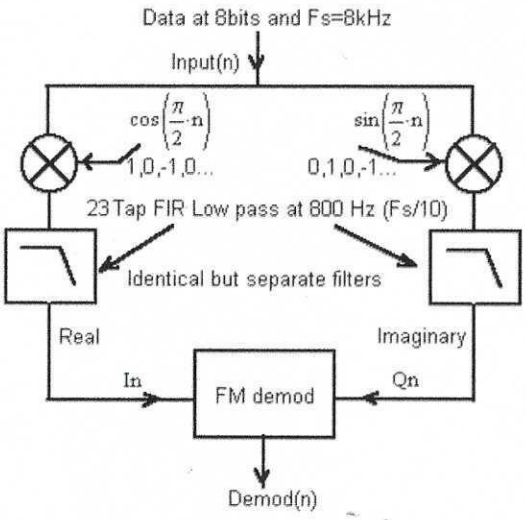
DSP FM Demodulator for SSTV

Lately there have been many excellent software programs developed that use an inexpensive and simple comparator circuit to send SSTV audio over RS232 to a computer where demodulation and display of the picture is done. In addition to this, there are programs that utilize the sound card in the computer to generate the same information. To date, all of these programs use the same basic demodulation technique of counting the time between zero-crossings of the received audio to determine frequency. Although this has worked well for many years, the SSTV community should catch up with the current technologies and take advantage of the available DSP techniques to do better filtering and demodulation. Not everyone is up to speed on these technologies, but for those who are, I'm writing this to spark some interest and experimentation.

I've been doing some development using an Analog Devices ADSP2181 DSP processor that's available on an evaluation board (called "EZ KIT Lite") and includes the software development tools for under $80 (no, I have no connection with this company...it just happens to be the processor I used). Included on the board are stereo A/D and D/A converters, RS232 interface and power comes from a supplied wall transformer. This board is pictured to the left. Similar boards are also available from Motorola for their DSP processors.

The results achieved to date look very promising with the filtering of the audio and demod being done entirely on the DSP board. If you want to see the technical details...read on. If you aren't familiar with DSP fundamentals...take some time to review in the ARRL Handbook before proceeding. Most of what I cover here is discussed on page 18.14 of the 1995 handbook (my latest copy).



The Quadrature Mix

The technique I used to demodulate the FM signal involves doing a quadrature mix of the audio down to DC where in-phase and quadrature signals are generated to give both amplitude and phase information of the signal. Refer to the block diagram at the left. Sampling of the audio from the transceiver is done at 8 kHz. This is then multiplied by a cosine signal at 2 kHz (Fs/4) to generate the "real" part of the signal. Similarly, the signal is multiplied by a sine signal at 2 kHz to generate the "imaginary" part of the signal (the multiplication is doing the same thing as the mixer in your radio). Together, the real and imaginary signals form a complex representation of the signal with what was at 2 kHz now "translated" to DC. With me so far?? Our signal of interest for SSTV was between 1200Hz (sync) and 2300 Hz (white). Now, our SSTV signal is between -300 and +800 Hz. If we had chosen a sample rate of 7 kHz we could have centered the signal around DC (which is what you would try to do) but the example would have been more awkward. At this point you may be asking how the sine and cosine signals were generated. Well... Fortunately, since the frequency we chose is 1/4 of the sample frequency, we can represent a cosine at that frequency very nicely by the sequence of samples 1,0,-1,0,1,0,-1,0..... This gives us 4 samples per cycle of a 2 kHz **cosine** wave where the sample sequence is at a 8 kHz rate. Similarly, we can generate the **sine** wave by using the sequence 0,1,0,-1,0,1,0,-1..... To illustrate how we multiply our input samples by the sine and cosine sequences, lets look at what happens to the first two samples that come in. The input sample is multiplied by 1 to generate our first "real" (in-phase) sample. This **same** input sample is multiplied by 0 to generate our first "imaginary" (quadrature) sample. The next input sample gets multiplied by 0 to generate the second real sample and by 1 to generate the second imaginary sample....and so on. Please don't leave out the data that comes out as 0 (it'll mess up the whole process).

The Filters

Both real and imaginary signals are run through identical FIR low-pass filters to remove images. Our filters for this example would be 800Hz low-pass FIR filters that effectively perform the job of a bandpass filter on the SSTV signal. FIR filters have a constant delay (linear phase) response (a big plus for FM demodulation where you are detecting changes in phase). We can use a 23-tap filter and be down by 40+ dB at 1500 Hz and have plenty of processing horsepower left to spare.

The Demodulator

The rate of change of phase of our SSTV signal will give us the SSTV FM demodulation data we are after. The phase is simply the arctangent of (Qn/In) where Qn is the quadrature (or imaginary part of the signal) and In is the in-phase (or real part of the signal). The rate of change of phase with respect to time (or samples in our case) is the frequency so all we have to do is figure out how much the phase changes between each sample and that number will be a representation of the frequency of our signal. Without going into the derivation, I will now give you an equation that can be used (without doing any trig. calculations) to accomplish just what we want. If you can do a little calculus and you're still interested, the derivation is included at the end of this article.

$$Demod_n=\{Q_n*I_{n-1}-I_n*Q_{n-1}\}/\{I_n^2+Q_n^2\}$$

Here "n" is used to indicate the current sample and "n-1" is used to indicate the sample just before the current sample. "I" is the real or in-phase part and "Q" is the imaginary or quadrature part. This is a "first order" detector that works very well for our slow-scan design. Higher order implementations can be done that use more input samples but it doesn't buy much in performance.

The Results

To see how it all works, I have implemented all of the above on the ADSP2181 and sent the result out through the DAC on the board so it can be seen on an oscilloscope. The demodulator performed very well and seemed to offer some improvement during noisy conditions (this type of FM demod has fairly good rejection of AM noise). The same data can be output on the RS232 port of the DSP board to your PC where it can be converted into a picture. I'm currently working on that part of the project. There are a couple of SSTV developers who are trying their hand at this method including one who is using a soundcard to generate the digitized samples. This should work just as well with the tradeoff being the CPU cycles that get consumed to do the DSP. Since the DSP code would be replacing the entire zero crossing code and all the Band-Aids required to make this technique work reasonably well, the tradeoff may be a reasonable one. There are many advantages to this approach including that you get several demod outputs per cycle of the input audio. In fact, the output data rate is much higher than is needed and can be reduced by doing "decimation" of the data after the FIR filter processing (as long as you maintain 2 samples per cycle at the frequency where the filter is down by 30 or 40 dB). Work continues on this project and I would be interested to hear from developers who plan to give this a shot. If you know folks who have been developing SSTV programs, please ask them to check this out. DSP techniques will be **the** way to do SSTV demod (and all other kinds of demod) in future ham equipment. Similar techniques are currently being used to do demod of digital modulation in commercial equipment (including TNCs and cell phones). Here is an opportunity to experiment with something that more hams should be having fun with. **See you on the air!**

Derivation of the equation:

$$Demod_n=\{Q_n*I_{n-1}-I_n*Q_{n-1}\}/\{I_n^2+Q_n^2\}$$

(Note that my original article had the sign flipped. It still worked but the output waveform was inverted.)
We know from the article that $Demod_n=d(\text{atan}\ (Q_n/I_n))/dn$

Since $d(\text{atan}\ u)/dn=\{1/(1+u^2)\}*du/dn$

Then $Demod_n=\{1/(1+(Q_n/I_n)^2)\}*d(Q_n/I_n)/dn$

Simplifying gives a general expression for FM demod (Equation 1):

$Demod_n=[\{I_n*d(Q_n)/dn-Q_n*d(I_n)/dn\}/I_n^2]/(1+(I_n/Q_n)^2)$

For a first order approximation of the derivative d/dn, we can use (Equations 2&3):

$d(I_n)/dn \sim I_n-I_{n-1}$

$d(Q_n)/dn \sim Q_n-Q_{n-1}$

For a better approximation of the derivative we could use a program that generates coefficients for a higher order FIR filter that is a differentiator. Linearity of the Demod output can be improved this way, if needed.

Back to the equation. Substitute Equations 2&3 into Equation 1 and we get:

$Demod_n=[\{I_n*(Q_n-Q_{n-1})-Q_n*(I_n-I_{n-1})\}/I_n^2]/(1+(I_n/Q_n)^2)$

Simplifying:

$Demod_n=\{I_n*(Q_n-Q_{n-1})-Q_n*(I_n-I_{n-1})\}/(I_n^2+Q_n^2)$

$Demod_n=\{Q_n*I_{n-1}-I_n*Q_{n-1}\}/\{I_n^2+Q_n^2\}$