PICAXE FAQ

Where can I purchase PICAXE microcontrollers?

All microcontrollers can be purchased from within the PICAXE section of the online store at www.tech-supplies.co.uk or from our distributors (see www.picaxe.co.uk)

There appears to be two PICAXE download cables - which should I use?

The standard PICAXE cable (part AXE026) ends with a stereo style 3.5mm plug. If making your own board we recommend this stereo cable cheaper as it is cheaper, better quality, and our sample PCB files use this connector (part CON039). The original PICAXE-28 cable (part AXE025) ended with a 3 pin in-line connector, but this cable is no longer used on any of our project boards or sample pcbs.

My laptop only has a USB connector - is there a USB version of the PICAXE?

To make a USB version of the PICAXE we would have to use a more expensive USB enabled microcontroller, which would increase the unit cost of each PICAXE chip significantly. To avoid this increased chip cost we instead supply a 'USB to Serial Adapter' (part USB010) which enables USB users to use the standard serial download cable. The one-off cost of purchasing this adapter is cheaper (in the long term) than the cost of buying multiple expensive chips.

I've built a second pcb (without the download circuit) and the PICAXE program will not run!

If you program a PICAXE chip in a different board, and then move the chip to a board without the download circuit, you must ensure that the 'serial in' pin is tied to ground (0V) on the second board for reliable operation.

I've bought some blank PICs and they don't work in the PICAXE system!

The PICAXE microcontroller is not a blank PICmicro! It is a microcontroller that has been pre-programmed with a 'bootstrap' program that enables the download via the direct cable link (the bootstrap program tells the microcontroller how to interpret the direct cable programming commands). Therefore you must buy 'PICAXE' microcontrollers, rather than blank microcontrollers, to use with the PICAXE system. However we sell PICAXE microcontrollers at approx. the same price as blank devices, so there is very little price difference for the end user, particularly if you purchase the multi-packs.

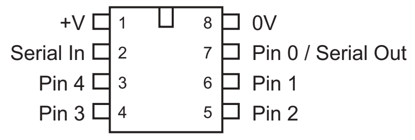
I've programmed a PICAXE microcontroller using a conventional programmer and it will now not work in the PICAXE system!

You have overwritten, and hence deleted, the PICAXE bootstrap program (see above). The microcontroller can no longer be used as a PICAXE microcontroller, but you can naturally continue using it with your conventional programmer.

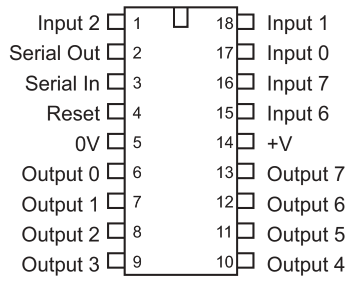
Can you reprogram microcontrollers (that I have accidentally erased) with the bootstrap program?

No. We do not accept microcontrollers from unknown sources due to the correct storage/handling procedures required by these devices. We use gang programmers costing several thousand pounds to program the bootstrap code into the blank microcontrollers, and so must protect this expensive equipment from damage. It is also likely that if we did offer this service the handling cost would end up more expensive than new PICAXE microcontrollers anyway!

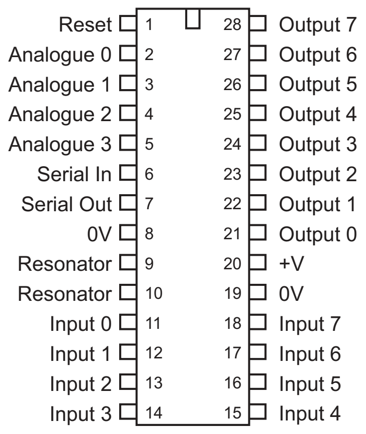
PICAXE-08



PICAXE-18



PICAXE-28



www.picaxe.co.uk
(c) Rev-Ed Ltd 2002

Can you supply the bootstrap program so that I can make my own PICAXE microcontrollers?

No. The small royalty made on each PICAXE chip sold is the only financial benefit to our company to support the PICAXE system - the software is free and the cables/development kits are sold at very low cost. Therefore we do not allow anyone else to manufacture PICAXE microcontrollers.

Can I see the assembler code that is downloaded into the PICAXE?

If you own a Revolution Serial PIC Programmer, you can convert PICAXE BASIC programs into assembler code, to program blank PICs or to just learn how assembler code works by 'disassembly'. However some of the more complex commands (e.g. serin) are not supported, and the assembler code program generated is optimised for sequential learning (not optimised for compactness as with the PICAXE system) and so the code is not 100% identical to that downloaded to the PICAXE.

Can you alter the input/output pin arrangement of the PICAXE microcontroller?

The PICAXE-08 has 5 pins that can be configured as desired. The PICAXE-28X/40X pin configuration can be altered. The PICAXE-18/18A/18X and PICAXE-28/28A input/output pin arrangements are fixed and cannot be altered.

How long a program can I download into the PICAXE microcontroller?

This varies on the commands used, as not all commands use the same amount of memory. As a general rule you can download about 40 lines of code into the PICAXE-08/18, 80 lines into the PICAXE-18A/28/28A and 600 into the PICAXE-18X/28X/40X. However some commands, such as sound and serout use more memory and so will reduce this count. In our experience most educational programs that are too long to download are generally badly composed, and can be greatly reduced in size by use of sub-procedures etc.

Do I need to erase the device?
How do I stop a program in the PICAXE microcontroller running?

Each download automatically overwrites the whole of the previous program. There is generally no need to erase the memory at any point. However if you want to stop a program running you can select the ‘Clear Hardware Memory’ menu to download an ‘empty’ program into the PICAXE memory.

How often can the PICAXE microcontroller be reprogrammed?

The manufacturer datasheets state the PICAXE-08/18/18A, and X parts can be reprogrammed at least 100,000 times. The -28 and -28A devices can be reprogrammed at least 1000 times. In practice this number may be much greater.

How vulnerable to damage are the microcontrollers?

The microcontrollers have a high level of static protection built into each pin and so handling them without any personal static protection in an educational environment is perfectly acceptable.

Can I control servos using the PICAXE?

Yes, the A and X parts have a ‘servo’ command that allows control of up to 8 servos (one on each output).

Can I control an LCD display?

Yes, the PICAXE supports serial LCD modules (like the Serial LCD/Clock Module AXE033) via the serout command. Note that the AXE033 module can also be pre-programmed with up to 8 messages to reduce the memory usage of the PICAXE microcontroller.

How fast does the PICAXE operate?

The PICAXE-08/18 microcontrollers have an internal 4MHz resonator, and the PICAXE-28 uses an external 4MHz ceramic resonator. This means the microcontroller processes 1 million assembler commands a second, which equates to roughly about 10,000 BASIC commands per second.
The M and X parts can be overclocked to 8 or 16MHz (multiplies speed by x2 or x4).

Does the PICAXE support interrupts?

Yes. The A. M and X parts support a polled interrupt on the input port. Use the 'setint' command to setup the desired interrupt port setting.

How do I create time delays longer than 65 seconds?

The best way of creating long delays is to do minute delays with a loop, e.g. to wait an hour (60 minutes)

```
for b2 = 1 to 60  'start a for..next loop
pause 60000       'wait 1 minute
next b2           'next loop
```

The PICAXE microcontroller works at a nominal 4MHz, but due to device manufacturing tolerances there is likely to be a drift of a few seconds over long time periods (e.g. a day). Note that the Serial LCD/Clock module (AXE033) has a precision clock and 'alarm clock' function that can be used to trigger the PICAXE at predefined interval or at certain time/dates with much greater precision. The X parts can also be linked to the i2c DS13097 real time clock.

My program is too long! What can I do?

Tips for reducing program length (see BASIC Commands help file for more details):
1) Use 'let pins =' instead of multiple high/low commands
2) Use sub-procedures for repeated code
3) Try to reduce the use of sound and serout commands, which use a lot of memory
4) If using an LCD, store the messages in the AXE033 Serial LCD Module, rather than in the program
5) Use eeprom and read commands to store messages in data memory (see next page)
6) Restructure your program to reduce the number of 'goto' commands
7) Use a PICAXE chip with the largest memory (X parts)
You can use the 'PICAXE>Check Syntax' menu to test the length of your program without a download.

Do symbols increase the program length?

No, all symbols are converted back to 'numbers' by the computer software prior to download and so have no affect on program length. You can use as many symbol commands as you wish.

What notes are generated by the sound command?

The sound command generates different 'beep' sounds for the values 1-127.
The tune and play commands on the PICAXE-08M are specifically designed to play tunes. See the PICAXE-08M music datasheet for more details.

I need more outputs - what can I do?

Use the PICAXE-28X or 40X which can have up to 16 outputs. Or connect a single output (e.g. output7) from a first PICAXE chip to input0 of a second PICAXE-18 chip. Program the second PICAXE-18 chip with this simple program:

```
main: serin 0,N2400,b1
      let pins = b1
      goto main
```

The eight outputs of the second chip can now be controlled with a serout 7,N2400,(b2) command by the first chip, where b2 contains the 'pins' value (0 to 255) desired on the second chip. This gives you a total of 15 useable outputs.

I need more inputs - what can I do?

Use a PICAXE-28X or 40X, which can be configured to have a large number of inputs. Remember that analogue inputs can also be used as digital inputs if required, just see if the 'readadc' value is greater or less than 100. In many applications switches can also be connected in parallel on a single input pin.

How do I test more than one input at once?

Use the following command to test both inputs `if pin0 = 1 and pin1 = 1 then...`
either inputs `if pin0 = 1 or pin1 = 1 then...`

What is the data memory?

The data memory is memory within the chip that can be ‘preloaded’ with data (e.g. messages for an LCD) by use of the eeprom command. It can then be accessed as the program runs by the ‘read’ command, or modified by the ‘write’ command.

On the PICAXE-08 and 18 the data memory is shared with the downloaded program, and so a longer program means that less data memory is available.. On all other chips the data is separate from the program, and so generally of more use. Correct use of the data memory can greatly reduce the length of programs when using commands such as serout.

The PICAXE-18A /18X have 256 bytes of data memory.
The PICAXE-28A / 28A have 64 bytes of data memory and a further 256 bytes (using readmem/writemem commands).
The PICAXE-28X / 40X have 128 bytes of data memory.
Therefore the addresses 0-63 are valid on all parts, and higher addresses are available on the 18A and X parts.

The following example demonstrates a program that stores two LCD messages in the standard data memory and then retrieves them for display on a Serial LCD module (part AXE033).

```
eeprom 0,(“This is message1”)
eeprom 16,(“This is message2”)

main: serout 7,n2400,(254,128)      ‘ move to line 1
            let b1 = 0              ‘ get message 1
            gosub display
            serout 7,n2400,(254,196)      ‘ move to line 2
            let b1 = 16             ‘ get message 2
            gosub display
            end

display:    b2 = b1 + 16            ‘ generate end address
            for b3 = b1 to b2       ‘ start loop
              read b3,b4            ‘ read letter into variable b4
              serout 7,n2400,(b4)   ‘ output letter
            next b3                 ‘ loop
            return
```

How do I use interrupts?

The interrupt is set by the setint command. The interrupt can be set to occur on any input pattern on the input pins by means of the port and mask bytes. For instance, to interrupt when input0 goes high, the setting is %00000001,%00000001. The first port byte indicates that input0 must be high, the second mask byte sets the microcontroller to only use input0 (ie ignore the condition of the other inputs).

In the following program the loop continually flashes output 4 on and off. As soon as input0 is activated the interrupt will occur, and therefore output3 will switch on for 1 second. Note that the setint command is re-activated in the interrupt sub-procedure so that the interrupt will re-occur if the input is still high.

```
main: setint %00000001,%00000001
loop: high 4                  interrupt:  high 3
      pause 1000                          pause 1000
      low 4                               low 3
      pause 1000                          setint %00000001,%00000001
      goto loop                           return
```