
      pause 200


' LED gets dimmer
      for mark = 8 to 0 step -1
        gosub flash
      next mark

' LED full off and check sensors
      readadc 1,light

' if light value low then goto bed
      if light < 80 then bed

' if switch pushed add petting value
      if pin3 = 1 then addpet
      pause 250
      goto loop

' ***** end of main loop ***** (continued overleaf)
' ***** add one to pet value *****
addpet:
      pet = pet + 1
      sound 2,(120,50)
      pause 200

' check pet value to see if it is three yet
' if under 3 go back to loop
      if pet < 3 then loop

' pet value is 3 so play a tune then reset
      high 0
      high 4
      sound 2,(80,50,100,50,120,50,100,50,120,50)
      let pet = 0
      goto main


' ***** bed routine when in dark ****
' in dark so switch off LEDs and wait till
' light again

bed:
      low 0
      low 4
      readadc 1,light
      if light > 80 then main
      goto bed

' ***** PWM routine to dim LEDs *****
' sub procedure to output PWM on LEDs
flash:
      for b0 = 1 to 10
         high 0
         high 4
         pause mark
         low 0
         low 4
         pause space
      next b0
      return
```

The PICAXE-08 Commands

The list below is a full summary of the commands supported by the PICAXE-08 system. For syntax details and example programs see the General>Basic COMMANDS help.

DIGITAL OUTPUT

HIGH	Switch an output pin high (on).
LOW	Switch an output pin low (off).
TOGGLE	Toggle the state of an output pin.
OUTPUT	Set a pin as output.
INPUT	Set a pin as input.
REVERSE	Reverse the input/output state of an pin.
PULSOUT	Output a pulse on a pin for given time.

ANALOGUE OUTPUT

PWM	Provide a PWM output pulse.
SOUND	Output a sound.

DIGITAL INPUT

IF... THEN	Jump to new program line depending on input condition..
PULSIN	Measure the length of a pulse on an input pin.

ANALOGUE INPUT

READADC	Read analogue channel into a variable.

PROGRAM FLOW

FOR.. NEXT	Establish a FOR-NEXT loop
BRANCH	Jump to address specified by offset
GOTO	Jump to address
GOSUB	Jump to subroutine at address.
RETURN	Return from subroutine
IF.. THEN	Compare and conditionally jump

VARIABLE MANIPULATION

{LET}	Perform variable mathematics.
LOOKUP	Lookup data specified by offset and store in variable.
LOOKDOWN	Find target's match number (0-N) and store in variable
RANDOM	Generate a pseudo-random number

SERIAL I/O

SEROUT	Output serial data from output pin. Up to 2400 baud.
SERIN	Serial input data with qualifiers on input pin. Up to 2400 baud.

INTERNAL EEPROM ACCESS

EEPROM	Store data in data EEPROM before downloading BASIC program
READ	Read data EEPROM location into variable
WRITE	Write variable into data EEPROM location

POWER DOWN

NAP	Enter low power mode for short period (up to 2.3 sec)
SLEEP	Enter low power mode for period (up to 65535 sec)
END	Power down until reset

MISCELLANEOUS

PAUSE	Wait for up to 65535 milliseconds
WAIT	Wait for up to 65 seconds