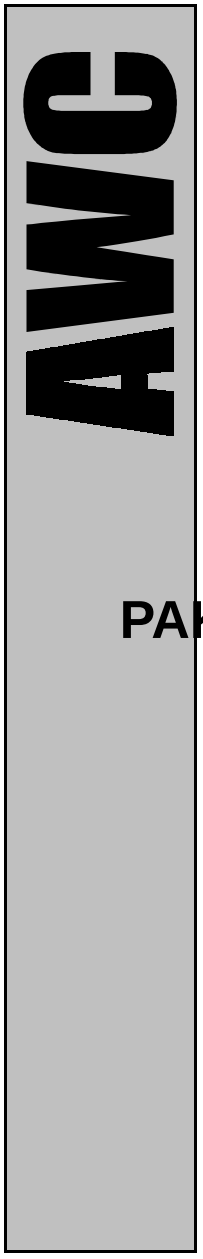


PAK-II Data Sheet

AWC
310 Ivy Glen Court
League City, TX 77573
(281) 334-4341
http://www.al-williams.com/awce.htm
V1.2

Overview

The Stamp Pak II is a 32-bit floating-point math coprocessor with a versatile interface that makes it easy to interface it with a variety of microcontrollers including the Basic Stamp from Parallax. However, you can easily interface the Pak II with practically any modern microprocessor. The Pak II can add, subtract, multiply, and divide numbers between +/- $6.80564693 \times 10^{38}$. The smallest number the Pak II can represent is +/- $1.17549435 \times 10^{-38}$. Here are some of the Pak II's major features:

- Robust, speed independent serial interface
- Uses as few as 2 pins to connect to the host
- Can connect multiple devices to the same 2 pins with one additional pin per device
- Can work with IEEE754 floating point numbers
- Works with Stamp II's ShiftOut and ShiftIn commands
- Synchronous operation – read results when you are ready for them
- Adds 16-bits of general-purpose digital I/O pins
- Fast 20Mhz speed (can be reduced to save power)
- Easy to use

The Pak II is a standard 28-pin .3” IC. In order to operate it must have a regulated supply of 5V and

connection to a clock element. The Stamp Pak II includes a 20MHz ceramic resonator that you can use to clock the chip.

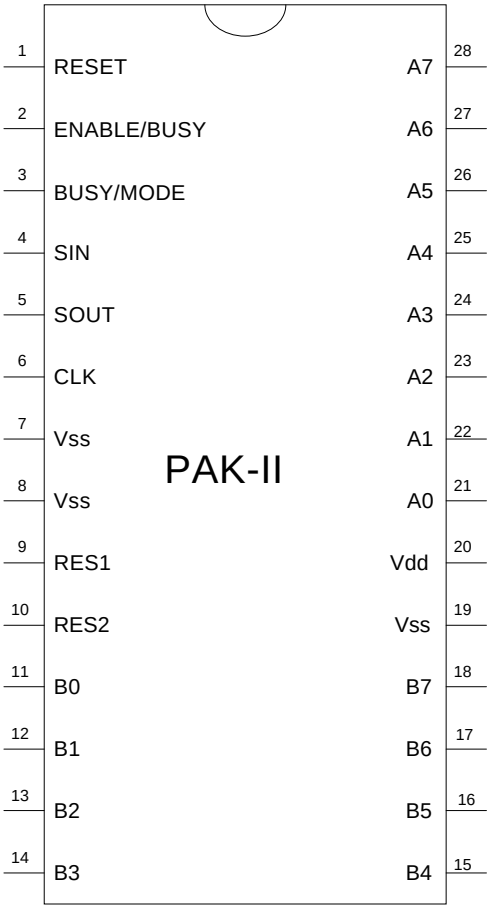
Other than the power and clock connections, the Pak II requires at least two wires to connect between your microcontroller (the host) and itself. Depending on the host's capabilities and your application, you may want to use more pins for I/O (up to 5 pins).

If You Need Help

If you require assistance with your PAK-I,I please feel free to contact us. The best way to get support is via e-mail (stamp@al-williams.com). However, you may also call between 9AM - 4PM Central Time at (281) 334-4341. You can also fax to (281) 538-2147. Be sure to check out our Web page for updates at www.al-williams.com/awce.htm.

Registering Your PAK-II

Please take a moment to register your e-mail address with AWC. Simply send an e-mail message to pakreg@al-williams.com. AWC will not make your address available to other companies, but we may periodically send you updated technical notes. You'll also receive information about new microcontroller products


PAK-II
1 RESET
2 ENABLE/BUSY
3 BUSY/MODE
4 SIN
5 SOUT
6 CLK
7 Vss
8 Vss
9 RES1
10 RES2
11 B0
12 B1
13 B2
14 B3
28 A7
27 A6
26 A5
25 A4
24 A3
23 A2
22 A1
21 A0
20 Vdd
19 Vss
18 B7
17 B6
16 B5
15 B4

and specials.

Pin	Name	Type	Description
3	Busy/Mode	Input/Output	This pin has two functions. At start up, if the pin is pulled up by a resistor, the PAK1 uses this pin as a busy indicator. The pin will read 0 when the PAK is not busy. However, if at startup the pin is pulled to ground, the PAK will indicate its busy status on the Enable/ Busy pin.
2	Enable/ Busy	Input or Open Collector Output	If Busy/Mode is pulled high at start up, the Enable/Busy pin is only an input. Bring it low to disable the PAK, or high to enable it. However, if Busy/Mode is pulled low at start up, this pin serves as an open-collector I/O pin. If the host pulls the pin low, it disables the PAK. If the PAK is busy, it pulls the pin low.
6	CLK	Input	The host pulses the CLK pin to shift data in or out of the PAK II.
4	SIN	Input	The PAK II reads data from this pin. You may short SIN and SOUT and use the same pin for input and output if the host is capable of treating a single pin as an I/O pin.
5	SOUT	Open Collector Output	The PAK II writes data to the host on this pin. Since the pin is open collector, you'll need a resistor to 5V on this pin. You may short SIN and SOUT together to use one pin for I/O
1	RESET	Input	Hardware resets the PAK II when low.
7/8/ 19	VSS	Power	Ground - please connect all ground pins.
20	VDD	Power	+5V
9	RES1	Clock	Connects to resonator
10	RES2	Clock	Connects to resonator
21- 28	A0-A7	I/O	General purpose I/O
11- 18	B0-B7	I/O	General purpose I/O

Pin Connections

Code	Name	Alternate	Argument	Returns	Description
$01	LOADX	N/A	FP	Nothing	Load X
$02	LOADY	N/A	FP	Nothing	Load Y
$03	READX	N/A	None	FP	Read X
$04	SWAP	N/A	None	Nothing	Swap X and Y
$05	DIGIT	N/A	DIGIT	ASCIIDIGIT	Find ASCII digit from X
$06	IEEECVT	N/A	None	Nothing	Convert X to IEEE format
$07	FLOAT	$87	None	STATUS	24-bit integer in X to floating point
$08	COMMCK	N/A	None	$2A	Communications check
$09	FPCVT	N/A	None	Nothing	Convert X from IEEE format
$0A	CHS	N/A	None	Nothing	X=-X
$0B	INT	$8B	None	STATUS	Convert X to 24-bit integer
$0C	MUL	$8C	None	STATUS	X=X*Y
$0D	DIV	$8D	None	STATUS	X=X/Y
$0E	SUB	$8E	None	STATUS	X=X-Y
$0F	ADD	$8F	None	STATUS	X=X+Y
$10	OPT	N/A	OPTS	Nothing	Sets option bits
$11	ABS	N/A	None	Nothing	X=|X|
$12	STO	N/A	REG	Nothing	REG=X
$13	RCL	N/A	REG	Nothing	X=REG
$14/$94	DIRA/B	N/A	DIRS	Nothing	Sets direction register
$15/$95	RIOA/B	N/A	None	IBYTE	Reads input byte from port
$16/$96	WIOA/B	N/A	OBYTE	Nothing	Writes byte to output port
$17	XTOY	N/A	None	Nothing	X=Y
$18	YTOX	N/A	None	Nothing	Y=X
$19	SQRT	$99	None	STATUS	X=SquareRoot(X)
$1A	LOG	$9A	None	STATUS	X=Ln(X)
$1B	LOG10	$9B	None	STATUS	X=Log(X)
$1C	EXP	$9C	None	STATUS	X=e**X
$1D	EXP10	$9D	None	STATUS	X=10**X
$1E	POW	$9E	None	STATUS	X=X**Y
$1F	ROOT	$9F	None	STATUS	X=Yth Root of X
$20	RECIP	$A0	None	STATUS	X=1/X
$21/$24	SIN/ASIN	$A1/$A4	None	STATUS	X=sin(X) (radians)
$22/$25	COS/ACOS	$A2/$A5	None	STATUS	X=cos(X) (radians)
$23/$26	TAN/ATAN	$A3/$A6	None	STATUS	X=tan(X) (radians)

Software Commands

Notes and Key:

- FP – 32 bit floating point number (see text for format)
- DIGIT – A digit number. Zero returns +, -, or space if the number is positive, negative, or zero. See reference for more details.
- ASCIIDIGIT – The ASCII representation of the requested digit
- STATUS – Floating point error flags
- OPTS – Options ($80 turns on saturation; $40 enables rounding)

- RREG – Temporary storage registers (0-23)
- DIRS – Byte indicating port directions (0=input, 1=output)
- IBYTE, OBYTE – Input or output byte
- Using the alternate form of a command prevents it from returning the status result.

Floating Point Numbers and Errors

The 32-bit floating point representation used by the Pak II is a modified version of the IEEE754 format. The first byte is the exponent (biased by 127). If the exponent is zero, the number is zero. The second byte contains the most significant bits of the mantissa. The most-significant bit is the sign bit (1 is negative) and the mantissa starts with an implied 1 before the decimal point. The next two bytes are the remainder of the mantissa.

This differs from the IEEE745 format (the one used by PC C compilers, for example) only slightly. In the IEEE745 format, the sign is in the most-significant bit of the exponent. The most-significant bit of the first byte is actually the least significant bit of the exponent. Otherwise, the two formats are identical. The Pak II has functions to convert back and forth between the two formats.

The accompanying disk contains a Windows program (fconvert.exe) that will convert between a normal floating point number, the Pak II's format, and the IEEE745 format. Simply enter the number you know and the program will display the other two formats.

The status return bits contain the following information:
Bit 0 – Integer overflow
Bit 1 – Floating point overflow
Bit 2 – Floating point underflow
Bit 3 – Divide by zero
Bit 4 – Not a number error
Bit 5 – Domain error
Bits 6-7 – Reserved; always 0.

A return of zero indicates no error.

Resetting

There are several ways you can reset the Pak II. It is a good idea to reset the unit before using it, or any time that you want to make sure it is in a known state. This is especially true when using the Stamp. Each time the Stamp resets or wakes up from sleep, the I/O pins briefly become inputs. This can fool the Pak II into starting a data transfer. Always reset the Pak II first.

The best way to reset a single Pak II is to send a special reset sequence over the clock and data lines. This has the advantage that it doesn't require any extra I/O from the host. To send a reset sequence, bring the data pin to 0 and raise the clock to a 1. While the clock remains in the 1 state, bring the data pin high. This will indicate to the Pak II that you wish to reset. When the clock returns to a zero state, the Pak II will reset. The reset doesn't change any register values or port pins, but it does reset communications to a known

state.

You can get the same result by bringing the enable pin low and then returning it to high to enable the Pak II. This is useful if you are connecting more than one Pak to the same data and clock lines. You'll need to use the enable pin then anyway, and it makes sure that the selected Pak is always in a known state. Using the data sequence method may be unreliable when using multiple Paks together.

Finally, you can force a hardware reset by bringing the reset pin low. This might be useful if your circuit generates a hardware reset signal based on a brown-out detector or other master reset circuit. Normally, you'll just connect the reset pin to the +5V supply and allow the Pak II to reset itself on power up. If you do want to drive this pin, make sure that it is at 5V for normal operation. You can use a reset switch or other device if you pull up the reset pin with a 10K-22K resistor.

Communications

There are several schemes you can use to communicate with the Pak II. All of them revolve around a synchronous protocol involving a clock pin and 1 or 2 data pins. Data is shifted most significant bit first, and samples at the rising edge of the clock. All signals are positive logic (that is, a 1 is a high logic level). The Pak II exposes a separate input (SIN) and output (SOUT) pins for hosts that can't easily handle bi-directional I/O lines. However, for hosts like the Stamp or PIC, it

is a simple matter to tie these lines together since SOUT is open collector.

Therefore, the minimum number of lines you need are two. An output for the clock and an I/O line to connect to SIN and SOUT. You'll connect the Enable/Busy pin to +5V and use a pull up resistor to 5V on the Busy/Mode pin. This allows you to reset the device and communicate with it. The only problem is, you will have to make sure not to ask the device for data while it is processing. Most operations are quite fast, but some operations (calculating digits and division, for example) can take as much as 200uS with the provided components.

For some applications, this isn't a problem. For example, a standard Basic Stamp II can't operate fast enough for this to be a problem except for division and digit calculations. However, you may want to know when the device is ready to process. There are two ways you can accomplish this.

If you need to disable and enable the device, you'll want to drive the Enable/Busy pin. Using this pin allows you to share the SIN, SOUT, and CLK lines with other devices that use a similar protocol. So to connect 5 Paks you'd only need 7 I/O lines (5 enables, 1 SIN/SOUT, and 1 CLK line). You drive the Enable/Busy pin low to disable the device. If you don't plan to use this capability, just pull the pin high with a pull up resistor. If you are using the Busy/Mode input (see below) you can still use

a pull up resistor, or just tie the Enable/Busy pin directly to +5V.

If you ground the Busy/Mode input at start up, the device will also use the Enable/Busy pin to tell you when it is unavailable. It does this by driving the line low when it is not ready. That means the host must drive the pin with an open collector output if it wants to disable the device. If you don't need to disable the device, you can just connect the pin to a pull up resistor and connect the Enable/ Busy pin to an input on the host.

If you want to use a normal output to drive the Enable/Busy pin, but you still want a busy indication, you can connect the Busy/Mode pin to 5V through a pull up resistor. Then the Pak II will use the Busy/Mode pin to signify it is busy (the pin is high when busy). In this case, the Enable/Busy pin is always an input and you can drive it normally.

Many commands have an alternate form. These commands are the ones that return the floating point error flags on completion. Using the alternate form suppresses the status return value.

One thing to note about the status returns. All of the commands that take any significant amount of time will return zeros in the top bit of the response. You can use this as a form of busy indication if you are unwilling to use any other pins. Simply read the data pin until it reads zero. This indicates

that the device is sending the response. The READX and RIOA/B commands do not follow this convention, but they take very little time to execute anyway (less than 4uS).

To summarize, here are the ways you can synchronize with the Pak II:
1. Delay long enough to insure the operation is complete

2. Monitor the Enable/Busy line while Busy/Mode is grounded. If you bring Enable/Busy low it indicates that the Pak II is disabled. If the Pak II brings the pin low, it means it is engaged in operations.

3. Place a pull up resistor on the Busy/Mode pin. This will cause the Busy/Mode pin to be high when the Pak II is busy.

4. On commands that return status or ASCII digits, you can wait until the data pin goes low.

Command Reference

ABS

Finds the absolute value of X. That is if X is negative, negate it. If X is zero or positive, leave it alone.

ADD

Adds X and Y leaving the result in X. Y is unchanged. This command may take

approximately 55uS in the worst case.

CHS

Negates the X register.

COMMCK

This command causes the Pak II to issue a $2A response. You can use this to check communications. The Pak II will always respond with $28 plus the version number.

COS/ACOS

Sets the X register to the cosine/arccosine of X (in radians).

DIGIT

The DIGIT command accepts a single byte as an argument and returns a single byte. If the input byte is a zero, DIGIT returns an ASCII +, and ASCII – or a blank character depending on if the X register is positive, negative, or zero. You can use this as a SGN function, but it is mostly useful when writing numbers in readable form.

If the input number is less than $7F, DIGIT returns the ASCII digit from the X register that you ask for. For example, if X contains 13.141 and you issue a DIGIT 1 command, the return will be '3'. DIGIT 2 will return

'1'.

Numbers greater than $80 return ASCII characters to the right of the decimal point. So using the same example, if you issue a DIGIT $81, the return will be '1'. DIGIT $82 will return '4'.

This command does not affect the X or Y registers. It may take some time to execute depending on the digit you request, and the number in the X register.

This command performs a complex algorithm which involves converting the number to an integer and back to a float in several places. Since integer to float conversion is limited to 24 bits, this places an upper limit on the size of the number you can convert with DIGIT. Note that this does not apply to using it as a replacement for SGN. If you operate on a large number (or a number with many significant digits) and an intermediate result exceeds 24 bits, you may get an incorrect answer, or the PAK may fail to converge.

DIRA/B

These commands allow you to set the directions of the two 8 bit I/O ports. On power up, all pins are inputs. Setting a bit to 1 in the DIR command causes the

corresponding I/O bit to be an output. So issuing a DIRA $83 will make A7, A1, and A0 outputs and all other pins inputs.

DIV

Divides X by Y leaving the result in X. Y is unchanged. This command may take approximately 190uS in the worst case.

EXP, EXP10

These commands raise e (or 10) to the power of X. This is the inverse of the LOG and LOG10 commands.

FLOAT

The float command converts the 24-bit integer in the X register's mantissa into a floating point number ready for calculations.

IEEECVT, FPCVT

These commands convert the X register to IEEE format, or to the Pak II's internal format. If you load an IEEE number, you must convert it before using it in calculations. Once you convert to IEEE, you must convert back before using the number in any calculations. These operations are very fast, so it isn't a problem to convert several times.

INT

Converts the X register to a 24-bit integer.

To read the integer, issue a READX. In order to continue calculations with the same number, you'll need to issue a FLOAT command, or reload the X register. It is often quicker to use a STO command to save the X register, perform the INT command, read the result, and then RCL the original contents.

LOADX, LOADY

These commands take the next four bytes you send and places them in the X or Y register. The format is exponent first, followed by the mantissa (most significant bit first). You can also load integers or IEEE format numbers for conversion using this command.

LOG, LOG10

These commands find the natural or common logarithm of X.

MUL

Multiplies X and Y leaving the result in X. Y is unchanged. This command may take approximately 120uS in the worst case.

OPT

Sets options. Send the option byte after this command. Using $80 turns on saturation (which represents over and underflows as minimum and maximum values). Using $40 enables rounding. If this option is not

on, results are truncated which is faster, but less accurate. On a power up reset, both flags are off.

POW

Use the POW command to compute a power. For example, to find 33 cubed, set X=33 and Y=3 then execute POW.

RCL

Recalls the specified register to X. The register is unchanged. Registers range from 0 to 23.

READX

This command sends the four bytes that comprise the X register to the host. The host must clock in 4 bytes. The first byte will be the exponent, and the following three bytes the mantissa (MSB first). Note that if you execute an IEEECVT command the result will be in IEEE format. If you execute an INT command, the exponent will be meaningless, and the mantissa will be the 24 bit binary integer result.

RECIP

Computes 1/X. The Y register is unchanged.

RIOA/B

This command reads 8 bits from the I/O port. Of course, bits set to outputs will read whatever you are writing to them. The Pak sends the byte immediately.

ROOT

Use this command to calculate roots. For example, to find the cube root of 10, store 10 in X, 3 in Y, and use the ROOT command. The SQRT command is much faster for square roots.

SIN/ASIN

Sets the X register to the sine/arcsine of X (in radians).

SQRT

This command calculates the square root of X.

STO

Stores X in the specified register (0-23).

SUB

Computes X-Y leaving the result in X. Y is unchanged. This command may take approximately 55uS in the worst case.

SWAP

Swaps X and Y

TAN/ATAN

Sets the X register to the tangent/arctangent of X (in radians).

WIOA/B

This command writes 8 bits to the I/O port. This only affects pins that are set as outputs. Send the byte immediately after the command.

XTOY

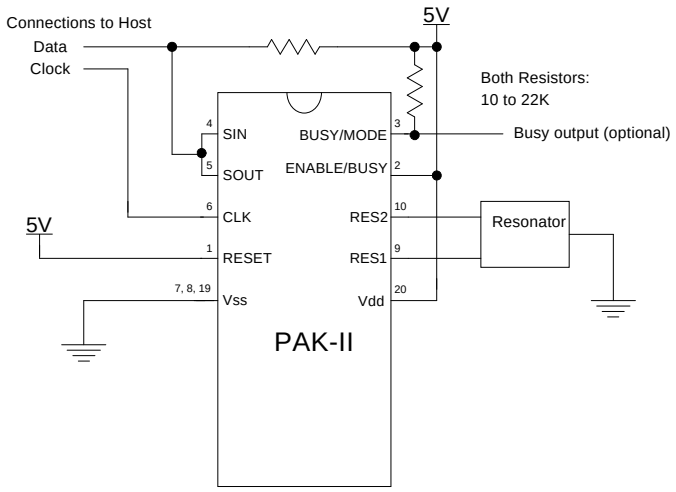
Stores X in Y without changing X.

YTOX

Stores Y in X without changing Y.

Typical Circuits and Software

Connecting the Pak-II to its external components is simple. Supply 5V to Vdd and ground the Vss pins. Connect the two outer pins of the ceramic resonator to the RES1 and RES2 pins (the order does not matter). Ground the center pin of the resonator. Connect the RESET pin to 5V. For the simplest mode of operation connect the ENABLE/BUSY pin to 5V and the BUSY/MODE pin to 5V through a pull up resistor (10K-22K). Of course, you'll also need to make connections to the host microprocessor. You may also want to make different connections to ENABLE/BUSY and BUSY/MODE if you want to use different methods of synchronizing.



Example 1. Connections to Stamp or similar host

If you are using the Basic Stamp II or Basic Stamp IISX, you'll have no problems using the ShiftIn and ShiftOut commands. Many Basic Stamp-compatible compilers have these commands too. If you don't have access to these commands, don't worry. Here is some simple code (written in simple PBasic) that shows the actions you have to take:

```
' B1 is the byte to shift
shiftoutput:
output datapin
for b7=0 to 7
' Set data pin to 0 or 1
    low datapin
    if (b1 & $80) <> $80 then so0
    high datapin
so0:
    b1=b1*2         ' shift byte left
```

```
    high clkpin
    low clkpin      ' could use pulsout
next
return

' B1 is byte to input byte
shiftinput:
  input datapin
  b1=0
  for b7=0 to 7
   b1=b1*2
   b1=b1 | in15
   high clkpin
   low clkpin       ' could use pulsout
  next
return
```

You'll find a library of Basic Stamp II routines that allow you to access the Pak II on the companion disk. Here is a simple program that counts from 1 to 1000 while blinking an LED connected to the Pak II's A4 pin:

```
i var word
gosub FReset    ' reset

fpx=$10           ' set B4 to output
gosub IODir

gosub FZeroX    ' x=0
gosub FSwap     ' x<>y
fpx=1
gosub FLoadInt  ' x=1.0
gosub FSwap     ' x<>y  so x=0.0, y=1.0
Debug "Start...", CR
for i=1 to 1000
  fpx=i//2*$10
  gosub IOWrite  ' Blink light
  gosub FAdd    ' x=x+y
  fpx=4
  gosub FDump   ' print it out
next
Debug "End",cr
```

Here is the same program, but instead of loading 1 as an integer, it loads .5:

```
i var word
gosub FReset    ' reset
fpx=$10
gosub IODir

gosub FZeroX    ' x=0
gosub fSwap     ' x<>y
fpxhigh=$7e00   ' x=.5
fpxlow=0
gosub FLoadX
fpx=3
gosub FDump
gosub FSwap     ' x<>y  so x=0.0, y=0.5
Debug "Start...",CR
for i=1 to 1000
  fpx=i//2*$10
  gosub IOWrite
  gosub FAdd    ' x=x+y
  fpx=4
  gosub FDump
next
Debug "End",cr
```

Frequently Asked Questions

Q: How can I load a particular number to the X or Y registers?

A: There are two methods. First, you can use the fpconvert program to determine the 32-bit number that corresponds to the number you want. Then

pass it to the FLoadX (or FLoadY) subroutine. You can also use the LOADX or LOADY command directly. The other way you can load a number is via the FLOAT or FLoadInt commands.

Q: Can I run the Pak II from a different clock source?

A: Yes. Any crystal or ceramic resonator up to 20Mhz may work. Remember, the ceramic resonator has capacitors that you will have to supply if you use a crystal or a resonator without capacitors. You'll use two capacitors (one on each RES pin). You'll have to experimentally determine the values. Start with about 15pf and work up to as much as 100pf for very low frequency crystals. You can also feed an existing TTL-level clock signal into pin 16.

Q: Will running the Pak II at a lower speed reduce its power consumption? Will it run faster?

A: Yes running at slower speeds will save power. You will probably not have success overclocking the Pak II. However, contact AWC for information on faster devices and other customizations.

Q: Why are my results backwards?

A: Make sure you are using MSBPRE with the ShiftIn command and not MSBFIRST. Use MSBFIRST with the ShiftOut command.

Q: Is there a second source for the Pak II?

A: No. However, if you have a high-volume application and you are concerned about availability, contact AWC about obtaining a license to produce your own Pak II's or obtain them from third parties.

Q: How can I compare two numbers?
A: To compare X and Y, store X away (using one of the STO commands). Subtract Y from X and then call DIGIT 0 to determine if the result is positive, negative, or zero (this operation is very fast). Then restore X. Of course, if you don't mind destroying X, you can skip the STO and RCL.

Q: Can I use more than one Pak at one time?

A: Yes. You can drive multiple Paks on the same data and clock lines as long as you only enable one at a time. Of course, you could keep them always enabled if you use separate data and clock lines. You can use a resonator for each, or you can use a single resonator and feed pin 15 of the Pak with the resonator to pin 16 of the other Paks (this will only work for a small number of Paks).

Basic Stamp II Library

On the enclosed disk you'll find FUTIL.BS2 which contains a simple library for the PAK-II.

The calls in that file appear on the next page.

Call	Function	Notes
FReset	Reset the coprocessor	Always call first
FAdd, FSub, FMult, FDiv, FLog, Flog10, FSqrt, FRecip, FExp, FExp10, FPow, FRoot, FSin, FCos, FTan, FArcSin, FArcCos, FArcTan	Basic operations	Operate on X and Y
FSquare	Square X	
FChs	Change sign	
FAbs	Absolute value	
FSto, FRcl	Store and recall scratchpad	fpx=register number
FLoadX, FLoadY	Load values to X and Y	Place hex numbers in fpxhigh and fpxlow
FZeroX, FZeroY	Load 0 to X or Y	
FPI	Load X with pi	
Fe	Load X with e	
FLoadInt	Load integer to X	Place integer in fpx
FReadX	Read X	Places result in fpxhigh and fpxlow
FInt	Read X as integer	Leaves result in fpxhigh and fpxlow
FSwap	Swap X and Y	
FGetDigit	Decode an ASCII digit or compute signum	Place digit number if fpx, returns in fpdigit
FDump	Dump number on debug terminal	Set fpx to number of digits before decimal point
FOption	Set options	Options in fpx
FXtoY	Y=X	
FYtoX	X=Y	
IODir/IODir1	Set I/O direction register (A/B)	Directions bits in fpx
IOWrite/IOWrite1	Write to I/O port (A/B)	Data out in fpx
IORead/IORead1	Read from I/O port (A/B)	Returns byte in fpx

Specifications

Absolute Maximum Ratings

Parameter	Minimum	Typical	Maximum
Supply voltage	4.5V	5V	5.5V
Vdd rise time on power up	.05V/ms	-	-
Supply current	-	10mA	20mA

DC Characteristics (at 20MHz)

Ambient temperature under bias	-55°C to +125° C
Storage temperature	-65°C to +150° C
Voltage on VDD with respect to VSS	-0.3 to +7.5V
Total power dissipation	1000 mW
Maximum current out of VSS pin	300 mA
Maximum current into VDD pin	250 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sink/source by A0-A7 combined	200 mA
Maximum current sink/source by B0-B7 combined	200 mA

Notes

When configured as inputs, all of the general-purpose I/O pins on Port A have weak pull up resistors (about 20K) internally configured.