A PAK-IX Example

The PAK-IX is a 28 pin IC (.3") and like the PAK-II operates at 20MHz. The A/D converter has 5 input channels that read from 0 to 5V, or you can use one or two channels for an external reference if you prefer. The PAK-IX adds 4 commands to the PAK-II's mathematics commands:

- AD - Reads from 1 to 16 samples from an analog channel and averages the result
- ADCONF - Sets up reference channels, if desired
- ESTO - Stores all PAK storage registers to EEPROM
- ERCL - Recalls all PAK storage registers from EEPROM

When the chip powers up, it automatically executes an ERCL. There's also 8 digital I/O spares, so while it takes 2 I/O pins to connect it to the Stamp, you gain 5 analog inputs and 8 digital I/O, plus floating point math capabilities. Not a bad trade for 2 pins!

It is very simple to use SHIFTIN and SHIFTOUT (or similar command on other processors) to communicate with the PAK. We provide a library to help you get started. In addition, we have example code on our Web site for how to use the PAK protocol with 68HC11s, AVRs, SXs, PICs, and more.

What can you do with it?

The easiest thing you can do with a PAK-IX is to read a voltage directly in volts. With 10 bit resolution, each count is worth about .0048828125V. Using the PAK-IX library, you can write:

```
fpxhigh=$7720 ' .0048828125
fpxlow=0
gosub floady

aloop:
fpx=0 ' channel #
fpb=8 ' # of samples
gosub fa2d
gosub fmult ' convert to volts
fpx=1 ' display
gosub fdump
debug cr
goto aloop
```

This will display the average of 8 readings from AD0 in volts. Of course, you might want to read this into the Stamp in, say 1/100 volt units (0-500). Then just use .48828125 ($7D7A0000) as a multiplier, convert to an integer, and read that integer into the Stamp. It is easy. You can also use the FInt subroutine to read the counts directly as an integer. You could use this program as-is as a voltmeter, for example.

Of course, normal people don't usually care about volts -- they want something like a temperature. Suppose you use a polynomial fit like this:

temp = C3 x Ct**3 + C2 x Ct**2 + C1 x Ct + C0

(where **3 is the 3rd power, **2 is the 2nd power). Ct is the counts (0-1023) from the A/D converter. You determine that:

C0 = 84.78281 = $852990CC

C1= -1.099857E-02 = $78B43359

C2 = 7.037247E-07 = $6A3CE798

C3 = -2.325349E-11 = $5BCC8A34

(The hex numbers are from the FConvert program.)

You could write:

```
fpxhigh=$8529
fpxlow=$90CC
gosub floadx
fpx=0
gosub fsto
fpxhigh=$78B4
fpxlow=$3359
fpx=1
gosub fsto
fpxhigh=$6A3C
fpxlow=$E798
fpx=2
gosub fsto
fpxhigh=$5BCC
fpxlow=$8A34
fpx=3
gosub fsto
' Don't need to repeat the above again

fpx=0 ' channel #
fpb=8 ' # of samples
gosub fa2d
fpx=3  ' 3rd degree polynomial
fpb=0 ' register 0
gosub fpoly

fpx=2 ' display
gosub fdump
debug cr
```

Again, you could compute the polynomial to give you results in 1/10, 1/100, or whatever units and read those as an integer back to the Stamp. You could also take the result, multiply by a scaling factor (x10, x100, x1000, etc.) and multiply using FMult.

You can find more about using PAK math coprocessors in the Document Library. You can fit your data online using http://kalamation.com/Fitter/ or use the GAUSFIT program provided by Parallax.