TECHNICAL PICAXE FAQ

This datasheet provides advanced technical data for users who wish to understand more advanced technical data about the PICAXE microcontrollers. This information is not required for normal PICAXE use.

These notes presume the user is familiar with PIC microcontrollers, their configuration fuse settings and programming in assembler code.

What is a PICAXE microcontroller?

A PICAXE microcontroller is a Microchip PIC microcontroller that has been pre-programmed with the PICAXE bootstrap code. The bootstrap code enables the microcontroller to be reprogrammed without the need for an (expensive) conventional programmer, making the whole download system a very low-cost simple serial cable!

The bootstrap code also contains common routines (such as how to generate a pause delay or a sound output), so that each download does not have to waste time downloading this commonly required data. This makes the download time much quicker.

Why use the PICAXE instead of assembler / C?

The PICAXE uses a simple BASIC langauge (or flowcharts) that younger students can start generating programs with within an hour of first use. It is much easier to learn and debug than either C or assembler code.

The second advantage is the direct cable download method. The software is free and so the only cost per computer is the download cable (£3). This enables students to buy their own cable and for schools to equip **every** single computer with a download cable. Other systems that require an expensive programmer are generally too expensive to implement in this way.

Finally as the PICAXE chip never leaves the board, all leg damage (as can occur when the chip is moved back and forth from a programmer) is eliminated.

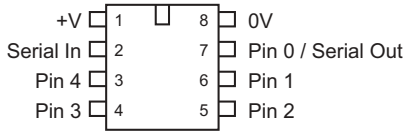
How is the program stored within the microcontroller?

The program is stored in either data or program memory depending on the microcontroller type. The following table shows how program, read/write/eeprom data and readmem/writemem data is stored.

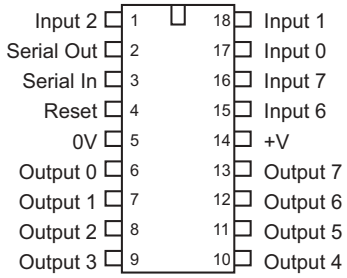
	Program	Read/Write	Readmem/Writemem
PICAXE-08	Data	Data	N/A
PICAXE-08M	Data	Data	N/A
PICAXE-18	Data	Data	N/A
PICAXE-18A	Program	Data (256)	N/A
PICAXE-18X	Program	Data (256)	N/A (use i2c)
PICAXE-28A	Program	Data (64)	Program (256)
PICAXE-28X	Program	Data (128)	N/A (use i2c)
PICAXE-40X	Program	Data (128)	N/A (use i2c)

The program and read/write memory is overwritten with every download. Use the EEPROM command to preload data (within the program) for the read/write commands. The readmem/writemem memory is not changed during a download.

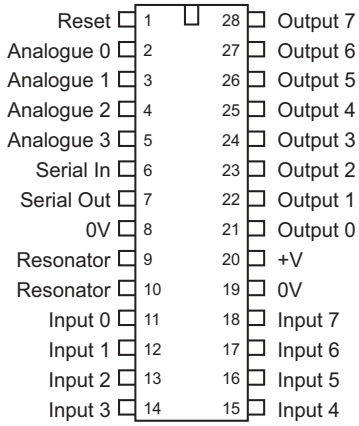
PICAXE-08



PICAXE-18



PICAXE-28



www.picaxe.co.uk

How many times can the microcontroller be reprogrammed?

PICAXE chips can be reprogrammed at least 100,000 times (apart from the PICAXE-28/28A - can be reprogrammed a minimum of 1,000 times). Note these are minimum values and the actual values may be much greater.

How is a download started?

When the computer starts a download an interrupt is generated on the serial input pin on the PICAXE. This interrupts the main program and puts the PICAXE into a state for a new download to be received. Therefore you must ensure that the 'serial in' pin is tied to ground (0V) via the 22k/10k resistors on ALL project boards for reliable operation of the microcontroller (to prevent unwanted 'floating pin' interrupt signals).

What are the electrical characteristics of the PICAXE (e.g. operating voltage range etc.)?

The electrical characteristics of the PICAXE microcontroller is dependant upon the base PIC microcontroller that is programmed with the PICAXE bootstrap code to create the PICAXE microcontroller. Therefore see the Microchip datasheet (from www.microchip.com) for the appropriate microcontroller characteristics. The lowest operating voltage from these datasheets is shown below (Note this is the 'operating voltage' only. You may require a higher voltage (minimum 4.5V recommended) whilst doing the actual serial download from the computer to ensure accurate memory programming of the chip).

PICAXE-08	is a pre-programmed	PIC12F629-I/P	minimum voltage	2.0V
PICAXE-08M		PIC12F683-I/P		2.0V
PICAXE-18		PIC16F627 (A)-04/P		3.0V (2.0V)
PICAXE-18A		PIC16F819-I/P		2.0V
PICAXE-18X		PIC16F88-I/P		2.0V
PICAXE-28A		PIC16F872-I/SP		3.0V
PICAXE-28X		PIC16F873A-I/SP		2.0V
PICAXE-40X		PIC16F874A-I/P		2.0V

Does the PICAXE set the watchdog timer fuse?

Yes, the watchdog timer is set and used within a number of commands such as sleep and nap. The user cannot alter it's settings.

Does the PICAXE set the power-up timer fuse?

Yes.

Does the PICAXE set the brown-out fuse?

Yes for the PICAXE-08, no for other parts. An unfortunate side effect of the brown-out fuse is that it restricts the lowest operating voltage of the micro-controller to about 4.2V (2.0V on PIC12F629). As many users wish to use 3V battery packs, the brown-out fuse is not set on the PIC microcontrollers with a 4.2V brown-out.

How does the PICAXE do ADC (analogue-to-digital) conversions?

The PICAXE-08 and PICAXE-18 use the internal comparator to do a low-resolution ADC step comparison, providing 16 discrete analogue values. The other PICAXE microcontrollers use the internal ADC to do a full 256 step (8 bit) conversion. Although the microcontrollers are technically capable of 10 bit conversions, this is converted by the PICAXE A parts into byte (8 bit) values for ease of use via the byte (b1 etc.) variables, which makes the maths easier for students. This gives a resolution of about 0.02V (at 5V supply) which is adequate for almost all educational projects. The M and X parts have a separate 10 bit adc read option (1024 steps), via the readadc10 command.

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 mix assembler in with the BASIC code?

No. The program and bootstrap code cannot be 'mixed' with assembler code, this is not good programming practice. However you can achieve the same goal by converting your BASIC into assembler code using the automatic conversion feature, and then editing the converted assembler code program. (see below).

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 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 and 40X can also be altered to give more inputs or outputs. The PICAXE-18 and PICAXE-28 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 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.

There is no fixed 'byte' formula as to memory usage e.g. pause 5, pause 50 and pause 500 will all take different amounts of memory space! To calculate memory usage use the 'Check Syntax' option from the PICAXE menu. This will report the amount of memory used.

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.

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.

Why is an 'empty' program still 3 bytes long?

Each downloaded program contains some configuration data, and an 'end' command is always added automatically to the end of each downloaded program. Therefore an 'empty' program on screen will not generate a zero byte program.

How vulnerable to damage are the microcontrollers?

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

Can I use i2c EEPROMs with the PICAXE?

The X parts support i2c parts via the i2cslave, readi2c and writei2c commands.

Can the PICAXE count pulses?

The X parts support the count command which can count the number of pulses in a defined period. All parts support the pulsin command to measure the length of a pulse.

Can I control servos using the PICAXE?
Can I do PWM control of a motor using the PICAXE?

The X parts have a dedicated pwmout command which acts on two of the pins (one pin on 18X) for full pwm control. The A nd X parts have a 'servo' command that allows control of up to 8 servos (one on each output). The servo command users the internal timer and an interrupt, so that the pulses are maintained 'in the background' all the time that the PICAXE is running the main program.
The servo command produces a pulse of length 0.01ms to 2.55 ms approximately every 20ms. Therefore it can also be used as a simple background PWM output with PWM mark:space ratios between 1:2000 and 1:8 (approx).

How fast does the PICAXE operate?
Can I overclock the PICAXE?

The PICAXE-08/18/18A/18X microcontrollers have an internal 4MHz resonator, and the PICAXE-28/40 family use 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. Different commands take different times to execute depending on how complex their 'assembler code' is.
The M and X parts can be overclocked to 8 or 16MHz (see X-parts datasheet for restrictions).

Why does the PICAXE only support up to 2400 baud rate on serout/serin commands?
Can I send and receive serial data via the download cable?

The maximum baud rates were originally selected for reliable operation with microcontrollers with internal resonator. The early internal resonators were not as accurate as an external device, and a slower baud rate ensures reliable operation. The X parts support a higher 4800 baud rate, and use of a faster resonator will increase the baud rates.
The PICAXE-08 can send data via the download cable via a 'serout 0' command. The X parts can send data via a 'sertxd' command. The 'serial in' pin is reserved for program downloads and cannot be used for receiving general data.

Does the PICAXE support interrupts?

The PICAXE uses the internal microcontroller interrupts for some of it's BASIC commands (e.g. servo). Therefore the internal interrupts are not available for general use. However the A, M and X parts all support a single 'polled' interrupt on the input port. Use the 'setint' BASIC command to setup the desired interrupt port setting to enable the polled interrupt. The polled interrupt scans the input port between every BASIC command (and constantly during pause commands), and so activates very quickly.

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 scan 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
```