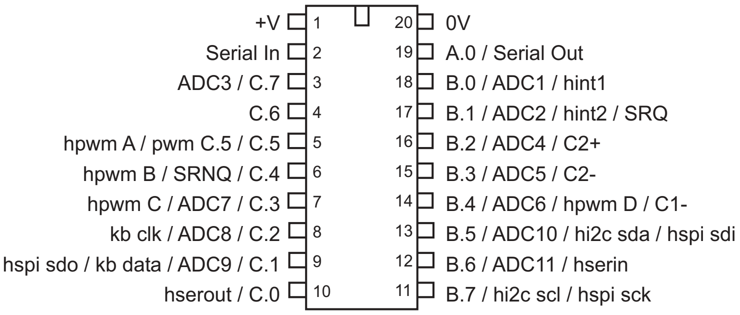
PICAXE X2 Product Briefing.

Introduction

This product briefing is designed to inform existing PICAXE users about the enhanced programming commands and features of the exciting new X2 range of PICAXE microcontrollers. Further details about each command and feature are available in the updated PICAXE Manuals (v6.8 or later). Programming Editor software must be v5.2.3 or greater.

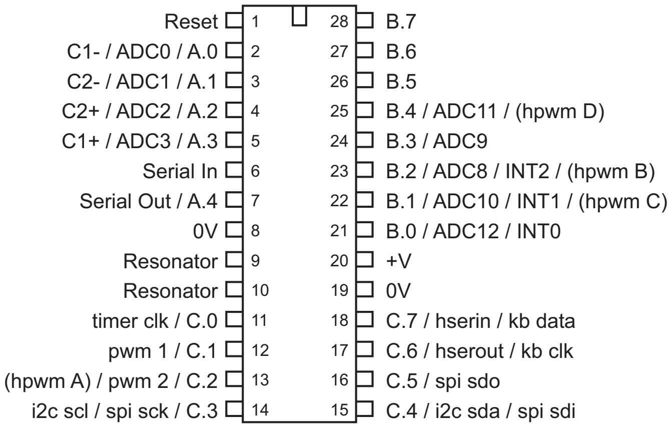
Pinouts

PICAXE-20X2



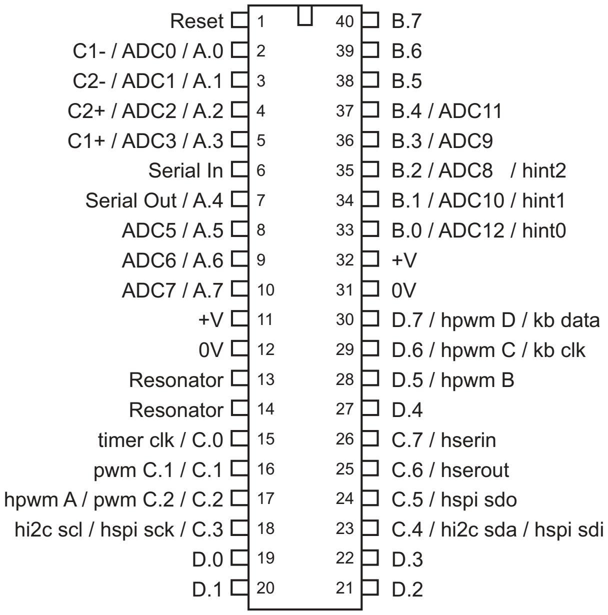
+V	1	20	0V
Serial In	2	19	A.0 / Serial Out
ADC3 / C.7	3	18	B.0 / ADC1 / hint1
C.6	4	17	B.1 / ADC2 / hint2 / SRQ
hpwm A / pwm C.5 / C.5	5	16	B.2 / ADC4 / C2+
hpwm B / SRNQ / C.4	6	15	B.3 / ADC5 / C2-
hpwm C / ADC7 / C.3	7	14	B.4 / ADC6 / hpwm D / C1-
kb clk / ADC8 / C.2	8	13	B.5 / ADC10 / hi2c sda / hspi sdi
hspi sdo / kb data / ADC9 / C.1	9	12	B.6 / ADC11 / hserin
hserout / C.0	10	11	B.7 / hi2c scl / hspi sck

PICAXE-28X2



Reset	1	28	B.7
C1- / ADC0 / A.0	2	27	B.6
C2- / ADC1 / A.1	3	26	B.5
C2+ / ADC2 / A.2	4	25	B.4 / ADC11 / (hpwm D)
C1+ / ADC3 / A.3	5	24	B.3 / ADC9
Serial In	6	23	B.2 / ADC8 / INT2 / (hpwm B)
Serial Out / A.4	7	22	B.1 / ADC10 / INT1 / (hpwm C)
0V	8	21	B.0 / ADC12 / INT0
Resonator	9	20	+V
Resonator	10	19	0V
timer clk / C.0	11	18	C.7 / hserin / kb data
pwm 1 / C.1	12	17	C.6 / hserout / kb clk
(hpwm A) / pwm 2 / C.2	13	16	C.5 / spi sdo
i2c scl / spi sck / C.3	14	15	C.4 / i2c sda / spi sdi

PICAXE-40X2



Reset	1	40	B.7
C1- / ADC0 / A.0	2	39	B.6
C2- / ADC1 / A.1	3	38	B.5
C2+ / ADC2 / A.2	4	37	B.4 / ADC11
C1+ / ADC3 / A.3	5	36	B.3 / ADC9
Serial In	6	35	B.2 / ADC8 / hint2
Serial Out / A.4	7	34	B.1 / ADC10 / hint1
ADC5 / A.5	8	33	B.0 / ADC12 / hint0
ADC6 / A.6	9	32	+V
ADC7 / A.7	10	31	0V
+V	11	30	D.7 / hpwm D / kb data
0V	12	29	D.6 / hpwm C / kb clk
Resonator	13	28	D.5 / hpwm B
Resonator	14	27	D.4
timer clk / C.0	15	26	C.7 / hserin
pwm C.1 / C.1	16	25	C.6 / hserout
hpwm A / pwm C.2 / C.2	17	24	C.5 / hspi sdo
hi2c scl / hspi sck / C.3	18	23	C.4 / hi2c sda / hspi sdi
D.0	19	22	D.3
D.1	20	21	D.2

Inputs and Outputs

One of the key new features of the X2 series is that almost every pin is configurable as input or output. This creates much more flexibility. Naturally the pins can be configured to the traditional PICAXE layout if desired.

The X2 range have up to 32 configurable input/output pins, which are arranged in 4 ports, labelled A to D. Each port has up to 8 pins (0-7). See the pinout diagrams for the specific 20 / 28 / 40 pin layouts.

Pins are referred to by the notation format PORT.BIT e.g.

```
high B.0
count C.2,1000,w1
```

When using input pin variables (e.g. within if..then commands) the notation pinPORT.BIT is used as the variable name.

```
if pinC.3 = 1 then
```

The whole port can be read or written by using the variable name pinsX

```
let b1 = pinsA              ‘ read the input pins
let b1 = outpinsA           ‘ read the state of the output pins
let outpinsB = %10101010    ‘ control the output pins
```

All pins (with the exception of the download serial output pin) are configured as digital inputs at power-up. Most output commands (high, low, pulsout, serout etc.) automatically convert the pin to an output. However the configuration of the pins can also be controlled by the dirsX variables or the input/output/reverse commands.

```
let dirsB = %11110000
input C.1
output B.2
```

Hardware Interrupt Pins

The X2 has up to 3 pins that can be configured as hardware interrupt pins. When correcty configured, these pins continuously background scan for an edge based trigger, even during sleep. When this trigger occurs a flag is set which can be used to trigger a ‘setintflags’ event. See the ‘hintsetup’ command for more details.

Analogue Inputs

Many more ADC channels, up to 12, are now available. Analogue pins are configured using the adcsetup variable

```
let adcsetup =  %00000011
```

Using the ‘readadc’ command does not automatically configure the pin as an analogue input. Pins must also be set as inputs (not outputs) for the analogue input to work correctly. The analogue voltage range can be the PICAXE power supply range or an alternate external voltage range. In this case two analogue pins are used to set the positive and negative reference for the ADC.

Due to the large number of ADC pins now available, each ADC is given a unique ADC channel number. This is the number used in the readadc command (e.g. use ‘readadc 0,b1’ not ‘readadc A.0,b1’).

A new feature of the X2 range is the addition of 2 internal comparators (C1 and C2) which constantly compare two analogue values. The two values can be two external ADC pins, or one external ADC pin and an internally generated, configurable, voltage reference.

The comparator outputs are always available in the ‘compvalue’ variable. If desired the comparators can be setup to trigger a flag, which can then be used in a ‘setintflags’ interrupt routine. See the compsetup command for more details.

Some X2 parts also have an accurate internal voltage reference (1.2V or 1.024V), for use with monitoring battery powered projects. See the calibadc command for more details.

Low Voltage Operation

The 28X2 and 40X2 are also available in a special 3V version, which operates from 1.8V to 3.6V. Note that using 5V on the 3V version will permanently damage it!

The low voltage options are known as 28X2-3V and 40X2-3V. It is recommended that these 3V parts are only programmed via the AXE027 USB cable, as this uses logic level signals as opposed to RS232 (up to +/-12V) serial signals.

The 20X2 takes a slightly different approach. It has an internal 3.3V silicon die, but also contains an internal Low Drop Out Regulator, which is automatically enabled when required. This means the normal 20X2 can be used across the entire 1.8 to 5V voltage range.

Clock Frequency

Much higher clock rates are now available. This greatly improves the PICAXE processing speed.

The default power-up operating frequency is 8MHz, using the internal resonator. Therefore pause commands are now calibrated at 8MHz, not 4MHz. This also means the default sertxd and serrxd baud rate is now 9600,n,8,1.

The 20X2 has internal clock frequency options up to 64MHz – 16x faster than 4MHz!
The 28X2/40X2 has external clock frequency options up to 40MHz (internal 8MHz).
The 28X2-3V/40X2-3V has external clock frequency options up to 64MHz (internal 16MHz).

The external clock frequencies make use of an internal 4x Phased Lock Loop (PLL).This means the value of the external resonator is actual 0.25 of the final operating speed –for instance an 8MHz external resonator gives a 32MHz operating speed. Therefore if an existing design currently uses an 8MHz resonator the X2 operating speed will instantly be 32MHz without any hardware modification.

Variables

The X2 have a much larger RAM area, up to 1280 general user bytes.

Part	*General RAM*	*Scratchpad*
40X2	256	1024
28X2	256	1024
20X2	128	128

General RAM

On the X2 parts there are up to 256 general purpose variables. 56 of these, known as b0 to b55, can be used directly in any command (as with all other PICAXE parts).

All bytes (0-127 / 0-255) can also be addressed both directly and indirectly.

To directly address the values the peek (read the byte) and poke (write the byte) commands are used. Note that peek and poke are now dedicated to the general purpose variables, to read the microcontroller peripheral registers the new commands peeksfr and pokesfr are used.

To indirectly address the values the virtual variable name '@bptr' is used. @bptr is a variable name that can be used in any command (ie as where a 'b1' variable would be used). However the value of the variable is not fixed (as with b1), but will contain the current value of the byte currently 'pointed to' by the byte pointer (bptr).

The compiler also accepts '@bptrinc' (post increment) and '@bptrdec' (post decrement). Every time the '@bptrinc' variable name is used in a command the value of the byte pointer is automatically incremented by one (ie bptr = bptr+1 occurs automatically after the read/write of the value @bptr). This makes it ideal for storage of a single dimensional array of data.

Scratchpad RAM

On the X2 there are up to 1024 scratchpad bytes.

To directly address the scratchpad values the get (read the byte) and put (write the byte) commands are used. Note that the address in these cases must be a word variable, as there are now 1024 possible addresses.

To indirectly address the values the virtual variable name '@ptr' is used. @ptr is a variable name that can be used in any command (ie as where a 'b1' variable would be used). However the value of the variable is not fixed (as with b1) , but will contain the current value of the byte currently 'pointed to' by the pointer (ptr, which is now a word variable, made up of two bytes ptrl and ptrh).

The compiler also accepts '@ptrinc' (post increment) and '@ptrdec' (post decrement).

Understanding Program Slots

Each X2 program slot can contain approximately 1000 lines of BASIC code.

The X2 range have up to 4 internal program slots, numbered 0 to 3. Each slot is completely independent of the other slots. When the microcontroller is reset the program in slot 0 automatically starts running. The other programs can then be started by using a 'run' command.

A new program download is, by default, into slot 0. To download into another program slot the #slot directive must be used in the program, .e.g.

```
#slot 1
```

will download the program into slot 1 instead of slot 0. All other slots are unaffected.

Note that when the download is complete the program will always start running from slot 0, not the slot just downloaded. If you wish to instantly test, for instance, a program downloaded into slot 1, the command 'run 1' must have been previously downloaded into slot 0.

As the microcontroller only has one internal EEPROM data area (used by the EEPROM, read and write commands) any download into any internal memory slot will always update the same EEPROM memory. To disable this update it is possible to use a #no_data directive in the downloaded program. This prevents the EEPROM data area being updated (i.e. any EEPROM command data is ignored).

The usual way to make use of the program slots is to test an input (e.g. jumper link) upon reset, and then run the different program according to the input condition e.g.

```
#slot 0

if pinC.1 = 1 then
   run 1
endif
if pinC.2 = 1 then
   run 2
endif
```

However program slots can be combined into one 'long program' as long as the following points are noted:
1) No gosubs (including the interrupt) can be shared between program slots
2) The gosub/return stack is reset when moving from one slot to another
3) Outputs and variables/scratchpad are not reset
4) The 'run X' command should be regarded as 'goto to the start of program X'

Note that 'run 0' is not the same as the 'reset' command, as the reset command will also reset all variables and convert all pins back to inputs.

External Program Slots

As well as the internal memory slots, 4 additional slots can be used by connecting an external i2c EEPROM chip (part 24LC128 or 24LC256). As up to 8 different I2C chips could be used on the same I2C bus, this gives a theoretical 32 additional program slots.

Running a program from external i2c has some restrictions
1) The i2c bus is reserved exclusively for the program reading
2) The i2c pins cannot be used for any other purpose
3) Any hardware i2c/spi commands are completely ignored
4) Program execution speed is reduced, due to the relatively slow speed of reading data from the external 24LC128
5) The external 24LC128 only stores the program memory space. Any download data memory information (ie from the EEPROM command) is not stored externally. Read and write commands continue to act on the internal EEPROM data memory space.

Booti2c Command

The booti2c command can be used to copy a program from an external 24LC128 memory slot into an internal memory slot. The booti2c command is only processed if the program revision number (set by the #revision directive during download) in the 24LC128 memory slot is greater than the revision number currently in the internal program slot. This means that the program copying will only occur once after a new 24LC128 is fitted.

See: run, booti2c, #revision, #slot

Other new X2 features

UNI/O Support

The X2 support the uniin / uniout commands to communicate with Microchip UNI/O EEPROM memory chips.
See: uniin, uniout

Servo

The servo command operation has been fully updated and revised, and is now more accurate. The command now also has an optional timer1 preload value, which allows the 20ms refresh rate to be altered if desired. Servo now operates at 8 or 32 MHz.
See: servo, servopos

Doze Command

The new doze low-power command is similar to the sleep command, but maintains peripheral (e.g. timers and pwm) operation.
See: doze

Internal Pullups

Some pins have a weak internal pullup resistor that can be enabled via the 'pullup' command.
See: pullup

Hardware Serial Port

The higher clock frequencies mean higher baud rates are now possible via the hardware serial port. The hsersetup command also has an additional configuration bit to set the polarity of serial receives.
See: hsersetup

Additional Interrupt flags

Extra interrupt flags are now also generated on hardware interrupt pins and comparator change condition.
See: setintflags, hintsetup, compsetup

NOB Unary Operator

A new unary operator, Number of Bits, counts the number of '1's within a variable. This is useful when, for instance, adjusting brightness of multiplexed displays.
See: Unary Operators

Read Revision Version

This command reads the user set program revision (set using the #revision directive at download time) into a variable.
See: readrevision, #revision

Read Firmware / Silicon Version

Readfirmware command reads the PICAXE firmware version into a variable.
Readsilicon command reads the chip type and silicon revision number into a variable.
See: readfirmware, readsilicon

Timer 3

An additional internal timer, timer 3, can be used for background timing purposes.
See: tmr3setup

SRLatch

The SR latch is a hardware feature that can be used in the background to control the SRQ and SRNQ latch output pins. This can be used to instantly control the output pins, independent of program operation. The latch can also be used to generate a 555 timer style pulsing output.
See: srlatch, srset, srreset

Converting 28X / 28X1 programs to 28X2 programs.

The PICAXE Programming Editor and AXEpad software contain an automated wizard to help convert existing 28X / 28X1 programs into 28X2 format. The wizard is accessed via the PICAXE>Wizards menu. Please note that the wizard is only a tool that helps simplify the conversion task, no wizard can accurately translate every single program, and so all translated programs should be carefully checked and manually adjusted if/as required.

The four main stages in converting a 28X1 program to a 28X2 program are:

1) Add the following lines at the top of the program, to set portB as outputs and portA 0-3 as analogue.

```
let dirsB = %11111111      ; set portB pins as output
let adcsetup = 4           ; set ADC0-3 as analogue pins
setfreq m4                 ; set frequency to 4MHz
```

2) The setfreq command above changes the default speed (8MHz on X2) back to the 4MHz used on X1 parts. This will ensure all count, pulsin etc. type of commands operate as they did on the X1 part. However this does then mean that all pause / pauseus commands are half the value they should be, as on the 28X2 pause is calibrated at 8MHz, not 4MHz. Therefore it is necessary to double all pause values when running at 4MHz. A quick and easy way to do this when pause uses a variable value is to just repeat the pause line twice.

```
pause 1000                        pause 2000
pause w1                          pause w1 : pause w1
```

3) Rename all output pins. The default outputs on X / X1 parts are portB, e.g.

```
high 7                            high B.7
serout 1,n2400,(b6)               serout B.1,n2400,(b6)
```

4) Rename all inputs pins. The default inputs on X/X1 parts are portC, e.g.

```
if pin3 = 1 then                  if pinC.3 = 1 then
count 0,1000,w1                   count C.0, 1000, w1
```

See the conversion wizard datasheet for more detailed information about the conversion process.

Appendix 1 - X2 Variations

Most X2 commands are supported on all of the parts in the X2 range.

However different variants of the PICAXE-X2 range have slightly different features and memory size. This is due to variants in the base PIC microcontroller used to generate the PICAXE chip. It is not possible for the PICAXE firmware to change these differences as they are physical hardware features of the PIC silicon design.

Feature	PICAXE Command	20X2	28X2	28X2 (3V)	40X2	40X2 (3V)
Voltage Range (V)	-	1.8-5.5	4.2-5.5	1.8-3.6	4.2-5.5	1.8-3.6
Max. Internal Frequency (MHz)	Setfreq	64	8	16	8	16
Max. External Resonator (PLL) Frequency (MHz)	Setfreq	n/a	40 (32*)	64	40 (32*)	64
Default PICAXE Internal Frequency (MHz)	Setfreq	8	8	8	8	8
ADC setup (seq=sequential, ind=individual)	Adcsetup	Ind	Seq	Ind	Seq	Ind
Internal ADC reference (V)	Calibadc	Yes (1.02)	No	Yes (1.2)	No	Yes (1.2)
Variables RAM (bytes)	Peek/Poke @bptr	128	256	256	256	256
Scratchpad RAM (bytes)	Put/Get @ptr	128	1024	1024	1024	1024
Internal program slots External i2c program slots	Run	1 32	4 32	4 32	4 32	4 32
Number of hardware interrupt pins	Hintsetup	2	3	3	3	3
Pwmout channels	Pwmout	1	2	2	2	2
Hpwm command support	Hpwm	Yes	No	Yes	Yes	Yes
Power steering support in single hpwm mode	Hpwm	Yes	No	Yes	No	Yes
Pull-ups can be individually controlled	Pullup	Yes	No	Yes	No	Yes
Contains SR latch module	SRLatch	Yes	No	No	No	No
Base PIC Microcontroller (18F series)	-	14K22	2520	25K20	4520	45K20

** 32MHz (8MHz resonator with x4 PLL) is recommended for programs using serial commands as 40MHz is not an even multiple of 8 and so does not produce valid serial baud rates.*