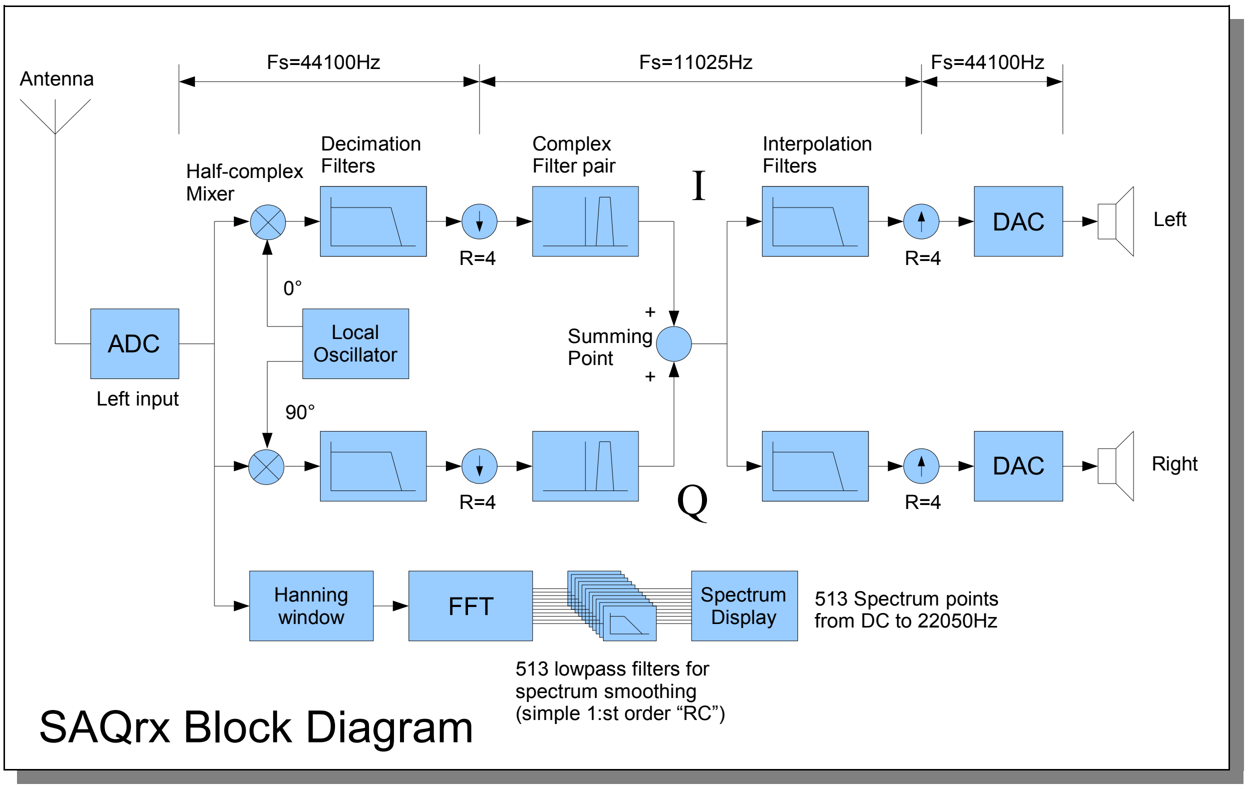
SAQrx

A VLF Receiver with Panoramic Tuning



RECEIVER

SAQrx is built almost like the schoolbook example of a phasing method SSB receiver, the main difference is that SAQrx is held together by DSP software instead of solder. Below is an attempt to describe the signal flow through SAQrx:

The input signal is sampled at Fs=44100Hz and fed to a half-complex mixer where the signal is mixed (multiplied) with a quadrature local oscillator signal, cos(ωLO * t) for the I branch and sin(ωLO * t) for the Q branch. The resulting complex I/Q signal may contain both positive and negative frequencies. A single frequency I/Q signal can be thought of as a rotating vector in the I/Q plane where the length of the vector is the signal amplitude and the rotation is the signal frequency. If the input signal has a higher frequency than the LO, the I/Q vector will rotate clockwise and vice versa. An input signal at the same frequency as the LO (zero beat) will produce a still standing vector whose "pointing direction" in the I/Q plane depends on the phase relationship between the input signal and the LO.

RECEIVER (cont'd)

The I/Q signal from the complex mixer is passed through two equal lowpass filters (decimation filters). These are 64-tap FIR filters that are flat from DC to 3000Hz. The filter stopband is about -90dB at 5512Hz which “happens” to be Fs/8. The purpose of these filters is to band-limit the I/Q signal to less than half the new sampling rate in order to avoid aliasing (Nyquist). In other words, only those antenna input signals that are within +/-3000Hz of the LO frequency will pass through the decimation filters. The decimation filters guarantee that the I/Q signal contains no frequency components above 5512Hz so we can safely decimate (reduce the sampling rate) by a factor of four (R=4) to 11025Hz. The FIR buffers of the decimation filters are filled at Fs=44100Hz but they are only “computed” at the output rate, Fs=11025Hz. The reason for decimation is to reduce the computational load. Not only is the FIR filter computing rate reduced, the filters can have less taps too without sacrificing performance!

Next step is the complex filter which provides the main selectivity in the SAQrx receiver. Both halves, I and Q, are initially designed as two exactly equal lowpass FIR filters having half the desired SSB bandwidth (DC to Fbw/2). This results in a passband +/-Fbw/2 around DC (zero beat) which is not exactly what we want. The receiver would sound like a direct conversion receiver with equal response either side of zero beat. So, in order to shift the passband center away from DC to a comfortable CW pitch, and true single sided reception, we multiply the coefficient sets for the two filters by cos(Foffs) and sin(Foffs), respectively, where Foffs is the desired shift from DC. The resulting (complex!) filter response now looks as shown in the complex filter boxes in the block diagram where the vertical line corresponds to zero beat. SAQrx actually contains three selectable complex filters although only one is shown in the block diagram.

Classic (hardware) phasing receiver designs often use a wideband phasing network that puts the relative phase of the I and Q signals 90 degrees apart, over the whole audio bandwidth of interest, and then use some kind of bandpass filter after the “Summing Point”. The 90° phasing network can be made as a FIR filter in DSP (Hilbert transformer) but I think the complex filter approach is more elegant since it “kills two birds with one stone” - bandpass filtering *and* phase shifting.

The I and Q components from the complex filters are added in the “Summing Point”. This point is where the actual sideband cancellation occurs. SAQrx is “wired” for USB reception. If you want LSB just subtract I from Q (change sign on one of them), or swap the I and Q lines before the complex filter or swap the two LO outputs. A two-phase AC motor is a good analogy – just swap the two phases anywhere along the powerline and the motor will run backwards ;-)

The signal from the summing point is a real (“single-wire”) signal almost ready for listening, the only remaining problem is to step up the sampling frequency from 11025Hz to 44100Hz.
This step-up is necessary since the DAC runs at the same speed as the ADC. This is accomplished in an interpolation filter. SAQrx happens to have two identical interpolation filters although a single filter would be enough since both filters carry exactly the same signal. The reason for using two filters is that I have plans for playing with “binaural pseudo-stereo” experiments in the future ;-)

The interpolation filters use exactly the same FIR coefficients as the decimation filters, they have exactly the same frequency response, but they are implemented differently to save processing time. The output of the interpolation filters is computed at Fs=44100Hz while the input is fed with 11025 signal samples each second, interleaved with three zeroes between each signal sample. The fact that three out of four input samples are zero makes it possible to use a trick. Even though the filter has 64 taps (theoretically), only 16 taps (multiply-and-accumulate operations) have to be computed at each 44100Hz sampling instant. In addition, the FIR data buffer size can be reduced to 16. The FIR coefficients are split into four groups that are used cyclically when computing the filter output.

RECEIVER (cont'd)

Oops! I forgot to draw the 0..50dB gain control in the block diagram. The signal is multiplied by a constant, 1 to 316, between the interpolation filters and the DACs. You'll have to imagine this...

SPECTRUM DISPLAY

Every input sample from the ADC is stored in the FFT input buffer. Whenever the buffer gets filled, the entire 1024 sample chunk in the buffer is multiplied by a Hanning window function in order to reduce the sidelobes in the spectrum plot. The windowed data chunk is then passed to the “real input” FFT routine which takes 1024 real (“single-wire”) input samples and generate a 513 point complex (I/Q) output spectrum, ranging from DC to 22050Hz, in the FFT output buffers. The audio processing and the screen drawing operations run in different and asynchronous threads; the FFT is run about twice as often as the screen redraw function, which runs at approximately 50ms intervals, so more than half of the FFT output is wasted. This is not a serious problem, just a bit ugly. You can see that the spectrum graph looks more “lively” when the filter passband is beeing dragged with the mouse because the screen is then updated continously to make the movement look smoother.

Anyway, the screen drawing routine sets a flag when it has finished the spectrum plot. If this flag is set at the end of the FFT calculation, the audio thread will compute the amplitude spectrum. We are not interested in the relative phase of each FFT frequency “bin”, only the absolute amplitude, so we use Pythagoras on each complex frequency “bin”: Amplitude = $\sqrt{(I^2+Q^2)}$. These 513 amplitude points are then individually lowpass filtered to get a smoother and less “nervous” display. These filters are very simple, they are actually the DSP equivalent of simple first order RC filters. After smoothing, the amplitude points are stored in a buffer that is read by the screen drawing routine where the amplitudes are converted to dB and scaled for screen output.

DEVELOPMENT TOOLS

IDE/compiler: DEV-C++, version 4.9.9.2, www.bloodshed.net
Soundcard interface: Portaudio, version 18, www.portaudio.com
FIR filter design: ScopeFIR version 3.7a, www.iowegian.com

I use "PortAudio" for audio I/O because it is extremely simple to use compared to accessing window$'s messy multimedia functions directly. Portaudio's application program interface (API) is fully described in the file portaudio.h. The included version of Portaudio uses the “classic” window$ MME interface only, there is no support for WDM-KS, ASIO etc. PortAudio source code is freely available on the web but you don't have to download it; the SAQrx zip file contains all Portaudio files needed to compile SAQrx. By the way, v18 is not the latest version of Portaudio but it works perfectly for basic audio I/O. v19 seems to add a lot of “bells and whistles” (and unwanted complexity...) which does not suit my QRP taste.

The FFT routine was written by Wolf DL4YHF (the author of SpecLab) and is shared with his permission.

ScopeFIR is not free but other FIR filter design programs can probably be found on the Internet.

How to compile SAQrx

1. Install the latest version of Dev-C++ (v4.9.9.2 at the time of this writing)
2. Unzip saqrx.zip into a folder of your choice
3. Double-click SAQrx.dev to open the project

You probably need to make a change in the "Project Options" because of a different directory structure on your PC (some file name paths stored in the .dev project file are relative). Most settings should be OK in my .dev file, except the path to the WMME link library. Open the "Project Options" dialog, click the tab "Parameters" and delete my old file path under "Linker". Then click "Add Library or Object", browse to the directory "...\lib" under the DEV-C++ installation directory and select the file libwinmm.a.

All settings under "Project Options"/Compiler should be set to "NO" except the following:

- Linker: Strip executable: YES (removes some garbage to make the .exe file smaller - QRP)
- Optimization: Perform a number fo minor optimizations: YES
- Optimization: Further Optimizations: Best Optimization: YES

Do *not* select a program icon in the "Project Options"! The icon is linked through a resource file.

Files in the SAQrx zip archive:

1. `saqrx.dev`	DEV-C++ project file
2. `main.c`	SAQrx main source file
3. `dspmath.c`	SAQrx FFT and windowing functions
4. `dspmath.h`	Header file for above
5. `coeffs.c`	SAQrx FIR filter coefficients
6. `coeffs.h`	Header file for above
7. `saqrx.ico`	SAQrx program icon
8. `progicon.rc`	SAQrx icon resource file
9. `portaudio.h`	Portaudio API (don't touch)
10. `pa_convert.c`	Portaudio stuff (don't touch)
11. `pa_host.h`	Portaudio stuff (don't touch)
12. `pa_lib.c`	Portaudio stuff (don't touch)
13. `pa_trace.h`	Portaudio stuff (don't touch)
14. `pa_win_wmme.c`	Portaudio stuff (don't touch)
15. `saqrx.pdf`	This file

Recommended reading:

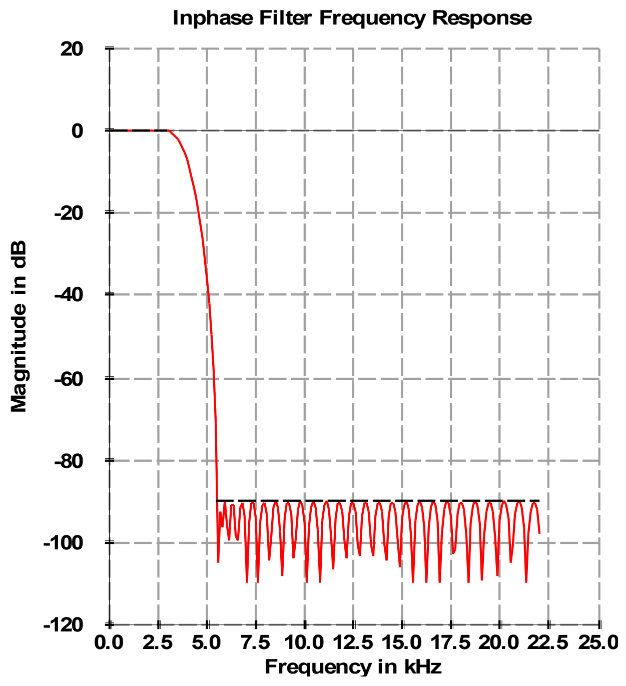
Digital Signal Processing in Communication Systems
by Marvin E. Frerking, ISBN 0-442-01616-6

The Scientist and Engineer's Guide to Digital Signal Processing
by Steven W. Smith, free download at www.DSPguide.com

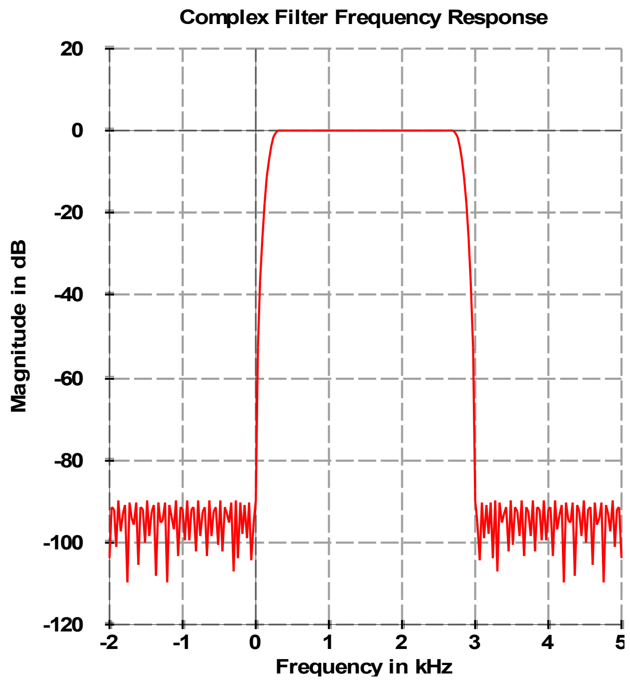
73 de Johan Bodin, SM6LKM
Daljöfors 2006-12-30

Appendix – Frequency Responses of SAQrx FIR Filters

Frequency response of decimation/interpolation filters (3000Hz lowpass):

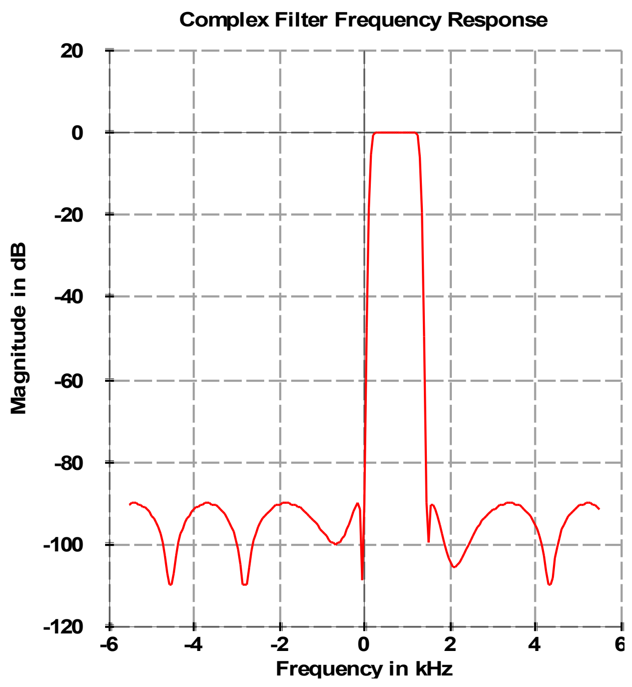


Frequency response of 2400Hz complex filter:

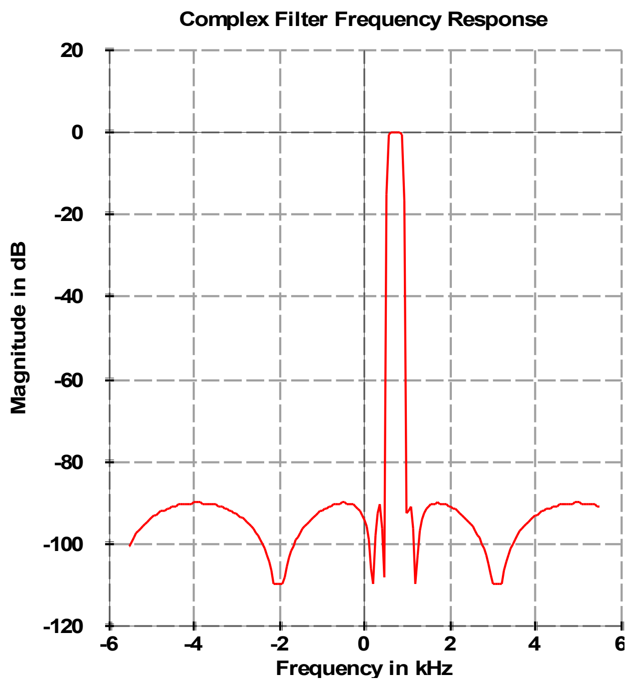


Appendix – Frequency Responses of SAQrx FIR Filters (cont'd)

Frequency response of 1000Hz complex filter:



Frequency response of 300Hz complex filter:


Complex Filter Frequency Response
Magnitude in dB
Frequency in kHz