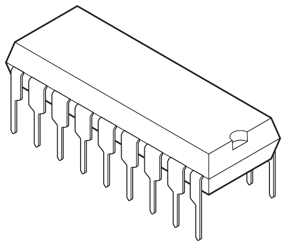
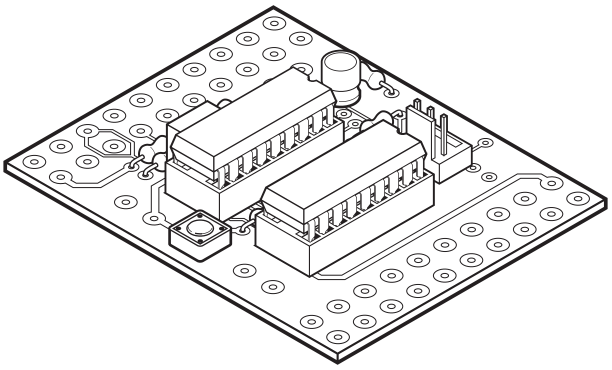
Rapid DATA SHEET

PICAXE

Order code	Manufacturer code	Description
13-0830	n/a	PICAXE-18 STARTER PACK

PICAXE-18 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-18 system is its simplicity. No programmer, eraser or complicated electronic system is required - the microcontroller is programmed (with a simple 'BASIC' program) 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 microcontroller provides 15 input/output pins - 8 digital outputs, 5 inputs and 2 serial interface pins.

- *low-cost, simple to construct circuit*
- *8 outputs and 5 inputs (3 inputs 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 with Crocodile Technology software.*
- *free manuals and online support forum*

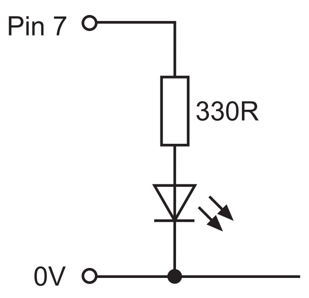
The comprehensive starter pack includes the following items:

- *standard interface board*
- *download cable*
- *CDROM containing software and manuals*
- *PICAXE-18 microcontroller chip*

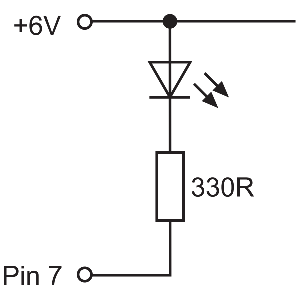
Downloading your first program

This first simple program can be used to test your system. It requires the connection of an LED (and 330R resistor) to output pin 7. If connecting the LED directly to a PICAXE chip on a home-made board, connect the LED between the output pin and 0V. When using the interface board (as supplied with the starter pack), connect the LED between V+ and the pin, as the output is buffered by the darlington driver chip on the interface board. (Make sure the LED is connected the correct way around!).

Direct connection to PIC



Connection to PIC via interface board



Note: The PICAXE microcontroller can be connected to a computer via the 18 pin project board OR via a simple self-made circuit board. Circuit connections are provided later in this 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 Options screen (this may automatically appear).
4. Click on the 'Mode' tab and select PICAXE-18
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 7
        pause 1000
        low 7
        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 7.
9. 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 7 should flash on and off every second.

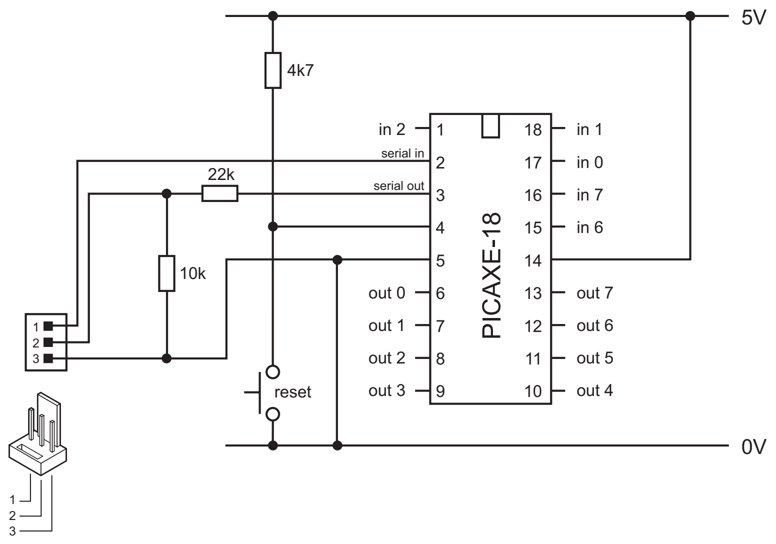
Trouble-shooting

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 selecting the correct COM port setting within the Programming Editor software.*
- *Not connecting, or trying to use a flat battery.*

The PICAXE-18 Circuit

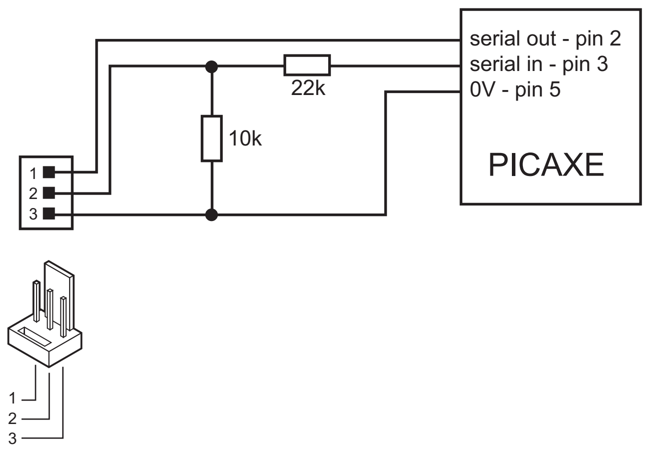
The basic PICAXE-18 circuit is shown below.



The 4k7 resistor is used to pull the PICAXE microcontrollers reset pin (pin 4) high. If desired, a reset switch can also be connected between the reset pin (pin 4) and 0V. When the switch is pushed the PICAXE microcontroller 'resets' to the first line in the program.

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.



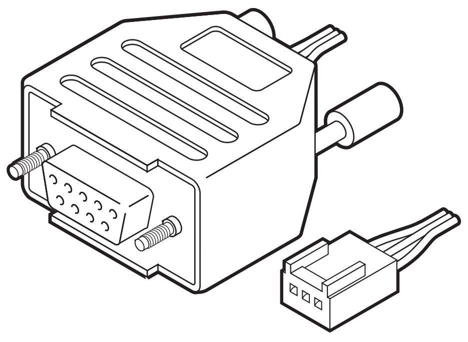
It is strongly recommended that this interfacing circuit is included on every PCB designed to be used with the PICAXE microcontroller. This enables the PICAXE microcontroller to be re-programmed without removing from the PCB.

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.

Making a Download Cable

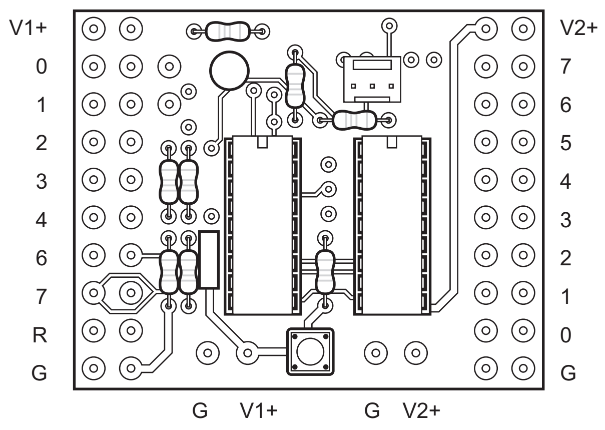
See the separate datasheet for information on how to make a download cable.



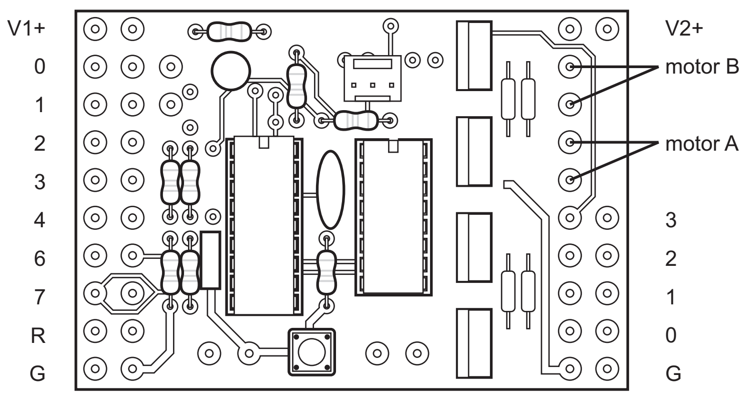
The PICAXE-18 Interfaces

The PICAXE-18 project boards are available in two configurations, standard and high-power. The input configuration on both boards is identical, providing up to 6 inputs.

The standard board uses a darlington driver IC to provide 8 digital (on/off) outputs. Each output is rated at 800mA.



The high power board uses 4 FETs to provide 4 high power digital outputs (rated at 1.5A each), and the option of a L293D motor driver IC to provide 2 reversible motor outputs, rated at 1A each. Please note that the L293D chip must be purchased separately.

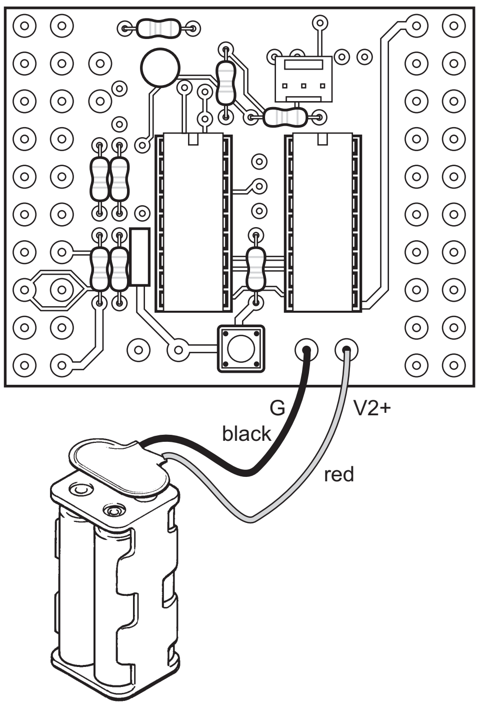


Both boards are supplied ready for use with the PICAXE-18 microcontroller. However, please note that you may have to solder the 3-pin header into position.

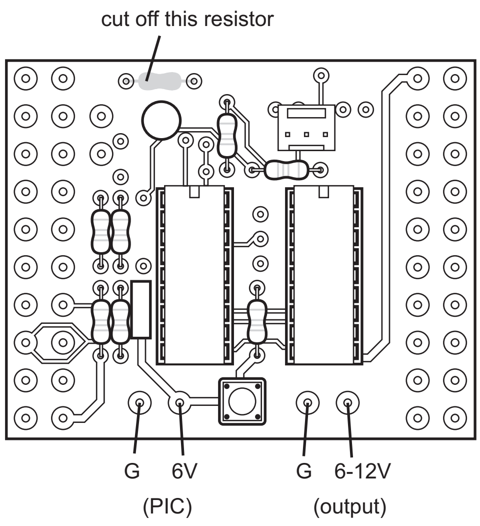
Project Boards Power Supply

The projects board require a single 3-6V power supply to operate. We recommend this is supplied via AA cell battery packs, connected to the V2+ terminal connections beside the reset switch. This pack will then power the microcontroller and the output devices.

The black wire is connected to the G (ground) connection and the red wire to the V2+ connection.

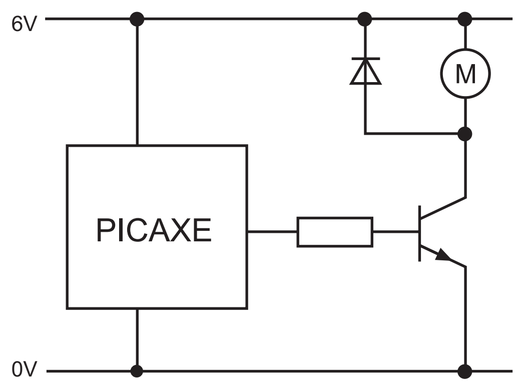


If a higher voltage (e.g. 12V) is required to drive the outputs, two separate power supplies may be used. In this case the second power supply only powers the output devices. The 3-6V power supply is connected to V1+ and the second 12V power supply is connected to V2+. When using two power supplies the resistor shown must be cut off the board to separate the two supplies.

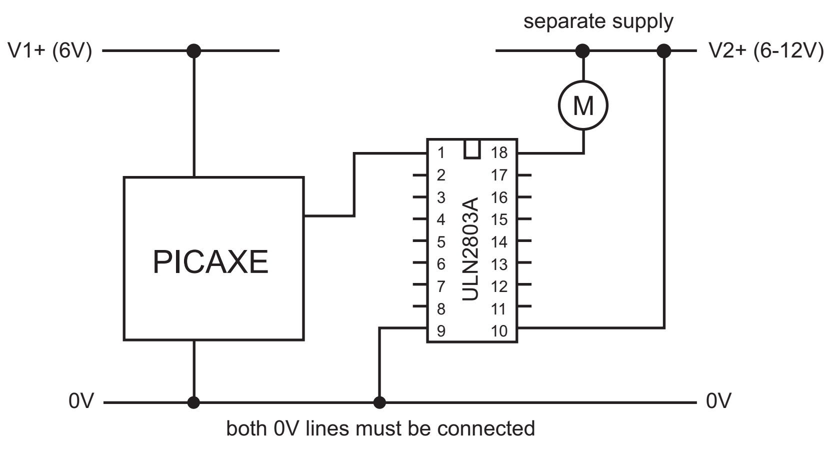


Digital Outputs

Each digital output is connected to a transistor as shown below.

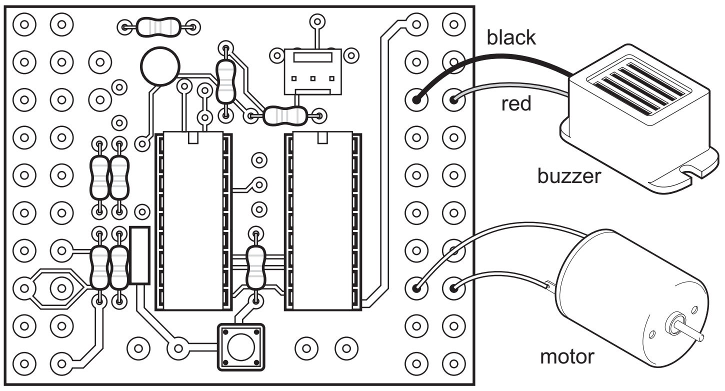


On the board the transistors are contained in a single ULN2803A darlington driver IC. The equivalent output circuit is shown below.

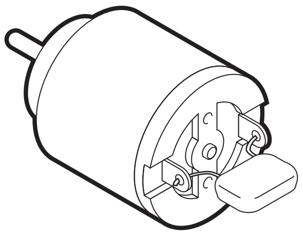


Output Devices

Output devices are connected between the pairs of holes on the pcb (pin and V2+) as shown below.

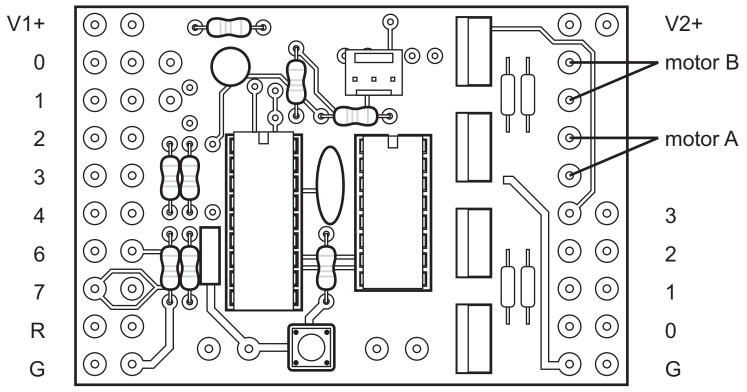


Note that motors should be suppressed by soldering a 220nF polyester capacitor across the motor terminals to prevent electrical noise affecting the circuit.



Controlling Motors on the High Power Board

To control motors an L293D motor driver IC must be fitted to the board as shown below.



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

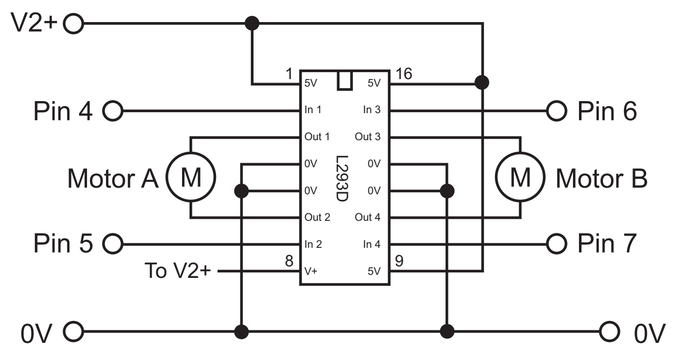
The chip is designed to run warm in use. This is normal.
The direction of rotation of the motors is defined as follows:

Pin 4	Pin 5	Motor A
off	off	off
on	off	forward
off	on	reverse
on	on	off

Pin 6	Pin 7	Motor B
off	off	off
on	off	forward
off	on	reverse
on	on	off

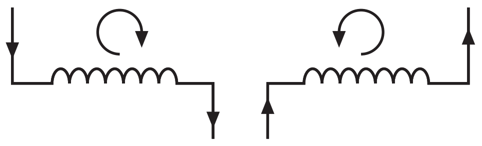
Motor Driver

An optional L293D motor driver IC (not supplied) may be added to the high power interface. This provides forward-backward control of two DC motors, controlled by outputs 4 to 7. Naturally, if only one motor is to be controlled then only two output lines are used.



Both inputs low	- motor halt
First output high, second output low	- motor forward
First output low, second output high	- motor reverse
Both inputs high	- motor halt

Changing the states of the input pins has the effect of altering the direction of current flow through the motor, as shown below.

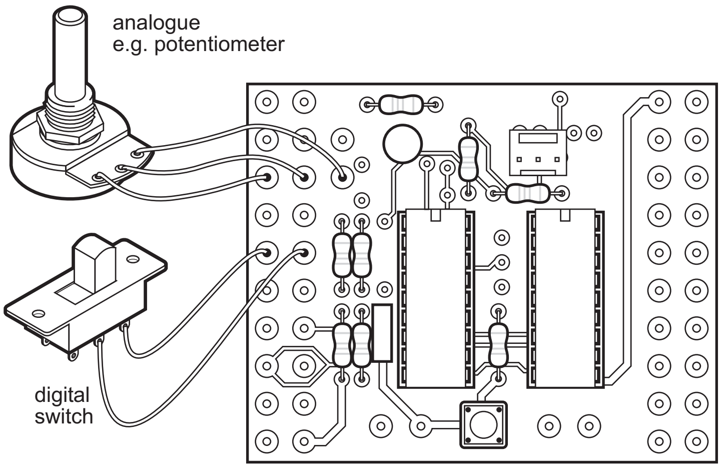


Note that the L293D will become warm with continuous use. A heatsink bonded onto the top of the chip will help keep it cool.

Motor A and B are connected to the interface via the pads provided. It is necessary to solder connectors or wires into these pads.

Input Devices

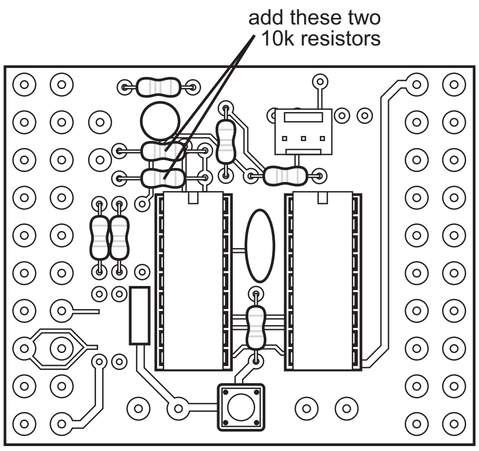
Analogue inputs are connected to input 0 (A) ands input 1 (B) as shown.
Digital inputs are connected between V1+ and the pin as shown below.



Note that inputs 3, 4 and 5 are not used with the PICAXE-18 system.

Reset Switch
A small reset switch is provided on the board. If desired an external reset switch can be connected in parallel between the R input and G (ground, 0V).

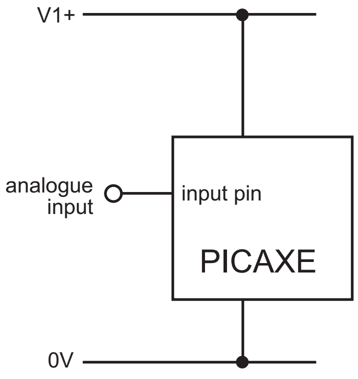
Note that when using inputs 0 and 1 as <u>digital</u> inputs you should add two 10k resistors to positions R7 and R8 as shown below.



Analogue Input Channels

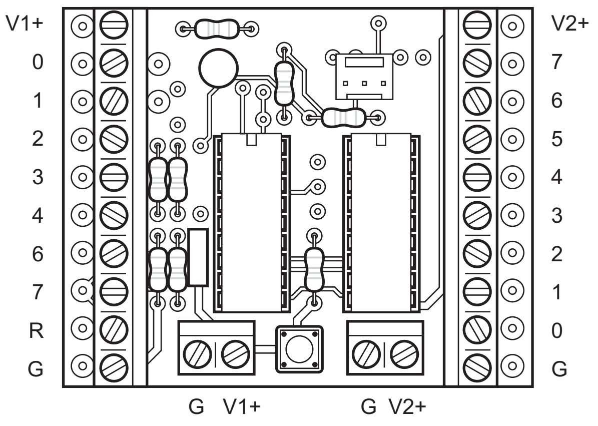
The analogue input channels are left floating (no connection). The equivalent input circuit is shown here.

Input pins 0, 1 and 2 can be used as digital or analogue inputs. 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.

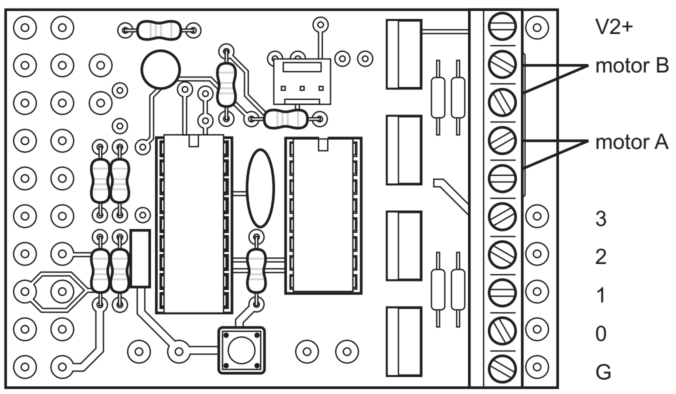


Connecting to the Project Boards

Inputs and outputs may be soldered via wires directly to the board. Alternately you may wish to purchase screw terminal blocks (5mm pitch) and solder these to the boards as shown below. This allows wires to be temporarily connected via the screw terminals.



Note that when using terminal blocks it is necessary to 'share' the V2+ output with all output pins and to 'share' the V1+ output with all inputs.



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 6 and 7, a switch to input 0, and a variable resistor to analogue channel 0.

For further details about each program see the Commands section.

Switching outputs on and off

The following program switches output 7 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 red LED on output pin 7 should flash on and off continuously.

```
main:                   ' make a label called 'main'
      high 7            ' switch output 7 on
      wait 1            ' wait 1 second
      low 7             ' switch output 7 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.

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 7 flash every time a switch on input pin 0 is pushed.

```
main:                               ' make a label called 'main'
      if pin0 = 1 then flash        ' jump if the input is on
      goto main                     ' else loop back around

flash:                              ' make a label called 'flash'
      high 7                        ' switch output 7 on
      pause 500                     ' wait 0.5 second
      low 7                         ' switch output 7 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 7 on for half a second before returning to the main loop.

Note carefully the spelling in the **if…then** line – **pin0** 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 = 7            ' rename output7 'red'
symbol green = 5          ' rename output5 '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 = 7                  ‘ rename output7 ‘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

The variables are commonly used to store ‘numbers’ as a program runs. This program lights up all the outputs in different combinations

```
main:                           ‘ make a label called ‘main’
      let b0 = b0 + 1           ‘ add one to b0
      let pins = b0             ‘ make outputs = value of b0
      pause 100                 ‘ wait 0.1 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.

For further details about the mathematical capabilities of the PICAXE microcontroller see the Commands section.

Reading Analogue Input Channels

The value of an analogue input can be easily copied into a variable by use of the 'readadc' command. The variable value (0 to 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 0,b0               ' read channel 0 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 7                      ' else switch off 7
      low 6                      ' and switch off 6
      goto main                  ' jump back to the start

red:                             ' make a label
      high 7                     ' switch on 7
      low 6                      ' switch off 6
      goto main                  ' jump back to start

green:                           ' make a label
      high 6                     ' switch on 6
      low 7                      ' switch off 7
      goto main                  ' jump back to start
```

Note that the PICAXE microcontroller has 3 analogue channels labeled 0 to 2. 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 0,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.

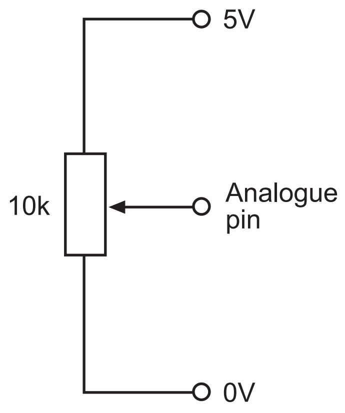
Using the Low-Resolution Analogue Input

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

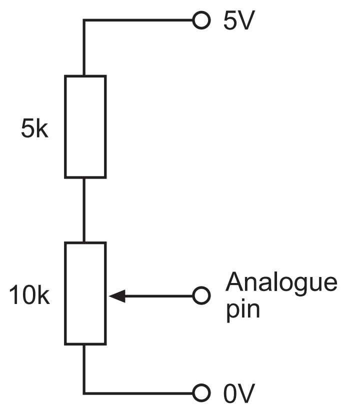
Advanced technical information on using the low resolution ADC

The low-resolution analogue pin within the PICAXE-18 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).

The PICAXE-18 Commands

The list below is a full summary of the commands supported by the PICAXE-18 system. For syntax details and example programs see the Commands section.

DIGITAL OUTPUT	
HIGH	Switch an output pin high (on).
LOW	Switch an output pin low (off).
TOGGLE	Toggle the state of an output 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
PEEK	Read from additional RAM registers
POKE	Write to additional RAM registers
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
SYMBOL	Rename variables.
DEBUG	Transmit variables to PC for debugging purposes.