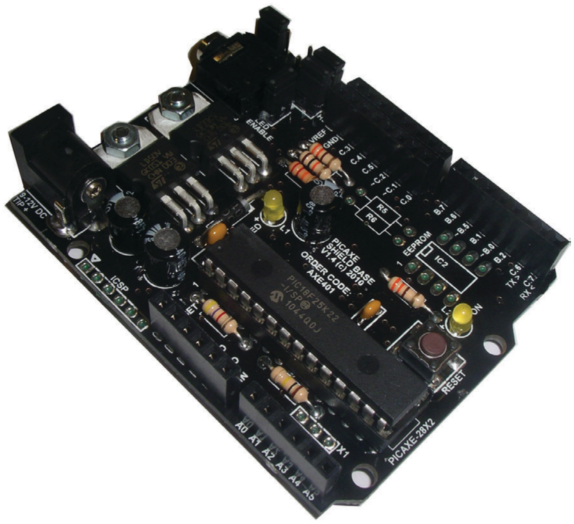
PICAXE-28X2 Shield Base (AXE401)

1.0 Introduction



Thank you for purchasing this PICAXE shield base. This datasheet is designed to give a brief introduction to how the shield base is assembled, used and configured.

For general tutorials and details on how to use the PICAXE programming language, please see the PICAXE website and PICAXE manuals **www.picaxe.co.uk** The completely free programming software may also be downloaded from this website. For individual project advice, and examples of how to connect your shield base to many different shields and electronic components, please consider joining the friendly online PICAXE community at **www.picaxeforum.co.uk**

A 'wiki' of how to use the PICAXE shield base with a wide range of shields is currently being developed at the PICAXE website.

1.1 Overview

The PICAXE shield base is an open-source hardware 'arduino inspired' controller board, designed to enable PICAXE use of the multiple different 'shields' that are now available. The shield base has been very carefully designed to be compatible with the vast majority of existing shields. Each PICAXE shield base is also provided with a free of charge AXE405 prototyping shield PCB for experimentation.
The PCB designs are released as 'open-source' and 'not for profit' designs.

1.2 Key Features

Firmware	PICAXE-28X2
Microcontroller	Microchip PIC18F25K22-I/SP
Programming Language	BASIC or flowcharts
Physical Size & Header Position	Standard Shield Size/Position
Open Source Hardware	Yes
Not for profit PCB	Yes
Development Software Cost	Free
Support for Windows	Yes
Support for Mac	Yes
Support for Linux	Yes
Downloads via USB port	Yes
Microcontroller System Voltage	5V or 3V (jumper selectable)
Recommended Supply Voltage	9-12V
5V Low drop out regulator	500mA
3V Low drop out regulator	500mA
Disconnectable LED on S.13	Yes
Stripboard compatible headers	Yes
On board i2c EEPROM socket	Yes
Separate programming and i/o pins	Yes
TX / RX select jumper headers	Yes
Through hole for easy self assembly	Yes
Program Slots	4 x 4k
User RAM bytes	1296 (256 + 1024 + 16)
Clock Speed	32KHz - 64MHz
Non-volatile EEPROM bytes	256
Run extra programs from i2c	Yes

Contents

Please download the latest full assembly instructions and datasheet from this web link:

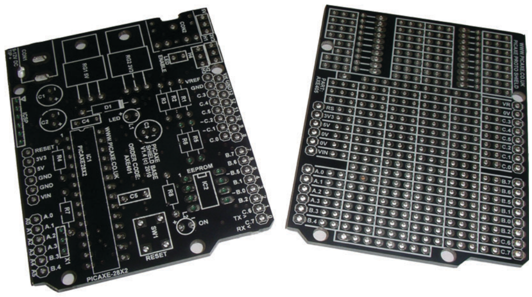
www.rev-ed.co.uk/docs/axe401.pdf

1.3 What does open source hardware mean?

It means we publish the shield base circuit schematic and PCB layout free of charge on our website. You can modify the PCB and build your own new PCB derived from it if you want to. You can also even download completely free PCB software (e.g. Eagle or DesignSpark) to do these modifications and generate new Gerber files! However as this product is also a 'not for profit' PCB, it is highly unlikely that a prototyping company could produce a PCB cheaper than we will sell you one!

1.4 What does 'not for profit' PCB mean?

As the name suggests we manufacture, stock and sell the unpopulated shield base PCBs at the lowest cost possible. A nominal handling charge is added to cover warehouse processing costs, but this is deliberately kept to a minimum. We can afford to do this as the shield base is just one item within a large range of PICAXE development project boards and chips, so by buying this permanently low priced 'promotional' shield PCB we hope you will consider purchasing further PICAXE microcontrollers and project boards in the future.



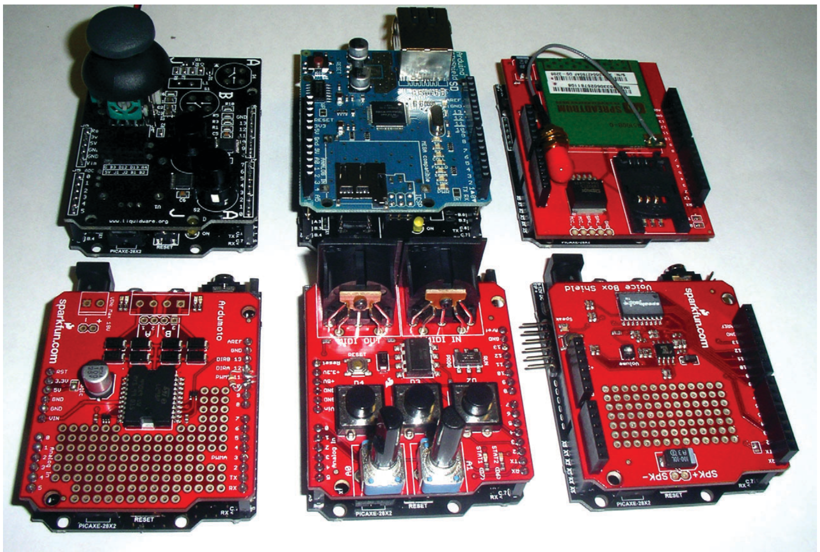
We will post the PCBs anywhere in the world by Airmail at a nominal postage cost.

1.5 PICAXE-28X2 microcontroller

The shield base is designed to work with the PICAXE-28X2 microcontroller, which is based upon the Microchip PIC18F25K22-I/SP microcontroller. Other PICAXE chips are also available in 8, 14, 18, 20 and 40 pin formats, but these other sizes will not fit on the shield base as they are a physically different size.

1.6 Do I have to use the shield base to use the PICAXE system?

No, not at all! In fact the only normal reason to use a shield base is to integrate with an existing third party shield (e.g. MP3 player shield or Ethernet controller shield). If you want to build a project yourself from scratch it may well be cheaper and easier to use one of our many PICAXE project or proto boards instead, particularly if you could use a physically smaller, lower cost 8 or 14 pin chip such as the PICAXE-14M2 instead. Or just buy a PICAXE chip by itself and build your own breadboard/stripboard/PCB circuit. The choice is yours!



2.0 PICAXE-28X2 shield base input/output pins

The PICAXE system provides great flexibility over individual pin use, and most pins can be configured to be used as a digital input or output, ADC analogue input or touch sensor input. Therefore the PICAXE system does not need separate 'analogue' and 'digital' headers, as most pins can be either. Some pins also have additional special hardware functionality, such as PWM or serial or SPI interfacing.

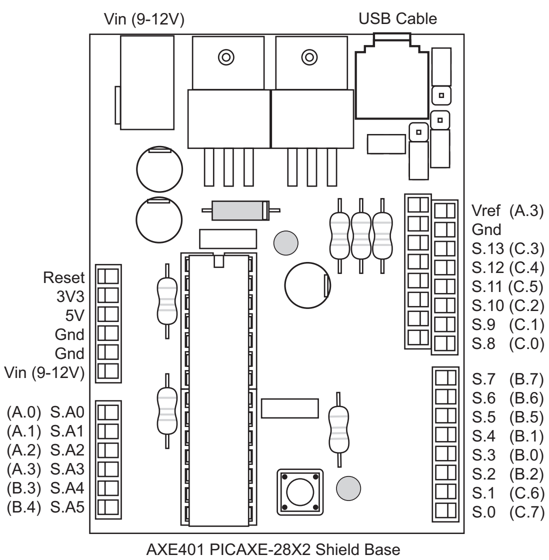
The PICAXE shield base has been very carefully designed to be compatible with almost all existing shields. Other PIC microcontroller orientated shield bases we have seen seem to have been designed without much thought for shield compatibility, as they simply wire the three i/o headers in sequence to the PIC microcontroller ports A, B and C. Although this may be the initial obvious route, it actually means that a large number of shields are then not compatible because, for instance, they require a hardware PWM or SPI special function which is now allocated to a completely different pin to the normal shield layout!

Therefore the PICAXE shield base has been very carefully laid out to ensure the special functions match the existing shield normal layout wherever possible. This gives support for a much larger range of shields (e.g. PWM, UART and SPI functions are on the same pins as expected).

To ensure a logical naming system can still be used, the PICAXE-28X2 compiler accepts the normal shield position numbers in the formal S.X, where X is the shield pin name (the original PICAXE pin name may also still be used if preferred).

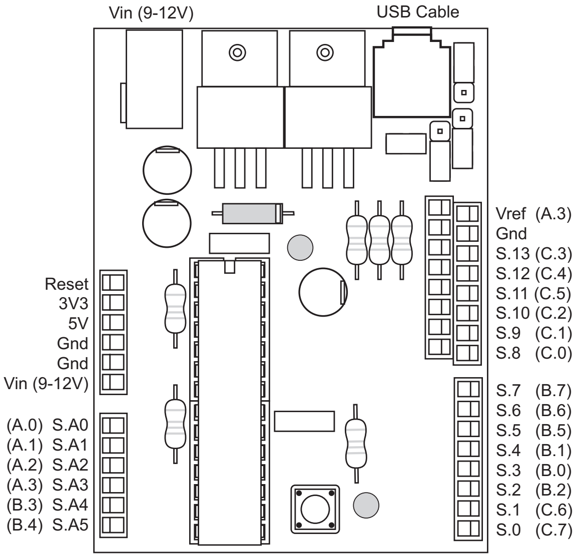
e.g. the shield header pins are named
S.A0 to S.A5, S.0 to S.13
the shield input pin variables are named
pinS.A0 to pinS.A5, pinS.0 to pinS.13

So users can select to use either the normal PICAXE pin name (e.g. 'high B.4') or the shield position nickname (e.g. 'high S.A5') in their program. Both act exactly the same way within the compiler. Naturally you can also rename the pins to easier to remember names such as 'high red_LED'.



AXE401 PICAXE-28X2 Shield Base

2.1 PICAXE Shield Pinout



AXE401 PICAXE-28X2 Shield Base

Shield Header	Shield Nickname	Primary Pin Function	Advanced Pin Function	PICAXE Pin Name	PICAXE ADC
RESET		Reset		Reset	
3V3		3.3V Supply Out		V+	
5V		5V Supply Out	5V Supply In	V+	
GND		0V	0V Supply In	0V	
GND		0V		0V	
VIN		Supply In (9-12V DC)			
A0	S.A0	In / Out / ADC / Touch	Comp1-	A.0	0
A1	S.A1	In / Out / ADC / Touch	Comp2-	A.1	1
A2	S.A2	In / Out / ADC / Touch	Comp2+ / DAC	A.2	2
A3	S.A3	In / Out / ADC / Touch	Comp1+ / Vref	A.3	3
A4	S.A4	In / Out / ADC / Touch		B.3	9
A5	S.A5	In / Out / ADC / Touch	hpwm D	B.4	11
0	S.0	In / Out / ADC / Touch	hserin / kb data	C.7	19
1	S.1	In / Out / ADC / Touch	hserout / kb clk	C.6	18
2S	.2	In / Out / ADC / Touch	hpwm B / hint 2	B.2	8
3S	.3	In / Out / ADC / Touch	pwm / hint0	B.0	12
4S	.4	In / Out / ADC / Touch	hpwm C / hint 1	B.1	10
5S	.5	In / Out / ADC / Touch	pwm	B.5	13
6S	.6	In / Out		B.6	-
7S	.7	In / Out		B.7	-
8	S.8	In / Out	timer clk	C.0	-
9	S.9	In / Out	pwm	C.1	-
10	S.10	In / Out / ADC / Touch	hpwm A / pwm	C.2	14
11	S.11	In / Out / ADC / Touch	hspi sdo	C.5	17
12	S.12	In / Out / ADC / Touch	hspi sdi / hi2c sda	C.4	16
13	S.13	In / Out / ADC / Touch (or LED via H4)	hspi sck / hi2c scl	C.3	4
GND		0V		0V	
VREF	S.A3	In / Out / ADC / Touch	Comp1+ / Vref	A.3	3

2.2 Modifying incompatible shields

Great care has been taken to align the PICAXE pins with the standard shield format. Therefore almost all shields will work directly out of the box with the PICAXE shield base without any modification at all.

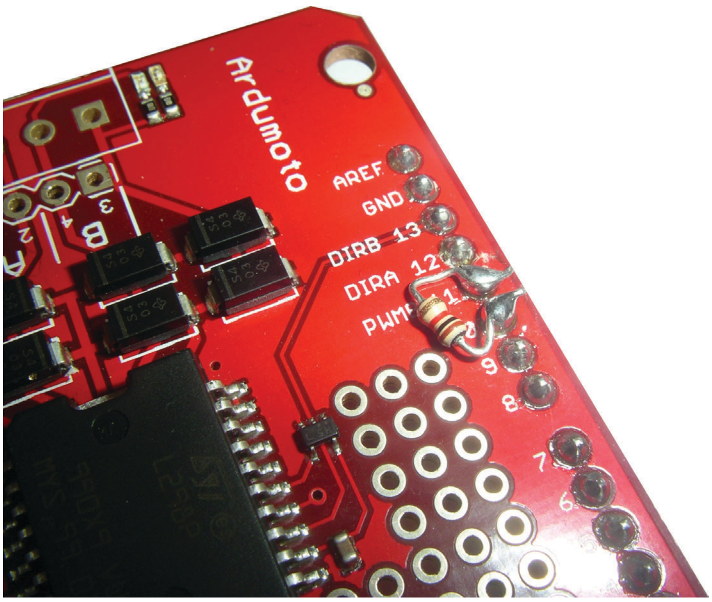
However there are some unavoidable very minor differences that may affect a couple of shields. The hardware differences to an Arduino system are:

- 28X2 has PWM on S.3, S.5, S.9, S.10, but not on S.6 and S.11
- 28X2 has I2C connections on S.12 (instead of S.A4) and S.13 (instead of S.A5)

This is due to unavoidable silicon design differences between the ATmega and PIC microcontrollers.

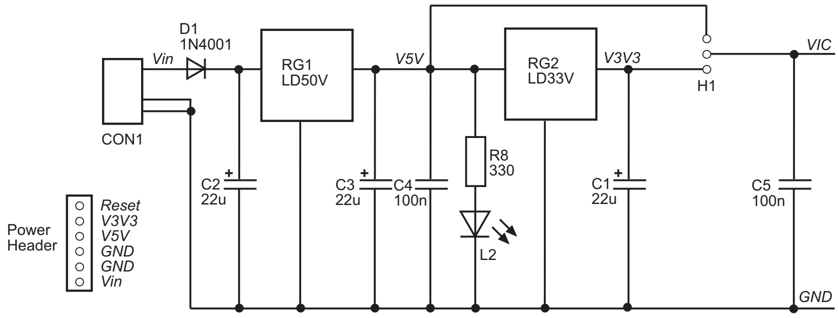
The PICAXE-28X2 system has 4, not 6, PWM channels. Therefore if a required PWM hardware feature is not on an available pin, a very simple modification may be required. As a real-world example, let us look at the Sparkfun Ardumoto motor controller shield. This requires PWM signals on pins S.3 and S.11 to operate. On the PICAXE shield base S.3 already has PWM so is no problem to use, but S.11 does not have PWM (although S.10 does). So we need to redirect the shield pin from using S.11 to using S.10 instead.

This is easily achieved by simply soldering a 1k resistor between pins S.11 and S.10 on the shield (either above the shield (as shown) when using normal headers, or below the shield when using stacking headers). As long as S.11 is maintained as an input, S.10 can now control the PWM to the motor driver. If desired you can even completely cut off the S.11 header pin, although this is not essential as the 1k resistor provides protection against accidentally setting both pins as outputs.



3.0 Power Supply

AXE401 PICAXE Shield Base - Power Circuit



The shield base is primarily designed to be used with an external 'plug in the wall' regulated power supply that provides 9-12V DC on a 2.1mm, centre (tip) positive connector. A suitable power supply for UK use (only) is part PWR009A.

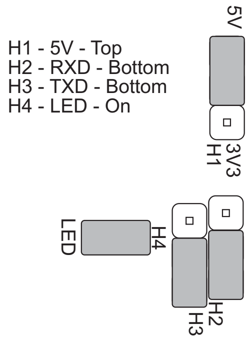
A minimum of about 6.5V is required via the 2.1mm power connector, and voltages above 12V may cause the regulators to get hotter (some warmth is normal). Do not use a PP3 9V battery via this connector, this type of 9V PP3 battery is designed for long term minimal current use (e.g. in a smoke alarm) and so is simply not suitable for long term use of any type of shield base.

The board can also be powered via batteries (e.g. 4.5V from 3 x AA alkaline cells) via the 5V power header. Connect the red positive wire to '5V' and the black negative wire to 'GND'. Alternately, if assembling your own board, leave off the 5V regulator and solder in a battery clip instead (see the minimum battery powered circuit assembly example).

The shield base contains both 5V and 3V low-drop out regulators, each rated at 500mA. Therefore for 3V work there is generally no need to build a separate 3V regulator onto the shield. However if the shield already has a 3V regulator fitted this may still be used.

3.1 Why the 5V / 3V Jumper H1?

Default positions of the 4 jumper links



Most shields are designed to work with a 5V system voltage, so jumper H1 should be left in the 5V position. However when building your own circuit using a proto shield with a third party 3V only device, for instance an XBee or GPS module, this third party device may be 3V tolerant only and hence damaged at 5V. So rather than build 5 to 3V conversion circuits on to the proto shield it is simply much easier to run the entire system voltage at 3V to start with. This is no problem for the PICAXE-28X2, which can be used at any system voltage between 2.1V and 5.5V.

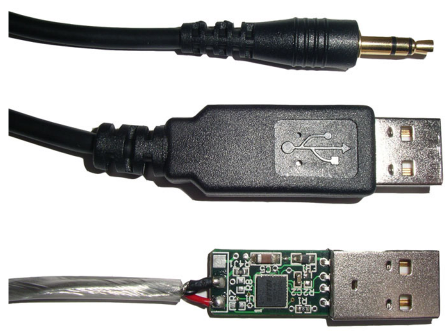
3.2 Can I power the board from the USB port of my computer?

We are not keen supporters of the USB powering method. The majority of PICAXE shield projects are only connected to the computer whilst programming, so generally require a separate power supply system for the actual project use anyway. Therefore we suggest that you might as well use this power supply all the time! Many users also fail to recognise the power limitations of USB (e.g. when a laptop goes to sleep), and the current limitations when using the more complex shields with additional on board regulators or motors. However if you so desire you may naturally choose to use a USB breakout board or USB breakout cable to provide 5V to the board (via the power header).

4.0 PICAXE USB download cable

The AXE027 PICAXE USB cable is recommended. This is widely available from various PICAXE distributors around the world. It works reliably with Windows, Mac and Linux. You may alternately also use universal USB-to-serial converter cables (make sure they have a Prolific or FTDI chipset) with a home made cable.

The AXE027 contains an internal FTDI USB-to-serial conversion circuit, specifically optimised for the PICAXE system. This FTDI chip is the same chip that is often found on other shield bases (e.g. the Arduino Duemilanove). However building the chip into the cable instead of the shield base has several advantages:



1) The cost of each shield decreases, as you only purchase the USB chip once when you purchase the cable (rather than buying a USB chip on every shield base).
2) The FTDI chip is only available in a very small surface mount format, so it would make a self assembly PICAXE shield base almost impossible.
3) A single download cable can be used for all your different PICAXE projects, even those not built upon shield bases.

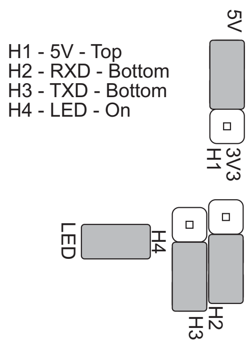
Therefore the initial cost of the 'intelligent' AXE027 download cable is slightly more expensive than a non-intelligent 'printer' style USB cable, but has the advantage that it can be used across all your PICAXE projects, and will also work out more economical in the long run if you purchase more than one shield base or project board.

4.1 Why the download cable jumper links H2 and H3?

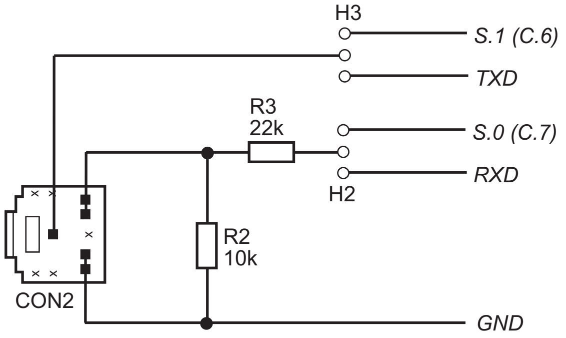
The PICAXE-28X2 system uses completely separate pins of the PICAXE chip for new program downloads. Therefore the digital hardware serial port pins (S.0 and S.1) have no effect on new program downloads. Data within the program can also be transmitted directly back to the computer (without using S.0/S.1) via the download cable with the debug, sertxd and serrxd BASIC commands.

However on occasions for some projects it might be desired to have the S.0 and S.1 i/o pins connected to the download cable. This can be achieved via the jumpers H2 and H3.

Default positions of the 4 jumper links



AXE401 PICAXE Shield Base - Download Circuit



5.0 Clock Speed

The PICAXE-28X2 can use a clock frequency between 32kHz and 64MHz. The clock speed can be changed at any point within the BASIC program via the 'setfreq' command. The PICAXE-28X2 uses an internal resonator for frequencies up to 16MHz, and then an external resonator for either 32, 40 or 64MHz.

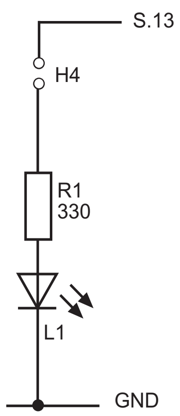
A higher frequency means a faster operating speed, but the power use of the microcontroller also considerably increases. Therefore the default PICAXE-28X2 power up frequency is 8MHz (internal resonator), which is a good compromise between speed and power consumption, and quite suitable for the majority of shield based projects. However very complex projects (e.g. using an ethernet shield) may benefit from selecting a faster operating speed via the setfreq command.

To use speeds greater than 16MHz a 3 pin ceramic resonator (not supplied, e.g. part RES037) must be soldered onto the board in position X1. As the PICAXE-28X2 contains an internal 4x PLL multiplier, an external 16MHz resonator provides a 64MHz operating frequency. An external 8 MHz resonator would provide a 32MHz operating frequency, an external 10MHz resonator gives 40MHz.

Note that each PICAXE command requires a number of 'operating instructions' to process. Different commands require different numbers of operating instructions, as some BASIC commands (e.g. readtemp) are much more complex to decode and process than other commands (e.g. high). Therefore processing speed is not the same as one 'BASIC command per operating instruction'. However the default 8MHz operating speed is still quite sufficient for the majority of shield based projects.

5.1 Why is the LED on pin S.13 connected via jumper H4?

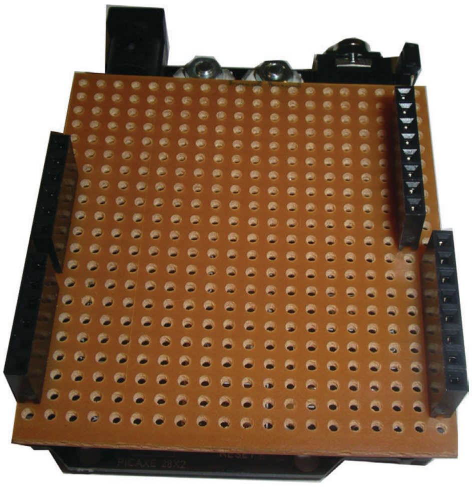
Many shield bases have an LED permanently connected to output S.13. However this can adversely affect using this pin for another purpose e.g. as a touch sensor input or for i2c communication. Therefore jumper H4 gives the best of both worlds - you can have the LED connected if you want to, or remove the jumper link and hence completely disconnect the LED from this pin.



5.2 Why the double parallel socket beside outputs S.8 to S.13?

You'll already know the answer to this question if you have ever tried to build your own shield using stripboard (veroboard). Because the gap between the two digital i/o connectors on the shield format is not on a 0.1" (2.54mm) grid, it is very difficult to make your own shield from stripboard to fit the normal shield connector positions. The second parallel header brings these solder joints onto the stripboard grid, to allow a stripboard shield design to be easily made.

When working with stripboard the stacking headers (parts CON060 and CON061) are recommended (as shown).

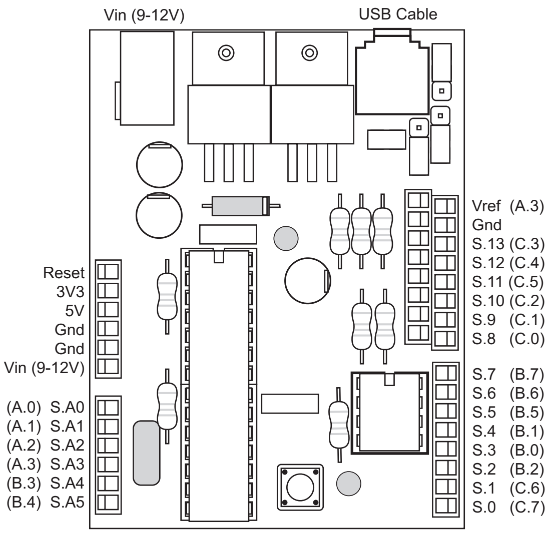


5.3 What is the EEPROM socket for?

The EEPROM socket is for users who wish to add an I2C memory EEPROM to their system. This could provide additional program or data storage memory space. The 28X2 program could also be updated using the 'booti2c' command using an EEPROM inserted into this socket. A suitable part would be the Microchip 24LC512-I/P. Two 4k7 pullup resistors must also be added in position R5 and R6 (i2c communication with the PICAXE system uses pins S.12 and S.13). The i2c slave address is fixed at %10100000 - see the hi2csetup/hi2cin/hi2cout BASIC commands for more details.

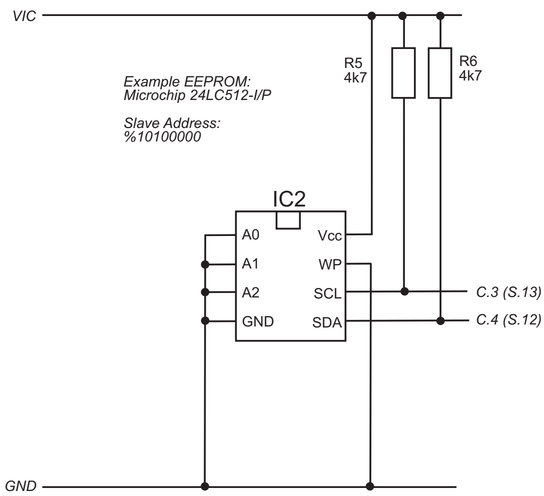
H4 *must* be disconnected to remove the LED from the circuit when using i2c.

The socket may also be used with a Microchip UNIO type memory EEPROM and the uniin/uniout commands.

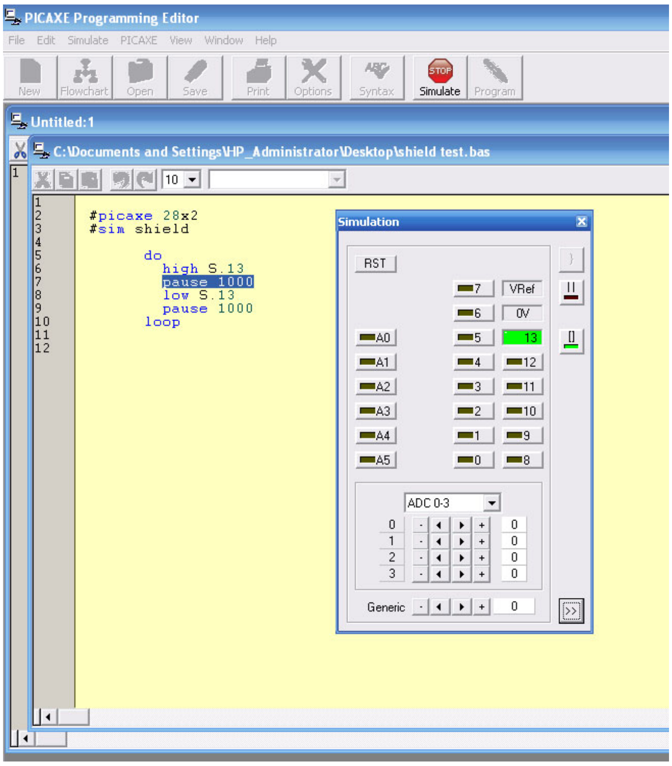


AXE401 PICAXE-28X2 Shield Base

AXE401 PICAXE Shield Base - Optional EEPROM Circuit



6.0 Programming the PICAXE



The PICAXE uses a 'BASIC' style interpreted language, which is often considered easier to use than the 'C' or 'assembler' programming languages. Of course if you are a commercial programmer by profession using 'C' may be no problem to you, but for many school students, hobbyists or artists starting out with no prior programming experience the 'C' language can sometimes prove difficult to use (and, in particular, fault find and debug). BASIC does not have these restrictions as it is designed to be simpler to use for beginners. However this does not mean that the BASIC language is not powerful - as complex programs can also be easily constructed, particularly with the simple to use in-built BASIC commands for interfacing protocols such as RS232, 1-wire, I2c and SPI.

PICAXE software is available as the 'Programming Editor' for Windows and 'AXEpad' for Mac and for Linux. All three titles are completely free to download and use. If you prefer to program in a different third party editor (e.g. Kate on Linux), free command line compilers are also available for all 3 computer platforms.

PICAXE can also be programmed 'graphically' in various other software applications such as flowcharts (e.g. the Logicator software) and graphical 'Scratch' icon based systems.

6.1 Can I on-screen simulate PICAXE programs?

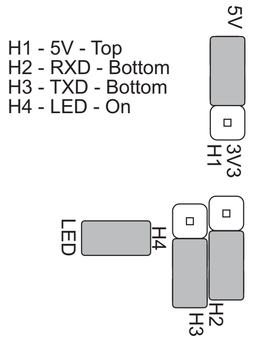
The free Programming Editor software supports a full on-screen simulation of the PICAXE program, so that the program can be stepped through line by line an analysed step by step. Breakpoints can also be set to stop the program at any point. A special '#sim shield' directive allows the on screen layout to match the pinout of the shield.

Another piece of software, PICAXE VSM, is a full featured SPICE and digital circuit simulator that supports circuit simulations. Through on screen animations you can see, for instance, LEDs light up when you flick a simulated switch or change a light level.

6.2 Getting Started - Test Program

This procedure assumes the PICAXE Programming Editor software and AXE027 USB cable drivers are already installed. See the PICAXE manuals for more details:
www.rev-ed.co.uk/docs/picaxe_manual1.pdf
www.rev-ed.co.uk/docs/axe027.pdf

Default positions of the 4 jumper links



1. Lay the shield base down so that the power and USB cable sockets are to the top. Do not connect any shield at this time.
   Ensure jumper H1 is in the top '5V' position.
   Ensure jumper H2 and H3 are both in the bottom position.
   Ensure jumper H4 is fitted (to connect the on board LED (L1) to pin S.13)
2. Connect a 9V regulated power supply (2.1mm, tip positive) into the power socket. The power LED beside the reset switch should light.
3. Insert the AXE027 download cable into the shield base download socket.
4. Within the PICAXE software select the View>Options menu
5. Ensure the mode is set to 'PICAXE-28X2' and the COM port is set to the 'AXE027 PICAXE USB' cable. Click OK.
6. Type in this program:

```
#picaxe  28x2
do
     high  S.13
     pause  1000
     low  S.13
     pause  1000
loop
```

and then click the PICAXE>Program menu. If all is well the LED L1 will flash on and off every second. Congratulations, your system is now up and running!

*NB: If your LED L1 is dimly on all the time check you have *not* fitted the optional I2C resistors R5 and R6 - as they must not be used at the same time as the LED!*

7. Now add this extra '#sim shield' line to the top of your program:

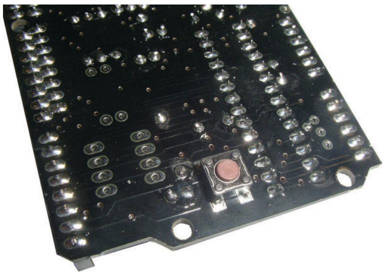
```
#picaxe 28x2
#sim  shield

do
     high  S.13
     pause  1000
     low  S.13
     pause  1000
loop
```

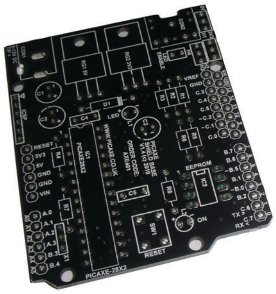
Now click the Simulate>Run menu. An on screen simulation of your program will now occur, demonstrating the on-screen simulation features of the PICAXE Programming Editor software.

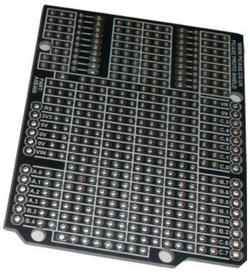
Great - you hopefully now have a working PICAXE shield base! The next thing to try out is a few tutorials in the PICAXE manual (part 1) to learn more about the PICAXE BASIC language. When using the examples remember that the shield pins are labelled using the format S.A0, S.A1, S.0, S.1, S.2 etc.

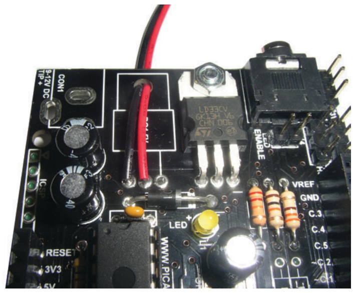
7.0 Kits - Why offer a self assembly kit as well as pre-assembled boards?



Although we also provide the shield base pre-assembled, many PICAXE users simply like soldering kits together (and saving some money at the same time!). As well as the satisfaction of making your own board, it also gives you the opportunity to customise and 'hack' the board, for instance you may want a blue LED as your power indicator, or you may want to use batteries and so can leave off the 5V regulator and solder in a battery clip instead. Many users also choose to solder the reset switch on the bottom side of the PCB instead of the top, so that it can still be easily pressed when a shield is in position. These modifications are all easily achieved when self assembling your own board.





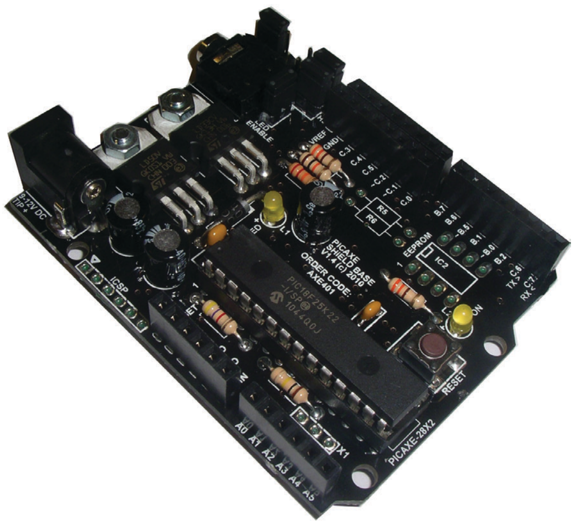


So the following options are available - a bare PCB (AXE401PCB), a kit of all parts including PCB and PICAXE-28X2 (AXE401KIT), or a fully assembled module (AXE401). All three versions are provided with a completely free AXE405 prototyping shield PCB. A starter pack including the AXE027 USB cable as well as the assembled shield base is also available (AXE410U).

The self assembly board is fairly simple to solder together by hand in about 20 minutes and does not use any small surface mount parts. It has a commercial grade solder resist 'lacquer' layer on both sides - this means solder only sticks where it is supposed to! We also design all our boards with thicker than normal tracks and large pads, this is deliberate to make self assembly via home soldering easier.

The pre-assembled board may be supplied populated with surface mount resistors and LEDs (this reduces manufacturing cost), but is exactly the same shape and functionality as the self assembly board. We never use surface mount microcontrollers on our pre-assembled boards – we know people sometimes do accidentally make short circuit wiring mistakes, and it is much cheaper (and environmentally friendly) to swap a chip in a socket than buy a whole new module!

Remember to peel off the green pad protection layer over the EEPROM and resonator pads before soldering these (optional) components. This protective layer is easily peeled off with a finger nail or edge of a small screwdriver.



7.1 AXE401 - Contents (pre-assembled shield base)

1	AXE401 PICAXE Shield Base (pre-assembled)	
1	AXE405 PICAXE Proto Shield PCB	
4	10 way 2.54mm headers	

7.2 AXE401Kit - Contents (self assembly kit)

	1	AXE401 PICAXE Shield Base PCB	
	1	AXE405 PICAXE Proto Shield PCB	
R1, R8	2	330 0.25W resistor	orange orange brown gold
R2	1	10k 0.25W resistor	brown black orange gold
R3	1	22k 0.25W resistor	red red orange gold
R4	1	4k7 0.25W resistor	yellow violet red gold
(R5,R6)	0	optional, not supplied	
R7	1	100k 0.25W resistor	brown black yellow gold
D1	1	1N4001 Diode	1N4001
RG1	1	5V 500mA LD1117V50 regulator	LD50V
RG2	1	3.3V 500mA LD1117V33 regulator	LD33V
C1,C2,C3	3	22uF 35V electrolytic capacitor	22u
C4,C5	2	100nF polyester capacitor	104 or .1
L1, L2	2	3mm yellow LED	
S1	1	6mm miniature push switch	
CT1	1	2.1mm power socket	
CT2	1	3.5mm stereo download socket	
IC1	1	28 pin pressed pin IC socket	
IC1	1	PICAXE-28X2 microcontroller	PIC18F25K22
(IC2)	0	optional, not supplied	
(X1)	0	optional, not supplied	
H1-4	5	10 way 2.54mm headers *(snap to length, see below)*	
H1-4	4	2.54mm jumper links	
I/O1-2	2	6 way 2.54mm socket	
I/O3-5	3	8 way 2.54mm socket	
	2	M3 bolts (6mm length)	
	2	M3 nuts	

7.3 AXE401PCB - Contents (PCB)

1	AXE401 PICAXE Shield Base PCB
1	AXE405 PICAXE Proto Shield PCB

7.4 Tools required for assembly (not supplied)

Soldering iron and 22swg solder
Minature side cutters / pliers
Small cross-head screwdriver

Basic soldering skills have been assumed.

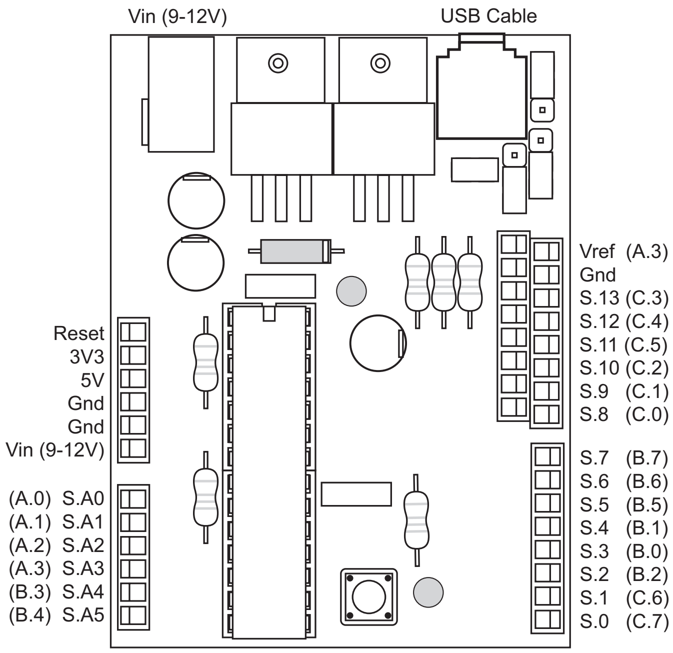
7.5 Self assembly kit preparation

Peel the protective covering from the rear of the PCB over the pads for X1, ISCP and IC2 (if present). This can be easily lifted with a finger nail or edge of a small screwdriver.

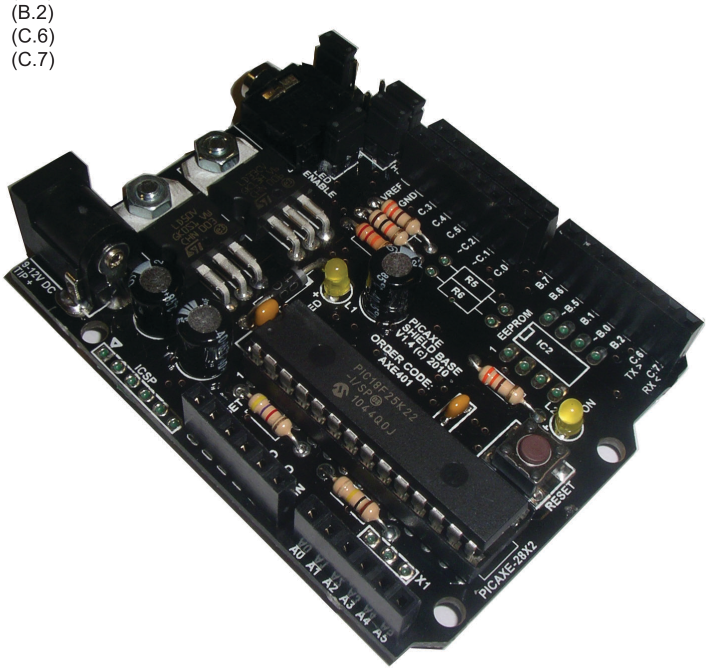
Carefully snap the 10 way headers into the following combinations:

8 + 2
8 + 2
6 + 3 (+ 1)
6 + 3 (+ 1)
7 + 3 *(not required on preassembled kit)*

The 1x 2 way and 3x 3 way lengths are used on the AXE401 PICAXE shield base.
The 2x 8 way and 2x 6 way lengths are used on the AXE405 PICAXE proto shield.
The extra 7 way and 2 way header are spares!



AXE401 PICAXE-28X2 Shield Base



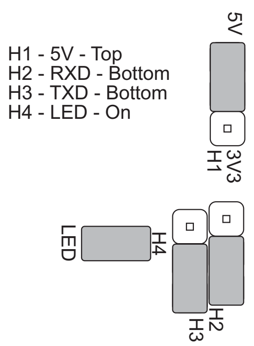
7.6 Assembly

1. Solder the six resistors in position (note that R5 and R6 are not used). Resistors may be soldered either way around.

R1	330	orange orange brown gold
R2	10k	brown black orange gold
R3	22k	red red orange gold
R4	4k7	yellow violet red gold
R7	100k	brown black yellow gold
R8	330	orange orange brown gold

   Don't fit R5 and R6 unless you are intending on using an I2C EEPROM!
2. Solder the 1N4001 diode in position. The grey band at one end of the diode must align with the white band marked on the PCB.
3. Solder the reset switch in position S1. Note that many users actually choose to position the switch under the board (so that the solder joints are on the top side). This is so that it can still be easily pressed when a shield is in position. However when using this layout ensure the board does not rest on (and hence hold down) the reset switch when in use!
4. Solder the 28 pin socket in position IC1. Note that the socket is soldered to the PCB (the microcontroller is not soldered, it is placed in the socket after assembly is complete).
5. 'Click' the stereo download socket into position so that it lies level (flat) on the PCB. Solder in position. It does not matter if the solder on the two 'pairs' of pins join, as the PCB design joins these pairs together anyway.
6. Solder the two LEDs in positions L1 and L2. The long leg (positive anode) of the LED must be in the hole nearest the '+' marking on the PCB.
7. Solder the two 100nF (marked 100n or .1) capacitors in positions C4 and C5. These can be soldered either way around.
8. Solder the three 22uF electrolytic capacitors in positions C1, C2, and C3. The long leg (positive anode) of the capacitor must be in the hole nearest the '+' marking on the PCB, so that the light blue stripe on the can (negative cathode) must be in the hole nearest the '-' marking.
9. Using a pair of pliers, bend the legs of the two voltage regulators at 90 degrees where the leg narrows. Place in position, but do not solder yet. Note that if the board is only to be powered by a 4.5V (3xAA) battery pack the 5V regulator is not required. Use a battery clip as shown instead. Double check that the 3V3 regulator (LD33CV) is nearest to the stereo download socket.Use an M3 nut and bolt to hold both regulators in position on the PCB - this is very important to prevent the two regulators accidentally touching.
10. Once the nut and bolt have been tightened then solder the regulator legs in position.
11. Solder the 1x 2 way and 3x 3 way headers in positions H1, H2, H3, H4. Solder one pin of each header first, and then check to ensure the sockets are neatly aligned. If necessary reheat the single joint and straighten the header. Solder the other pins.
12. Place the 4 jumper links onto the headers H1-H4 as shown in the diagram.

Default positions of the 4 jumper links



13. Solder the 2 x 6 way and 3 x 8 way sockets in the shield i/o positions at the edge of the boards. Solder one pin of each socket first, and then check to ensure the sockets are neatly aligned before soldering the other pins. Note that the two staggered headers of i/o pins S.8 to S.13 are joined in parallel, so do not worry if the solder joints between the pairs of parallel pins also joins together. Tip - if you already have an assembled shield it can be temporarily inserted to hold the sockets in place whilst soldering.
14. Solder the 2.1mm power connector in position, ensuring it is aligned squarely to the PCB.
15. Insert the PICAXE-28X2 (PIC18F25K22) microcontroller in the IC socket, ensuring pin1 (the end with a dent) is nearest the diode.

Congratulations, assembly is now complete.

7.7 Optional Components (not supplied)

Peel the protective covering from the rear of the PCB over the pads for X1, ICSP and IC2 (if present). This can be easily lifted with a finger nail or edge of small screwdriver.

Optional Resonator (X1)

An 8, 10 or 16MHz 3 pin ceramic resonator (e.g. part RES037) may be optionally soldered in position X1. This gives a possible faster operating frequency of 32MHz, 40MHz or 64MHz (setfreq em64) respectively. By default the PICAXE-28X2 uses it's internal 8MHz resonator and so the external resonator is not required.

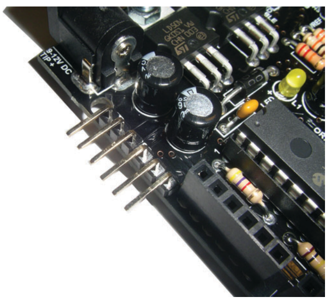
Optional I2C EEPROM (IC2, R5, R6)

1	RES-4K7	4k7 resistor (pack 100)
1	ICH008	8 pin IC socket
1	MIC051	24LC512-I/P EEPROM

Solder a 4k7 pull-up resistor (yellow violet red gold) in positions R5 and R6. Solder an 8 pin socket in position IC2 and fit with an EEPROM (e.g. Microchip 24LC512-I/P). Note H4 'LED' jumper link must be removed when using I2C.

Optional ICSP Header for Microchip PICKit2 (or similar)

Solder a 5 or 6 pin right angle 2.54mm (0.1") header in the ICSP position (pin 6 is not used, so a 5 way header may be used if desired).



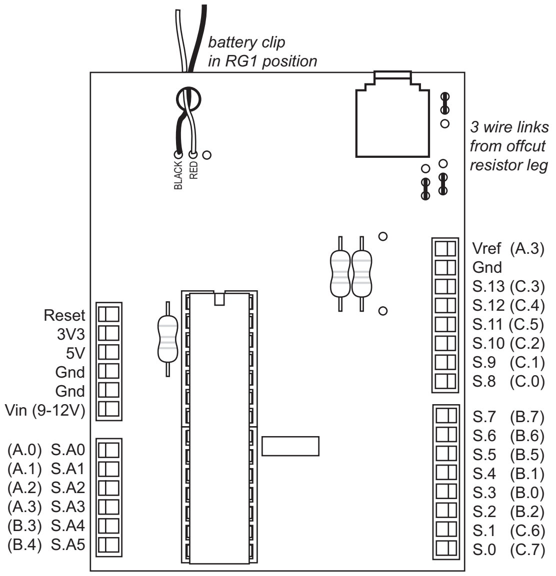
7.8 Alternate assembly - low budget 4.5V battery powered setup

	1	AXE401 PICAXE Shield Base PCB	
	1	AXE405 PICAXE Proto Shield PCB	
R2	1	10k 0.25W resistor	brown black orange gold
R3	1	22k 0.25W resistor	red red orange gold
R4	1	4k7 0.25W resistor	yellow violet red gold
RG1	1	Battery Clip	
C5	1	100nF polyester capacitor	104 or .1
CT2	1	3.5mm stereo download socket	
IC1	1	28 pin pressed pin IC socket	
IC1	1	PICAXE-28X2 microcontroller	PIC18F25K22
I/O1-2	2	6 way 2.54mm socket	
I/O3-4	2	8 way 2.54mm socket	

This is the minimum parts list required for a minimalistic, very low cost solution. Here the shield base is configured for permanent battery power, either 4.5V or 3V. The battery clip is soldered in the RG1 position.

Permanent wire links (resistor leg offcuts) are used instead of jumper links H1,2,3.

If desired the reset switch and LEDs/330 resistors may also be added.

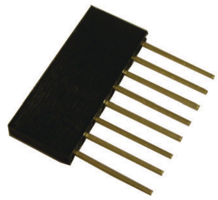


AXE401 PICAXE-28X2 Shield Base (Minimum 4.5V Battery Circuit)

8.0 AXE405 PICAXE Proto Shield

1	AXE405 PICAXE Proto Shield PCB
2	6 way 2.54mm header
2	8 way 2.54mm header

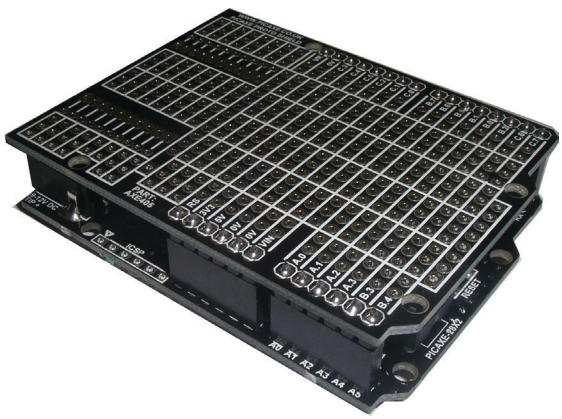
1. Place the short end of the headers through the PCB, so that the solder joint will be on the top (white text) side of the PCB. Solder in position.
2. If desired the headers may be replaced by ‘stacking’ headers as shown here (must be purchased separately, parts CON060 and CON061). In this case the solder joints will be on the bottom side of the PCB.



8.1 Using the proto shield

The proto shield is a ‘tri-pad’ design, where most of the prototyping solder pads are joined together in sets of three as indicated by the white boxes. This enables simple connection of electronic components - to join two components together simply make sure they are both soldered to pads within the same box. This system also allows easy use of other chips (ICs), simply place each row of pins in the centre of a set of pads.

There are also two long linear boxes where multiple pads are joined together. These are normally used for power rails - e.g. so you would connect via a jumper wire to 0V, 3.3V or 5V as required.



8.2 Using the proto shield with an XBee Wireless Module

The proto shield also contains two sets of 10 x 2mm spaced oblong pads, these are designed to allow use of the popular XBee wireless modules. To use the protoboard with an XBee module 2 x 10 pin 2mm sockets should be purchased (part CON040) and soldered in position. Alternately the XBee module may be directly soldered to the PCB.

The recommended minimum configuration of an XBee module is:

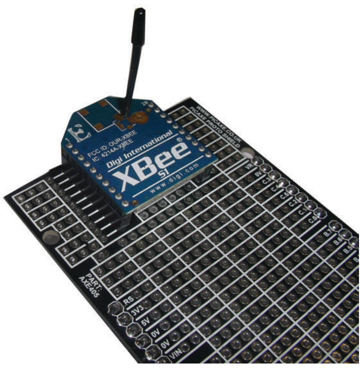
1	V+	3V3
2	TXD	to a PICAXE input (any I/O pin, S.0 recommended)
3	RXD	to a PICAXE output (any I/O pin, S.1 recommended)
9	SLEEP	to a PICAXE output (any I/O pin, S.2 recommended)
10	0V	GND
14	Vref	3V3

It is also common to connect an LED (and 180R) resistor between these XBee pins and GND so that the status of the XBee module may be observed:

6	RSSI	(indicates the received signal strength)
11	TX	(indicates module is transmitting)
13	ON	(indicates whether the module is active or sleeping)
15	ASSOC	(indicates if the module has 'associated' with another module)

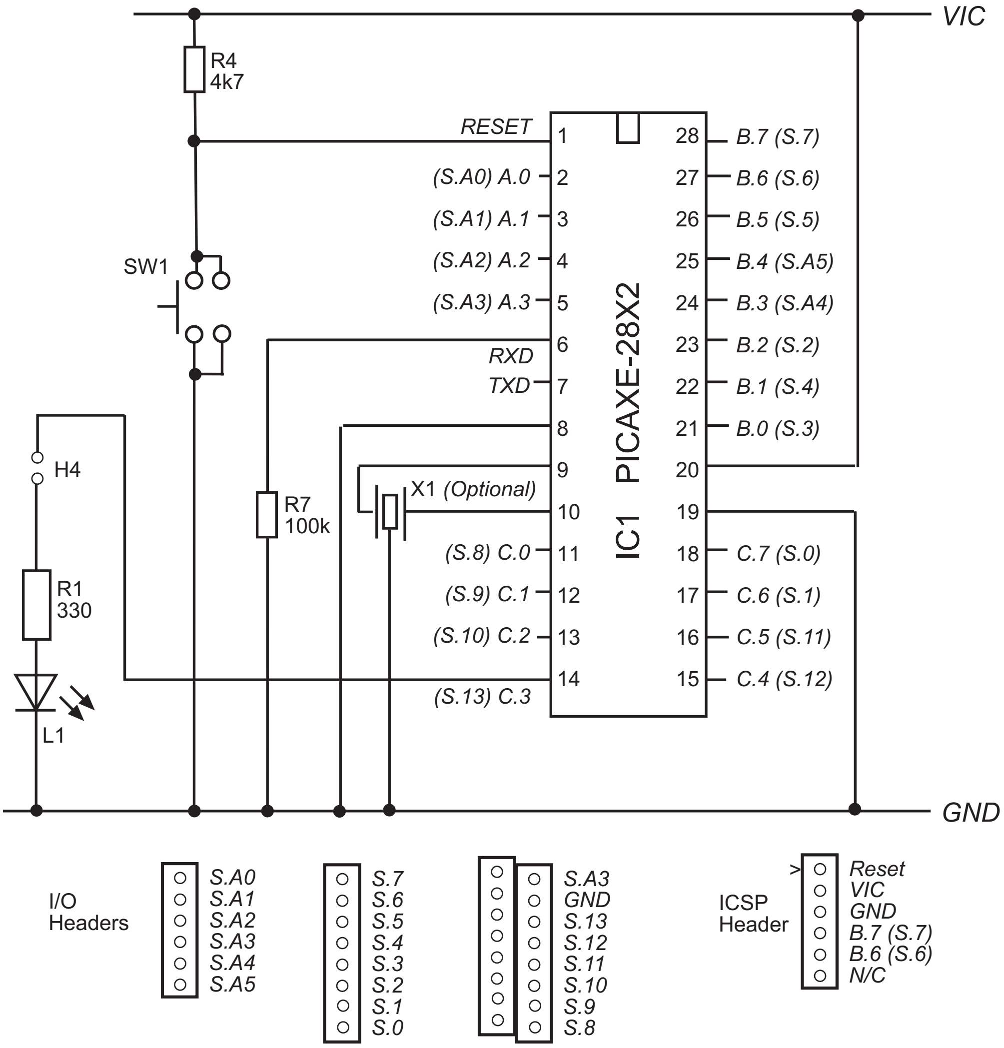
Note that when using an XBee module the PICAXE shield base *MUST* be used with a 3V system voltage i.e. jumper H1 on the PICAXE shield base must be in the 3V3 position. The 3V3 500mA regulator on the PICAXE shield is suitable for use with both 'XBee' and 'XBee Pro' modules.

To communicate between an XBee proto shield and a computer, the AXE210 project board is recommended. The shield base and Xbee module is then the remote item, the AXE210 and Xbee is connected to the computer (via the AXE027 USB cable).

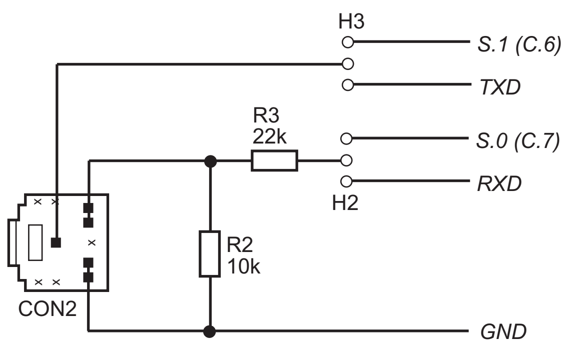


Appendix A: Circuit Schematics - PICAXE and Download Circuits

AXE401 PICAXE Shield Base - PICAXE-28X2 Circuit

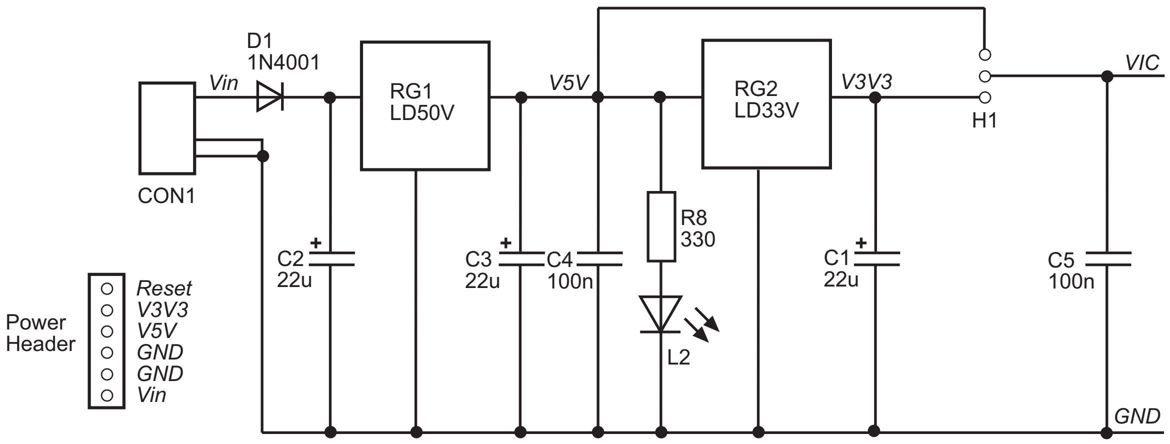


AXE401 PICAXE Shield Base - Download Circuit

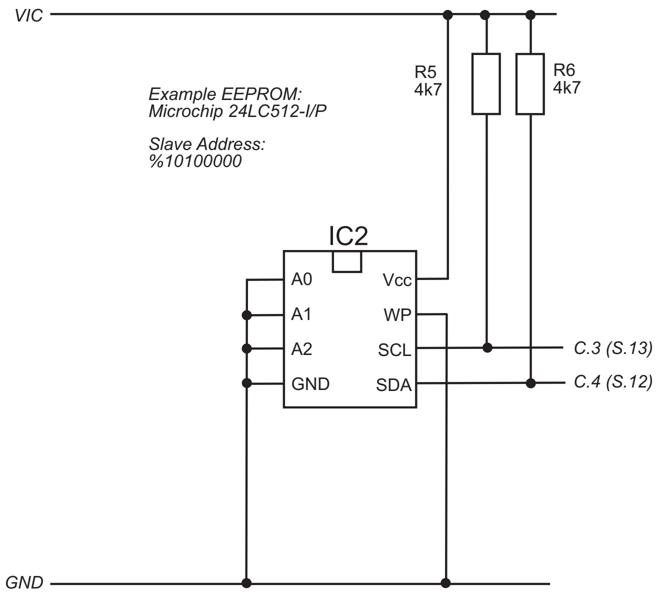


Appendix B: Circuit Schematics - Power and optional EEPROM Circuits

AXE401 PICAXE Shield Base - Power Circuit

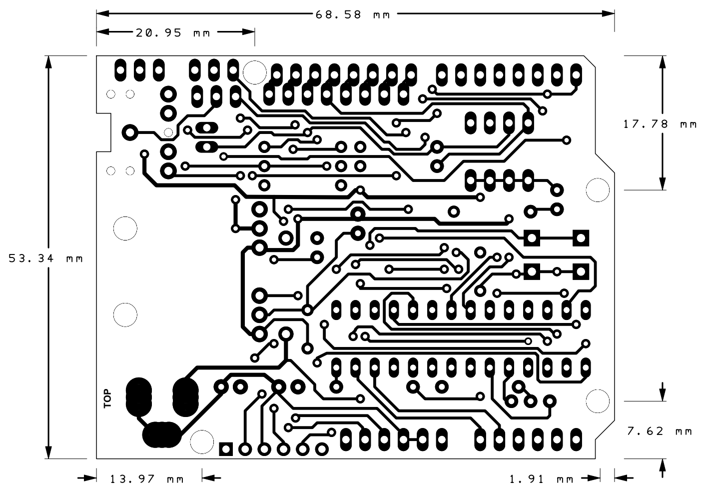


AXE401 PICAXE Shield Base - Optional EEPROM Circuit



Note H4 'LED' jumper link must be removed when using I2C.

Appendix C: Dimensions



Appendix D: What are the principle differences between using a PICAXE shield base and an Arduino?

The PICAXE shield base has been designed to be compatible with almost all existing third party shields. So it is very similar is shape and connector layout, and if you already have a shield the chances are it will work straight way. If not the shield is probably very easily modified to work.

However there are three main important differences between the systems:

Firmware	PICAXE-28X2	Arduino
Microcontroller Brand	Microchip PIC	Atmel ATmega
Programming Language	BASIC or flowcharts	C

There are also lots of similarities:

Physical Size & Header Position	Same	Same
Open Source Hardware	Yes	Yes
Software Cost	Free	Free
Support for Windows	Yes	Yes
Support for Mac	Yes	Yes
Support for Linux	Yes	Yes
Downloads via USB port	Yes	Yes

The microcontroller comparison is:

Program Slots	4 x 4k	1 x 14k/30k
User RAM bytes	1296	1024/2048
User EEPROM bytes	256	512
Run extra programs from i2c	Yes	No
System Voltage	5V or 3V	5V
Clock Speed	32kHz - 64MHz	16MHz
Digital I/O	Up to 20	14
Analogue Sensors	Up to 16	6
Touch Sensors	Up to 16	0
PWM Channels	4	6
Support for UART, SPI, I2C	Yes	Yes

There are also a few hardware advantages to the PICAXE shield base hardware:

Microcontroller System Voltage	5V or 3V	5V
3V Low drop out regulator	500mA	50mA
Disconnectable LED	Yes	No
Stripboard compatible headers	Yes	No
On board i2c EEPROM socket	Yes	No
Separate programming and i/o pins	Yes	No
TX / RX select jumper headers	Yes	No
Through hole for self assembly	Yes	No
Editor code syntax highlighting	Yes	No
Editor on-screen simulation	Yes	No

Comparison made between PICAXE-28X2 and official Arduino Duemilanove ATmega168/ Uno ATmega368 with 2k bootstrap. E. & O. E.

What is the difference between a Microchip PIC and an Atmel ATmega microntroller?

As a real world comparison, if you want to drive from your house to the local shops, it doesn't really matter if you use an Audi or a Mercedes car, they are both good quality cars and will comfortably transport you the distance. Of course Audi enthusiasts will always argue that their car is better, whilst Mercedes enthusiasts will do the same!

Likewise the web is full of heated forum discussions of why a Microchip PIC is better than an Atmel ATmega and vice versa! But the bottom line is that they are both microcontrollers and quite similar for most shield based electronics projects. For almost all shield projects both chips will do a very good job. In global terms Microchip have a larger market share of the commercial microcontroller market, and in 2008 attempted an (unsuccessful) buyout of Atmel.

Each chip has certain minor advantages and disadvantages, for instance the PIC has more ADC channels with better in-built touch sensor support, whilst the ATmega has a couple more PWM outputs than the PIC. But as a summary they both do the same job – a single chip 'brain' for your control program.

When you buy both systems the microcontroller is pre-programmed with the bootstrap 'firmware' before you buy the shield base. The Atmel ATmega chip is preprogrammed with the 'Arduino' bootstrap firmware, the Microchip PIC18F25K22 chip is preprogrammed with the 'PICAXE-28X2' bootstrap firmware. This firmware enables your control program to be downloaded to the chip via the USB cable.

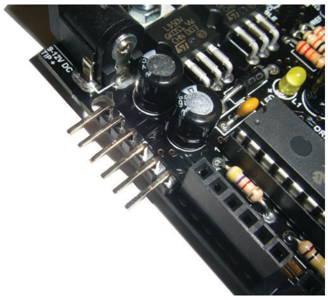
However the programming language / style between the two systems is very different – PICAXE uses interpreted 'BASIC' or graphical flowcharts whilst Arduino uses 'C'.

Appendix E: FAQ

FAQ - Can you provide a sewable 'e-textiles' version PICAXE controller like the Lilypad?

Yes, the 'DaisyPICAXE' system uses a surface mount PICAXE-20M2 microcontroller to provide a circular e-textiles 'sewable' controller system, programmed in BASIC via the AXE027 USB cable. The system was designed by Paul Gardiner with the approval of Leah Buechley, designer of the original Lilypad. The PICAXE-20M2 microcontroller allows up to 8 parallel program tasks to run at the same time, greatly simplifying the programming of complex lighting displays.

FAQ - I want to use a PIC microcontroller, but program it in assembler or C (or other language) instead of using the PICAXE system. Can I do this?



Of course, you can use whichever PIC programming system you like! On the side of the shield base are 6 pads for soldering in an optional 6 pin programming header (a 6 pin right angle 0.1" (2.54mm) header, not supplied), so that the board can be used with any 28 pin PIC chip and a low-cost programmer such as the Microchip PICKIT2. Just make sure your PIC type is voltage tolerant of the system voltage (3 or 5V) that is in use via jumper H1 (the PIC18F25K22 supplied can use either voltage, but not all PIC microcontrollers have this large operating voltage range).

Note that programming a PICAXE chip via the PICKit2 will permanently erase the PICAXE firmware, so the chip will no longer function as a PICAXE chip (but can naturally still be reprogrammed via the PICkit2 system).