

**AWC**

**PAK-IX Floating  
Point A/D  
Data Sheet**

© 1998-2003 by AWC

AWC  
310 Ivy Glen  
League City, TX 77573  
(281) 334-4341

<http://www.al-williams.com/awce.htm>

V1.2 30 Aug 2003



# Table of Contents

Overview .....	1
If You Need Help .....	2
Registering Your PAK-IX .....	2
Pin Connections .....	4
Software Commands .....	5
Floating Point Numbers and Errors.....	6
Resetting .....	7
Communications .....	8
Command Reference .....	10
A2D .....	10
A2DCONF .....	10
ABS .....	10
ADD.....	10
CHS .....	11
COMMCK .....	11
COS/ACOS .....	11
DIGIT .....	11
DIRA/B.....	12
DIV .....	12
ERCL/ESTO .....	12
EXP, EXP10.....	12
FLOAT.....	13
IEEECVT, FPCVT .....	13
INT.....	13
LOADX, LOAY .....	13
LOG, LOG10 .....	13
MUL .....	13
OPT.....	14
POLY .....	14
POW .....	14
RCL .....	14
READX.....	14
RECIP.....	15
RIOA/B.....	15
ROOT .....	15
SIN/ASIN.....	15

SQRT .....	15
STO.....	15
SUB .....	15
S WAP .....	15
TAN/ATAN .....	16
WIOA/B.....	16
XTOY .....	16
YTOX .....	16
Typical Circuits and Software.....	16
Frequently Asked Questions .....	19
Q: How can I load a particular number to the X or Y registers? .....	19
Q: Can I run the PAK IX from a different clock source?.....	19
Q: Will running the PAK IX at a lower speed reduce its power consumption? Will it run faster? .....	20
Q: Why are my results backwards? .....	20
Q: Is there a second source for the PAK IX? .....	20
Q: How can I compare two numbers? .....	20
Q: Can I use more than one PAK at one time? .....	20
Basic Stamp II Library .....	21
Online .....	21
Other Processors.....	22
Specifications .....	24
Absolute Maximum Ratings .....	24

## Overview

The PAK IX is a 32-bit floating-point math coprocessor and analog to digital converter 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 IX with practically any modern microprocessor. The PAK IX can add, subtract, multiply, and divide numbers between +/-  $6.80564693 \times 10^{38}$ . The smallest number the PAK IX can represent is +/-  $1.17549435 \times 10^{-38}$ . Here are some of the PAK IXs 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
- Five 10-bit A/D channels
- Automatic analog input averaging
- Adds 8-bits of general-purpose digital I/O pins
- Fast 20Mhz speed (can be reduced to save power)
- Easy to use

The PAK IX 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 PAK IX includes a 20MHz ceramic resonator that you can use to clock the chip.

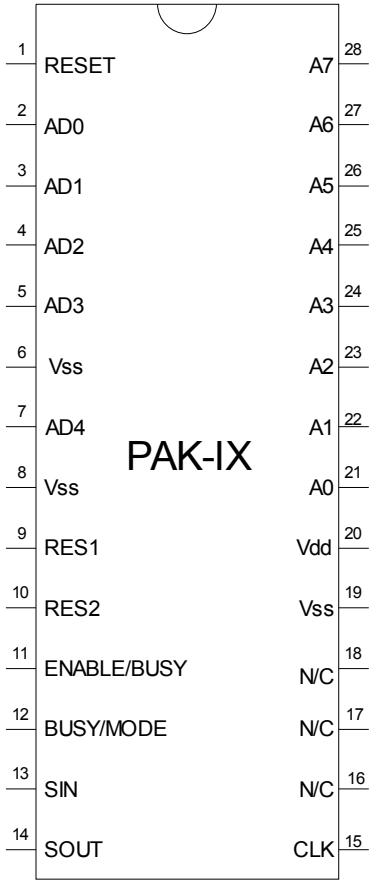
Other than the power and clock connections, the PAK IX requires at least two wires to connect between your microcontroller (the host) and itself. Depending on the host's capabilities and your application, you may want to use more pins for I/O (up to 5 pins).

## **If You Need Help**

If you require assistance with your PAK IX, please feel free to contact us. The best way to get support is via e-mail ([stamp@al-williams.com](mailto:stamp@al-williams.com)). However, you may also call between 9AM - 4PM Central Time at (281) 334-4341. You can also fax to (281) 754-4462. Be sure to check out our Web page for updates at [www.al-williams.com/awce.htm](http://www.al-williams.com/awce.htm).

## **Registering Your PAK-IX**

Please take a moment to register your e-mail address with AWC. Simply send an e-mail message to [pakreg@al-williams.com](mailto: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 and specials.



## Pin Connections

Pin	Name	Type	Description
12	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.
11	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.
15	CLK	Input	The host pulses the CLK pin to shift data in or out of the PAK II.
13	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.
14	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.
6/8/ 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
2-5, 7	AD0-AD4	Analog	Analog inputs



# Software Commands

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	ASCIIIDIG	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	\$41	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	DIRA	N/A	DIRS	Nothing	Sets direction register
\$15	RIOA	N/A	None	IBYTE	Reads input byte from port
\$16/	WIOA	N/A	OBYTE	Nothing	Writes byte to output port
\$17	XTOY	N/A	None	Nothing	Y=X
\$18	YTOX	N/A	None	Nothing	X=Y
\$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)
\$27	POLY	\$A7	COEF	STATUS	Compute polynomial
\$28	A2DCONF	N/A	ADOPT	Nothing	Sets A/D options
\$29	A2D	\$A9	ADSEL	STATUS	Reads A/D channel
\$2A/\$2B	ESTO/ERCL	\$AA/\$AB	None	0	Stores or recalls all registers to EEPROM

## Notes and Key:

- FP – 32 bit floating point number (see text)
- DIGIT – A digit number. Zero returns +, -, or space if the number is positive, negative, or zero (see text)
- ASCIIIDIG – The ASCII representation of the requested digit
- DIRS – Byte indicating port directions (0=input, 1=output)
- ADOPT - Byte for A/D options (see text)
- ADSEL - A/D channel and samples
- STATUS – Floating point error flags
- OPTS – Options ( \$80 turns on saturation; \$40 enables rounding)
- RREG – Temporary storage registers (0-23)
- IBYTE, OBYTE – Input or output byte
- COEF– Polynomial degree and starting coefficient (see text)
- 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 IX 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 IX 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 IXs 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 IX. 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 IX into starting a data transfer. Always reset the PAK IX first.

The best way to reset a single PAK IX 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 IX that you wish to reset. When the clock returns to a zero state, the PAK IX 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 IX. 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 IX 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 IX. 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 IX 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 IX 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 IX:

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 IX is disabled. If the PAK IX 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 IX is busy.

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

## **Command Reference**

### ***A2D***

Reads and averages a specified number of analog samples on a channel. Send a byte where the top 4 bits indicate the number of samples-1 (so 0 is 1 sample; %1111 is 16 samples). The bottom 4 bits indicate the channel number. So to read 10 samples on channel 2, send \$92. The result is left in the X register. The status returned is the result of the floating point operations.

### ***A2DCONF***

Sets A/D options. To set the default, send a zero byte following this command. To use AD3 as the positive voltage reference, send 1. To use AD3 as the positive reference and AD2 as the ground reference, send 8. No return value.

### ***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 IX to issue a \$41 response. You can use this to check communications.

## **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.

### ***ERCL/ESTO***

When the PAK resets, it automatically reads the data registers (the registers accessed with STO and RCL) from EEPROM. You can save the registers at any time using ESTO or restore them from EEPROM with ERCL. These commands access EEPROM, so you should wait for the status return to avoid over running the PAK. In addition, note that EEPROM has a limited number of writes it will accept (approximately 1,000,000 write cycles).

### ***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.

## **POLY**

This command requires you to send a single byte. The lower five bits indicates which register (from 0-23) holds the first coefficient for the polynomial. The top three bits contains the degree of the polynomial (less 1, so that 0 is a first degree polynomial). Say you store .5 in register 0 and 10 in register 1. You can then issue a POLY command with an argument of 0 (degree 1, register 0) to computer  $.5X+10$  (handy for  $y=mX+b$  computations). If you put, for example, -3 in register 2 and issued a POLY command with argument 00100000 (in binary), you'd compute  $X=.5X**2 + 10X - 3$ .

## **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. to save the X register, perform the INT command, read the result, and then RCL the original contents.

## **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**

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.

### **S WAP**

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**

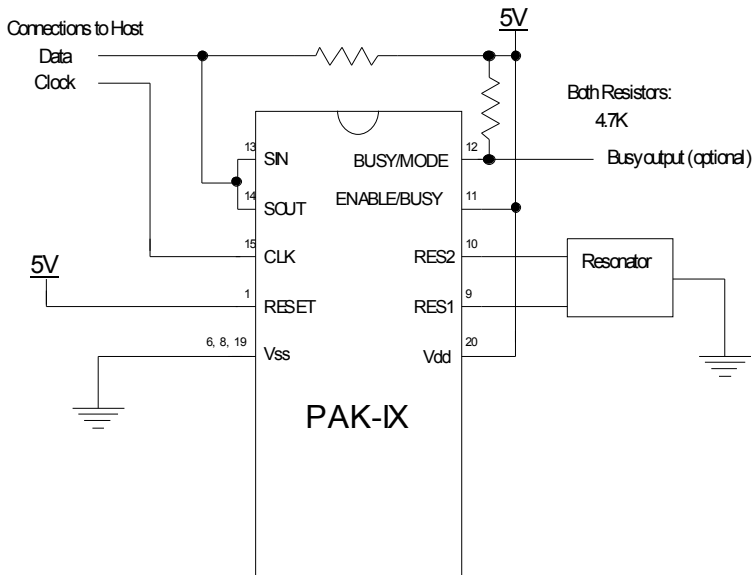
Stores X in Y without changing X.

## **YTOX**

Stores Y in X without changing Y.

## **Typical Circuits and Software**

Connecting the PAK-IX 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: Sample connection to Basic Stamp

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:

```
' iobyte is the byte to shift in or out
datapin con 15
datainput var in15
clkpin con 14
i var byte
iobyte var byte
shiftoutput:
output datapin
for i=0 to 7
' Set data pin to 0 or 1
  low datapin
  if (iobyte & $80) <> $80 then so0
```

```

    high datapin
so0:
    iobyte=iobyte*2 ' shift byte left
    high clkpin
    low clkpin      ' could use pulsout
next
return

shiftinput:
    input datapin
    iobyte=0
    for i=0 to 7
        iobyte=iobyte*2 ' shift left
        iobyte=iobyte | datainput
        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 IX on the companion disk. Here is a simple program that counts from 1 to 1000 while blinking an LED connected to the PAK IX'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 fconvert 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 IX 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 9.

***Q: Will running the PAK IX 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 IX. However, contact AWC for information on faster devices and other customizations.

***Q: Why are my results backwards?***

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

***Q: Is there a second source for the PAK IX?***

**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 IXs 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 (using RCL). 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 PAK9LIB.BS2 which contains a simple library for the PAK-IX. The calls in that file appear on the next page.

For some applications, you can use this file as-is. However, in many cases you'll want to trim away the parts you are not using, and you may want to customize some of the routines.

At the top of the file are constants (datap, datapin, and clock) that define the pins used to connect the PAK-IX. You'll also find a short example program near the top of the file.

## **Online**

Be sure to check out the online examples at:

<http://www.al-williams.com/awce/doclib.htm>.

Although many of the examples and documents are for the PAK-I or PAK-II, the PAK-IX will also work with them. You'll also find code for interfacing the PAK-I (and therefore, the PAK-IX) to other processors including the 68HC11, AVR, PIC, SX, and other processors.

## Other Processors

You can find code to drive the PAK using many different processors on our Web site at <http://www.alwilliams.com/awce/doclib.htm>.

In general, interfacing a PAK to a new host will require 4 routines:

1. A routine to reset the PAK
2. A routine to send a byte to the PAK
3. A routine to receive a byte from the PAK
4. A routine to wait for a ready condition (may be an optional part of the receive code)

The host connects to the PAK with a clock line (always an output) and either 1 or 2 data lines. If your processor can't support a line that is sometimes an input and sometimes an output, you'll need to use 2 data lines. If you can switch a pin between input and output you can use a single pin and connect it to both the SIN and SOUT pins on the PAK. In any event, SOUT needs a pull up resistor as shown in the manual.

To send a byte to the PAK, you set the most significant bit and pulse the clock from low to high. The data must be valid on the rising edge of the clock line. Then return the clock low and repeat until all bits are complete.

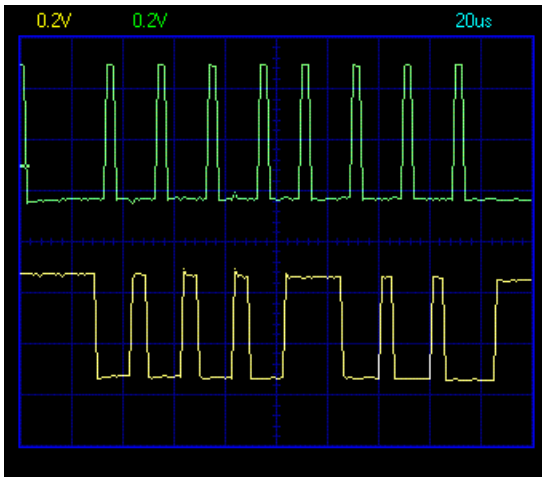
If the data line goes from low to high while the clock is high then the chip will reset its communications system.

Reception is similar. The host samples the data line and then pulses the clock. The first sample is the most significant bit and you sample until

you've read all 8 bits. Some operations require you to wait for the data line to go low. So if the host sees the data line is high, it

should not clock the PAK in these cases . Some commands don't take an appreciable amount of time to execute and therefore they don't follow this convention. For the PAK-IX running at 20MHz, keep the data clock below 100kHz. Slower speeds are fine but don't maintain the clock high for more than 2 seconds or the PAK will reset. For reference, the Basic Stamp II uses a 16kHz frequency and the Stamp IISX uses 42kHz.

Below is a scope capture (with x10 probes) of a 68HC11 sending a command \$8 to a PAK-I. The clock is on top and the data is the lower trace. Notice that the short pulses on the data line are ignored because the clock is low at those times. Only the wide pulse under the fifth clock pulse is a legitimate 1.



# Specifications

## *Absolute Maximum Ratings*

Ambient temperature under bias	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on VDD with respect to VSS	-0.3V to +6.5V
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

## *DC Characteristics*

<b>Parameter</b>	<b>Minimum</b>	<b>Typical</b>	<b>Maximum</b>
Supply voltage	4.5V	5V	5.5V
Vdd rise time on power up	.05V/ms	-	-
Supply current @ 5V	-	4mA	7mA