Ray Mack, W5IFS

17060 Conway Springs Ct, Austin, TX 78717; w5ifs@arrl.net

SDR: Simplified

Mea Culpa

I made a serious mistake in preparing the text for the July/August issue. I believed that the tools and documentation were sufficient to actually allow us to produce a working program that would implement a software defined radio. I have spent much time attempting to find the necessary *C* program materials, but with little success. I am still working on the process and hope to have an updated zip file for the July/August issue on the ARRL Web site by the time you read this. As I sort out what to do and why, I will update the Web site with instructions and clarifications.

Michel Barbeau, VE3EMB, has created a new *Linux* kernel that can be loaded into the board.[1] The new kernel includes a proper SPI driver that is not in the board as shipped.

I have received feedback that some of the three letter initials that I thought were in common use are foreign to some readers. I have provided a short Glossary as a sidebar to this installment. I find the use of two and three letter initials to be generally antisocial behavior by engineers, but some have been used for so long that they are a reasonable part of our language and replace their English language equivalents. I will avoid initials as much as possible in future columns. Some initials have such wide use, however, that the original English has actually been replaced. NASA, FBI, and IBM are examples where no one uses the English, and most folks probably do not know the origin of the initials. DSP, SDR, DFT, and FFT are ones from this installment that fall in the common use category. (See the Glossary for definitions.)

Hardware Update

I have been devoting significant time (when not fighting the software tools) to developing reference boards that we can use for our lab experiments. I have enough requests that Dave Pollum and I developed a combination DAC08/AD7476 board. We have Gerber files if you want to buy your own board from a circuit board manufacturer, or you can contact me directly to buy a board.[2, 3] The board uses through hole parts except for the AD7476A and AD8027A, and has a silk screen for component placement. The design of the AD7476 portion is different from the Analog Devices version with respect to power and voltage reference. Our board is for demonstration purposes rather than a reference design for a precision component, so we removed some of the power supply components to reduce cost and complexity. Figure 1 is the schematic that Dave created for this board.

[1]Notes appear on page 48.

Resource Update

I do a sanity check on all of the topics I write for you, using the books I have in my library and other resources. I have a new book to recommend, based on my research for this column. *Multirate Digital Signal Processing* has a fair amount of math, but the descriptions are understandable even if you totally ignore the equations.[4] The authors worked at what used to be Bell Labs. The focus of the book is on telecommunication applications, so it fits well with our interest in software defined radios.

If you haven't discovered Wikipedia, you should give it a whirl. It is one of those other resources I use. For example, type "wiki frequency leakage" in the search bar of your Web browser, and you will see a link to a description of spectral leakage as described below. It is a great place to look if I describe something you don't understand.

In digging through the documentation for the *C* compiler, I found that it implements DSP specific functions and data types that are a part of the *Visual DSP* product from Analog Devices. The help files on the CD for the Gnu *C* compiler point you to the documentation for *Visual DSP*, but the link is very old. Instead, you will want to go to the Analog Devices Web site to get the manual.[5] You will also likely want to look at the site for additional manuals for the Blackfin processors.[6] I also recommend downloading the manual for the BF537 EZ-Kit Lite.[7] Our Stamp boards only implement a subset of the EZ Kit hardware. Even so, the manual is a good source of information on our hardware.

Sampling, Nyquist and Spectrum

We need to dig a little deeper into the math at this point. The sequence of digital numbers that we get from the ADC in our real hardware is a process called sampling. The math that is involved in sampling multiplies a sampling function times the input analog signal. This multiplication changes the continuous analog input signal into a sequence of individual samples that are equally spaced in time.

The math sampling function that we use doesn't really exist. It is a mathematical trick, but its properties are important for conversions we can do in the real world. The basis of the sample waveform is called an impulse. It is a signal that has infinite height, an area of one, and zero width. The sampling waveform is actually a sequence of impulses with the time between impulses equal to the period of the sampling frequency. The math that deals with the implications of an infinite-height pulse gets really messy, and, thankfully, we can ignore it. The most important consequence is that an infinite sequence of impulses in the time domain produces a Fourier Transform that is an infinite sequence of impulses in the frequency domain. This is the basis for all of the discrete transformations. The Fourier Transform of a sample function of 1 Volt-second at 500 kHz has a spectrum containing 1 V at dc plus a 1 V sine wave at 500 kHz and a 1 V sine wave at every positive and negative harmonic of 500 kHz, from negative infinity to positive infinity. In the RF world, we call such a time domain signal a comb generator, since the output spectrum looks like a comb with equal height tines. It is the zero width of the time domain signal that causes the "comb" to have equal heights at all frequencies.

Our hardware generates the sampling sequence and multiplies it times the input analog signal. The result is the sequence of digital numbers corresponding to the input analog signal. If you could actually produce the sampled sequence in the real world and then look at it on a spectrum analyzer, you would see the spectrum of the input around dc and a double sideband set of the input spectrum around each harmonic of the sample frequency. This is shown in Figure 2.

Even though we cannot really produce this exact spectrum in the analog world, it does exist in the DSP world, and is very real when we do the math. The spectrum between dc and one half the sample frequency is called the first Nyquist zone. It holds the exact representation of our input signal. The next Nyquist region contains the same spectrum, but it is inverted since it is the lower sideband. We can easily apply a band pass filter to the spectrum of any of the Nyquist zones to produce a frequency translated version of our baseband signal. We can also use the equivalent of a product detector to take an inverted spectrum version and invert it a second time to get back to a true version of the original signal.

Discrete Fourier Transform and Filters

We talked a little about the discrete Fourier transform (DFT) in the July/August issue. We were interested in using the DFT to generate the coefficients for our filters. We need to look at the relationship between the number of samples and the

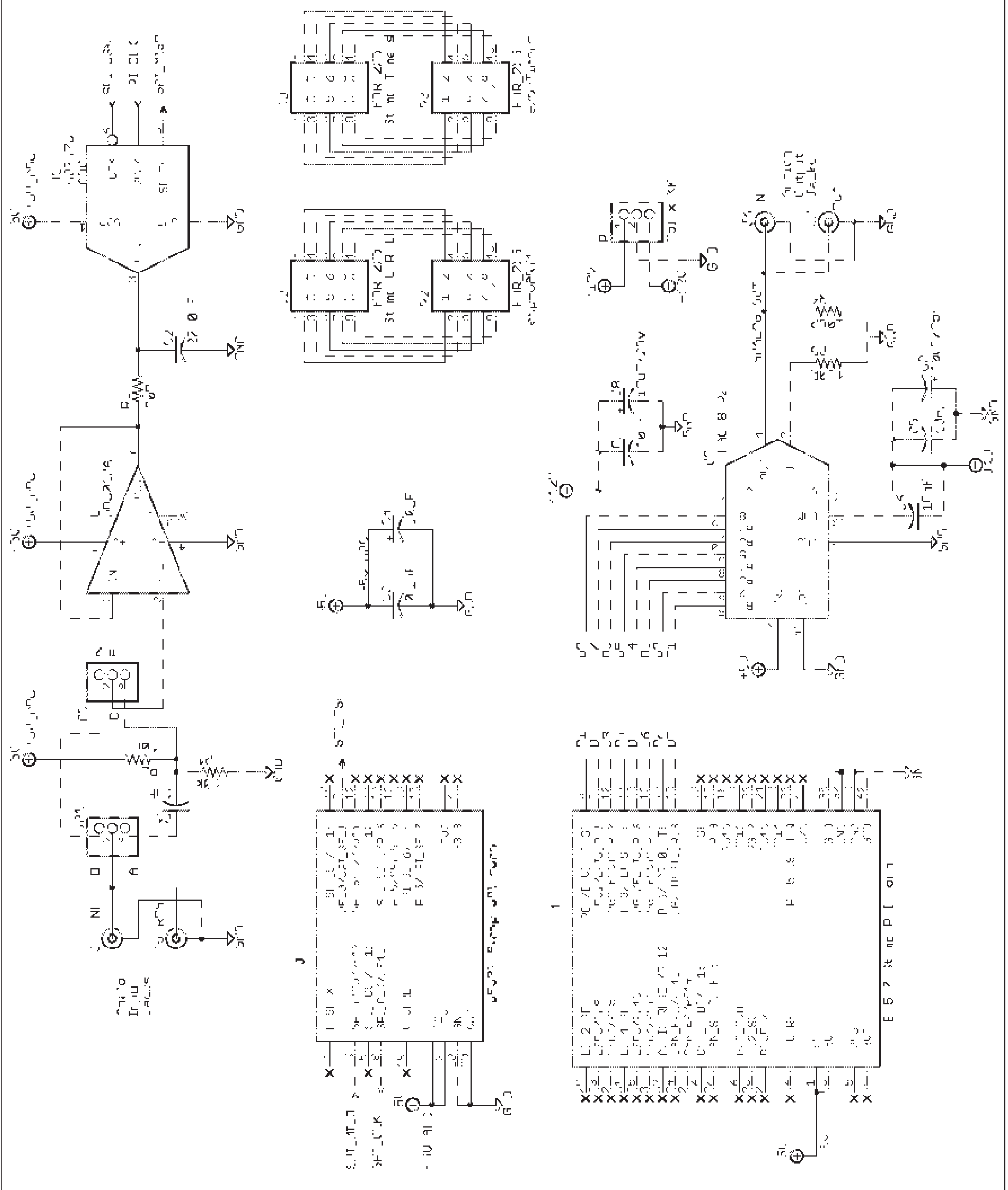

Figure 1 — Schematic of the combined AD7476 ADC and DAC08 DAC board.

sample rate and how that affects the representation of the data.

Mathematicians call the time domain signals that we use a "real" signal. Such a signal requires two wires: ground and signal. The signal from your microphone or your speaker is a "real" signal. This signal inherently contains all of the amplitude and phase information for a human to use it. A Fourier transform converts this signal into what mathematicians call a "complex" signal. All that really means is that the Fourier transform takes the single dimension real signal (voltage versus time) and creates two new signals that contain the phase and amplitude information in the frequency domain. We call the complex data I and Q.

A discrete Fourier transform starts with the conversion of our real signal into a sequence of evenly spaced samples in time. The Fourier transform action converts the time sequence into two sequences of samples in the frequency domain. One is the I sequence and the second is the Q sequence.

Our example samples the input at 50 kHz and the input data is bandwidth limited to 22 kHz (to meet the Nyquist criterion). We will do our discrete Fourier transform on a sequence of 1000 samples. The output of the transform always produces an I sequence and a Q sequence, where the input energy is split between the two sequences. The result is that the I sequence holds 500 frequency samples and the Q sequence holds the other 500 frequency samples. There is a small subset of real signals, where either the I sequence or the Q sequence will contain only zeroes, but the general case always has some energy in each sequence.

The center of the data is at time zero, with half of the samples before our arbitrary zero point and half after, so that the math works correctly. The symmetry about zero causes the frequencies of the transformed data to be centered around zero hertz, where half the samples represent negative frequencies and half are positive frequencies. Our input samples are spaced exactly 20 microseconds apart and they represent only the voltage at each instant that is measured; no values in between exist. Each transformed frequency sample also represents energy at ***exactly*** one frequency; no frequencies in between exist. Throwing away the information in between is the essence of the "discrete" in the discrete transform. The frequencies of each output transform sample cover the frequencies from –25 kHz through +25 kHz for a total frequency span of 50 kHz (the same as our sample rate). We have 500 I samples that cover 50 kHz. The math is pretty easy: 50 kHz / 500 samples = 100 Hz per sample. So the first I sample is –24900 Hz. The second sample is –24800 Hz and so on to +24900 Hz. The same frequency sample positions occur for the Q sequence. The DSP term for each of these sample positions is a ***bin***. ***Bin*** comes from the concept that the energy is stored in a container just as if it were an actual container on the wall. In our case we would need 1000 real containers. All of the input energy gets stored in one or more of the bins. There is energy, however, between the discrete 100 Hz locations of the bins. Another example can show us what happens to the energy that is not exactly at a bin frequency.

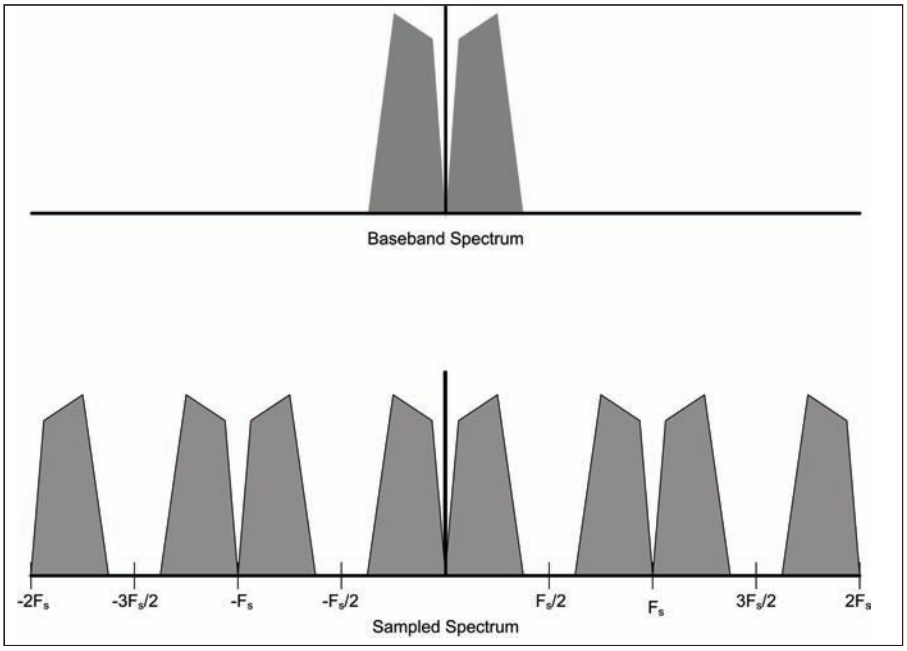

Figure 2 — The baseband spectrum of an analog signal before sampling is given at A, showing both positive and negative frequencies. Part B shows the spectrum of the baseband signal after it is sampled by an ideal sampling waveform. Note that there is an upper and lower sideband version of the baseband for each harmonic of the sampling frequency, just as if it were a double sideband suppressed carrier signal.

Let's set up a system with three crystal oscillators spaced 500 Hz apart. We choose frequencies of 100.0 kHz, 100.5 kHz, and 101.0 kHz with amplitudes of 1 V RMS.

Modern synthesized transceivers (like the ICOM IC-706) can be set up to look very much like a discrete spectrum analyzer. A discrete Fourier transform also implements a discrete spectrum analyzer. Let's set up the radio for CW with 100 Hz filter bandwidth. Let's also set up the tuning steps to 200 Hz and start tuning at 99.8 kHz. We will measure very little energy since the 100.0 kHz signal is quite far down on the filter skirt. The next step will tune the radio to exactly 100.0 kHz and we will measure the full strength of that signal. The next step will produce almost no energy. The next step tunes the radio to 100.4 kHz which puts the 100.5 kHz signal significantly up the slope of the filter skirt. The next step tunes the radio to 100.6 kHz, and we will measure the same value we read at 100.4 kHz due to energy on the other side of the filter skirt. Neither of those values will show the exact value that is at 100.5 kHz. We will once again get an exact value when the radio tunes to 101.0 kHz. The discrete tuning has created an output that is inexact for one of the signals in the input spectrum. It shows up at two frequencies and does not have an accurate amplitude. This is exactly what happens with a discrete Fourier transform if we do not use enough samples.

The energy that shows up in "wrong" bins is called spectral leakage. Spectral leakage is the frequency domain equivalent of quantization error in the time domain. Spectral leakage and quantization error are the math consequences of converting an infinite and continuous signal into a discrete signal with limited sample size and resolution. We'll look at quantization noise and other ADC and DAC errors in another column.

When we created a band pass filter in our AM radio experiment, we only used a 10 tap filter. Since the 10 taps come from doing a DFT of the filter response to an impulse, that means we only have filter bins every 500 kHz / 10, or 50 kHz per bin. That is not a very good filter. Since we mostly wanted to get rid of the dc component and energy at 180 kHz, the filter will do a moderate job. If we needed to resolve frequencies with better precision, we would need more taps to narrow the size of the bins. More taps means that we need to do a lot of math in each 2 microsecond interval. We will likely run out of CPU cycles very quickly. We only have to deal with energy over about 40 kHz in our AM radio example because it was filtered pretty well by our input filter. The signal is oversampled by a factor of 500 kHz / 40 kHz or 12.5 / 1. We could do all of the work we need and stay within the Nyquist criterion by sampling at 80 kHz if we manage the sample rate correctly. A lower sample rate would also allow us to have much smaller bins for our filtering.

Multirate Signal Processing

The sound of it is intimidating, but like everything we have looked at so far, it is not nearly as complicated as the mathematicians make it sound. The topic we are interested in now is sample-rate down conversion (decimation). Sample-rate up conversion is another multirate signal processing topic that we will cover when we look at transmitters.

If we want to operate directly at the 40 meter band for instance, we will need to sample above 14.6 MHz in order to satisfy the Nyquist criterion. We do not need a sample rate that fast to actually manipulate the information on receive, though, since the bandwidth of the widest signal will be on the order of 7 kHz or less. Even if we want to look at the entire band and generate a spectrum display, we only need about 650 kHz for the sample rate. Any higher sample rate on receive is a waste of processor resources.

There are two main reasons why we would want to match the sample rate closely to our intended bandwidth. The first reason to lower the sample rate is that the transition band and ripple of our filters are dependent on the ratio of the filter length (the number of taps) to the sample rate. The second reason is to allow more CPU cycles for processing each sample.

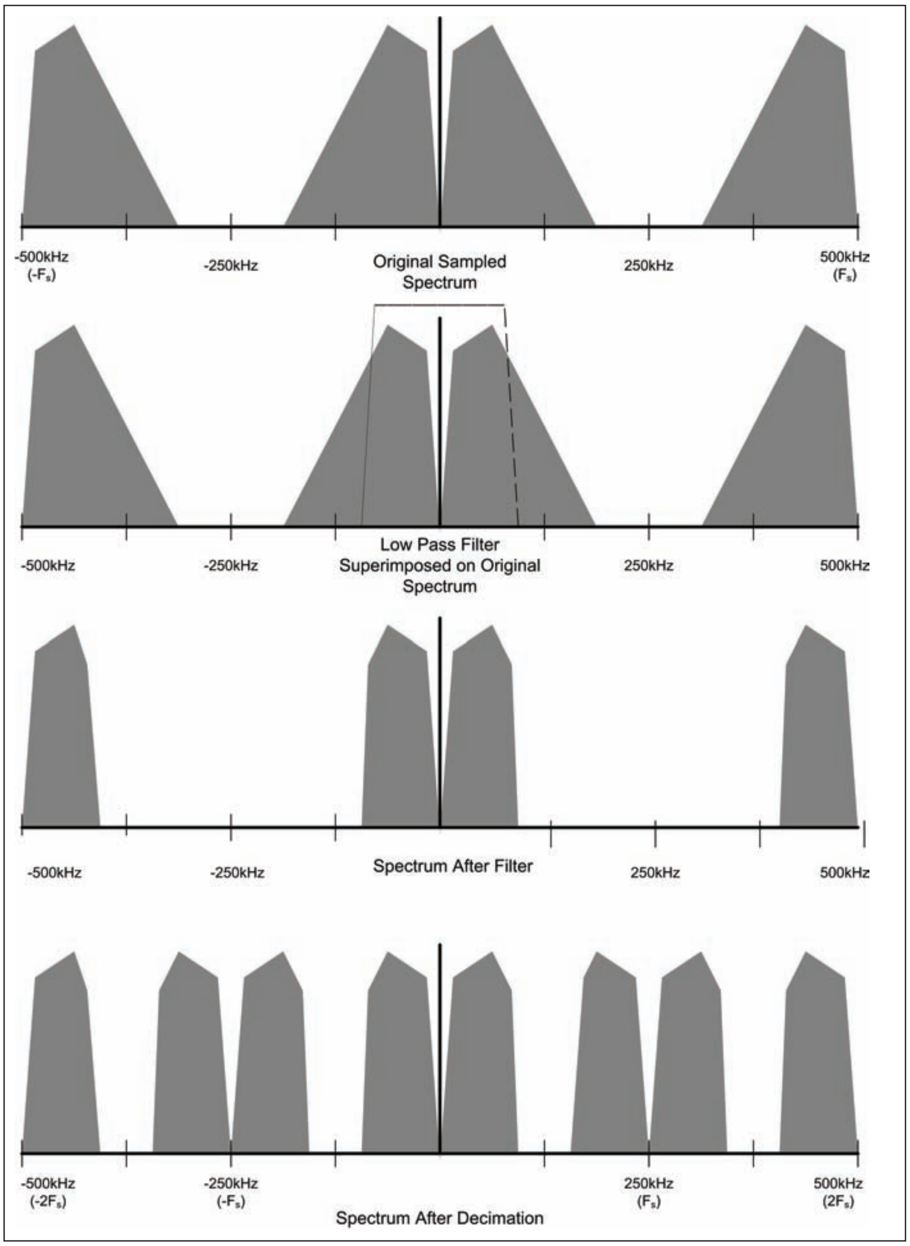

Figure 3 — Part A shows the spectrum of our sampled signal. Note that the frequency only extends from –500 kHz to +500 kHz since that is the extent that the math will manage. Part B shows the digital low pass filter response superimposed on the sampled spectrum. The Spectrum of the filtered signal, which is still sampled at 500 kHz, is shown at C. Part D shows the spectrum of the resulting sampled waveform after decimation by 2 (every other sample discarded). This is the spectrum of the signal when it is sampled at 250 kHz.

Sample-Rate Down Conversion

The super heterodyne version of our AM radio is a good candidate for sample rate conversion. Once we under sample the input and create the 90 kHz signal, the energy is all within the range of dc to 125 kHz. The input data is oversampled by 2 times the required amount. If it were possible to reduce the sample rate by 2, we would have two times as much CPU power to filter and demodulate the signal. This is low pass filter decimating.

The first step in the decimation process is to filter the input signal so that there is no energy above one fourth the sample frequency. A DSP low pass filter with cutoff at one fourth of the sample frequency has a very easy set of coefficients and can be implemented with a small number of taps. We sampled our AM radio at 500 kHz, so we need 125 kHz cutoff for the low pass filter. We now have a signal that has no energy above 125 kHz and is sampled at 500 kHz. We can throw away every other sample at this point to create a signal that is sampled at 250 kHz, with energy up to 125 kHz. The Nyquist criterion is satisfied with this new signal. Figure 3 shows the spectrum of the process.

The signal at 90 kHz is only about 40 kHz wide (70 kHz to 110 kHz), so it would be nice if we could drop the sample frequency even further. It turns out that our signal will fit into the band from dc to 62.5 kHz if it were translated down in frequency and would require a 125 kHz sample rate. This is integer band pass decimating. If we throw away every 3 samples of the original data, we can accomplish both frequency translation and sample rate reduction. The energy now spans from 15 kHz to 55 kHz. The signal spectrum is also inverted. In our AM radio case, we had band pass filtered the signal before sampling. It is possible to do the band pass filtering digitally with a small number of taps, and then do the decimation.

We succeeded in reducing the sample rate by a factor of 4. We cannot further reduce the sample rate using straight decimation. Each integer sample rate reduction requires that the band limited data fit completely within one of the Nyquist regions for the new sample rates. The next integer sample rate would reduce the sample rate by 5. The new sample rate would be 100 kHz. The data is contained in both the k = 1 (50 kHz to 100 kHz) and k = 2 (100 kHz to 150 kHz) Nyquist bands. Figure 4 shows the overlap. The overlap into the third Nyquist zone prevents further rate reduction. The requirement that all of the energy fits into one Nyquist zone limits the usefulness of this technique.

Integer sample rate reduction is a very useful tool because the filters are easily realized in hardware such as a field programmable gate array (FPGA) or other dedicated hardware. Texas Instruments (TI), Intersil (formerly Harris) and Analog Devices make dedicated ICs that do digital sample rate conversion using these techniques. My understanding is that the TAPR software defined radio project uses an FPGA for decimation. I recommend taking a look at the SDR resources on the TAPR site for additional information.

Next Issue

My plan for the next issue is to return to our work with the software on the Stamp board and do some more experiments. I should have sorted out the location of the necessary files and tools to be able to create *Linux* programs using a *Windows* system. We will also go through the process of loading a new *Linux* operating system on the Stamp, so you can take advantage of newer device drivers for our work.

Support Group

I get e-mail from time to time from folks with corrections, offers of help, and questions. You have seen the fruits of two contributors in this issue. For now I am putting folks into a group for my e-mail and sending out updates between issues as well as trying to keep the ARRL Web site up to date.

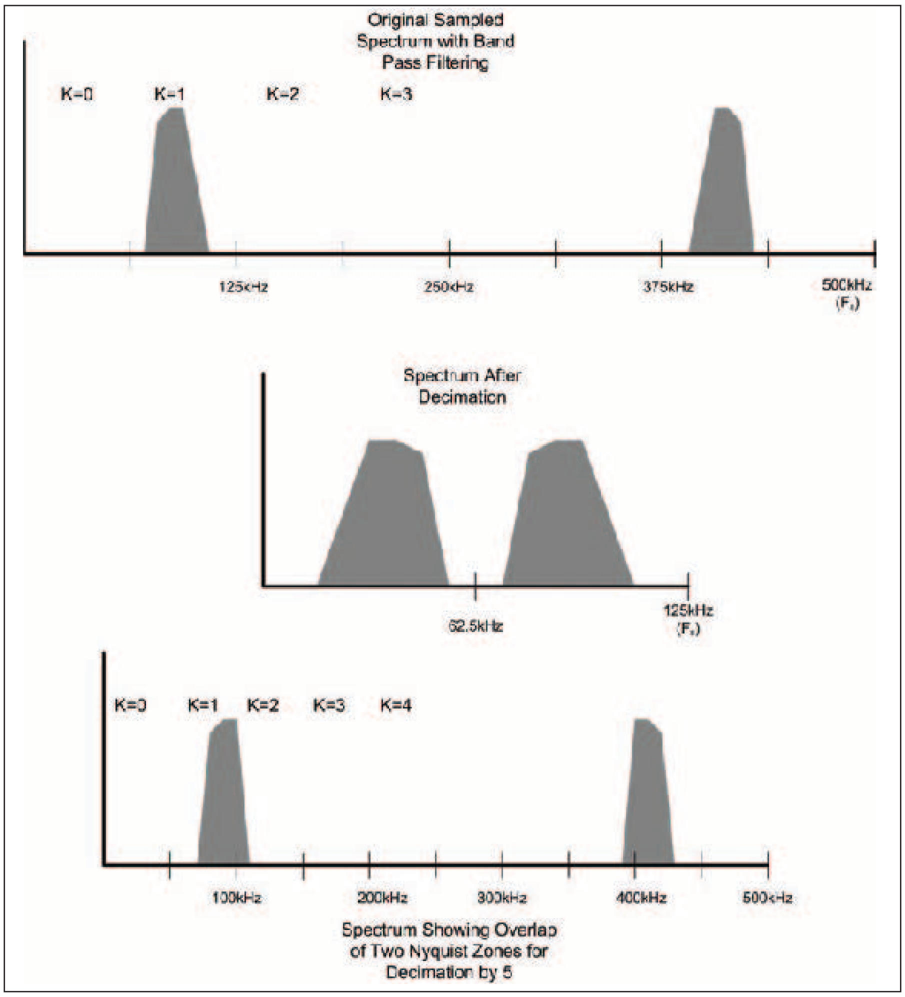

Figure 4 — Part A shows the spectrum of a 90 kHz signal that was band pass filtered and then sampled at 500 kHz. (only ½ of the spectrum is shown for clarity). The resulting spectrum when the signal is decimated by 4 (3 of every 4 samples discarded) is shown in Part B. Notice that this causes the signal to be aliased and the signal in the first Nyquist zone (k = 0) has its frequencies inverted just as a lower sideband signal is inverted. The signal is now sampled at 125 kHz. The initial sampled signal, showing the Nyquist zones that would occur if a decimation by 5 were attempted, is shown at C. Note that input energy exists in both the second and third Nyquist zones. This prevents proper decimation by 5.

Glossary

ADC (analog to digital converter) — An electronic circuit that converts an analog signal into a digital number with discrete values.

Bin — The DSP term for each of the discrete positions in a set of samples. Bin comes from the concept that the energy is stored in a discrete container.

CPU (central processing unit) — The part of a computer that does the actual math. Each operation is an instruction. The speed of the CPU is measured in millions of instructions per second (MIPS). It is normal for computers designed for digital signal processing to execute more than one instruction per clock pulse. Our Stamp board runs the CPU at approximately 400 MHz for the clock pulse rate, so we get at least 400 million instructions per second.

DAC (digital to analog converter) — An electronic circuit that converts a digital number into a corresponding voltage or current.

DFT (discrete Fourier transform) — The general case of a discrete transform from the time domain to the frequency domain.

DSP (digital signal processing) — The field of study in which signals are a sequence of data samples. Those samples are processed individually to perform operations such as filtering and conversion from time domain to frequency domain. The signals are not required to represent electrical signals.

FIR (finite impulse response) — A DSP operation that implements a transform on a sequence of samples with a fixed number of multiplications and additions. For example, an FIR transform with a length of 7 will do 7 multiplications and 7 additions on each input sample as it produces each output sample. Each output sample depends on only the last 7 input samples. There is no feedback from the output to the input.

FPGA (field programmable gate array) — An integrated circuit that can be programmed to glue multiple logic gates into new hardware functions.

FFT (fast Fourier transform) — A special case of the DFT. It requires a set of samples that is a power of 2 (2, 4, 8,… 1024, and so on). It takes advantage of the exact symmetry of the transform when all samples can be processed as pairs.

IIR (infinite impulse response) — A DSP operation that implements a transform on a sequence of samples using all of the preceding samples. Each output sample is presented to the input at the same time as each input sample. This feedback causes the output to be a function of all preceding input samples.

MAC (multiply/accumulate) — A special DSP instruction that implements a combined math operation that multiplies two numbers and adds the result of the multiplication to one of the original numbers.

SDR (software defined radio) — A radio where most (as close to 100% as possible) of the processing (filtering, gain, modulation, and demodulation) is done using digital hardware and software.

Notes

[1]Go to Michel Barbeau's Web site for a new Linux Kernel that includes SPI support:
www.scs.carleton.ca/~barbeau/SDR/

[2]The Gerber circuit board files are available for download from the ARRL *QEX* Web site. Go to **www.arrl.org/qexfiles** and look for the file **11x09_Mack_SDR_Gerber.zip**

[3]The author had a limited quantity of the circuit boards made, and is offering them at his cost to interested readers. Contact him at the address listed at the beginning of this article for details.

[4]Crochiere and Rabiner, *Multirate Digital Signal Processing*, ISBN 0-13-605162-6. Available from Amazon, **www.amazon.com**

[5]Go to the Analog Devices Web site to get the *VisualDSP* manual: **www.analog.com/static/imported-files/software_manuals/50_blackfin_cc.rev5.1.pdf**

[6]The Analog Devices Web site includes additional manuals for the Blackfin processors: **www.analog.com/en/embedded-processing-dsp/blackfin/processors/manuals/resources/index.html**

[7]Downloading the manual for the Blackfin BF537 EZ-Kit Lite from the Analog Devices Web site: **www.analog.com/static/imported-files/eval_kit_manuals/ADSP-BF537_EZ-KIT_Lite_Manual_Rev_2_4.pdf**