Ray Mack, W5IFS

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

SDR: Simplified

This issue begins a new column dedicated to software defined radios and digital signal processing.

Gerald Youngblood, K5SDR, has done a fantastic job of bringing SDR to the reach of many amateurs with his series of *QEX* articles, and the Flex Radio products.[1] Others have implemented designs like the Soft Rock that use simple RF circuits to get the signals down to "audio" for work with a sound card. We will be looking at techniques that move the RF as close as possible to the antenna. The "Holy Grail" of software defined radios is hooking a DAC and an ADC directly to the transmit/receive switch. Components are getting faster and allowing designs to move higher in frequency and closer to the antenna.

We will start with some experiments using the PC sound card as an arbitrary waveform generator (a fancy signal generator) and using some software tools to look at signals.

Intended Audience

I hope you are included in my target audience. I am writing this column for readers who have heard of software defined radios, and who have read about some of the concepts, but who still feel a bit "in the dark" about how this all works. If you are anxious to take your understanding to the next level, then I believe this column will help you. Here are the skills and tools that will make it possible to fully participate in the presentations and the laboratories:

1) You will need some exposure to the *C* language or *Pascal/Delphi*, but the examples can also help you learn these
2) Analog RF experience
3) A PC with a sound card
4) Average high school math ability (trigonometry and algebra)
5) Windows media player or other WAV file player
6) An oscilloscope is nice to have but not required
7) Roxio *Wave Editor* is also nice to have but not required

Most of the folks doing DSP work tend to be math or computer science wizards rather than interested in RF electronics. Usually by the fifth page of most DSP books or articles by those types of authors, my eyes glaze over from the complex math involved. We can't get away from the math entirely, but we can do almost all of the work with just basic high school trigonometry and algebra. We will use some of that trig in our first lab.

Software and Hardware Resources

We are really lucky that almost all of the software tools we need are free. Here is a list of packages you should download:

1) *Visual C++* — **www.microsoft.com/express/product/default.aspx**
2) *Visual Basic* — **www.microsoft.com/express/product/default.aspx**
3) *uCLinux* for the Blackfin DSP IC — **//blackfin.uclinux.org/gf/**
4) SDR: Simplified Utilities in the *QEX* files section of the ARRL Web site, **www.arrl.org** (These files will be updated concurrent with each installment).[2]
5) *Gnuplot* — **www.gnuplot.info/**

In future installments, we will be doing laboratories with the Analog Devices Blackfin 537 IC. This is a digital signal processor with hardware support for high speed signals and also some regular computer CPU features. This is in contrast to a PC and its sound card, where it is a computer first and signal processor second. The Blackfin BF537 is available from DigiKey on the BF537 Stamp evaluation board, which costs $212.50. We will also need the AD7476 ADC board ($32) and we will make a DAC output board. There are boards for various processors in the EZ Kit Lite series, but those are higher priced. The Blackfin family is used by Yaesu and Icom for IF DSP. TenTec uses the Sharc family for IF DSP.

We could have chosen DSP evaluation boards from TI or Freescale, but Analog Devices has "student priced" boards that are readily available from DigiKey, and there are open source tools available for the Analog Devices boards. There are open source tools for the other manufacturers, too. It is good that there are open source tools because buying most of the manufacturers' proprietary software tools would require a second mortgage!

Paul Decker, KG7HF, is a manager in the Analog Devices DSP software tools group, and he has volunteered to help us to learn DSP. Many thanks to him for his help so far, and for his continued support.

Introduction to I and Q

We need to start with some background on the basics of signal processing before we get into the nitty gritty of doing the processing digitally. If you have done much reading on DSP or software defined radios, you have seen mention of I and Q. There is significant mention of I and Q in Chapter 16 of the 2009 edition of *The ARRL Handbook*.[3] The "I" stands for "in-phase" and the "Q" stands for "quadrature." ***Quadrature*** is simply a mathematical term that means 90° out of phase, so our first trig identity is:

$$\cos(A) = \sin(A + 90°)$$

The general convention is backwards to the way most of us think of trigonometry. The cosine wave is considered the "in-phase" reference and the sine wave is the "quadrature" term. By convention, RF folks use *sine wave* to talk about our waveforms, but the math works better when we think in terms of cosine waves. The math reason for using the cosine as the reference is that the cosine is symmetrical about time zero, while the sine is not.

The next identity is the basis for everything related to quadrature analysis:

$$X \cos(A + y) = I \cos A + Q \sin A$$

where $A = 2 \times \pi \times \text{frequency}$

$$X = \sqrt{I^2 + Q^2}$$

X = amplitude of the new waveform

y = arctan (Q / I) = phase offset from the reference cosine wave

Translated: you can create a new sine wave of any amplitude and any phase relative to the reference cosine wave by adding a sine wave and a cosine wave of the same frequency.

When we modulate a signal, we can vary the phase of the modulated signal (changing A + y) or vary the amplitude (changing X) or vary both at the same time. Varying I and Q simultaneously to modulate a carrier is called Quadrature Amplitude Modulation (QAM). All forms of "traditional" modulation such as SSB, FM, and digital phase modulation can be created by applying the proper I and Q signals to a QAM modulator. When we want to recover the information from a modulated signal, we may only need amplitude information (full carrier AM) or phase information (FM and PM). Some modulation schemes require knowing both the phase and the amplitude to recover the information. Double Sideband-suppressed carrier AM is an example. Many digital modulation methods use complex combina-

[1] Notes appear on page 56.

tions of phase and amplitude to transmit multiple bits simultaneously.

We can use the I and Q signals in receivers to recover both the amplitude and the phase of the information of modulated signals in a process that is the mathematical inverse of the modulation process. When we start doing the math using DSP, we can frequently do the inverse operation digitally, where an analog method is either impossible or very expensive.

Using Gnuplot

We can use *Gnuplot* (which is also a part of the *Octave* math package) to see that our quadrature formula is actually true. *Gnuplot* turns your computer into a graphing calculator. Download the *Gnuplot* installation file from the **www.gnuplot.info/** Web site and install it on your computer. Search your C: drive for **wgnuplot.ini** after you have installed *Gnuplot*. Save the original **wgnuplot.ini** file in a safe place on your C: drive (or use an alternate name to backup the file). One of the files in the zip file available for download from the ARRL *QEX* Web site is **iq.plt**. After you download the zip file and extract the files, copy the **iq.plt** file to the bin directory of your *Gnuplot* installation. There is also a file named **wgnuplot.ini** in the *QEX* files download zip file. Replace the original version on your computer with this new **wgnuplot.ini** file from the *QEX* files. Launch **wgnuplot.exe**. At the command prompt, type **pwd** and verify that the directory is **C:\gnuplot\bin**. If the path is not **C:\gnuplot\bin**, type the command **cd "c:/gnuplot/bin."** Then type **load "iq.plt."** Note that the quotes around the file names are required for proper operation. This command runs the commands in the batch file **iq.plt** and displays a graph, as shown in Figure 1. These commands take a cosine of amplitude 3 (shown with long dashes) and a sine of amplitude 4 (shown as a solid line) and create a new wave of amplitude 5 (shown as a dotted line). Notice that the dotted-line waveform is shifted with respect to the reference cosine wave by positive 53°, which is (arctan (4/3)).

Gnuplot comes from the *Unix* world, so interactions with the computer may seem unusual. The first place this is true is file paths. You need to use the *Unix*/*Linux* convention for path names. In *Windows*, you would type **C:\gnuplot\bin**. In *Gnuplot*, the same directory is entered as **C:/gnuplot/bin**. Notice that the **pwd** command produces a path with Microsoft backslashes but you must use the *Unix* style forward slashes for any command options you enter. Fortunately, the designers have added some features from the *Windows* command system, so repeating and modifying commands is easy. The most useful are the command repeat and editing keys. Up arrow and down arrow will cycle

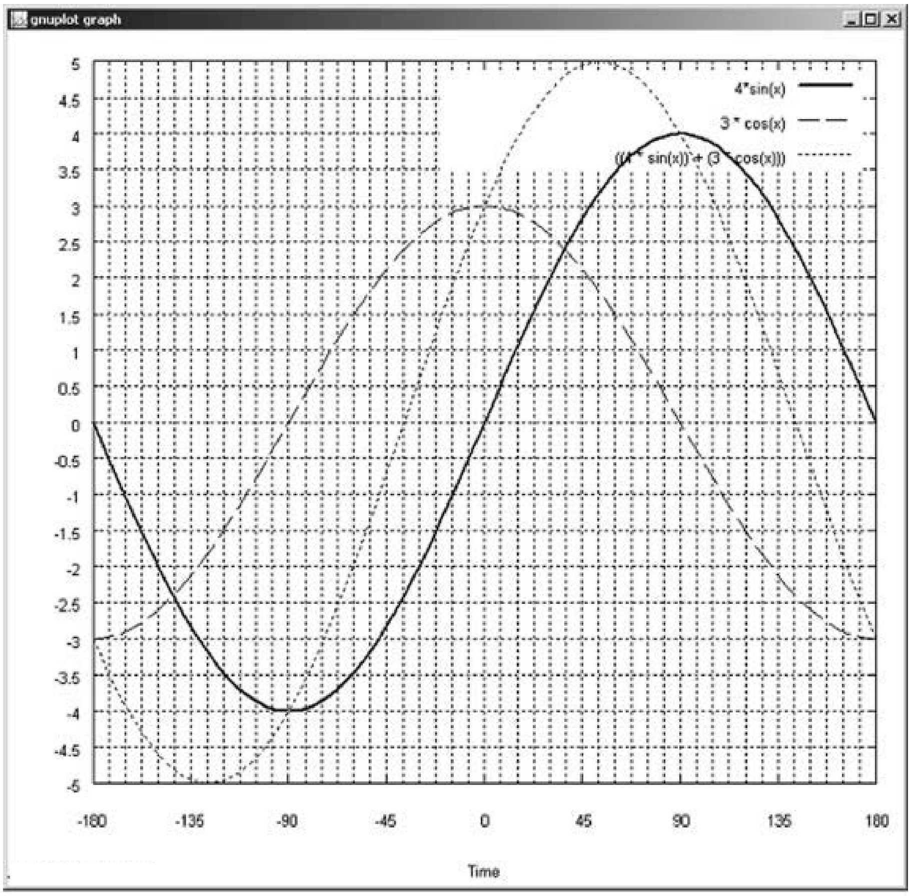

Figure 1 — This screen capture shows the *Gnuplot* graph from a file that plots a cosine with an amplitude of 3 (dashed line), a sine with an amplitude of 4 (solid line) and the sum of those waveforms (dotted line).

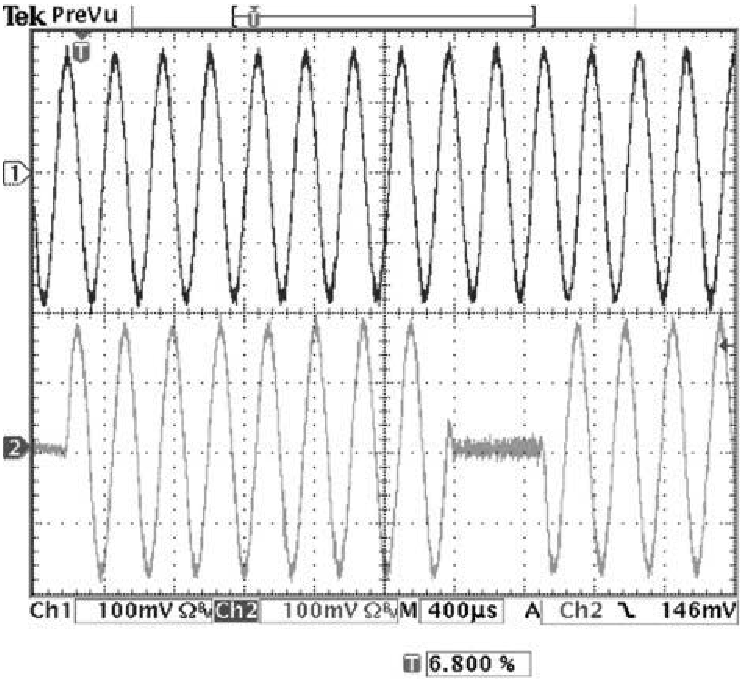

Figure 2 — This oscilloscope display shows a cosine waveform at trace 1 and a phase shift keyed waveform at trace 2. Notice that trace 2 is delayed 90° from the cosine reference at first, and then the phase shifts part way through the sweep, so that the trace is now shifted 90° ahead of the cosine reference signal.

you through the command history. This is very useful as you edit your batch file and rerun it to see the results.

You set the appearance of the plot with the set commands. The last command beginning on line 15 of **iq.plt** tells *Gnuplot* what to plot in the graph window. Notice in our example that there are "\" characters at the end of lines 15, 16, and 17. These allow you to split a single command across multiple physical lines (another Unix/Linux concept). The second thing to notice is that each waveform equation is separated from the next by a comma. If you leave out the comma, you will get an unhelpful error message. You can see this if you take out the comma on line 17 of **iq.plt**. The error says "line 18: undefined variable: x" even though the error is really on line 17. You just have to struggle through trying to find what the problem is. This is a "normal" programming problem where the error is on line 17, but the software doesn't see the problem until the next line. You may encounter similar issues when we start writing programs in *C*.

There are some aspects of plots that must be set in **wgnuplot.ini**. Each computer and operating system has a different "terminal." The lines that you use for plotting in Windows are set up with the variables "Line1" through "Line15." The first 3 numbers set the color of the line (R, G, and B). The fourth number sets the line density for solid lines and the fifth sets the type. The types are 0 — solid, 1 — dash, 2 — dot, 3 — dashdot and 4 — dashdotdot.

Gnuplot has a reasonable set of help pages, especially when you consider that the folks who write the software do it for free. If you have good programming experience, the commands are pretty easy to figure out. There are a very large number of features for 2D and 3D plotting, as well as both polar and rectangular presentations.

Experiments with I and Q

We can generate I and Q signals with the PC sound card by creating WAV files and playing them. Again, all of the files are on the ARRL Web site under *QEX*. Our experiment for this issue will be to create a Binary Phase Shift Keyed waveform. We will define a Zero to be a +45° angle with respect to the cosine and One to be a –135° angle with respect to the cosine. The amplitude of the waveform during each bit will be 1. The file **iq_wave.c** creates a *C* program that will generate and play a demonstration of I-Q modulation. The left channel produces the reference cosine, and the right channel produces our modulated signal. You can see the two waveforms in both the oscilloscope plots of Figures 2 and 3, and the Sound Editor screen shot of Figure 4. Notice that the screen shot shows the individual samples. The PC sound card

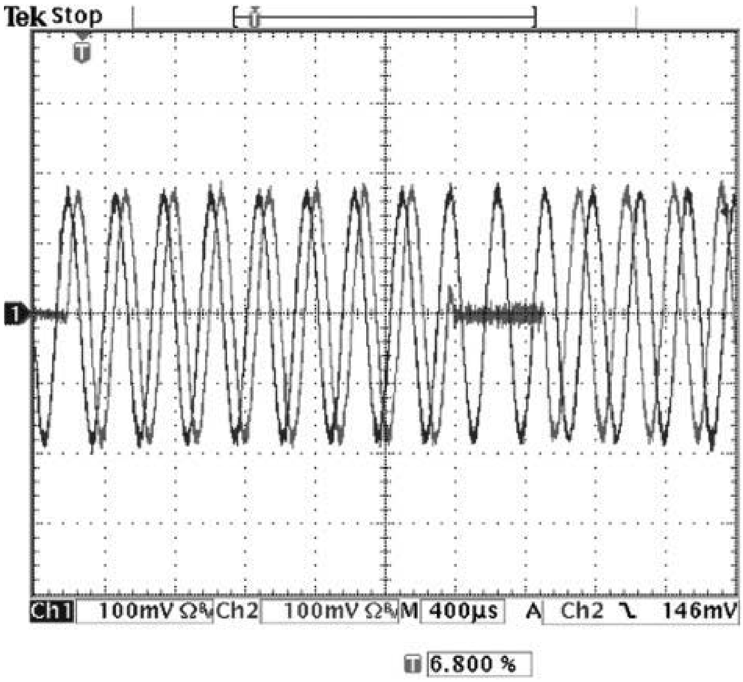

Figure 3 — This oscilloscope display makes it easier to see the phase shift that occurs in trace 2 as compared to the reference cosine waveform.

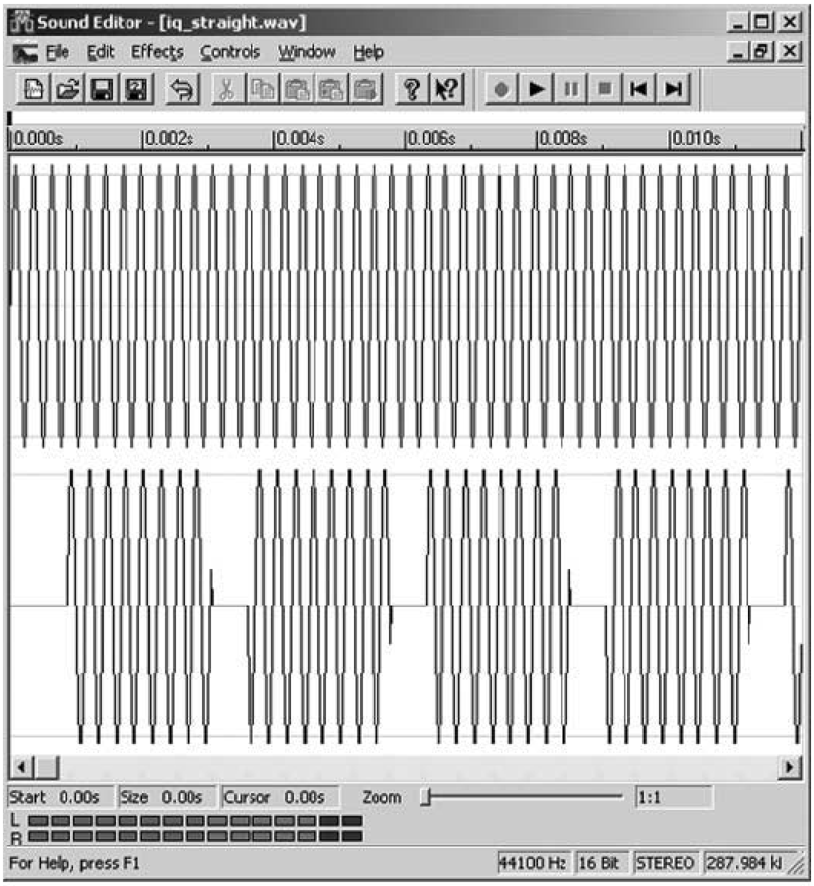

Figure 4 — This screen capture from the Roxio *Wave Editor* display shows a cosine reference signal at the top and a phase shift keyed signal at the bottom.

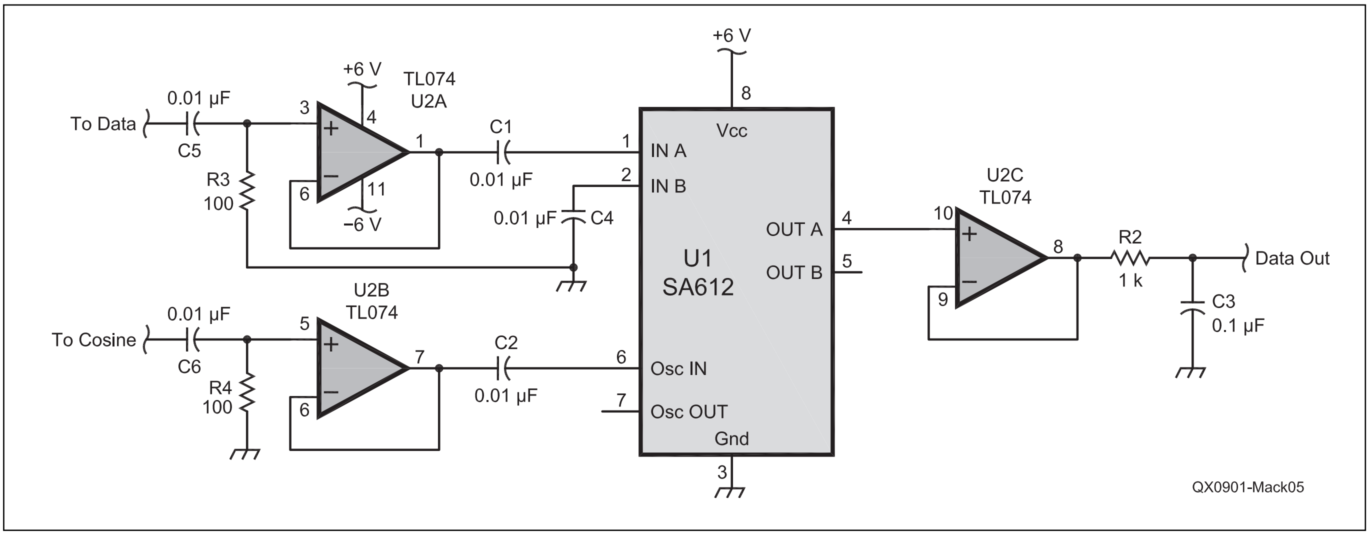

Figure 5 — This schematic diagram shows a simple demodulator that will recover the I signal from a modulated waveform. In the next installment of SDR: Simplified, we will use this circuit to analyze some signals.

filters the samples and smoothes the samples into sinusoids.

The *C* program is rather simple. It uses data arrays of interleaved left and right samples that I write to the output file in sequence. The logic is much simpler, since the program is just a sequence of writes. I used the values in an *Excel* spreadsheet to calculate the samples for the cosine, the sine and the one and zero data values. If you don’t have *Excel*, you can see the equations I used under each heading in the PDF file. The *C* program can be set up for either 48 kHz sampling to get an exact 8 kHz carrier, or you can use 44.1 kHz, to create a CD-compatible WAV file. The *Excel* spreadsheet computes the sample values of ((cos(angle) + sin(angle)) / 1.414) for a one. The sample values for a zero are ((–cos(angle) – sin (angle)) / 1.414). Each of these values is rounded to the nearest integer value. The range of values is restricted to –32767 and +32767 to maintain symmetry around zero.

Don’t be surprised if your waveforms don’t match up. My Dell computer running *Windows 2000* allowed *Musicmatch Jukebox* and Roxio *Wave Editor* to play the 48 kHz file. My HP computer running *Windows XP Professional* would only play the 44.1 kHz file using *Windows Media Player*. Another difference between computers was that the Dell played the signals with exact phase and the HP played the signals with a 180° phase shift. The differences can be attributed to driver differences and hardware differences.

We can build simple hardware that will use the reference cosine to convert the modulated BPSK signal back to the original binary signal. This hardware is your laboratory assignment until the next column. The schematic of Figure 5 shows a simple demodulator that just recovers the I signal. We don’t have enough channels in the PC sound card to generate both the I and Q reference. With BPSK, we really only need one clock that is synchronized to the data, and our reference cosine provides that sync.

In the next issue, I will explain the Nyquist criteria and begin our work with the Blackfin part.

Notes

[1] Gerald Youngblood, AC5OG (now K5SDR), A Software Defined Radio for the Masses, Parts 1 – 4,” *QEX*, Jul/Aug 2002, Sep/Oct 2002, Nov/Dec 2002, Mar/Apr 2003.

[2] The software files associated with this column are available for download from the ARRL QEX Web site. Go to **www.arrl.org/qexfiles** and look for the file **1x09_SDR_Simplified.zip**.

[3] Mark Wilson, K1RO, Ed, *The ARRL Handbook*, 2009 Edition, ARRL, 2008 ISBN: 0-87259-139-5; ARRL Publication Order No. 1395, $44.95. ARRL publications are available from your local ARRL dealer, or from the ARRL Bookstore. Telephone toll-free in the US 888-277-5289, or call 860-594-0355, fax 860-594-0303; **www.arrl.org/shop**; **pubsales@arrl.org**.