John Petrich, W7FU

1629 E. Lake Sammamish Pkwy NE, Sammamish, WA 98074; **petrich@u.washington.edu**

Tom McDermott, N5EG

3950 Southview Ter, Medford, OR 97504; **n5eg@arrl.net**

Digital Signal Processing (DSP) Projects: Examples of GNU Radio and GRC Functionality

After introducing us to GNU Radio Companion and leading us through the software installation in Part 1, the authors will lead us through several design examples and show us some of the things we can do with the software.

GNU Radio is the open source DSP software library that uses the GNU Radio Companion (GRC) graphical interface to implement DSP systems depicted as a visual flow graph on a computer screen. The examples of DSP projects begin with a visual orientation to the GNU Radio Companion graphical user interface (GUI) and detailed explanation of basic GNU Radio Companion terminology. Following the orientation to the GNU Radio Companion graphical user interface are illustrations of typical DSP flow graph applications that highlight some of the capabilities of GNU Radio Companion: a basic DSP that generates a 1 kHz tone internally and outputs the tone to the computer sound card, and more complex DSP back ends for FM and AM SDR receivers.

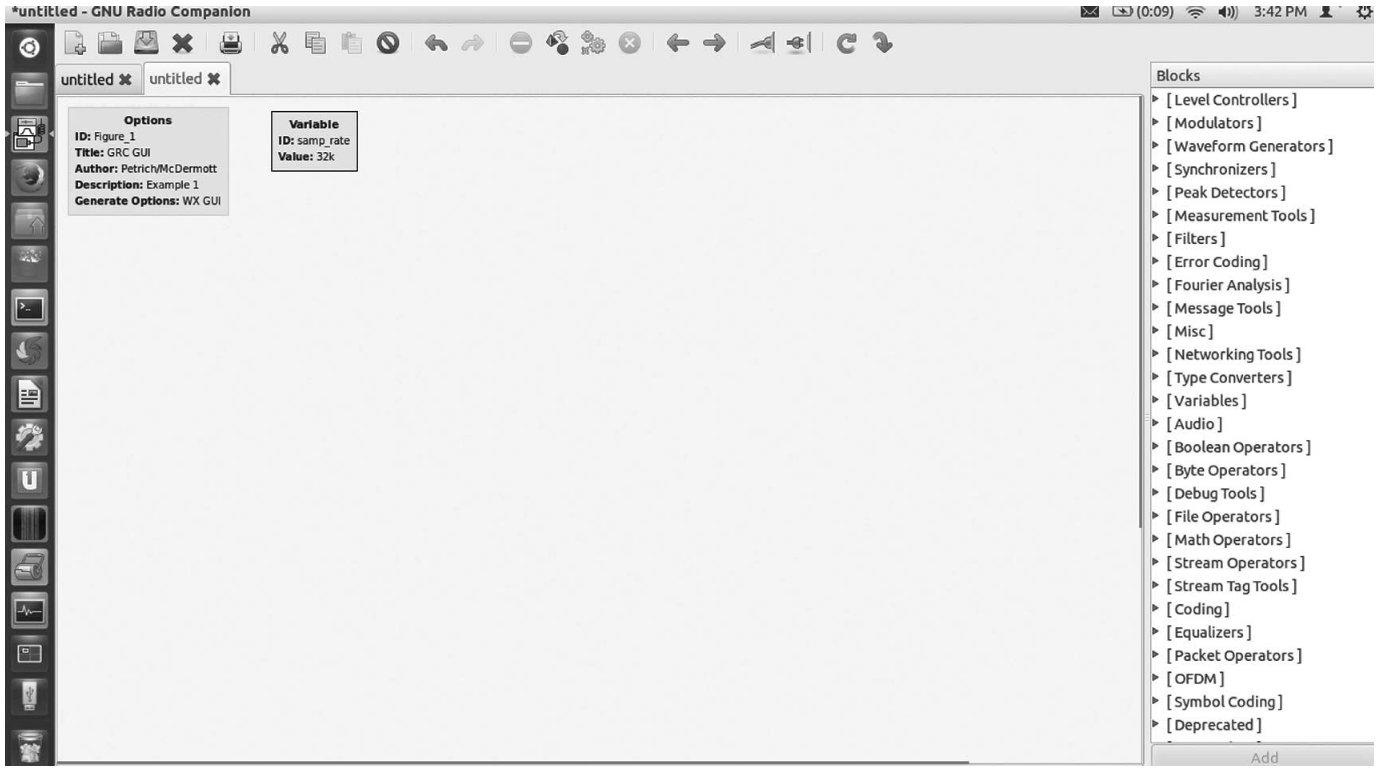
Example 1: Orientation to the GNU Radio Companion GUI

GNU Radio Companion opens on the computer screen with the graphical user interface depicted in Figure 1. The relatively empty center of the screen is the "active" work area. The active work area is the location where flow graphs are constructed and executed. The far right hand margin of the screen is the location of the DSP library which contains the DSP blocks. The library is organized into broad functional categories arranged vertically. Across the top of the screen is a menu bar that contains the buttons that reflect the available tools and operational commands for GNU Radio Companion. Of particular importance are the colorful "generate" button in the center of the menu bar, and the grayed out "execute" button just to its right. The generate button compiles the graphical flow graph into *Python* code and the execute button activates the DSP for signal processing. In the upper left hand corner of the active area is a small "Variable" block where the user can enter the DSP sample rate to reflect the desired digital bandwidth for the signal being processed.

To use this graphical user interface, you author a DSP flow graph in the active work area in a stepwise and logical manner. The first task is to enter the desired DSP sample rate variable for the bandwidth of the signal you want to process. The Variable block contains a variable named "samp_rate" that can be used within the DSP flow graph wherever a numeric value might be entered (rather than hard coding the sample rate in all block locations). Other variables can be defined and even linked to additional GUI blocks, GUI sliders or GUI choosers, which the user can change from the GUI while the DSP is operational. The next step is to place the desired DSP block from the right hand library into the active work area. A double click on the desired DSP library block is all that is needed to cause the desired block to automatically appear in the active work area. The DSP block can be moved within the active work area with a simple drag and drop technique. To open the parameter menu for any block in the active area, the user simply double clicks on the block.

Once double clicked, the parameter menu for that block appears in the center of the screen. With the parameter menu open, the user must enter the "Type" (data stream format — complex, float, or integer), and other desired block parameters into the appropriate spaces in the menu. Clicking "ok" or <Enter> closes that block parameter menu and allows the user to continue authoring the DSP flow diagram. The process of moving desired library blocks into the active work area, followed by entering the needed numeric values or variable names into the block parameter menu is repeated until all the needed blocks for the desired DSP flow graph are placed and configured within the active area.

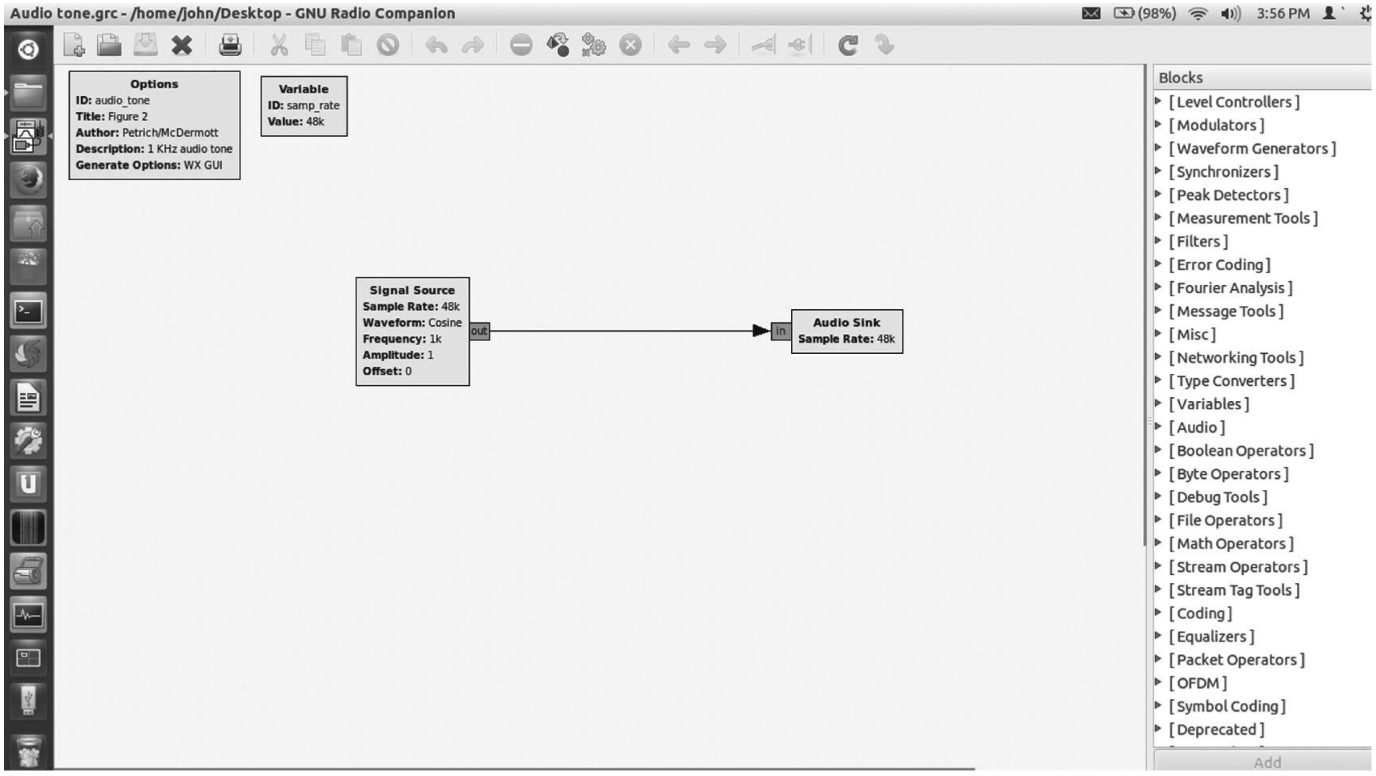
When all the necessary blocks are placed and configured in the work space, the block ports must be connected in a series fashion with arrow links. Check to make sure there are no errors in your intended flow diagram. GNU Radio Companion has an automatic error message system that identifies errors in real time. If there is an error in the block parameters, or a missing input or output connection, GNU Radio Companion colors



Options
ID: Figure_1
Title: GRC GUI
Author: Petrich/McDermott
Description: Example 1
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Figure 1 — This screen shot is the opening screen of the GNU Radio Companion graphical user interface.



Options
ID: audio_tone
Title: Figure 2
Author: Petrich/McDermott
Description: 1 KHz audio tone
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 48k

Signal Source
Sample Rate: 48k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Audio Sink
Sample Rate: 48k

Figure 2 — This GNU Radio Companion screen shot shows a basic Source and Sink flow graph.

the title of the block red. Blocks can also write error messages to the GNU Radio Companion status window located in a panel below the active work area. Once errors are eliminated, compile the underlying code using the "Generate" button from the menu bar at the top of the screen. To make the DSP flow graph operational, the "Execute" button to the right of the generate button executes the flow graph. An excellent video series by Balint Seeber depicts the GUI and the steps necessary to create functional DSP flow graphs.[1]

Example 2: A Basic "Source" and "Sink" Flow Graph

Begin using GNU Radio Companion with a simple DSP project such as an audio waveform generator driving a speaker. Figure 2 illustrates such a DSP flow graph. The exact same GNU Radio Companion graphical user interface is rendered as was shown in Figure 1, but now, the work area contains the DSP blocks necessary to generate and output the tone. The sample rate in the Variable box at the upper left of the active area was set to 48000 Hz to match the sample rate of the sound card in the test computer. The Waveform generator is found in the intuitively named "Waveform Generators" category in the DSP library. This will create the Signal Source block in our flow graph.[2] You can find the Audio Sink block in the "Audio" category of the DSP library, and it is connected internally to the computer sound card.[3]

[1]Notes appear on page 30

The source parameters are set for a "float" digital format "Type," a cosine waveform, a frequency of 1 kHz and a sample rate to the same value as is set in the sample rate "Variable" box, 48 kHz. The parameters for the "Audio Sink" are set complementary to the source parameters in this example. The DSP is activated with the execute button in the menu bar. If you are following along with this example on your computer, you will hear a 1 kHz tone from your computer speaker. You can adjust the output volume using your computer sound card volume control.

Example 3: A VHF FM Receiver DSP Using a TV Dongle Front End

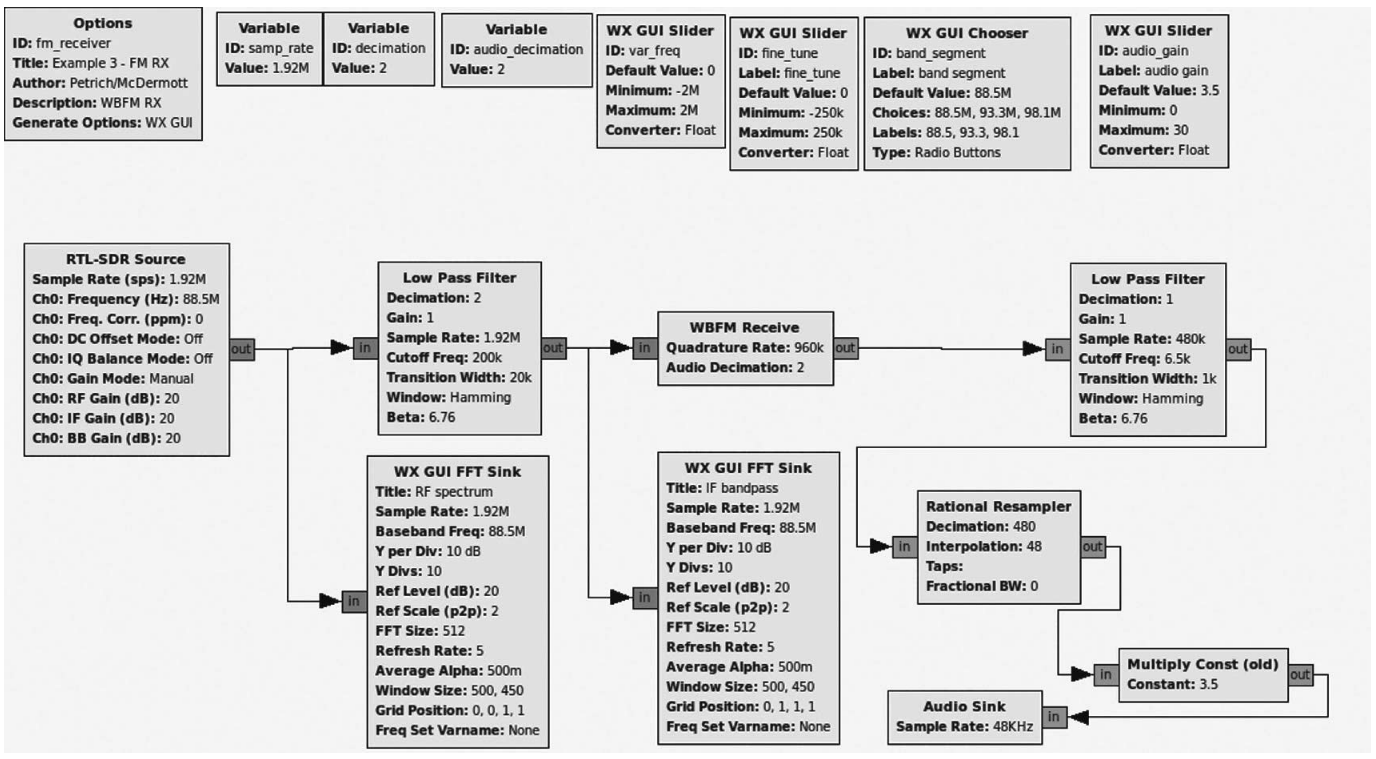
Figure 3 differs from the previous figures, and only shows the active work area of GNU Radio Companion, without the library on the right side of the screen. This example presents a more complex DSP flow graph: the DSP and control panel component blocks of an FM receiver that uses the inexpensive TV Dongle as the SDR front end.

This DSP flow graph demodulates the received signals and adds provisions for a visual display of the RF and IF passband, as well as a receiver control panel. These features are visible when the flow graph is executed and the receiver is operational.

The flow graph was constructed using the process described in the previous example. The TV Dongle block is found in the "Sources" library category and named "RTL-SDR" block. The parameter menu is set to tune the receiver to the FM broadcast band. Note the frequency parameters visible in the RTL-SDR Source block: Ch0 Frequency = 88.5 MHz. The RF and IF gain parameters for the RTL-SDR Source block are set by the user for optimum performance.

Following the digital stream flow graph from left to right, the digital output from the source goes through a "Low Pass Filter" block, and then the information from the signal is extracted using a wide band FM demodulator ("WBFM Receive" block). The demodulated digital information continues through another "Low Pass Filter" block and a "Rational Resampler" block before going to a volume control ("Multiply Constant" block) and finally to an "Audio Sink" that completes the audio chain.

Fast Fourier Transform (FFT) display blocks are positioned before and after the



Options
ID: fm_receiver
Title: Example 3 - FM RX
Author: Petrich/McDermott
Description: WBFM RX
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1.92M

Variable
ID: decimation
Value: 2

Variable
ID: audio_decimation
Value: 2

WX GUI Slider
ID: var_freq
Default Value: 0
Minimum: -2M
Maximum: 2M
Converter: Float

WX GUI Slider
ID: fine_tune
Label: fine_tune
Default Value: 0
Minimum: -250k
Maximum: 250k
Converter: Float

WX GUI Chooser
ID: band_segment
Label: band segment
Default Value: 88.5M
Choices: 88.5M, 93.3M, 98.1M
Labels: 88.5, 93.3, 98.1
Type: Radio Buttons

WX GUI Slider
ID: audio_gain
Label: audio gain
Default Value: 3.5
Minimum: 0
Maximum: 30
Converter: Float

RTL-SDR Source
Sample Rate (sps): 1.92M
Ch0: Frequency (Hz): 88.5M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 20
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Low Pass Filter
Decimation: 2
Gain: 1
Sample Rate: 1.92M
Cutoff Freq: 200k
Transition Width: 20k
Window: Hamming
Beta: 6.76

WBFM Receive
Quadrature Rate: 960k
Audio Decimation: 2

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 480k
Cutoff Freq: 6.5k
Transition Width: 1k
Window: Hamming
Beta: 6.76

WX GUI FFT Sink
Title: RF spectrum
Sample Rate: 1.92M
Baseband Freq: 88.5M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 20
Ref Scale (p2p): 2
FFT Size: 512
Refresh Rate: 5
Average Alpha: 500m
Window Size: 500, 450
Grid Position: 0, 0, 1, 1
Freq Set Varname: None

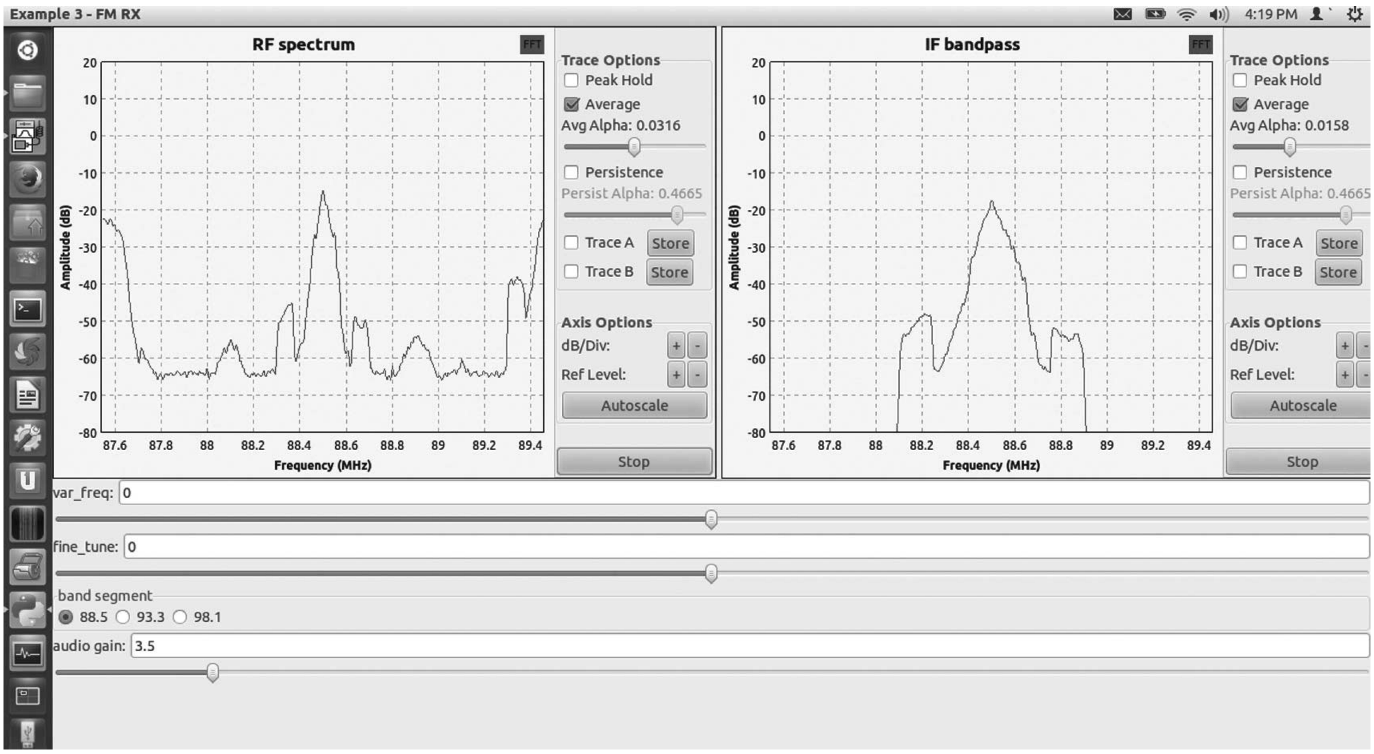
WX GUI FFT Sink
Title: IF bandpass
Sample Rate: 1.92M
Baseband Freq: 88.5M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 20
Ref Scale (p2p): 2
FFT Size: 512
Refresh Rate: 5
Average Alpha: 500m
Window Size: 500, 450
Grid Position: 0, 1, 1, 1
Freq Set Varname: None

Rational Resampler
Decimation: 480
Interpolation: 48
Taps:
Fractional BW: 0

Multiply Const (old)
Constant: 3.5

Audio Sink
Sample Rate: 48KHz

Figure 3 — Here is the DSP flow graph of a simple VHF FM Receiver.



RF spectrum: Avg Alpha: 0.0316, Persist Alpha: 0.4665

IF bandpass: Avg Alpha: 0.0158, Persist Alpha: 0.4665

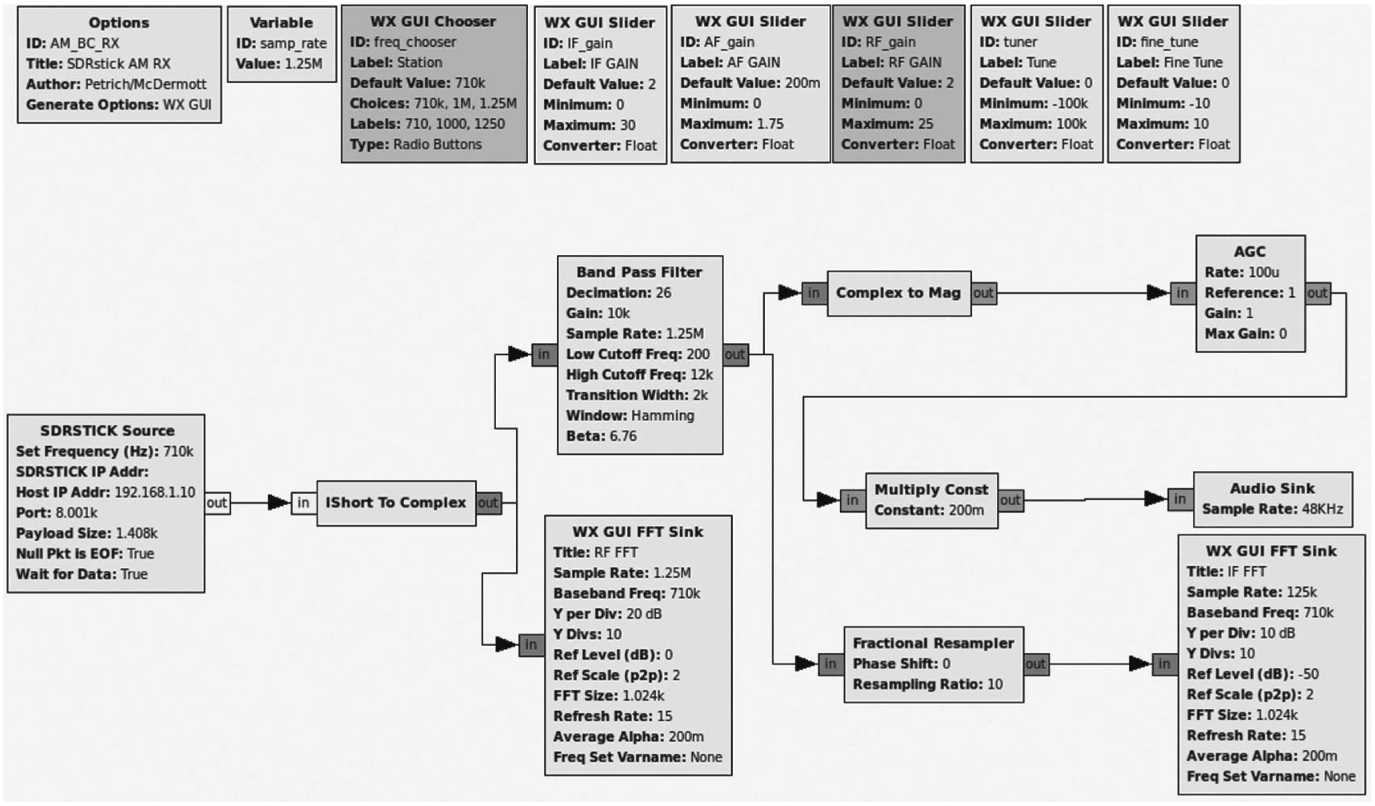
var_freq: 0

fine_tune: 0

band segment: [✓ 88.5] 93.3 98.1

audio gain: 3.5

Figure 4 — This screen shot is of the Display and Control Panel for the VHF FM receiver while it is operating at 88.5 MHz.



Options
ID: AM_BC_RX
Title: SDRstick AM RX
Author: Petrich/McDermott
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1.25M

WX GUI Chooser
ID: freq_chooser
Label: Station
Default Value: 710k
Choices: 710k, 1M, 1.25M
Labels: 710, 1000, 1250
Type: Radio Buttons

WX GUI Slider
ID: IF_gain
Label: IF GAIN
Default Value: 2
Minimum: 0
Maximum: 30
Converter: Float

WX GUI Slider
ID: AF_gain
Label: AF GAIN
Default Value: 200m
Minimum: 0
Maximum: 1.75
Converter: Float

WX GUI Slider
ID: RF_gain
Label: RF GAIN
Default Value: 2
Minimum: 0
Maximum: 25
Converter: Float

WX GUI Slider
ID: tuner
Label: Tune
Default Value: 0
Minimum: -100k
Maximum: 100k
Converter: Float

WX GUI Slider
ID: fine_tune
Label: Fine Tune
Default Value: 0
Minimum: -10
Maximum: 10
Converter: Float

SDRSTICK Source
Set Frequency (Hz): 710k
SDRSTICK IP Addr:
Host IP Addr: 192.168.1.10
Port: 8.001k
Payload Size: 1.408k
Null Pkt is EOF: True
Wait for Data: True

IShort To Complex

Band Pass Filter
Decimation: 26
Gain: 10k
Sample Rate: 1.25M
Low Cutoff Freq: 200
High Cutoff Freq: 12k
Transition Width: 2k
Window: Hamming
Beta: 6.76

WX GUI FFT Sink
Title: RF FFT
Sample Rate: 1.25M
Baseband Freq: 710k
Y per Div: 20 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Average Alpha: 200m
Freq Set Varname: None

Complex to Mag

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 0

Multiply Const
Constant: 200m

Audio Sink
Sample Rate: 48KHz

Fractional Resampler
Phase Shift: 0
Resampling Ratio: 10

WX GUI FFT Sink
Title: IF FFT
Sample Rate: 125k
Baseband Freq: 710k
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): -50
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Average Alpha: 200m
Freq Set Varname: None

Figure 5 — Here is a screen shot of the DSP flow graph of an AM Broadcast Band receiver using the SDRstick™ HF1 SDR hardware.

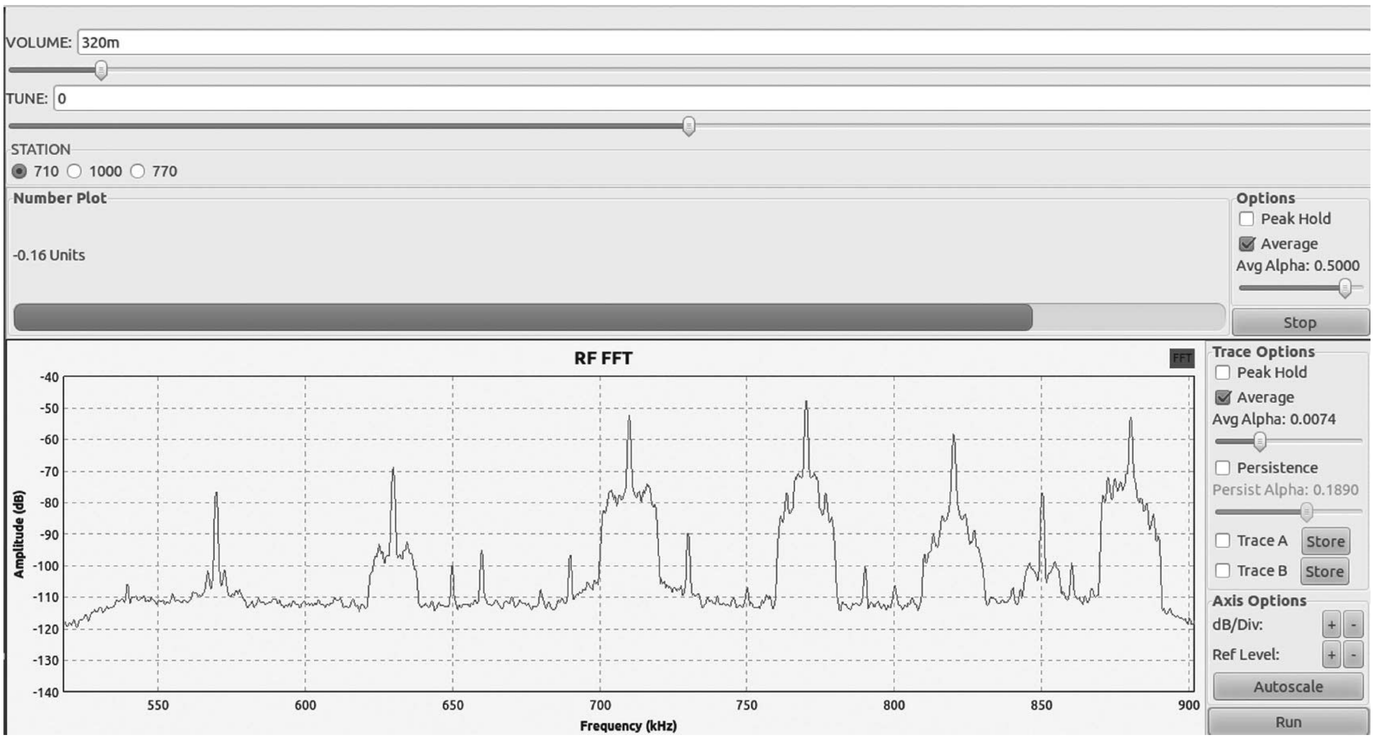


Figure 6 — Here is the RF FFT Display and Control panel for an SDRstick™ HF1 AM Broadcast Band receiver while it is operating at 710 kHz.

first low pass filter to display the RF and IF passbands. Upon executing the DSP flow graph, the FFT and control screen shown in Figure 4 is displayed, overlaying the flow graph.

Figure 4 shows the receiver control panel and both the RF FFT and IF FFT display for the received FM. Looking at the RF FFT display on the left, the main audio channel for the selected station is located in the center of the FFT display. The transmitted information sidebands, visible to the right and left of the main audio channel, contain the stereo and radio data system (RDS) information that is broadcast by the station. Adjacent FM stations are partially visible on the extreme margins of the display. The IF FFT display is on the right. The FM signal is centered in the IF passband.

The sliders and buttons below the FFT displays are user configured controls that permit changing the received frequency and adjustment of the audio volume. Imagine viewing the FFT display, tuning the receiver, and simultaneously hearing the demodulated audio in your headphones or speaker using a custom DSP that you authored!

Example 4: An HF AM Receiver DSP Using the SDRstick™ HF-1 Front End

Figure 5 depicts the flow graph in the active work area for the final example: a DSP authored to receive and demodulate AM broadcast band signals using the SDRstick™ HF1 as the SDR front end.[4] The SDRstick™ HF1 is a recently available HF SDR front end that was designed for use with a GNU Radio Companion DSP back end. The arrangement of this flow graph is similar to the arrangement of the flow graph for the FM receiver. In this flow graph, the AM demodulator, "Complex to Mag" block, replaces the FM demodulator, along with adding refinements such as AGC and a custom calibrated IF FFT display. Note the numeric variables visible in the "Band Pass Filter" block, which define the bandwidth, stop band, and filter shape. Variables are visible in other blocks as well. The digital stream "Type" is also different than Example 3. In this example the "Type" is "Interleaved Shorts" (IShort), which is the digital stream "Type" from the SDRstick™ source block to the remainder of the DSP. This digital stream Type is changed from Interleaved Shorts to complex using the "IShort To Complex" block.

Figure 6 illustrates the RF spectra of an AM broadcast station centered in the display at 710 KHz. Adjacent AM Broadcast Band signals are visible at lower and higher frequencies. The GUI sliders permit control of audio volume and the receiver tuning frequency as in the previous example of the FM receiver.

Conclusion

Flow graph authoring is easy but not totally intuitive. Developing DSP projects with GNU Radio Companion flow graphs leverages the user's foundational knowledge of signal flow and demodulation theory. GNU Radio Companion is a tool that builds on that knowledge and allows the user to easily implement practical DSP systems. To really learn digital signal processing, you must study DSP concepts and have a reasonable understanding of complex numbers.[5, 6, 7, 8] GNU Radio with GNU Radio Companion makes it easier to learn DSP by allowing you to focus on blocks rather than arcane source code and to develop functional DSP systems with a hands on approach.

More complex DSP flow graphs can be constructed to implement full featured communications oriented DSP systems, receivers and transmitters, for every imaginable modulation type and frequency. The potential is constrained only by the SDR front end hardware and the author's imagination.

There is a lot to learn and we urge you to embrace the journey, because it is very rewarding!

Note: One of the inevitable consequences of writing a "How To" article about dynamic and rapidly changing Open Source software is that changes to the software and the interface often come in rapid succession. Since the publication of Part 1 of this article in the Jul/Aug 2014 issue of *QEX*, there have been at least five version revisions to GNU Radio Companion, and some aspects of GRC have changed. In an effort to provide the latest information and installation instructions for GNU Radio and GNU Radio Corporation, the authors have created a website. Go to **www.w7fu.com** for the latest information.

ARRL member, Official Observer, and Amateur Extra class licensee John Petrich, W7FU, was first licensed as K6OJV in 1955 and then as W7HQJ after moving to Seattle. He is a practicing physician, and Clinical Associate Professor of Psychiatry, School of Medicine, University of Washington. John is active in community affairs, enjoys family life, sea kayaking and cycling. John's radio experience parallels the evolution of radio communications technology over the past century. He started with a homebrew crystal receiver followed by a much loved and modified single tube regenerative receiver in a cardboard box. Upon earning his license, he graduated to operating QRP using a crystal controlled 6V6 tube transmitter constructed on a wooden chassis. Subsequently his rigs evolved from modified surplus gear to home constructed to full featured analog receivers and transmitters. John's first love is CW, the prototypical digital QRP mode. He credits radio with endless opportunities for engaging learning opportunities and long lasting friendships. At present John's rig is an experimenter's station built around the HPSDR Atlas bus system. The station is supplemented with back-up rigs using Ettus and SDRstick™ SDR "front ends" and DSP "back ends" built using GNU Radio Companion. John is interested in communicating with others who have similar interests.

ARRL Life Member, and Amateur Extra class licensee Tom McDermott, N5EG, has been licensed 45 years. He is a member of TAPR, IEEE, and Internet2, and has been involved in the development of fiber optic transmission and switching systems since the initial deployment of single-mode fiber in positions ranging from ASIC designer to CTO. He currently is a participant in the IEEE 802.3 Ethernet 100GE and 400GE standards projects. Tom has a BSEE degree from the University of California, Berkeley, and has written one textbook on wireless digital communications. He's been involved in many computer-related ham projects, from the TEXNET layer 3 packet radio system, to a VNA project, network simulation, and other efforts. His current interest is using a HPSDR Hermes SDR transceiver and GNU Radio to experiment with DSP algorithms.

We Design And Manufacture To Meet Your Requirements
*Protoype or Production Quantities

800-522-2253

This Number May Not Save Your Life...

But it could make it a lot easier! Especially when it comes to ordering non-standard connectors.

RF/MICROWAVE CONNECTORS, CABLES AND ASSEMBLIES

- Specials our specialty. Virtually any SMA, N, TNC, HN, LC, RP, BNC, SMB, or SMC delivered in 2-4 weeks.
- Cross reference library to all major manufacturers.
- Experts in supplying "hard to get" RF connectors.
- Our adapters can satisfy virtually any combination of requirements between series.
- Extensive inventory of passive RF/Microwave components including attenuators, terminations and dividers.
- No minimum order.

NEMAL
Cable & Connectors for the Electronics Industry

NEMAL ELECTRONICS INTERNATIONAL, INC.
12240 N.E. 14TH AVENUE
NORTH MIAMI, FL 33161
TEL: 305-899-0900 • FAX: 305-895-8178
E-MAIL: INFO@NEMAL.COM
BRASIL: (011) 5535-2368

URL: WWW.NEMAL.COM

Notes

[1]Balint Seeber has posted a YouTube video tutorial about how to set up some DSP functions: **www.youtube.com/watch?v=N9SLAnGlGQs**.

[2]A "source block" is defined as having only output ports and is the origin of the digital stream for the DSP.

[3]The term "sink block" is reserved for DSP blocks that only have input ports. The digital stream does not extend out of the sink, only into the sink. An arrow connects the output port of the source to the input port of the sink.

[4]You can learn more about the SDRstick™ HF1 on the SDRstick website: **http://sdrstick.com/**. Scotty Cowling, WA2DFI, also described the SDRstick HF1 and HF2 boards in his article "Hardware Building Blocks for High Performance Software Defined Radios," in the Jul/Aug 2014 issue of *QEX*.

[5]There is a free DSP textbook available on the Internet: *The Scientist's and Engineer's Guide to Digital Signal Processing* by Dr. Steven W. Smith. **www.dspguide.com/**.

[6]*The ARRL Handbook For Radio Communications*, includes a chapter on "DSP and Software Radio Design. H. Ward Silver, NØAX, *Ed*, *The ARRL Handbook For Radio Communications*, ARRL, 2013, ISBN: 978-1-62595-001-7; ARRL Publication Order No. 0007, $49.95. 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; pubsales@arrl.org. **www.arrl.org/shop/2014-Handbook-Softcover-Centennial-Edition/**.

[7]Wes Hayward, W7ZOI, Rick Campbell, KK7B, Bob Larkin, W7PUA, *Experimental Methods in RF Design*, ARRL, 2003, ISBN: 978-087259-923-9; ARRL Publication Order No. 9239, $49.95. **www.arrl.org/shop/Experimental-Methods-in-RF-Design/**.

[8]Doug Smith, KF6DX, *Digital Signal Processing Technology*, ARRL, 2001, ISBN: 978-0-87259-819-5; ARRL Order No. 8195, $34.95. **www.arrl.org/shop/Digital-Signal-Processing-Technology/**.