Contents

Chapter 14 — Online Content

*Also see the Downloadable Content for the **Receivers** and **Transmitters** chapters*

Software Defined Radio and DSP Articles and Projects
- Build Your Own IF SDR by Alex Schwarz, VE7DXW and Guy Roels, ON6MU
- Chapter 14 — Audio Oscillator example.grc (GNU Radio design file)
- Digital Signal Processing and GNU Radio Companion by John Petrich, W7FU and Tom McDermott, N5EG
- Digital Signal Processing (DSP) Projects: Examples of GNU Radio and GRC Functionality by John Petrich, W7FU, and Tom McDermott, N5EG
- Hands On SDR — FPGAs by Scotty Cowling, WA2DFI
- Hands On SDR — Introduction by Scotty Cowling, WA2DFI
- Hands On SDR — More on FPGAs by Scotty Cowling, WA2DFI
- Hands On SDR — Sharing Radios on the Network by Scotty Cowling, WA2DFI
- Hands On SDR — Using FPGAs in SDR Designs by Scotty Cowling, WA2DFI
- Hardware Building Blocks for High Performance SDRs by Scotty Cowling, WA2DFI
- SDR Simplified — Demystifying PID Control Loops by Ray Mack, W5IFS
- SDR Simplified — Step One Toward a Working SDR by Ray Mack, W5IFS

Transverter Articles and Projects
- A 222 MHz Transverter for the Yaesu FT-817 by Paul Wade, W1GHZ
- A 222 MHz Transverter for the Yaesu FT-817 — Revisited by Paul Wade, W1GHZ
- Assembling a Microwave Transverter System by Paul Wade, W1GHZ
- Cheap & Simple 1296 MHz Transverter Update by Paul Wade, W1GHZ
- Corrections and Improvements for Simple and Cheap Multi-band Microwave Transverters by Paul Wade, W1GHZ
- Sequencers for Transverter Control by Paul Wade, W1GHZ
- Smart Fool-resistant Conditional Sequencer by Paul Wade, W1GHZ
- The Transverter — An Introduction to a Useful Device by Bill Wageman, K5MAT
- The Transverter by Tom Williams, WA1MBA
- Universal MMIC Preamp by Paul Wade, W1GHZ

Chapter 14
Transceiver Design Topics

This chapter includes high-level material on SDR design. Steve Hicks, N5AC, contributed the sections on signal chains and interfaces. John Petrich, W7FU, updated the introductory material on *GNU Radio* and expanded the list of reference materials. The additional short introduction to FPGA data engines is based on the series of *QEX* columns "Hands-On SDR" by Scotty Cowling, WA2DFI.

Material on transverters was contributed by Paul Wade, W1GHZ, and Wayne Overbeck, N6NB. Material on adding panadapters to older radios was contributed by Ed Krome, K9EK.

There is a great of deal of material that doesn't fit neatly into the categories of receiving, transmitting, DSP/SDR fundamentals, or any of the major radio functions. This information is still required, however, to integrate all of the functions into a single package — the transceiver. There are questions of architecture, interfaces, development tools, and many other topics that must be addressed to build an effective transceiver or communications system. That is the purpose of this general-purpose chapter, to create a home in the *Handbook* for material that would be inappropriate for a chapter focused on a particular aspect of the radios themselves. Amateur Radio is poised at the threshold of great innovation and change — it's a bright future!

14.1 Signal Chains in SDR Transceivers

In an SDR, most work in both transmit and receive signal processing is performed in a computing or processing element that operates on digital samples. **Figure 14.1** shows a simplistic diagram of a transceiver setup for phone operation. Looking first at the transmit side of the radio, the analog microphone audio is converted to digital samples in an analog to digital converter (ADC). These digital samples are passed to the processing element where they enter the *transmit signal chain*. The transmit signal chain typically has a series of functional blocks that perform various functions on the signal, including the modulator. While it is possible to use a purely digital exciter, most radios then pass the digital samples into a digital to analog converter (DAC) which turns the digital signal back into analog form. The resulting analog signal is amplified by an RF power amplifier and transmitted through the antenna.

In the receiver, the direction of flow is the opposite: analog RF data enters through the antenna and is turned into digital signals in an ADC. This data is passed to the receive *signal chain* where it is filtered, demodulated and processed into a signal we can hear. In some digital modes, such as RTTY, the demodulated signals are filtered and turned directly into digital symbols without ever being turned back into analog. For modes that we are expected to hear, the digital samples are sent into a codec where a DAC turns them back into analog for playback through headphones, speakers, or via a line-level output.

The transmit signal chain and receive signal chain are at the heart of an SDR. Let's take a closer look at how they operate.

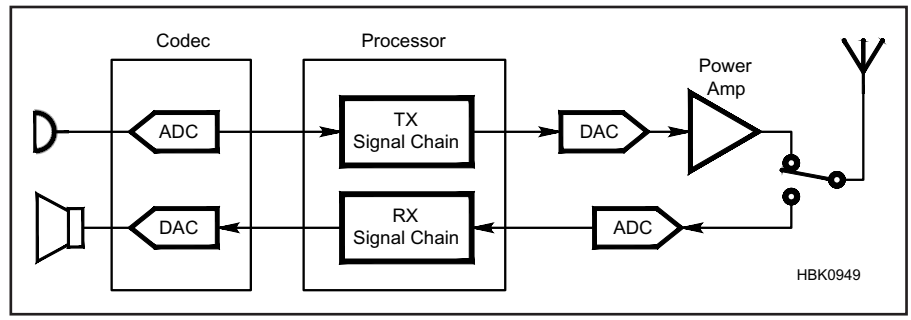


Figure 14.1 — A high-level block diagram of an SDR transceiver configured for phone operation.

14.1.1 Receive Signal Chain

Most receivers will capture more digital samples than are needed for a single receiver. This can happen for many reasons. For example, we might have a wideband sampling receiver that captures a substantial portion of the spectrum and places several receivers in that spectrum. In this case, each individual narrowband receiver only requires a portion of the captured data. When we use the term narrowband receiver, we're talking about a receiver that will be used to demodulate signals in a 50-20,000 Hz slice of the RF spectrum. In another example, the receiver might be designed using an analog or digital mixing technique that reduces the bandwidth to an "audio" range (such as 0-192 kHz). The radio may preserve this larger bandwidth to plot a spectrum display, but will reduce it for narrowband receiver processing.

To understand the next steps, it's important to first understand the relationship between sampling rate and available bandwidth. Generally, in a sampled RF system we will either be using real or complex samples. (See the **DSP and SDR Fundamentals** chapter) *Real samples* will be used for audio input and output. Real samples consist of a single fixed or floating point number that represents each sample.

According to the Nyquist sampling theorem, we can represent at most one half of our sampling rate in bandwidth. For example, if we sample a microphone signal at 48 ksps (real samples), we can represent audio from 0-24 kHz. As a practical matter, the filters we use to process the audio will have a transition band that will further restrict this to something around 20 kHz.

If we sample using the in-phase and quadrature sampling (I/Q) commonly used in RF systems, then each sample consists of a pair of fixed or floating point numbers, one for I and one for Q. This is also commonly referred to as *complex sampling*. In a complex sampled system, sample speed and sampled bandwidth are equal, ignoring filter transition bands. For example, if the sample rate is 48 ksps (complex samples), the samples can represent almost of 48 kHz of RF spectrum (again accounting for filter transition bands). For a better understanding of the reasons behind these general rules there are several excellent references, including Richard Lyons' book *Understanding Digital Signal Processing*, that can be consulted. (See the Reference sections of the **DSP and SDR Fundamentals**, **Receiving**, and **Transmitting** chapters for a comprehensive set of references.

In either situation, the primary need to reduce the bandwidth for the receiver signal chain is driven by the processing resources required to process the receiver signals. The more bandwidth that is captured digitally, the higher the sampling rate and consequently the more processing power required in our signal chain. If we plan to ultimately discard the extra bandwidth in a filter, then we are wasting valuable processing resources by processing a wider bandwidth than we ultimately need.

In any of these cases, our first step is generally to reduce the bandwidth of the incoming signals through decimation. (See the **DSP and SDR Fundamentals** chapter) By decimation, we reduce the number of samples in the receiver by combining multiple samples and producing an output that has fewer samples, but with more bits per sample (greater bit width). After decimation, the receiver will typically have a sampling rate of somewhere between 24 and 48 ksps. We will then process all elements of our signal chain at this rate. While this is not strictly necessary, it is the most convenient. If our sampling rate varies in each block of our signal chain, we will be constantly buffering and resampling, again wasting valuable time and computing resources.

A typical receiver signal chain is shown in **Figure 14.2**, again computing from right to left. (This is the same direction as signal flow in the block labeled "Receive Signal Chain" in Figure 14.1.) Each DSP block in the chain will take input samples, generally in a buffer or block of samples, process the samples using the algorithm for the block, and then pass the resulting samples to the next block. It is a requirement of the receive signal chain that all blocks are processed in less than the time it takes before another buffer of samples arrives.

What does this mean? Let's say that our system is running at 48 ksps and our sample buffers are 512 samples each. We will receive a new buffer every 10.67 ms. If we are not able to process the next sample buffer because we are still processing the last buffer we received, then the buffers will stack up, forming a queue of samples needing processing. Even worse, our DAC at the output of the signal chain has the responsibility for turning a stream of sample buffers into a continuous audio signal. If we "starve" it by not sending a buffer when it has exhausted the previous buffer, it will be forced to play empty audio or repeat previous buffers. If you've ever used digital electronics built this way, you've likely experienced the effects firsthand. Here are some examples of how other engineers dealt with a similar problem:

1. In XM satellite radio, when the satellite signal is briefly lost, some satellite radios play "comfort noise," a low-level hiss that sounds like a normal AM or FM signal that has faded away. This noise is completely synthetic and doesn't represent any actual noise in the receiver. Since the source signal is digital, the receiver knows when the signal has faded and could easily play "blank audio," but it was probably judged that this sound would best convey to the consumer that the radio is working but has lost a signal.

2. Many CD players, when starved for digital data from the disc due to a scratch and the inability to recover the signal, will replay the contents of the last digital buffer, resulting in a repetitive sound reminiscent of a skipping vinyl record player. This sound conveys to traditional (analog) vinyl users that there is a problem with the disc. (Data, however, does not warp when left in the sunlight on a vehicle's back seat!)

It is, therefore, important to budget processing power to ensure all blocks can be processed in the time it takes to collect a new buffer. If our radio has more than one receiver but a single processing element, our requirement will be to process all the DSP blocks for all receivers in the requisite time for a single buffer. While beyond the scope of this discussion, it's worth mentioning that multicore processors, both symmetric and asymmetric, can provide a platform for some unique solutions that stagger or pipeline DSP blocks and receivers and allow spreading the processing requirements across different cores.

PASS-BAND FILTER

Next, we filter the buffered data. The filter removes unneeded signals from the buffer, limiting it to just our bandwidth of interest, called the *receiver pass-band*. Since the samples entering the filter are 48 ksps of I/Q data, we start out with something on the order of 40 kHz of audio bandwidth. If we are using sideband, we're only typically interested in a bandwidth of 2-5 kHz. If we're using CW we might want to reduce the bandwidth down to 50-100 Hz. The filter achieves this reduction in bandwidth by employing DSP filtering techniques. (See the **Analog and Digital Filtering** chapter)

The filter may need to accumulate several buffers of time-domain samples before performing the filter processing depending on how steep or sharp the filter skirts are required to be in order to create the filter's required

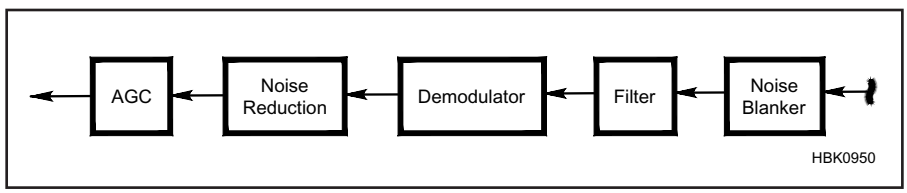


Figure 14.2 — Typical functional blocks that make up a receive signal chain.

shape factor. As mentioned previously, it is also required that once we have accumulated the requisite buffers, we must process the filter within the duration of our buffering. For example, if our buffer size is 512 samples and we are using an overlap-and-save FIR filter with 2048 taps, we will need to accumulate 2048 samples before processing the filter. After this, 1024 samples will be added to the filter for the next processing, since 1024 samples are saved from the last processing. In aggregate, we must be able to process this filter in under 1024/48 ksps = 20.3 ms of CPU time (including all other signal chain operations).

Filters used in receivers can be of many different topologies, including finite impulse response (FIR) and infinite impulse response (IIR). Even within these two broad categories there are plenty of additional variations that affect passband and stop-band ripple, phase distortion (detrimental to some digital modes), group delay, and other parameters. Also, this filter, which is often called the *final filter* since it is the last filter before the signals are presented to the operator, will determine a significant part of the receiver latency (the delay between the input of an RF signal to the output of the data or audio). Why is this? The slower the sampling rate, the more impact collecting samples and filtering them will have on latency.

Some radios allow the operator to make tradeoffs in filter parameters, favoring low latency at one time and favoring the best filtering at another. A sizable FIR filter using 4096 taps will add roughly 1.5 x 4096 / 48 ksps = 128 ms of latency. This is not a large amount for phone conversations, but would be significant for CW contesting.

One of the great triumphs of SDR is that we have eliminated our dependency on needing a collection of electronic or mechanical filters. We now have continuously variable filters that let us tailor our pass-band to meet our operating demands without worrying about the design cost or space for our old electronic and mechanical filter collection. In some radios, we can also adjust the taps in the filter to select better filter shape factor or lower latency as our operations dictate.

DEMODULATOR

The demodulator performs the conversion of on-air signal to the baseband audio signal that we will be hearing. For example, with FM we must convert a fixed-amplitude carrier changing frequency according to the amplitude of the audio into a time-domain amplitude-varying signal that we can understand with our ears and brain. The output of the demodulator is audio that we can hear and understand.

The demodulator block is selected based on the mode being used. When you change the mode of the radio on the front panel, you are switching one demodulator with another for the mode you have just selected. Of course, the filtering and noise mitigation techniques may vary by mode also, and so we may be switching other blocks in the DSP chain based on the requirements of each mode.

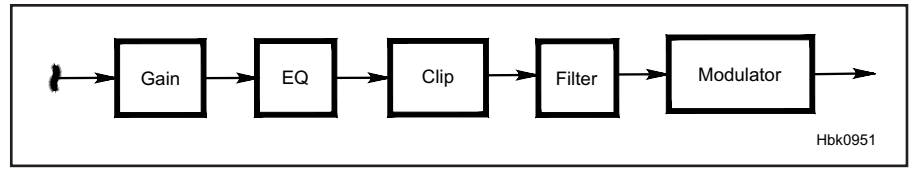


Figure 14.3 — Typical functional blocks that make up a transmit signal chain.

14.1.2 Transmit Signal Chain

For phone, the transmit signal chain will prepare spoken audio for transmission over the air. If our mode is CW or other digital modes, the transmit signal chain will be very different, perhaps just a single or a pair of software oscillators that produce the tones that we will send on air. **Figure 14.3** shows a typical signal chain for phone transmission.

GAIN

Depending on how the designer has implemented the ability to change output power in the radio, a gain stage may exist at the start of the signal chain that can be adjusted to raise or lower the power of the transmitter. This gain stage could be implemented at other places in the radio as well. A gain stage will multiply all samples in the buffer by an input level value, raising or lowering the level of all audio samples in the buffer.

EQUALIZATION

In a software defined radio, it is easy to isolate frequency bands in an audio buffer and allow each band's relative power to be raised or lowered. This equalization function can be used to tailor a person's voice to produce the best sound on the receiving end. This function is known as equalization or simply EQ. The input to the equalization function is the relative increase or decrease for each band of interest. A typical equalizer panel with vertical sliders for each audio band is shown in **Figure 14.4**. At the output of the equalization stage, we have audio prepared for transmission, but it may exceed required output levels that have been set in the radio. For example, we may independently have set a maximum output power for the radio to prevent overshooting an amplifier input. This is where a clipper becomes useful.

CLIPPER / ALC

A clipper may be employed to reduce the level of any signals that exceed required power levels. It generally does this by checking each sample in the buffer for the maximum allowable level and replacing any samples that exceed the threshold with a corresponding sample of the maximum level, thereby clipping the peak value of that sample.

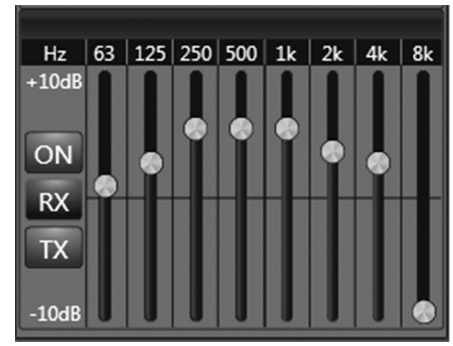


Figure 14.4 — An equalizer control window showing the adjustments for each frequency band of interest.

While clipping can produce undesirable effects, it may be necessary to prevent the radio from operating outside the limits provided by the operator. Automatic level control (ALC) may also be used to reduce signal levels if a downstream amplifier provides voltage to the radio requesting a lowering of the levels.

Generally, a series of meter displays or values derived at different stages in the signal chain will provide the operator with information on undesired effects occurring within the signal chain. This allows the operator to make adjustments to correct any undesired effects. Since we have absolute control over the signal level by altering samples in the radio, it is generally better to set a maximum power out in the radio and allow the signal chain to make the appropriate adjustments than to rely on ALC to adjust the levels when they are too high for an amplifier.

An ALC subsystem that is too active can modulate the transmit signal. This results in distortion that could be prevented by adjusting the proper levels in the signal chain to start with. This is why many digital mode guidelines specify that the ALC system be switched off to avoid unintentional distortion of the signal by power control systems.

FILTER

Since our audio input may contain frequency components well above those required or

that comply with good practices, a filter is employed to remove them. For normal SSB operation, we could set the filter to pass audio from 100 Hz–2.8 kHz. For Extended Signal Sideband (ESSB), we may set the filter to pass audio from 50 Hz-4 kHz or more. The transmit filter setting may also be set to comply with communication laws in the country where the transmission is occurring.

The filter will remove the unneeded frequency components of the signal before transmission using similar DSP techniques to those in the receive signal chain filter.

MODULATOR

The modulator will then take the audio signal and create a modulated RF or IF signal required by the operating mode. The modulator takes care of placing the audio in the proper sideband for SSB, producing the carrier for AM and performing frequency modulation for FM. After modulation, the signal is ready to be transmitted.

SUMMARY

Signal chains are the heart of the software defined radio. This overview provides the basics of how a signal chain is put together and the requirements on each of the DSP functions. A typical signal chain in an SDR may have many more blocks including blocks to meter samples, providing valuable feedback to the operator, more audio tailoring and filtering blocks, oscillators for in-air testing, speech compressors, and so on.

14.2 User Interfaces

Many years ago, the knob we used to tune our radios was often attached to a frequency indicator stenciled on a piece of plastic that would rotate into view as the knob was turned. The plastic marker displayed an offset into a band selected by a band switch. The strength of any incoming signal would be displayed on an S meter, a magnetic meter movement excited by the voltage from the AGC circuit. In the 1970s and 1980s, the *liquid crystal display* (LCD) and *light emitting diode* (LED) display came into popularity, and most radio manufacturers adopted some form of display device that used LCD and or LEDs. Both technologies were used to show frequency, mode selections, and simulated S meters without their mechanical counterparts.

These technologies of the past provided only minimal *situational awareness*. We only had a narrow view of the spectrum limited to the instruments at hand showing us the signal strength in our narrow receiver, along with a display of our current frequency. There was no way to see how many signals were on the band, how strong they were, whether the noise floor in the band was rising, or if we might just be listening where some local noise was causing us interference.

With *panadapters* now available on modern transceivers, we can see all the signals across the band. Finding a place to call CQ can be difficult when the bands are busy. With a panadapter, we can glance at the band and notice where there are gaps, quickly tune to an empty frequency, and call CQ. We can also instantly tell where the strong and weak signals are in the band. If we're looking for signals arriving at our location through particular propagation, we can search for signals that look alike or share the similar power levels.

The *waterfall display* provides the same information with a time-historical twist. In the waterfall we can see how signals and noise are changing over time. Imagine working a DX pileup where a few hundred amateurs are all trying to work the same station. We can watch the DX station systematically contact every station and observe the operating patterns. We know where the DX station is listening because we can see each station in the pileup respond to the DX station as they are called. Observing this pattern, we now know where the DX will listen next and can simply drop our call in this location and make the contact. Those enjoying panadapters today lament: how did we survive without this capability?

The tools available today for gaining a situational awareness of the bands and our receiver have dramatically changed. Let's look at each of these tools in some more detail.

14.2.1 The Panadapter Display

The *panadapter*, or spectrum display, shown in **Figure 14.5** displays the amplitude

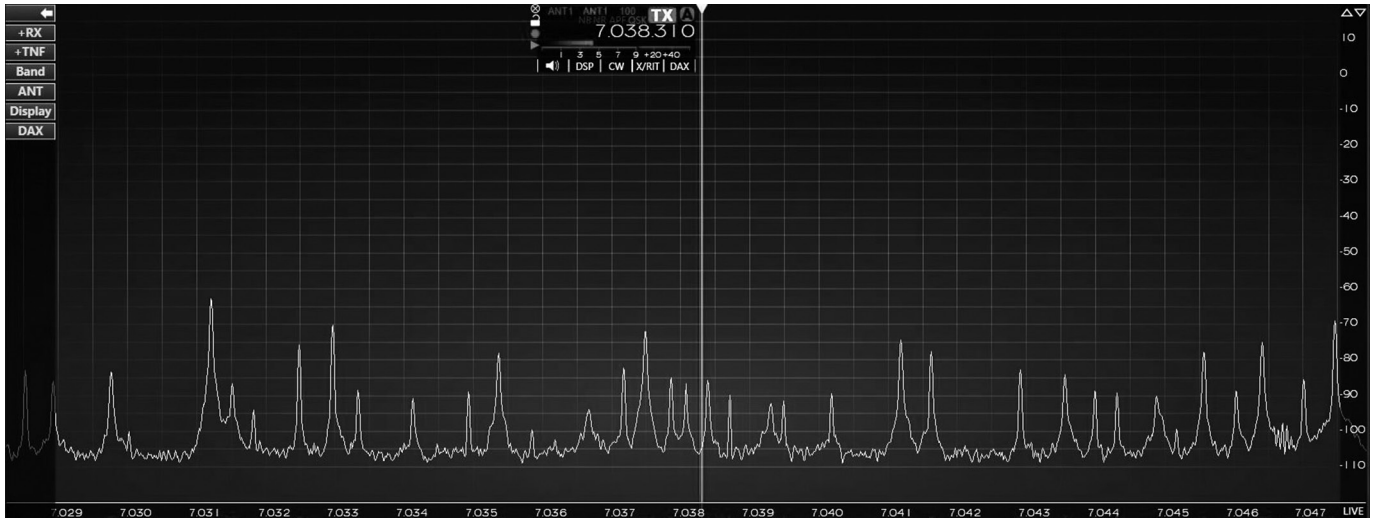


Figure 14.5 — A typical panadapter display showing a number of signals over about 18 kHz of the 40 meter CW segment.

of signals across a range of frequencies. The panadapter's design roots come from spectrum analyzers that were designed in the 1950s and 1960s. These instruments featured a swept receiver that tuned across the band. As the receiver tuned the band, the signal level in the receiver was captured and displayed. The resulting display is a graph with frequency on the x-axis and amplitude on the y-axis.

The spectrum analyzer has proven its worth in many areas, from radio design and troubleshooting to the analysis of on-air interference problems. The almost continuous view of the spectrum allows the operator to quickly understand the spectrum and act accordingly. A spectrum analyzer's controls allow alteration of the bandwidth of the sweep receiver, known as *resolution bandwidth (RBW)*, the *sweep rate* of the receiver, the total bandwidth displayed, and other parameters. (See the **Test Measurements and Equipment** chapter for more details about spectrum analyzers.)

In the absence of a signal, the swept receiver would return the higher of the noise from the input port of the analyzer or the spectrum analyzer's noise floor. This noise floor visualized or painted is partly a function of the bandwidth of the sweep receiver itself: the wider the bandwidth, the more noise was input to the receiver and the higher the noise floor appears. If the noise floor visualized on the sweep was higher than signals of interest, the resolution bandwidth of the receiver could be lowered, reducing the noise let into the receiver and lowering the noise floor of the display.

In the absence of a signal and assuming evenly distributed noise, the noise floor will drop by 3 dB each time the resolution bandwidth is halved, since the total noise in the receiver was cut in half by the bandwidth filter. Typically, as the display is "zoomed in" to show less bandwidth on the screen, the instrument will automatically reduce the receiver bandwidth in the sweep (resolution bandwidth) and the noise floor of the display will drop. The swept receiver approach to a spectrum display is used in radios designed in the early 2000s, such as the Icom 756 Pro-series.

The swept receiver design has some key drawbacks, though. First, time is required for the receiver to settle, a measurement to be taken, and the receiver tuned to the next frequency. This limits the speed at which the receiver can tune and often results in a slow "painting" of the display that lacks the full-motion look of modern panadapters.

The slower sweep rate reduces situational awareness. For example, imagine someone sitting sending a series of CW dits. In the swept design, the receiver will typically hear a signal or no signal as it passes the CW dit signal. We will either see a nice signal out of the noise or nothing, depending on the timing of the dits in relation to our sweep receiver. With a modern panadapter refreshing many times per second, we can see the signal rise and fall with the CW dits, giving us a clear view of what's going on.

The second drawback of the swept receiver design is the tradeoff between sweep speed and resolution bandwidth. As we reduce RBW to see more detail in the spectrum, the sweep rate must slow because the measurement takes longer, primarily because the narrower filter requires more time for signals to pass through and stabilize before the measurement takes place. The more detail we want to see, the slower our display updates, and the lower our situational awareness.

Today's transceiver panadapter displays are built on many of the same principles of the spectrum analyzer with one notable exception: today's receiver panadapters are almost always built from a Fast Fourier Transform or FFT (See the **DSP and SDR Fundamentals** chapter). The FFT accepts a wide bandwidth of time domain samples from an analog-to-digital converter and translates the time-domain signals into the frequency spectrum.

PANADAPTER TYPES

There are two main panadapter types available in current amateur products, the *dependent* and *independent* panadapter. In a dependent panadapter design, the data for the panadapter is from the receiver's IF signal. In this scenario, the panadapter will always show the spectrum to either side of our received frequency, which is shown in the center. For example, if the panadapter dis-play spans 192 kHz, our receive frequency will be in the center and we will see 96 kHz of spectrum on either side of the receiver.

In contrast, an independent display operates independently of the receiver. The panadapter display frequencies can be moved, and the received frequency can be moved without being in the center of the display. This allows showing a DX pileup with the DX station on the left and the pileup on the right, and we can tune our transmitter across the pileup without shifting the spectrum display. Internally, this requires a separate receiver just for the panadapter that is tuned independent of the receiver(s) in the radio.

The FFT used to build the panadapter is always measured in *points* or *bins*, each of which is essentially a collection of receiver channels like our sweep receiver in the spectrum analyzer. The bins are spaced evenly across the range of frequencies output by the FFT. The bandwidth of each bin is determined by the total sampling width presented to the FFT divided by the number of FFT points. For example, if our input is 192 kilosamples per second (abbreviated ksps) and we use a 4096-point FFT, each bin of the FFT would represent a receiver that is 192,000 / 4096 = 46.875 Hz in width. In other words, each point in the display of our FFT represents the signal and noise heard by a receiver with a bandwidth of about 47 Hz. In this way, you can think of the panadapter as a series of equally spaced S meters, laying on their sides continually showing the signal level in each of our FFT bins.

To compute an FFT for a panadapter, the FFT must first collect enough time-domain samples to fill the input registers of the FFT. Using the previous example, a 4096-point FFT required 4096 time-domain input samples in order to create a 4096-point frequency-domain output. For a receiver operating at 192 ksps, the time required to collect these samples is 4096 / 192,000 = 21.3 ms. After this time, the FFT can be computed and a *frame* can be displayed for the operator. The maximum possible frame rate for the display is the reciprocal of the time to fill the input. In this example, 1/21.3ms = 46.875 frames per second (46.875 Hz refresh or update rate). Note that this frame rate is exactly equal to the bin width of the FFT.

Considering a typical sideband signal with a bandwidth of 3 kHz, the panadapter just described would show 3,000 Hz / 46.875 Hz = 64 different points across the audio range of a received signal. Since a sideband signal is the audio in the operator's voice shifted in frequency to the RF spectrum, we can see the energy in the operator's voice at each of 64 different frequencies as the operator talks. Outside of the operator's intended transmit bandwidth we can also observe any RF that is transmitted unintentionally. This allows us to observe the cleanliness of signals on the band in significant detail.

Figure 14.6 shows a transmitted SSB signal that is fairly clean and has little RF outside of the transmitted band of interest. **Figure 14.7** shows a significantly less clean signal displaying splatter caused by an improperly configured amplifier. This capability is particularly useful for assisting another amateur to diagnose a transmitter that has signal anomalies. The visual representation of their transmission can make quick work of identifying intermodulation distortion, splatter, and other problems.

For CW or narrowband digital mode operation, a panadapter with a higher resolution than 47 Hz is desired. There are two ways to achieve higher resolution:

1. An increase in the number of points in the FFT directly reduces the bandwidth of each FFT bin, both lowering the noise floor and increasing the resolution.

2. A reduction in the bandwidth of the samples presented from the ADC also directly affects the resolution. For example, if we simultaneously reduce the ADC bandwidth to 96ksps and increase the FFT to 8192 points, our bin size is now 96,000 / 8192 = 11.7 Hz, a reduction factor of four in the bin size. The noise floor of the narrower bin will decrease by 6 dB as well.

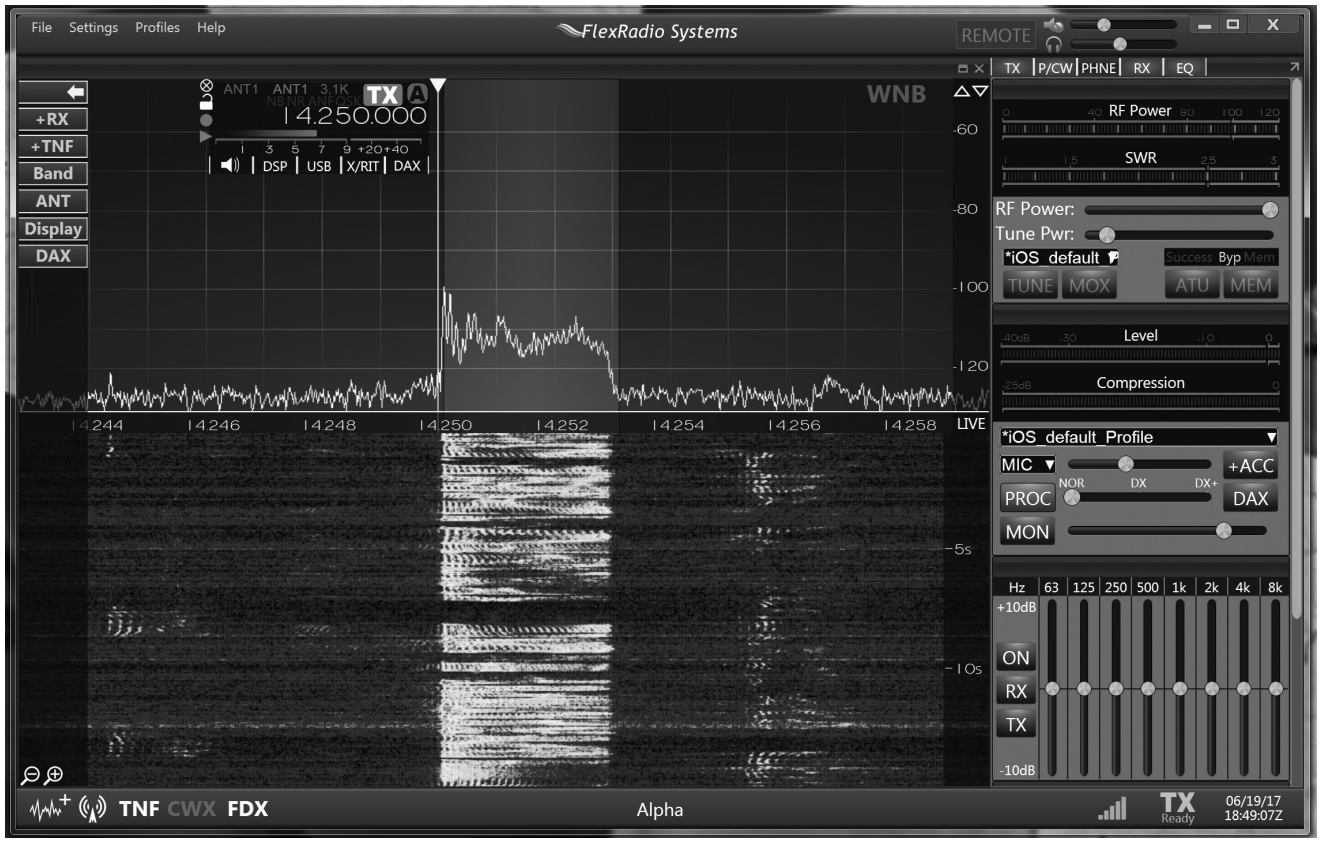


Figure 14.6 — A panadapter display showing a cleanly transmitted SSB signal.

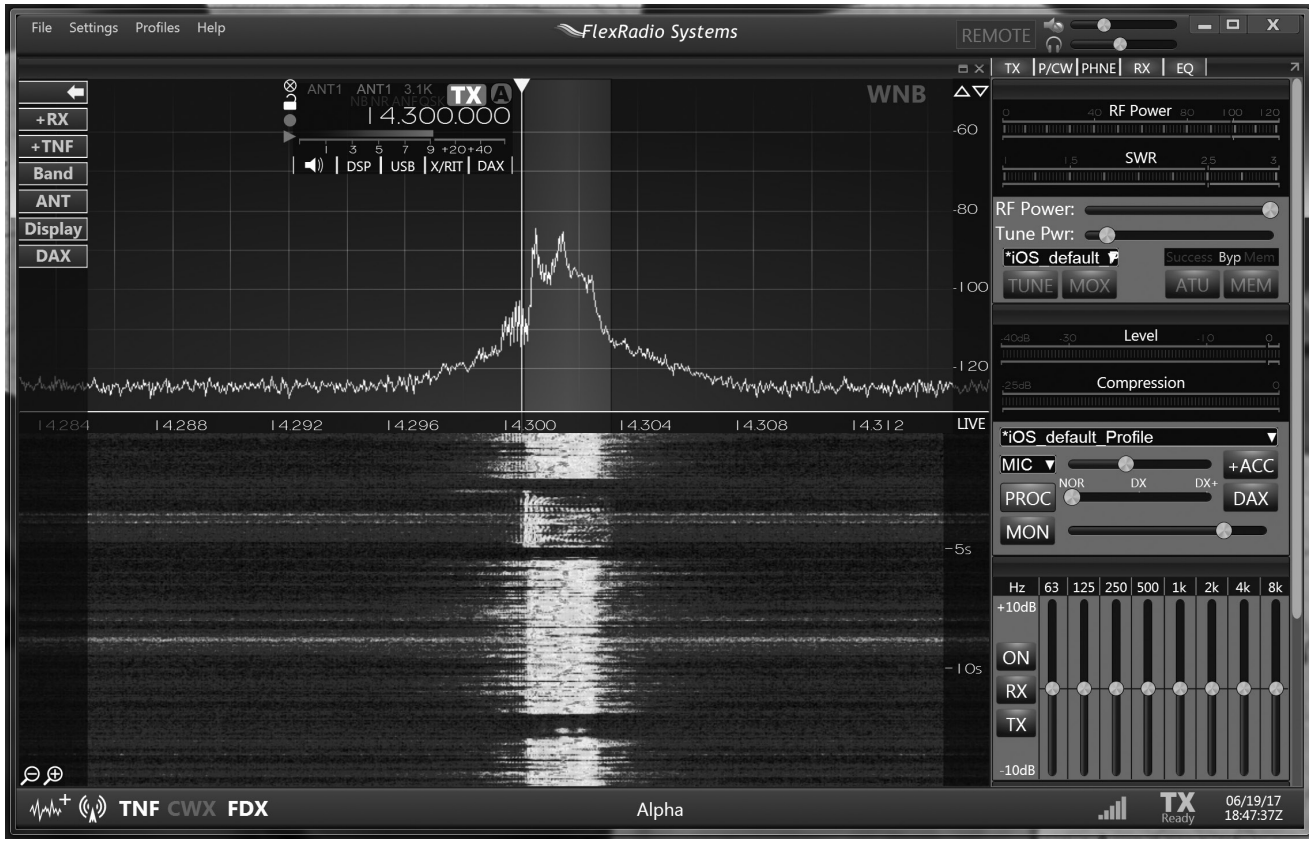


Figure 14.7 — A panadapter display showing an SSB signal with splatter and other distortion products.

The resulting display shows higher resolution, but this comes at a price. First, the frame rate of our FFT is now 11.7 frames per second. This is because we must now collect 8192 / 96,000 = 85.3 ms of data before calculating the FFT. Also, because the FFT is now based on a larger time period of data, fine temporal (moment-to-moment) details are lost in the display. For example, a 60 WPM CW signal is composed of 20 ms dits and 20 ms spaces between elements. A string of dits at this speed could be visible as individual pulses in our first FFT example, but would be somewhat lost in the new FFT since several transitions would occur during each FFT frame. This time limit tends to place practical limits on how fine-grained an FFT can be used in a panadapter. The other practical limit is computational complexity. As the number of points in the FFT is increased, more computing power is required to calculate the FFT that is the basis for the panadapter.

How far we might "zoom in" to a panadapter display is a function of what we want to observe. For SSB with signals approximately 3 kHz wide, we often want to observe many tens of kilohertz of spectrum to get a good view of other signals on the bands. In contrast, with CW we might only want to see a few kHz of bandwidth to determine the exact frequency of each CW signal. With two-tone FSK (RTTY) and PSK (PSK31) signals, it can be helpful to see both components clearly to aid in tuning, but not at such a fine resolution that the display operates slowly or incurs temporal distortion. A properly designed modern panadapter will allow us to make these tradeoffs to get the panadapter tuned "just right" to give us the view of the spectrum we need to help most in our operating mode.

PANADAPTER CONSTRUCTION

There are several methods for building panadapter displays, but the two key pieces of a panadapter display are the receiver and the FFT. As mentioned earlier, dependent panadapters (**Figure 14.8A**) are derived from the same receiver we use to demodulate and copy stations we are trying to work. In this case, the primary decision is how wide to make the receiver (sampling rate), since this will determine the bandwidth of our panadapter.

In radio systems with dependent panadapters for which the processing of the receiver signal chain is performed on a PC, the designer must be careful to limit the panadapter bandwidth, since all this data will need to be sent to the PC and processed using the PC's main CPU. The dependent panadapter architecture was invented with *PowerSDR* and first shown on the original FlexRadio transceivers such as the FLEX-5000. This panadapter architecture is also used in the HPSDR and several other radios.

With the independent panadapter architecture shown in Figure 14.8B, we see that the receiver and panadapters are separate. This architecture allows tuning the panadapter independently of the receiver and gives rise to more flexibility in setting up the panadapter display and receivers. The other key consideration is the rate of the sample stream from the receiver into the FFT. As discussed above, this affects the bin width of the display along with the refresh rate of the display. It also affects the amount of processing power required to process the raw data from the receiver. The wider the panadapter, the higher the sampling rate data is required to produce the display and the more computing power is required, assuming no loss of data.

To implement a "simple" panadapter display using the independent panadapter architecture, we could use a signal receiver with a bandwidth of 1 MHz, running into a FFT with 16,384 FFT points (the FFT algorithm requires the number of bins to be a power of two, in this case 2^{14}). This would give us bin sizes of 61 Hz. If our radio display has a horizontal width of 600 pixels we can implement a zoom-in and zoom-out feature by simply discarding or using a MAX function across bins we are consolidating into a single pixel. This type of display will also have a consistent noise floor, commensurate with a 61 Hz receiver bandwidth.

To implement a more advanced panadapter using the independent panadapter architecture, we could implement a variable rate decimation function as shown in **Figure 14.9**. The decimator or resampler can take in a sample

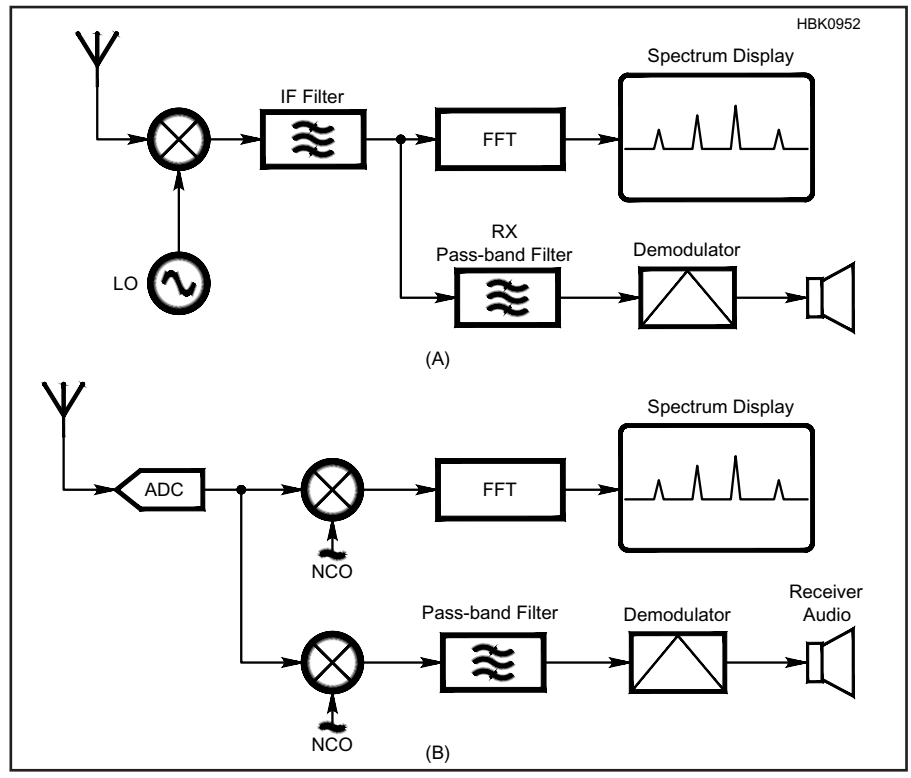


Figure 14.8 — Panadapter architecture. (A) Dependent architecture in which the panadapter and main receiver signal chain share common tuning and filtering mechanisms. The panadapter and receiver are tuned together. Dependent panadapters can be implemented as both superheterodyne and direct sampling receivers. (B) shows an independent architecture in which the panadapter and receiver have their own signal chain. The panadapter and receiver can be tuned independently of each other.

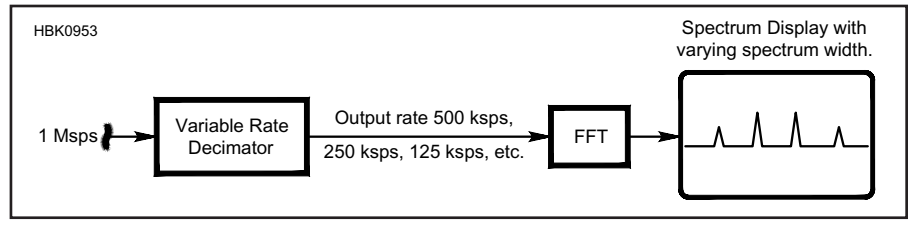


Figure 14.9 — Variable rate decimation.

stream at one sample rate and output a stream at other rates. In this way, we can zoom in and zoom out by varying the decimation rate. For example, let's assume out input rate to the decimator is 1 Mbps. We could output 1 Mbps, 500 kbps, 250 kbps, and so on, down to any rate we needed. This data could then be input to the same FFT function.

The resulting FFT output frame rate will be the input sampling rate divided by the number of FFT points and will have a bin width equal to the sampling rate divided by the number of bins (assuming I/Q input data). This architecture is used in the FLEX-6000 Signature Series transceivers and requires substantially more computing resources than the simple implementation, but it provides a more capable panadapter that can expand to many MHz of spectrum or zoom in to just a few kHz.

PANADAPTER APPLICATIONS

The benefits of a panadapter can be wide-reaching. In the simplest of cases, the instantaneous view of the spectrum can quickly demonstrate where the band is and is not in use. Watching the band for just a few seconds gives the operator an idea for a good place to call CQ. It's always a good idea to check if the band is in use before calling since only one side of a conversation may be audible from your QTH, but this same awareness of the band is difficult to achieve by spinning the dial.

When operating on microwave weak signal bands, it can be hard to find other signals. Rover (mobile) stations using crystal-controlled rigs in very cold or very hot weather may have drifted many kHz from their dial frequency due to the multiplicative effect of crystal error on higher bands and temperature changes in the rover. A panadapter makes quick work of locating these operators. A common methodology is to ask a rover to "send dashes" on a microwave band and then watch the waterfall for the telltale dashes to appear. Because signals deep into the 500 Hz noise floor can be seen on the waterfall display, we can see signals from stations before we can hear them, and this can also be used to assist in pointing antennas. As a fixed station, we can tell the rover operator on a liaison frequency to rotate his antennas as we watch the panadapter for a rise or fall of signal strength. Some panadapters have increased their FFT bins to 2^{17} to reduce their bin size and allow them to see even further into the noise floor.

In contesting, there are often advantages to working stations in a particular order. For example, weaker stations that have reduced signals in the panadapter may be DX stations that would result in a multiplier. Stronger stations are easier to work, and bagging these stations first might be a strategy worthy of consideration. In some sprint-style contests *running*, continually calling CQ on the band and working stations that respond to your CQ, is not permitted. Finding other stations quickly becomes paramount in this type of operating event. With a panadapter, finding stations quickly is simplified to looking for peaks in the panadapter display.

Interference in contests is a frequent problem, and rapid identification and resolution of interference sources is important. An adjacent splattering station is obvious in a panadapter and can allow an operator to make a quick comment to the right station, asking for the other operator to check the transmitting equipment. In some situations, changing frequency may be required, and again the panadapter can help locate where to move. From a competitive standpoint, having a panadapter is akin to having extra mirrors on a race car to see where other competitors are and what they are up to.

For panadapters displayed on a computer monitor, tuning can often be achieved by placing the a cursor on a signal or specific frequency and clicking. Quickly working stations in this manner bypasses the required tuning on traditional receivers. In this situation, the panadapter can provide a significant advantage. This is also true of panadapters on touch-screen devices where tuning can be accomplished just by tapping on a signal seen on the panadapter display.

Interference and noise are common in today's RF landscape. The panadapter offers the same capabilities of a spectrum analyzer by showing instantaneous views of the spectrum to help with isolation and characterization of noise. Some noise is limited to a narrow range of frequencies, while other noise sources may repeat on a specific frequency interval. Knowing the repeating frequency interval of a noise source or the similarities of the noise across the band can aid in diagnosis of the type of noise. Some types of noise drift over time, while others remain fairly constant with time. Other noise is broadbanded and raises the noise floor of the entire band. All of these different characteristics of noise can be easily observed in a panadapter, giving the amateur a thorough view of the type of noise under investigation.

14.2.2 The Waterfall Display

The *waterfall* display is similar to the panadapter in that it has frequency along the x-axis. The y-axis in a waterfall is time rather than amplitude, as shown in **Figure 14.10**. The waterfall display is typically created line-by-line, starting at the top of the display. A new line is added to the display with each unit of time, and all previous lines of the display are shifted down one line. In this way, the waterfall shows what has happened in the spectrum over time. It is this flowing motion of the waterfall that gives the display its name, since it gives the impression of a band of slowly falling water.

Let's build on the earlier analogy of a panadapter as equally spaced S meters lying on their sides. Similarly, the waterfall display can be envisioned as a series of panadapter frames, equally spaced in time and lying on their sides so we view them "from the top." What we see in the waterfall is color or intensity representing the height of the signal (amplitude) in the panadapter. Each line of the waterfall has either different colors or different intensities on each frequency that represent the signal level from the panadapter during that period of time. Typically, brighter or more intense colors represent stronger signals. By looking at the waterfall display, we can see the occupation of frequencies across the spectrum over time. The historical view of the waterfall can assist

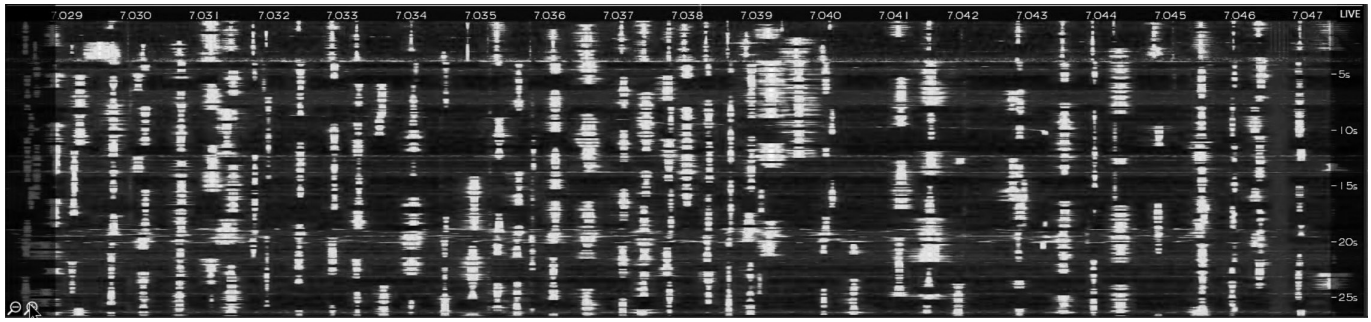


Figure 14.10 — A waterfall display showing the same spectrum as the panadapter in Figure 14.5 with CW and digital signals. More recent data is at the top of the display, and signal strength is indicated by brightness.

in locating a frequency that either has or has not been in use depending on the need.

Like its cousin the panadapter, waterfall displays are also calculated from an FFT, sometimes from the same data used to produce the panadapter. The waterfall also uses the same FFT bins and follows the same rules as the panadapter. The noise floor visible in the waterfall is determined by the bin size of the FFT. The narrower the bins in the FFT, the less noise in the bin and the more weak signals can be observed in the waterfall. In Amateur Radio, we generally refer to the noise floor as the noise observed in a receiver with a 500 Hz bandwidth. If our FFT bin width is 50 Hz, our noise floor would be 10 dB lower, calculated as $10\log_{10}(500/50)$. Similarly, a 5 Hz bin would have a noise floor 20 dB lower than for a 500 Hz receiver bandwidth. To see signals "below the noise floor" is a useful capability in both the panadapter and waterfall displays.

The historical view of a waterfall can provide an additional level of information not always visible in a panadapter. Using the same microwave operating example, assume that we are attempting to hear 10 GHz signal from a microwave rover with the dish antenna slightly off our bearing. A common technique is to ask the rover to send a series of slow CW dashes "please send dashes" using a VHF liaison frequency. By watching the noise floor in the waterfall carefully, we may be able to see a faint line representing those dashes. We can then ask the rover to rotate the dish one way or the other and see if the intensity increases. The final outcome would hopefully be to align the dish where we can hear the rover.

Using an HF example, the waterfall can provide unique insight into the patterns used by a DXpedition station operator. Knowing these patterns and where the DX station is transmitting can help the DXer pick a frequency that is likely to be heard, and worked, by the DX station. For example, by observing the DX station working several stations across a pileup, the DXer can immediately put the information to use and "tailgate" the last contact hoping to work the DX entity before the listening frequency changes. Consistently locating the DX entity on a traditional receiver by spinning the dial is futile in comparison.

COMBINING THE PANADAPTER AND WATERFALL

A common way to show both a panadapter and a waterfall display at the same time is to position the panadapter above the waterfall, both lining up in frequency so that a common frequency axis is used for both. In this way, we can see what's happening now in the panadapter and what happened in the past in the waterfall.

A vast amount of information is presented to the operator with both displays present simultaneously. A short burst of noise heard in the receiver could be anything, but close examination of the panadapter and waterfall can show that it is a signal slowly moving across the spectrum (perhaps an ionosonde) or maybe it's a spread-spectrum signal that is 10 kHz wide moving around the ham bands. The difference will be apparent looking at a wider view of the waterfall and seeing the same signal move around the band. Most operators find that after using a panadapter and waterfall for a while they are not able to easily go back to using a radio without these features, which enhance situational awareness.

14.2.3 User Control Interfaces

The front panel of the radio, referred to here as the *user interface* or simply *interface*, provides for control, meters, status, frequency indicators, and displays such as a panadapter and/or waterfall to visualize the spectrum. Typical controls allow for changing the frequency, mode, mode-specific settings, and other features of the radio.

THICK CLIENTS

The SDR-1000, an amateur SDR designed by Gerald Youngblood, K5SDR, was featured in a 2002 series of *QEX* articles listed in the references for the **DSP and SDR Fundamentals** chapter. The SDR-1000 hardware did not have an interface itself, but relied instead on a computer program as its interface. This first SDR interface, *PowerSDR*, was designed to display the spectrum and control the radio. *PowerSDR* is not only the interface for the SDR-1000, it also performs all of the signal processing required to both modulate and demodulate audio and produce all of the spectrum and other displays in the software. This type of interface is known as a *thick client* in the computer industry. The term thick client is used when substantial work must be done in the client software. **Figure 14.11** shows how

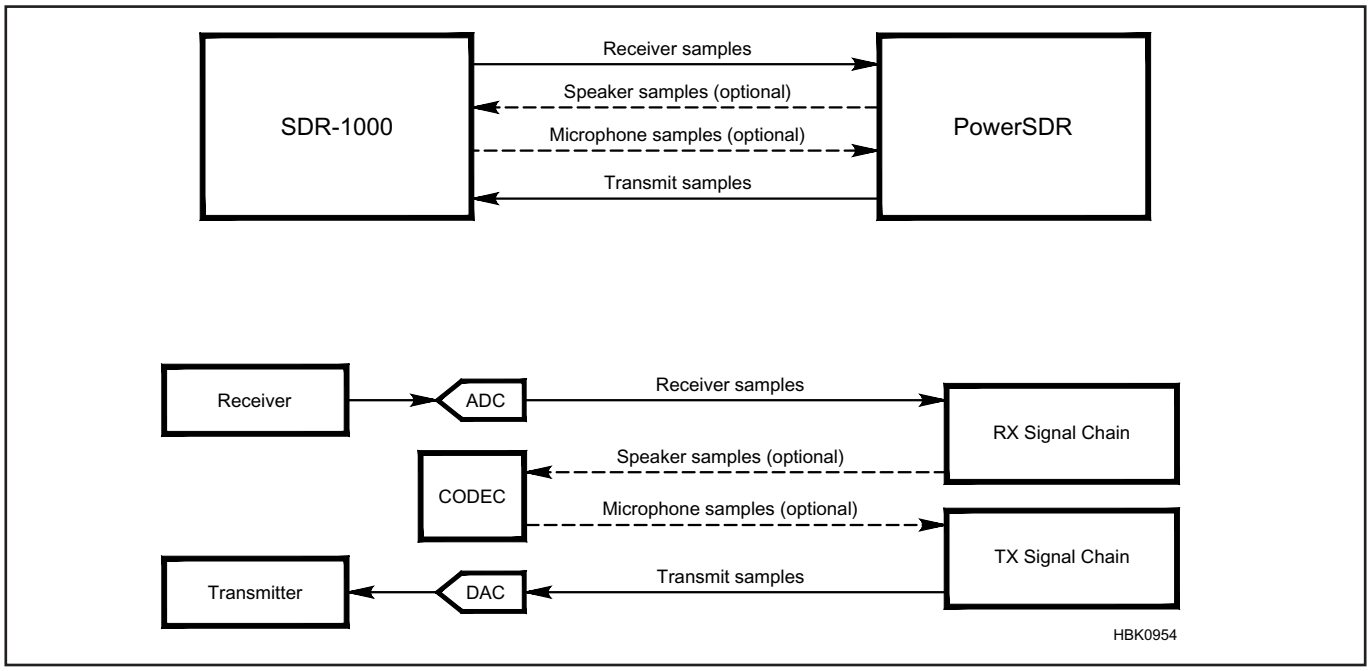


Figure 14.11 — A block diagram of the SDR-1000 showing the main data flows for signal processing.

the signal processing software in *PowerSDR* works with the radio.

There are generally four streams of real-time data that flow between a thick client and the SDR hardware:

1) Receiver samples are time-domain samples collected by the SDR hardware. The hardware itself can be direct-sampling, a superheterodyne (followed by an ADC), or direct conversion. The samples sent to the software for processing are generally in the range of 48 ksps to 384 ksps. Since the samples are often sent as both in-phase and quadrature (I/Q) and are often 16-32 bits each, a 384 ksps stream can contain as much as 25 Mbps of data.

2) The receiver samples are tuned, demodulated, filtered, and ultimately provided to the radio hardware for audio output, usually from headphones or speakers.

3) Microphone samples are captured in the radio hardware and sent to the software for processing and preparation for transmission.

4) Transmit samples are sent from the software back to the radio for transmission when the radio is transmitting.

Some radios require both the speakers and microphone to reside on the computer rather than the radio. In this case, both the speaker and microphone streams are generally not sent to and from the radio, respectively.

Once the interface software has the samples collected from the receiver, it can perform the DSP functions required to create the audio signal. (See the sections on Receive and Transmit Chains earlier in this chapter.)

Similarly, the transmit signal chain commonly has a modulator, filter and limiter. When the radio is transmitting CW or FSK, the SDR software generally synthesizes the waveforms for both of these modes directly without the need for many of the other transmit signal chain elements. For this reason, it is common for there to be a separate signal chain for each mode.

The thick client SDR interfaces such as *PowerSDR* place a higher demand on the computing hardware, since many of the DSP functions are performed in the computer. There is also a stringent set of requirements around timing to ensure that the constant flow of samples on both the receive and transmit sides is not interrupted. If an interruption occurs on the receive side, it will result in audio dropouts. Interruptions on the transmit side can result in discontinuities in the transmit signal. Depending on the hardware involved, discontinuities in the transmit signal can result in high frequency components that cause the transmitted signal to reside outside of the desired transmit passband. There are advantages to the thick client architecture, specifically:

1) Since computing and software requirements for the radio itself are minimal, the radio hardware can be less expensive.

2) Software that runs on the PC is typically easier to develop, test, and debug.

3) As PC hardware has continued to achieve higher price-performance ratios, more power for more capabilities can be added without changing the radio..

The thick client architecture is now common, due in a large part to the availability of open source *PowerSDR* and the ease of adapting it to other radios. More than one HPSDR radio has been designed using this architecture, as have the ANAN radios, adapted from the HPSDR designs.

THIN CLIENTS

In contrast to a thick client interface are *thin client* interfaces. In a thin client, all or most of the work required to modulate, demodulate, and produce the displays for the radio is performed inside the radio hardware. The thin client is used, then, to display the interface and accept control commands, which are passed to the radio for processing. An example of the thin client approach is the FLEX-6000 Signature Series radios by FlexRadio Systems.

In the FLEX-6000, all modulation, demodulation, and radio control are performed in the radio in the *SmartSDR* software running under *Linux*. This reduces the performance requirements on the client system used for the interface. In turn, the interface can easily be run on cellular phones, tablet computers, and other less expensive computing hardware. Key advantages of the thin client architecture are:

1) A wide array of computing hardware can serve as the interface due to the low computing demands.

2) Reduction of network bandwidth between the client and radio due to the processing occurring in the radio. This allows remote operation over low-speed network interfaces making remote connection directly to a cell phone possible, for example.

3) The client software is considerably simpler since it is just responsible for display and control and can, therefore, be easily ported to new computing devices resulting in a wide array of available interfaces and interface platforms.

4) Availability of an applications programming interface (API) due to the necessity of an API for the client. This API can then be used by other amateur applications wishing to use the radio as a server or utility.

14.2.4 Panadapters for Older Radios

Panadapters can also be added to an older analog or analog-digital hybrid transceiver. The concept is straightforward: use an SDR receiver to monitor the radio's IF spectrum while the radio operates normally. The radio provides the front-end preselector, band-pass filtering, and initial IF filtering. The SDR and software displays signals in the IF passband. As the radio is tuned, signals move across the display. This is an example of *dependent architecture* as described in this section (see Figure 14.8). Panadapter software demodulates signals independently of the radio, so different signals can be monitored simultaneously on the radio and the panadapter.

Adding a panadapter to an older radio requires 1) an SDR device with display software that will cover the receiver's IF; and 2) a method of interfacing the SDR to the IF of the radio. SDR devices come in all capabilities and price ranges. The two most common and least expensive types are:

1) An external commutating mixer (also

Software for External Panadapters

Software for display and demodulation is dependent on SDR type and PC operating system. All listed titles below have versions available for multiple operating systems. The following packages support the Windows operating system:

- *HDSDR*: (**hdsdr.de**) Powerful, many customizable features. Can record RF or AF on a timer. Mild learning curve. Widely used.
- *SDR Sharp*: Powerful. Easy to use. Lots of add-ons and customizations. The port driver *Zadig* (included with the distribution) is required for using *SDR#* with RTL-SDR. Specific instructions on how to properly load *SDR#* on the **RTL-SDR.com** website must be followed exactly. (**sdrsharp.com** redirects the browser to AirSpy website where *SDR#* software is available under downloads)
- *SDRuno*: (**sdrplay.com**) Supports all *SDRplay* and other packages.
- *SDR Console*: (**sdr-radio.com**) Extremely powerful, multiple windows. More advanced learning curve.
- *RTL-SDR*: (**rtl-sdr.com**) Supports the RTL2832+R820. Detailed installation instructions, much information.
- *Rocky*: (**dxatlas.com**) For SoftRock (or other I/Q type devices). Simple and works well.

Two packages for the Android operating system include *RF Analyzer* and *SDR Touch*. Both are designed for use with RTL-SDR connected to the Android device by an OTG ("On-The-Go") cable. More software is available from Google Play.

called a Tayloe mixer — see the Receiving chapter) such as a SoftRock. These use a fixed or variable LO to switch a multiplexer IC that samples the IF signal in quadrature (every 90 degrees). The result is a pair of audio frequency I and Q signals that are fed to a computer soundcard for the display and demodulation software.

Limited by the performance of inexpensive components and the requirement for an LO that usually runs at four times the IF, these are generally limited to a maximum IF frequency of 30 MHz. An additional limitation is the sample width and quality of the soundcard. Built-in PC soundcards are often limited to 48 kHz, so 24 kHz is the limit of the displayable RF bandwidth. Higher performance external or add-in soundcards are available with sample rates up to 192 kHz with 24-bit resolution.

2) USB "dongles" such as the FunCube Dongle Pro+, a USB plug-in with a 0.15 to 1900 MHz RF range and 192 kHz bandwidth. These have automatically selected internal RF filters and offer surprisingly good performance. Other popular types are SDRPlay, AirSpy, and HackRF. For an IF higher than 25 MHz, the RTL2832U+R820 European TV tuner dongle, commonly called RTL-SDR, includes software that converts it into a 24 – 1700 MHz receiver of 2 MHz bandwidth and 8-bit resolution. These have no internal filtering but if applied properly, work OK for IF monitoring.

Interfacing to the IF requires modifications to the radio. Making the modifications to add the panadapter could void the warranty and can damage the radio if done incorrectly. Refer to a skilled transceiver technician if you are not comfortable making the necessary modifications on your own.

Adding an IF monitoring tap is not a simple task. Modern analog radios have low-impedance circuitry in the IF, so simply adding a 50-Ω device in parallel with the existing IF circuitry will be detrimental to stage gain and balance. Some SDR devices are known to generate switching noise from their input port where it is injected into the receiver's IF, affecting all signals received by the main radio.

SDR Panadapter Hardware

- JFET buffer amps from **SDR-Kits.net** and **huprf.com**
- Funcube: FCD Pro+ from **funcube-dongle.com**
- SDRplay: (**sdrplay.com**) Several versions available; 10 kHz to 2 GHz with up to 10 MHz bandwidth.
- Airspy: **airspy.com**
- HackRF: (**greatscottgadgets.com**) Frequencies to 6 GHz. Half duplex transceiver. Open source.
- RTL2832+R820 (**rtl-sdr.com**) Also called RTL-SDR. Available from Amazon, Nooelec, other sources.
- SoftRock: (**fivedash.com**) simple kits and pre-built.

There are several ways of tapping IF signal out of the radio. The radios for which it's easiest to add the panadapter have buffered IF outputs already built in. Examples include the Icom IC-R7000 receiver and Elecraft K2 and K3 transceivers. Simply connect the SDR device input to this output port. Some radios add a dc voltage to the IF output for supplying power to an external device. If the input of the SDR device you are using is dc-coupled, a dc blocking capacitor must be added. A value of 0.01 µF will work in most cases. Use a ceramic capacitor with low lead inductance to minimize reactance at VHF and higher frequencies.

It may be possible to find a point in the IF signal path that is buffered. The Kenwood TS-430S and TS-700SP transceivers both have splitters and buffers in the IF path that feed separate FM demodulator circuits. (In these radios, a signal was always present at the buffer output, regardless of mode selected, but that may not be true of all radios. Check the in-depth User's or Service Manual for the transceiver.) An SDR connected to that point should have little effect on the overall receiver performance, although dc blocking may still be required. Careful testing before and after adding the separate IF output is required.

Another widely used method of interfacing is to tap into the IF with a high impedance buffer amplifier. An excellent design is available from SDR-Kits (**www.sdr-kits.net**) utilizing a JFET input buffer (100 kΩ input impedance) followed by a buffer amplifier and low-pass filter to drive the SDR device. These interfaces provide reverse isolation to keep SDR noise out of the IF chain. Information from the interface's original designer (G4HUP) is maintained on the SDR-Kits website.

It is also very useful to remove dc power or disable the IF output during transmit to prevent spurious audio noise and display artifacts during transmit. A source of dc in the radio that turns off during transmit or a control signal to enable and disable the IF output is useful. See the information from G4HUP on the SDR-Kits website for suggestions.

Examples of radios evaluated by the author include:

• Icom IC-R7000: An easy radio for adding a panadapter. The buffered 10.7 MHz IF is brought out to a jack on the back panel. No buffer amp required, but adding a dc blocking capacitor is required.

• Kenwood TS-430S: 8.83 MHz IF. Worked well with SoftRock.

• Kenwood TS-700SP: A buffered path in the 10.7 MHz IF, similar to that in the TS-430S.

• Yaesu FT-817: No buffered path in the 68.33 MHz first IF so it requires a high-impedance buffer such as the G4HUP "PAT." There is a version with a 70 MHz low-pass filter that removes artifacts before the following SDR. If the IF is tapped before the first IF bandpass filter, the SDR input bandwidth is approximately ±75 kHz from the center operating frequency. After the first IF filter the SDR input has a bandwidth of ±10 kHz.

14.3 Configuration and Control Interfaces

Computer control interfaces for radios play a key role in use of our radios beyond all but the simplest of operation. These interfaces allow us to check the status of our radios, change their settings, tune them, and operate from a distance. Logging programs often use them during contests to automatically acquire frequency and other data for the contest log. In combination with logging programs, the interfaces can be used to automatically select antennas, turn rotators, and control other equipment in the station.

Each radio manufacturer has selected or designed an interface to work with their radios. Typically a manufacturer will use a common interface across their product lines. The user's manual and service manual contain complete information on the commands and data that are exchanged via the interface protocol.

14.3.1 Icom CI-V

Icom's control interface, CI-V, named as the fifth (V) Communications Interface (CI), is a serial "one-wire" protocol that uses the RS-232 electrical interface (generally with 5 V rather than the RS-232 standard ±12 V), but allows multi-drop or multiple transceivers on a single CI-V communications line.

By using CSMA/CD (carrier sense, multiple access, collision detect), multiple radios can avoid using the communications interface at the same time, detecting when a collision occurs and retransmitting data. This is the same technique used in 10Base2 Ethernet in which coaxial cable was used and multiple computers used the same communications bus.

In CI-V, each radio has an address (which can be changed) and the computer can communicate with any of the radios on the CI-V bus. The CI-V protocol allows tuning the radio, changing modes, selecting memory channels, and other functions. CI-V generally treats the RS-232 data as bytes and references commands using their hexadecimal values. It is essentially a binary communications protocol. CI-V is also used in newer Ten-Tec radios, but with additional commands specific to Ten-Tec transceivers.

14.3.2 CAT (Computer Aided Transceiver) Interface

CAT is a control interface based on the RS-232 electrical standard. Most manufacturers such as Kenwood, Yaesu, Elecraft, and FlexRadio implement CAT via RS-232 or via a USB interface that emulates RS-232 signals. The protocol used by each manufacturer is proprietary to that line of equipment. Unlike CI-V, CAT is a point-to-point protocol, meaning the expectation is that there is a single computer controlling a single radio on the line. CSMA/CD protocols are no longer required, but an additional communications port must be used for each radio. CAT is not a binary communications interface, but uses ASCII characters. As with CI-V, each radio implements only commands that are meaningful to that radio.

With the advent of USB, many radios now make their CAT interfaces available over USB in the form of a USB communications port. When the USB cable is plugged into a computer, the USB protocol recognizes the device as a communication port and adds a new port to the computer. In this way, many radios can be added to a single computer using a USB hub.

As Ethernet and the Internet have been added to our computing palette, CAT has been implemented over TCP/IP, an Internet standard communications protocol pair. For example, with the FLEX-6000's *SmartSDR* CAT program, a TCP/IP port can be added that responds to CAT commands. The base protocol remains the same, but the communications transport mechanism is TCP/IP over Ethernet rather than RS-232. This allows a relatively simple conversion for programs attempting to access radios over Ethernet rather than either RS-232 or USB.

FlexRadio Systems, although adopting CAT to allow for interoperation with legacy software such as existing loggers and digital mode programs, also added a new computer control interface, the *SmartSDR* API. CAT is restricted to commands relating to two variable frequency oscillators (VFO), typically called VFO A and VFO B. Since the FLEX-6000 can currently have up to eight slice receivers (synonymous with a VFO for the purposes of controlling the radio), the CAT standard could not sufficiently provide for control of the radio. The SmartSDR API is also based on ASCII commands and allows full control of the radio over a TCP/IP port or using a Microsoft *Windows* .NET DLL interface. FlexRadio's *SmartSDR* thin client running on a *Windows* PC achieves all of its display and control capabilities using the *SmartSDR* API.

14.4 SDR Design Tools

A description of how SDR works and the opportunities it presents are interesting but not of much use unless the reader can acquire and use the necessary tools to work with SDR technology. This section discusses *GNU Radio*, a popular software development environment for working with digital signal processing (DSP) and applying it to SDR. A short introduction to working with FPGA technology is also included. DSP software is key to working effectively with SDR at the level to which amateurs are accustomed with traditional analog technology and a soldering iron.

14.4.1 GNU Radio

INTRODUCTION TO *GNU RADIO*

The ability to create and work with software for SDR projects is a new skill for many hams. Professional programmers have many options to pursue. For hams without a programming background, software packages are now available that create a gateway for ordinary hams to build custom DSP software. Software authoring has been vastly simplified. The software packages feature graphic user interfaces (GUI) and offer useful DSP functions in DSP source code libraries.

The ham experimenter has choices of graphically based DSP software packages: the proprietary packages *Matlab / Simulink* by MathWorks and *LabVIEW* by National Instruments, or the open source *GNU Radio / GNU Radio Companion* (*GRC*) DSP library package. This section introduces the combination of *GNU Radio* and *GRC* package as a powerful, "ham friendly" development environment for SDR projects. The material is based on a pair of *QEX* articles by John Petrich, W7FU (see the *GNU Radio* references), which are included in the online supplemental information for this book.

GNU Radio was conceived as a development tool intended for advanced educational programs and commercial product development labs. With a little effort, hobbyists can use these tools as well. *GNU Radio* can be used as a simulation tool to study DSP without being connected to an SDR. *GNU Radio* can also be used to develop practical and operationally satisfying DSP software for SDR transmitters, receivers, and transceivers as well as test equipment.

To make DSP authoring accessible for all levels of users, the *GNU Radio* developers include the user option of *GNU Radio Companion (GRC)*. *GRC* is the graphical DSP

library interface, an integral part of *GNU Radio*. The foundational *GNU Radio* DSP library is accessed with character-based command line entries. *GRC* is a graphical overlay to the *GNU Radio* DSP library with the same full functionality as the command line approach. *GRC* emulates the command line entries and represents each DSP function as a graphical icon without command line entries. (For the sake of completeness, a very few of the *GRC* DSP parameters are expressed as simple Python code commands. These Python commands are covered in the section **Learning Resources: DSP Competency and *GNU Radio***)

An arrangement of DSP icons and digital data links is termed a *flow graph.* When executed, the *GRC* flow graph is compiled to a Python code file, which itself is a wrapper for C++ executable DSP code. This artful layered construction, like a set of nesting dolls, provides a high level, user friendly, graphical interface for DSP authoring with the speed and efficiency of lower level code for DSP process speed and efficiency. More information from the developers about the construction of this DSP library environment is available at **wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F**

GNU RADIO AND THE GRC TOOL

GRC, the easy to use graphical interface, makes this DSP library particularly appealing. Reduced to the basics, the step-by-step design of a functional DSP begins with envisioning or drawing a block diagram pattern of a traditional analog signal processing circuit. Once the general signal processing pattern is conceived, select the desired DSP function(s) from the DSP library (filter, mixer, oscillator, amplifier, etc.), arrange the DSP function(s) as graphical blocks in the work screen according to the design pattern, connect the blocks with an appropriate type of data link, add the DSP parameters in each block, and execute the program. The details of constructing a DSP flow graph are covered in the **Success with *GNU Radio* Tools** section and in the online tutorials referenced in the section **Learning Resources: DSP Competency and *GNU Radio***. The process is straightforward.

Let's examine the *GRC* work environment and flow graph composition at the highest level or block level. The *GRC* work environment consists of four parts: the work area, the control bar, the library, and the console. **Figure 14.12** illustrates the *GRC* user graphic user interface (GUI). The DSP is authored in the work area in the center of the GUI screen. The *GRC* control bar is located across the top of the work area. The *GNU Radio* DSP library runs along the right margin of the screen. The operating console is visible at the very bottom of the work environment and becomes active when a DSP function is executed.

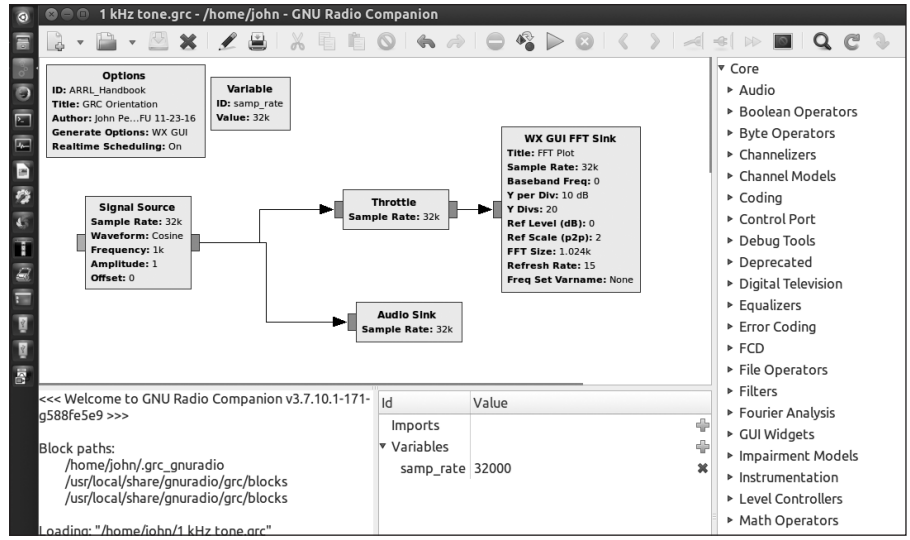


Figure 14.12 — The development window user interface for *GNU Radio* functional blocks.

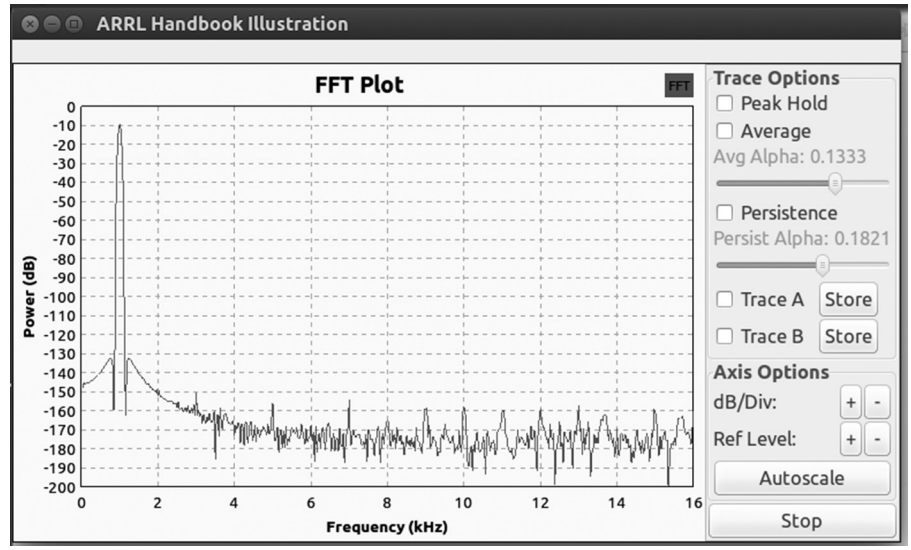


Figure 14.13 — The user interface window showing an operating 1 kHz audio tone FFT.

Centered in the work area are six DSP blocks, which in aggregate constitute a DSP flow graph. The flow graph implements, in DSP, a 1 kHz audio tone output to both an audio sink (computer speaker) and to an FFT panadapter display (on screen). **Figure 14.13** is the FFT display output once the flow graph has been executed. Note the signal spike is accurately centered at 1 kHz on the FFT display and the 1 kHz tone would be heard via the computer sound card.

Continuing to the parameter level, one step below the block level, each DSP library item is displayed as a rectangular shaped "block" with input / output ports. Each block must be "opened" with a mouse right-click to gain access to the parameter fields that define that DSP function. Each parameter has a field in which the values are either added or selected from a menu.

Figure 14.14 displays the parameter screen for a low-pass filter DSP block. Note how the digital data type (*FIR Type*), sample rate, filter bandwidth and filter shape, and other parameters can be selected or specified. Some parameter values are intuitive, direct carry-overs from analog design. For example, the bandwidth and shape factor parameters for a DSP filter block are specified in kHz in the same way an analog filter is specified. Other parameter values are not as intuitive and refer to parameters unique to DSP. Examples of these parameters are the DSP *Sample Rate* and *FIR Type* choices. More detail about the most commonly used DSP blocks and appropriate parameter choices are explained in

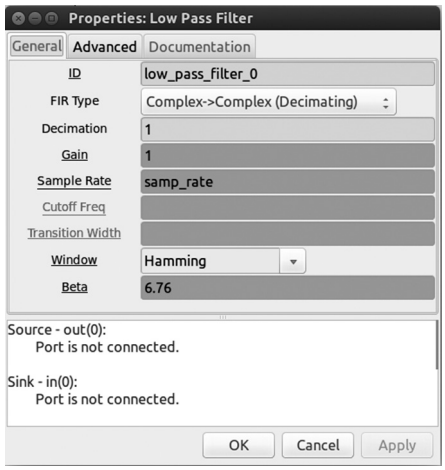


Figure 14.14 — A screen showing how parameters are managed for a typical DSP filter block.

greater detail in the section below, **The Rest of the Story: Basic DSP Theory Applied to a Flow Graph Segment**.

Figure 14.15 depicts the more complex flow graph of an SDR broadcast FM receiver. This flow graph uses an RTL-SDR as the receiver source. Other SDR receiver hardware can be substituted easily as explained in the section ***GNU Radio*, Radio Hardware, and UHD**.

The RTL-SDR is represented by the *RTL-SDR Source* block on the left. The *Audio Sink* on the right outputs the audio sound. The *WX GUI FFT Sink* in the center generates a panadapter display. The chain of DSP blocks in the center of the work screen perform the filtering and signal demodulation functions for this radio. Compare this flow graph with a block diagram for a simple analog FM receiver to appreciate the fundamental similarity. **Figure 14.16** illustrates the operating receiver GUI that appears when the flow graph is executed. Shown are the radio tuning controls for frequency and audio output, and the FFT panadapter display.

SUCCESS WITH *GNU RADIO* TOOLS

Just as success at woodworking comes from competency with woodworking tools, success with *GNU Radio* comes from learning and using its tools. Viewing a flow graph is the first step is learning the tools. Success with *GNU Radio* inevitably proceeds to DSP competency as well.

An important part of success with *GNU Radio* comes from an organized and disciplined authoring approach. Lacking organization and discipline, flow graph construction can easily become a chaotic, non-functional mess. This section describes one general disciplined approach. Following a description of the general approach, a detailed example flow graph creates entry level DSP competency through flow graphs construction.

DSP competency for new users is not necessary for initial success. Many of the automatic default settings in *GRC* are adequate for entry level flow graphs. Ultimately, DSP competency comes from incrementally constructing more advanced DSPs with simple flow graphs.

GENERAL APPROACH: THE FOUR STEPS TO FLOW GRAPH SUCCESS

The disciplined approach involves following a pattern of four basic steps. These basic steps are useful at getting started the first time with DSP flow graph construction and as an efficient approach to construct more advanced and complex flow graphs.

How does one begin a DSP flow graph using the *GNU Radio GRC* graphical interface? Four steps are required to construct any flow graph: the "block" step, the "parameter" step, the "logical interconnection" step, and the "DSP" step. Each step is explained below.

Step 1 — the Blocks: Arrange the appropriate DSP blocks from the *GRC* library to form a diagram of the desired circuit in the work area. See how the functional blocks are arranged in the example flow graph in the work area of **Figure 14.17**.

Step 2 — the Parameters: To perform the desired DSP function, each DSP block re-

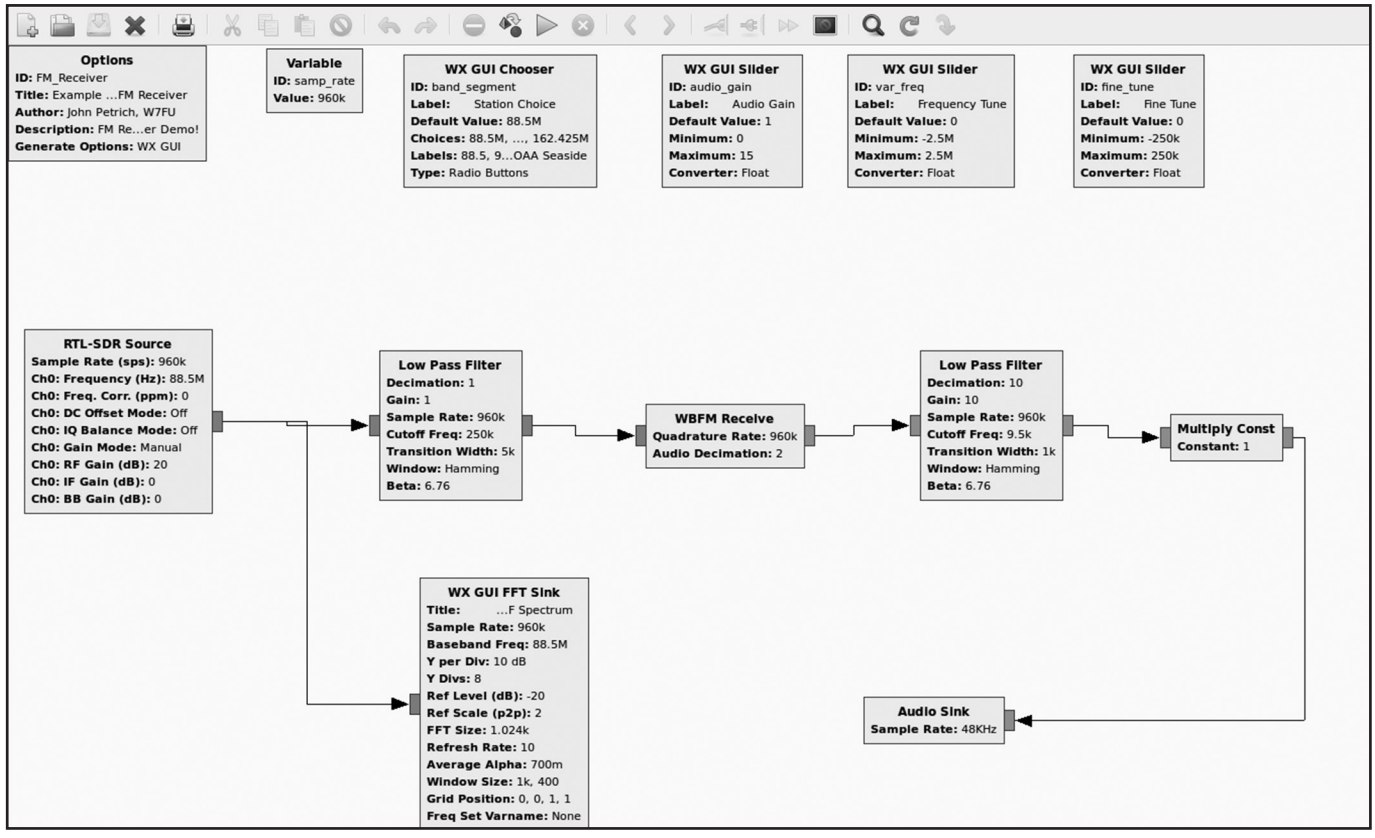


Figure 14.15 — Flow graph for a broadcast FM receiver showing the various function blocks involved.

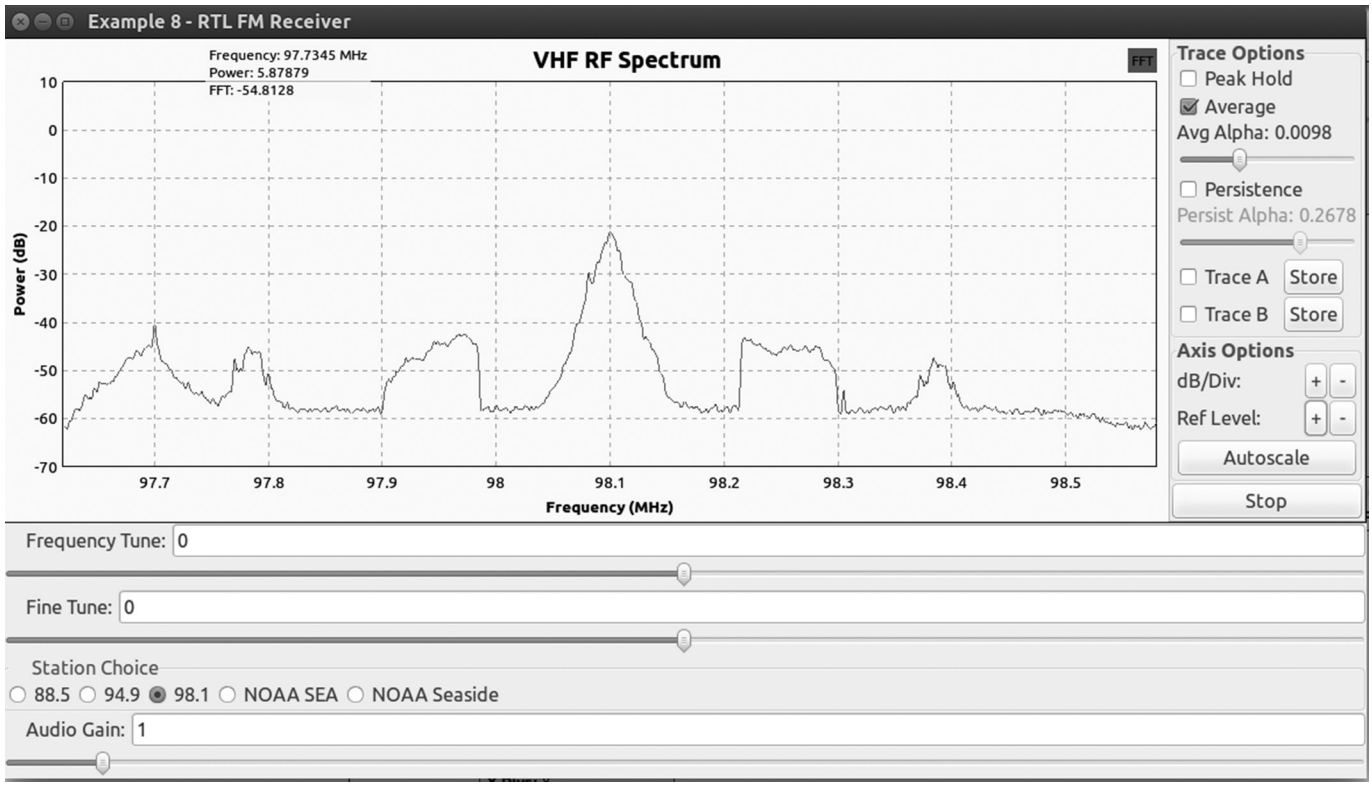


Figure 14.16 — The user interface window for the operating FM broadcast receiver created from the flow graph in Figure 14.15.

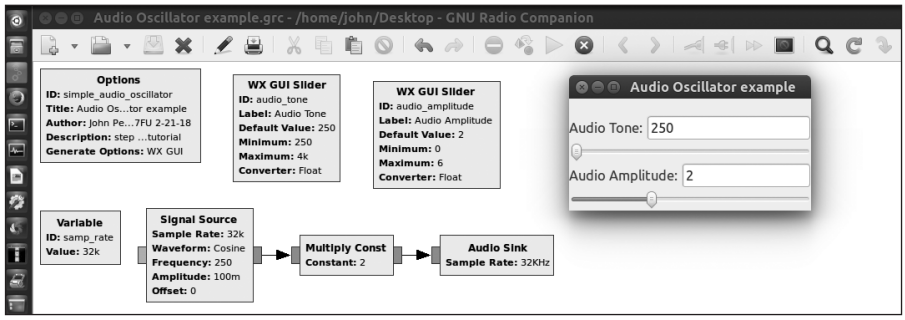


Figure 14.17 — The flow graph for the audio oscillator.

quires user-specified parameters. In the parameter step, right-click each block, select *Properties*, and enter the parameter values in the available fields. Each block in the flow graph diagram is opened in turn, and the desired parameters are entered in each block. Typical parameters are familiar values such as gain, frequency, bandwidth, and so on, as well as some less familiar values such as *Sample Rate* and *Output Type*. The *Sample Rate* and *Output Type* parameter values are unique to DSP as discussed below. These DSP parameters are set by default in many flow graphs and automatically propagated into every DSP block. For one to meaningfully change these default parameters, a basic understanding of DSP theory is required as described in Step 4.

Step 3 — the Logical Interconnections: Each DSP block must be connected to the other blocks by a data link arrow in a logical manner. The data links begin with the "input" or *Source* location in the flow graph and end with the "output" or *Sink* in the flow graph.

Step 4 — DSP Parameters: It is in this fourth and final step that authoring a flow graph involves DSP issues. Let's begin with the special *Variable* block located in the left-hand corner of the work area. That block is automatically present whenever *GRC* is opened for programming. The *Variable* block is the location for the base line *Sample Rate* parameter for each block in the DSP. The default parameter is 32,000 samples per second (SPS). That default parameter is automatically propagated to the *Sample Rate* field in each DSP block in the flow graph work area. The default *Sample Rate* value is a good starting place for many DSP flow graphs and can be used as a default for many designs. Advanced DSP designs may require different sample rates and adjustments to the sample rate parameter in the DSP blocks.

Another default DSP parameter is the *Output Type* parameter for each DSP block. The default parameter is termed *Complex* or I and Q. This default parameter is useful for many but not all radio-related DSPs. The flow graph example described below illustrates a flow graph where the default *Output Type* must be changed for the simple flow graph to function. (See the **Modulation** chapter for more information about complex signals and modulation.)

FLOW GRAPH EXECUTION

In practice, the above outlined flow graph is created in the *work area* of the *GRC*. To operationalize the flow graph, select the *Execute flow graph* control (a green arrow on the screen) from the tool bar above the *work area*. After the software compiles, the DSP will spring to life. The operational GUI appears on the screen with the control functions displayed as shown on the right side of the screen in Figure 14.17.

SPECIFIC FLOW GRAPH EXAMPLE: SIMPLE SINGLE-TONE AUDIO OSCILLATOR

Step 1 — the Blocks: Figure 14.17 is a screen capture of the audio oscillator flow

graph. The operating GUI is superimposed on the right side of the figure. The operating GUI only appears only after the *Execute flow graph* is selected from the tool bar.

The flow graph blocks are selected from the library off the right-hand edge depicted in Figure 14.12 (not Figure 14.17) and placed in the work area. At a glance, the flow graph block diagram is constructed to look essentially the same as a traditional analog radio block diagram with the same components. The diagram and component content reflect standard signal processing practice. A traditional circuit block diagram for a simple sine wave oscillator with audio output would consist of series of blocks linked by lines: an oscillator block, a frequency control, an audio amplifier, an audio volume control, all connected one after the other, to an audio output block.

In this audio oscillator example, the signal processing pattern is the same for the traditional analog and the *GNU Radio* digital implementation. With a DSP, the interconnections between blocks are no longer signal links (wires in traditional circuits) but digital data links, as we will see in Step 3 below.

Step 2 — the Parameters: Most of the parameters for each DSP block are visible in Figure 14.17. Examine the parameters of each block with a right-click on the block and select *Properties*. Right-click the *QT GUI Range* control block with the *ID audio_tone* located at the top of the work space. This block is the control block for the oscillator frequency or audio tone. Note the block ID, *audio_tone*, and the tone's tuning range from 250 Hz to 4 kHz in 1 Hz steps. This block connects to and controls the *Signal Source* DSP block output frequency, which is the audio oscillator tone in this example.

The *ID audio_tone* comprises the software link between the control block and the *Signal Source* DSP block. The term *audio_tone* is entered into both the *Signal Source frequency* parameter field and the *QT GUI Range ID* field, thereby connecting the two blocks in software.

In the same manner, the audio volume control is linked via software to the audio amplifier. Select *Properties* in the *QT GUI Range* control block with the *ID audio_amplitude*. The multiplication values range from 0 to 10. These numbers are factors that tell the *Multiply Const* DSP block to multiply the digital data stream amplitude for audio volume control. The ID *audio_amplitude* denotes the software link to the *Multiply Const* DSP block. That ID is entered in the *constant* parameter field in the *Multiply Const* DSP block. The two blocks are linked in the same way as the audio tone frequency is linked to the audio oscillator. The *Sample Rate* value from the *Variable* block, 32k, is automatically propagated to the other DSP blocks but not to the two control blocks.

Step 3 — the Logical Interconnections: To complete the basic DSP flow graph, each block is logically interconnected to the others. Additional attention to details are important in this step. Not appreciated in the screen capture is how the *Multiply Const* DSP block became configured to have only single input and single output ports. This block and many other blocks from the DSP library are pre-configured for multiple input and output ports. When placed in the *GRC* work area, pre-configured blocks from the library are reconfigured for the desired number of ports to fit the flow graph. To reconfigure the number of ports, right-click the block, select *Properties,* and enter a value for the number of desired ports in the appropriate port parameter field. In this example, a value of 1 was entered in the *Input Port* field of the *Multiply Const* DSP block. Select OK, the window closes, and the block appears on screen ready to use with a single input and single output port.

Once the block ports are configured, the DSP blocks must be linked with the data stream. To form these data stream links, left-click on the output port of the *Signal Source* DSP block, followed by a left-click on the input port of the *Multiply Const* DSP block. A data link arrow appears, and those two blocks are digitally connected. The output of the *Multiply Const* DSP block connects in a similar manner to the input port of the *Audio Sink* DSP block.

The flow graph is almost complete. The two control blocks have no visible data link arrows. They are automatically connected via the software *ID* parameters described above. Next, we consider the DSP parameter step.

Step 4 — DSP parameters: The audio oscillator flow graph functions fine with the default *Sample Rate* parameter in conjunction with modern computer sound cards. No change is required for the *Sample Rate* parameter. The *Output Type* parameter is different and must be changed from the default setting for this flow graph to function.

The *Output Type* parameter for each DSP block is displayed in a drop-down menu, found by selecting *Properties* for that DSP block. The *Output Type* is also visually displayed in the flow graph by the color of the input and output port tabs on the blocks. (This color code is a useful bug fix hint with advanced DSP programing and not central for this description. The change in color is only visible when viewing the screen capture or flow graph on a computer screen.) The desired *Output Type* for this flow graph is "Float" (digital data using floating-point values). The *Output Type* parameter for each DSP block in this example must be the same: "Float." Right-click and select *Properties* for each DSP block in the flow graph, navigate to the *Output Type* parameter, and select "Float" from the drop-down menu. The two control blocks don't require any DSP parameters.

Once all steps are completed, the flow graph is ready to be executed. Upon execution, a new GUI appears on the screen with the controls displayed. That GUI screen is superimposed in the upper right-hand corner of the screen capture in Figure 14.17. Listen for the 250 Hz audio tone in the speaker. Adjust the audio volume and audio frequency with the GUI controls. You can also experiment with adjusting the sample rate in the *Variable* block or parameter values in any block and observe the effect these changes have on the operation of the flow graph. (The GRC flow graph for this example is included in the downloadable supplemental package for this book. The file, Chapter 14 - Audio Oscillator example.grc, can be executed in the *GNU Radio* application.)

THE REST OF THE STORY: BASIC DSP THEORY APPLIED TO A FLOW GRAPH

To better understand this DSP flow graph example and to aid authoring more advanced DSP flowgraphs, basic information about the relationship of sample rate, data stream output type, and DSP is important. (See the **DSP and SDR Fundamentals** chapter for a detailed discussion of the analog/digital conversion process.)

Sampling is a foundational aspect of DSP. DSP comes into action in the A/D converter where an analog signal is converted to a digital data stream. The DSP acts to manipulate the series of data points which form a replica of the sinusoid signal as stream of digital data. The greater the number of samples, the more closely the replicated digital data corresponds to the original sinusoid signal. After digital processing, the digital data stream can be reconstructed to generate the familiar analog sound or radio signals with a D/A converter.

Connecting the sampling concept to a functioning flow graph, each DSP block in the flow graph must function with a compatible sample rate parameter. *GRC* automatically enters the default sample rate for each DSP block, which is fine for many DSP processes. Advanced DSP techniques using *GNU Radio and* GRC call for a wide range of sample rate parameters beyond the default value. The sample rate parameters can also be individually adjusted for each DSP block in the flow graph. These sample rate changes are accomplished using *Resampler* DSP blocks. Sample rate adjustments within a single flow graph are termed "multi-rate" sampling, a common practice with advanced DSPs such as those for SDR applications.

Another foundational concept regarding DSP is to understand that the sole purpose of each DSP block is to mathematically manipulate the digital data stream. Fortunately, with *GNU Radio* and the *GRC* graphical approach

to programming, the user is not required to understand or program the underlying mathematics. The DSP functions in *GNU Radio* DSP library are preloaded with the necessary mathematical algorithms to perform their intended DSP function.

The data stream math algorithms are unique in each DSP block. The operation can be single purpose, such as for the *Multiply Constant* DSP block, which functions as an amplifier or attenuator by simple multiplication of the digital data stream amplitude. Other DSP blocks provide filtering or mixing functions and frequently combinations of these functions. All operations are performed mathematically based on algorithms programed by the developers into each DSP block.

Not only is the data stream manipulated in a mathematically determined manner, but the mathematical structure of the digital data stream has an impact on how a flow graph can function. Return to the flow graph in Figure 14.17 where the *Output Type* DSP parameter reflects the structure of the digital data stream that communicates among the DSP blocks. The mathematically processed digital data stream *Output Type* can be customized as necessary for advanced DSP processes.

As a practical matter, the default *Output Type* digital data structure for most *GNU Radio* DSP blocks is set by default to "Complex" or I and Q data. The default setting "Complex" is the preferred data type for many, if not most, ordinary radio related DSP applications. Some DSP flow graphs, such as those employing phasing techniques and the audio oscillator example in this case, require an *Output Type* that is different from the default "Complex." (Some SDR hardware digital data output types are derived from different math conversions and require a non-default data stream structure conversion in the flow graph, as well.) In other words, one type of digital data stream doesn't fit all. The *Output Type* data stream parameter can be quite specific for the some SDR hardware and quite specific for some DSP process. Mastering the details of *Output Type* for your flow graph constructions will come with more in-depth understanding of DSP fundamentals. For further information on sampling rate choices and *Output Type* digital data streams, review the information on the GNU Radio website.

LEARNING RESOURCES: DSP COMPETENCY AND *GNU RADIO*

This introduction to *GNU Radio*, *GRC* and flow graph construction presents DSP at the high level of blocks and parameters, consistent with *GNU Radio*'s high-level graphical environment. But that is only the beginning of a journey into the world of DSP. The **DSP and SDR Fundamentals** and the **Receiving** chapters contain much useful information about understanding DSP. That information offers important background and context to more in-depth understanding of DSP to apply to *GNU Radio*, although not specific to *GNU Radio*.

DSP competency is readily learned using *GNU Radio* as a tool for experiments and simulation. The online educational resources specific to *GNU Radio* are great learning resources. Detailed examples of DSP applications and parameter selection, awkward and sometimes confusing in a text format, are much more easily understood via online video tutorials. The Ettus Corporation-sponsored series of DSP tutorials (**files.ettus.com/tutorials/labs/Lab_1-5.pdf)** and the related video series (**www.youtube.com/playlist?list=PL618122BD66C8B3C4**) are highly recommended for this learning format.

For a more academic treatment of DSP consider both the **DSP and SDR Fundamentals** chapter in this *Handbook* and the DSP texts by Richard Lyons. The Lyons texts provide a particularly "ham friendly," intuitive approach to understanding DSP. (See the ***GNU Radio* References** section.)

INSTALLING *GNU RADIO*

The very first step for a new user is to review the *GNU Radio* web portal for authoritative information (**gnuradio.org/redmine/projects/gnuradio/wiki**). The *GNU Radio* developers have published a rich source of detailed information to help the new user get started using *GNU Radio*. From that portal page one can link to installation instructions, tutorials, latest news, and in-depth information about the software development process. Refer to it frequently, since the content changes with the ongoing software development.

To get started with *GNU Radio*, a particularly appealing option is to use the *GNU Radio Live SDR Environment* (**wiki.gnuradio.org/index.php/GNU_Radio_Live_SDR_Environment**). The *Live SDR Environment* allows the user to implement *GNU Radio* without installing *GNU Radio* on the computer hard drive. The latest versions of the *Ubuntu* operating system and *GNU Radio* software are downloaded from the Internet to a storage medium (for example, DVD or USB memory stick) as an executable file. Upon executing from the medium, the *Live Environment* opens in a complete *Ubuntu* operating system with the *GNU Radio* application pre-installed. *GNU Radio* and *GRC* can be immediately executed to run at essentially full capability.

To install *GNU Radio* directly to a computer hard drive, installation from the official Source with the "*build-gnuradio script*" is far and away the most dependable and highly recommended approach (**wiki.gnuradio.org/index.php/InstallingGRFromSource**). Installing *GNU Radio* from the Source with the *build-gnuradio* script automatically installs the most recent *GNU Radio* version, all add ons, and the latest driver for compatible SDR hardware, the *Universal Hardware Driver* (*UHD*). The *build-gnuradio* script is a preferred method for initial installation and subsequent updates of *GNU Radio* and UHD as needed. Step by step "ham friendly" help with the installation process is also available at **www.w7fu.com**

GETTING THE MOST OUT OF *GNU RADIO*

Support for *GNU Radio* is available online via the "discuss-gnuradio" message reflector (**lists.gnu.org/mailman/listinfo/discuss-gnuradio**). Questions can be posted on the reflector, and the *GNU Radio* developers respond via email. The reflector is very active and a rich source of information.

The most efficient and user-friendly approach to get full benefit of what *GNU* Radio has to offer is to implement *GNU Radio* in the Linux *Ubuntu* operating system. The *GNU Radio* developers develop and maintain *GNU Radio* in *Ubuntu*. Potential cross-platform bugs and incompatibilities are avoided by adopting the same *Ubuntu* operating system that the developers use. *GNU Radio* is reported to operate on Windows and Mac OSX, but there are challenges with full implementation with these operating systems, which make them best avoided by beginners.

Computer users unfamiliar with Linux operating systems shouldn't be intimidated by using the *Linux Ubuntu* operating system. *Ubuntu* has the look and feel and straight forward ease of use of MS Windows. The *Ubuntu* operating system is open source, downloadable via the internet, and can easily be installed in a separate hard drive partition if necessary. Installing *Ubuntu* via the internet is straightforward and quick. The official *Ubuntu* website provides the latest downloadable *Ubuntu* versions and instructions for using *Ubuntu* (**www.ubuntu.com/download/desktop**).

GNU Radio is a dynamic application with version updates on an irregular but frequent basis. It is important to keep abreast of the changes via the web portal and learn to update the application re-using the *build-gnuradio* script as necessary. Follow the "Latest news" for updates and other information (**gnuradio.org/redmine/projects/gnuradio/wiki**).

For serious DSP authoring and most SDR projects, avoid installing the Linux "distribution" versions of *GNU Radio*. The "distribution" versions of *GNU Radio* are built into the *Ubuntu* operating version at the time of that operating system's release. These "distribution" versions can be installed easily from within the *Ubuntu* version. Because *GNU Radio* is built into the *Ubuntu* version at the time of initial release, these *GNU Radio* versions are typically out-of-date, have outdated *UHD* drivers, and are installed dif-

ferently within the file system from the versions downloaded from the official Source. The UHD might not work with newer SDR hardware. Updating and adding DSP blocks to distribution versions is not possible. For all of these reasons it is best to avoid installing a "distribution" version of *GNU Radio.* Instead install a full installation of the current *GNU* Radio version.

A capable computer is necessary to use *GNU Radio* to its full real-time DSP potential. The developers recommend a dual-core CPU at the i3 level or above to users. Single-core and early dual-core CPUs lack the speed and capacity for many ham DSP designs.

GNU RADIO, RADIO HARDWARE, AND UHD

To implement real-time SDR radio projects with *GNU Radio*, it is necessary to use compatible SDR hardware. The *Universal Hardware Driver* (*UHD*) is a dedicated software package to interface *GNU Radio* and compatible SDR hardware. Of late, newer generation SDRs have appeared that interface with a multi-layered PPA system of *GNU Radio* packages or a suite of applications. The documentation for compatible hardware will refer to the *GNU Radio UHD* capability or to a PPA *GNU Radio* package. Given the popularity of *GNU Radio* with the DSP developer community, *UHD* or the *GNU Radio* PPA is commonly specified for modern SDR hardware.

The latest version of *UHD* is installed automatically whenever *GNU Radio* is installed via the *build-gnuradio* script from Source. The *GNU Radio* PPA requires a separate and additional installation. The ever-expanding list of compatible hardware is published at **wiki.gnuradio.org/index.php/Hardware.** Searches of the internet and close reading of the specifications can sometimes find *UHD*-compatible SDR hardware that hasn't yet reached the official list.

GNU RADIO REFERENCE MATERIAL

J. Petrich, W7FU, "Digital Signal Processing and GNU Radio Companion," Jul/Aug 2017, *QEX*, pp. 41-46.

J. Petrich, W7FU, T. McDermott, N5EG, "Digital Signal Processing (DSP) Projects: Examples of GNU Radio and GRC Functionality," Sep/Oct 2014, *QEX*, pp. 25 – 30.

R. G. Lyons, "Understanding Digital Signal Processing" ISBN 0-201-63467-8

"GNU Radio Workshop," Microwave Update, October 2019, North Texas Microwave Society (**microwaveupdate.org**)

www.w7fu.com — step by step information on installing and maintaining *GNU Radio*, with examples of ham radio oriented SDR flow graphs.

www.reddit.com/r/GNURadio — a very active *GNU Radio* development community.

Suggested reading: **wiki.gnuradio.org/index.php/SuggestedReading**

14.4.2 FPGA Data Engines

What actually implements the DSP functional blocks referenced in the previous chapters? In the early days of amateur SDR and DSP, the only suitable hardware platform with a reasonable cost was a high-end PC or specialized DSP or graphics processing microprocessors (microcontroller units – MCU). These are still used today but after some years in transition, amateur SDR is largely implemented by FPGAs — Field-Programmable Gate Arrays.

An MCU executes instructions from a predefined instruction set in sequential order; the hardware is fixed, but the sequence of instructions is programmable. An FPGA, on the other hand, has no fixed instruction set or sequence of instructions, operates on data in parallel, and has programmable logic and interconnections. Software defined radio (SDR) implementations benefit greatly from the FPGA parallel hardware architecture, since they can be configured to implement the required math operations directly in hardware at very high speed. FPGA ICs are now available inexpensively and with a sufficient number of gates (in the 10s of 1000s) to implement high-performance SDR functions.

The FPGA as typically used by non-professional developers is part of a general-purpose computing unit, also known as a *data engine*. Support ICs are included in the package to allow the developer to interact with the FPGA and program it. Other data ports are available, depending on what applications the data engine is designed to handle. When combined with external RF hardware and either an on-board user interface or the ability to stream data to an external user interface, the result is an effective SDR development environment. The example in **Figure 14.18** combines the SDRStick receiver and transmitter with a BEMICROCV-A9 data engine. An Ethernet port provides enough bandwidth to implement the user interface and display on a PC. This is a thin-client implementation as described in the previous section on User Interfaces.

In a series of Hands-On SDR columns in *QEX* magazine beginning in 2014, Scotty Cowling, WA2DFI, described several data engines and the software development environment needed to program the FPGAs. Several of the key columns dealing with FPGA development are included with the online supplemental information for this book.

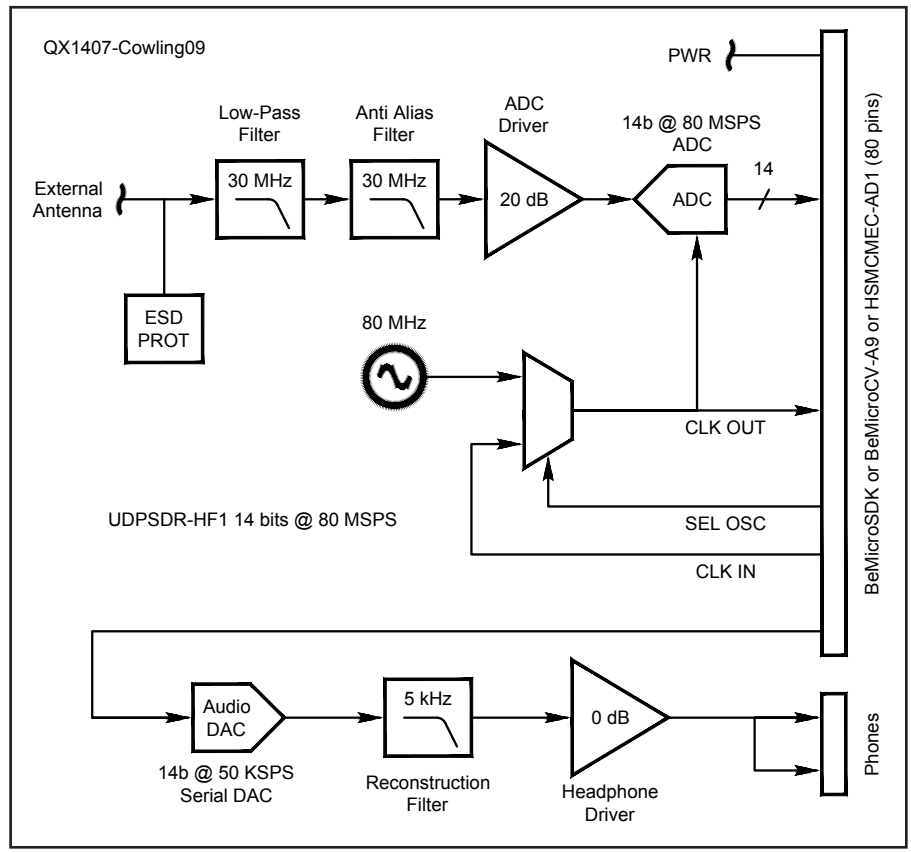


Figure 14.18 — An SDR development system based on the BEMICROCV-A9 data engine and separate SDRStick receive and transmit modules.

14.5 Transverters

The material in this section is largely taken from material provided by Paul Wade, W1GHZ, and Wayne Overbeck, N6NB. Along with downloadable materials in the supplemental information package, the print and online articles in the references provide much more detail and technical depth.

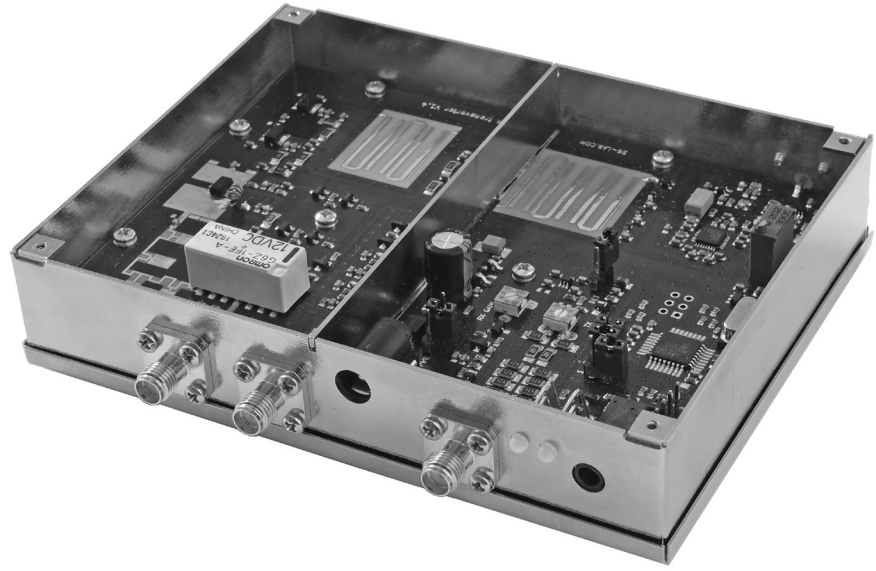
Transverters allow an HF or VHF transceiver to operate on a band it doesn't support by converting both the transmitted and received signal. As a result, the full features of the transceiver are available on the converted band. At one time, building and installing transverters presented a serious challenge for all but the most experienced and technically adept amateurs, but today's off-the-shelf transverters are almost as easy to install as a commercial transceiver. For those who prefer the challenge of building their own transverters from kits or from scratch, those options offer a real sense of accomplishment, as well.

Non-repeater operation on the VHF/UHF/microwave bands is usually CW or SSB, requiring a multimode transceiver. (FT8, JT65, and similar digital modes use the SSB or SSB DATA features.) Some transceivers cover 6 meters, 2 meters, and 70 cm in addition to HF. Although some VHF/UHF transceivers such as the TS-2000 and IC-9700 cover 23 cm (1296 MHz), multimode radios that cover 222 – 225 MHz, 33 cm (902 MHz), and the bands above 2.3 GHz are not available as of early 2020. Wideband SDR receivers that cover from VHF through several GHz are widely available, but companion transmitters are just starting to become available. That limits the equipment choices to surplus equipment that can be modified — and transverters.

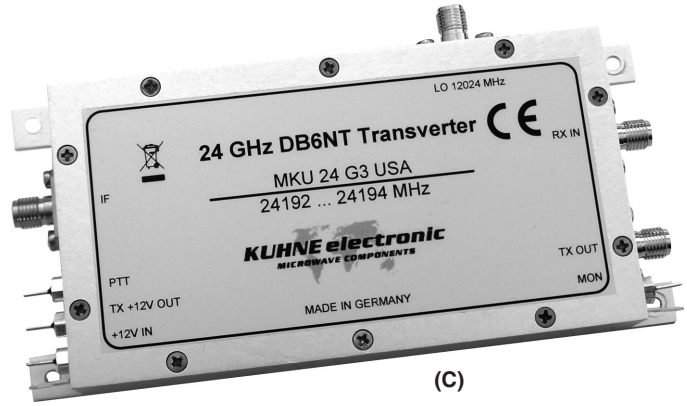
Similarly, for the amateur MF and LF bands at 630 meters (472 – 479 kHz) and 2200 meters (136 kHz), commercial transceivers covering these bands are uncommon. Surplus equipment can be adapted to amateur use, but availability is quite variable and the equipment is likely to be designed for operation on fixed channels. As a result, transverters are likely to be used on those bands for the foreseeable future.

Commercial transverters such as those in **Figure 14.19** are available as finished packages for 6 meters, 2 meters, 222 MHz, 432 MHz, 902 MHz, 1296 MHz, 2304 MHz, 3456 MHz, 5760 MHz, and 10 GHz. Recently, transverters have also become available for much higher frequency bands, including 24 GHz, 47 GHz and 76 GHz. Manufacturers that offer complete transverters include DB6NT (**shop.kuhne-electronic.com**); Down East Microwave (**www.downeastmicrowave.com**); Q5 Signal (**www.q5signal.com**); and SG-Lab (**www.sg-lab.com**). A number of small shops also produce kits or finished prod-

(A)



(B)



(C)

Figure 14.19 — The Q5 Signal 222 MHz transverter (28 MHz IF) is shown at A. At B is the SG-Lab 2.3 GHz transverter (430 MHz IF), and at C is the Kuhne MKU 24 G3 transverter for 24 GHz (432 MHz IF).

ucts as single PC boards that can be mounted in an enclosure and integrated into a system. For the kits and single-board products, an internet search may turn up vendors.

Becoming a member of a VHF+ club is also a good way to find used and surplus equipment along with information about vendors and their products. These groups are typically great for mentoring and technical advice on transverters, as well as tips on operating.

14.5.1 Transverter Basics

A transverter is both a receive converter and a transmit converter for a conventional HF or VHF transceiver, usually operating on 10 meters or 2 meters. The HF/VHF transceiver acts similarly to the intermediate frequency (IF) sections of a conventional superheterodyne radio and is referred to as the *IF radio*. For those using a full-featured commercial transceiver, the various receive and transmit functions are also available on the converted band, creating a high-performance VHF/UHF/microwave (or MF/LF) transceiver. Some transceivers also have the ability to display the converted frequency so that the operator does not have to convert the lower frequency display to the higher band. (From this point on, discussion will assume the transverter is supporting a VHF/UHF/microwave band.)

An IF radio operating on the 28 MHz or 144 MHz band will cover either 2 or 4 MHz of the band that is being converted if the operator tunes from 28 to 30 MHz or 144 to 148 MHz. That is adequate to cover the entire weak signal portion of a VHF/UHF/microwave band. However, the amateur bands above 400 MHz are much wider than the 10 meter and 2 meter bands, so the entire band will not be covered. For FM repeater coverage of 432, 902, or 1296 MHz, a dedicated mobile or handheld FM transceiver is needed, since FM activity may cover a span of 20 MHz or more.

The 28 MHz band is commonly used as an IF for 50 MHz, 70 MHz (in Europe), and 144 MHz transverters, and sometimes for transverters on higher bands. The most common IF band for the higher bands is 144 MHz. For transverters on higher bands, 432 MHz is becoming more popular as an IF both to provide coverage of more of the band and to achieve better technical performance.

Even older transceivers and single-band 10 meter and 11 meter transceivers can provide reasonably good performance as IF radios. Low-power transceivers such as the Yaesu FT-817/818 are widely used because most transverters are now designed for up to 5 W of IF drive, nicely matching the output power of such transceivers.

14.5.2 Integrating a Transverter with a Transceiver

Figure 14.20 shows the two most common ways of connecting a transverter and transceiver IF radio. Some transceivers have dedicated low-level transmit outputs as in Figure 14.20A, often available through an auxiliary connector, that can be connected directly to a transverter regardless of the power available at the main antenna connector. (A dummy load may have to be connected during transverter operation.)

If the transverter is rated to handle the power, a simpler approach is to turn the transceiver output power down to 5 W and connect its antenna connector to the IF port on the transverter that handles both transmit and receive signal paths. The transverter internally switches the IF port between the transmit and receive sections of the transverter. Most modern transverters have built-in attenuators that can handle up to 5 W of transmitter power from the transceiver. That's one reason why 5-W transceivers are so popular for use with transverters. This method is shown in Figure 14.20B. Of course, the operator has to remember to reduce the transmitter power of a higher power transceiver from the full output to 5 W when switching from "straight through" operation on 10 meters or 2 meters to transverter operation. Be aware that some transceivers transmit a full-power transient at the beginning of a transmission before the ALC system can control the output power. If this is the case, turning the power down manually is probably a poor option. **Table 14.1** shows some values for attenuators you can build to reduce transmit power.

If the transverter does not have the option to operate directly with the transceiver's main RF port, a TX/RX switch is required. Switching is controlled either by RF sensing or by the transceiver's transmit control line, such as a grounded PTT line or a positive-true TX ON signal. Performing switching control through the IF radio's RF port does eliminate one cable and connector set.

Optional preamps and power amplifiers may be mounted directly at the antenna for best performance. Because feed line losses are so high at microwave frequencies, many amateurs mount transverters and amplifiers in a weatherproof enclosure at the antenna and only send the IF signals, dc power, and any control lines from the transceiver.

Transverter users must also consider frequency stability, especially on the microwave bands. Most transverters are now designed to accept a highly stable 10 MHz "reference signal" from a *GPS-disciplined oscillator* or a *rubidium oscillator* (see the **Oscillators and Synthesizers** chapter). Commercial transverters usually have built-in local oscillators (LO) that are designed to be fairly stable. However, the best LO circuits will drift several kilohertz or more on microwave bands during warmup. Even after a long warmup, an internal LO may drift too much for modern digital modes. Most amateurs who want good performance will add an external GPSDO or Rubidium reference to "lock" their transverters on the correct frequency. One outboard reference signal can be fed into a distribution amplifier to serve several transverters.

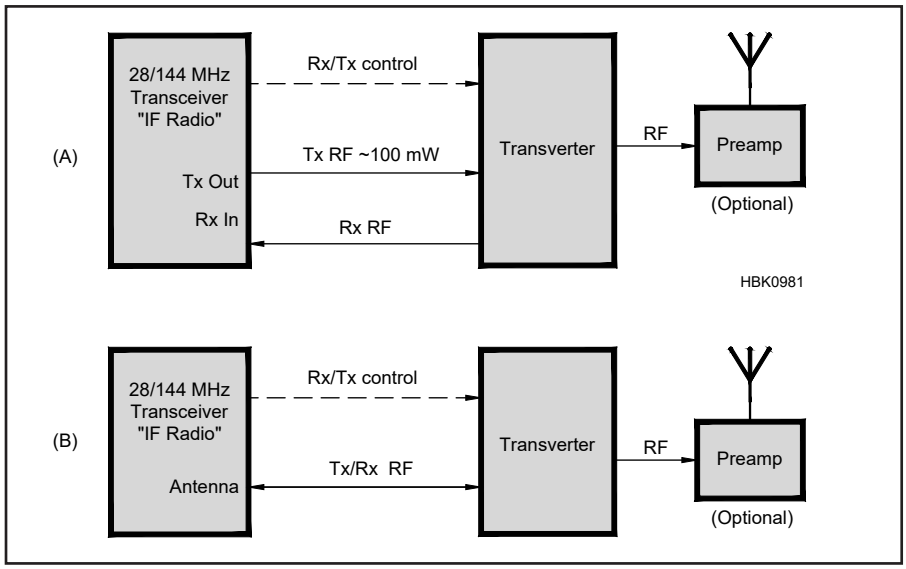


Figure 14.20 — The illustration at A shows a transceiver with receiver input and separate low-power transmit output connections. At B, the RF path is through a direct connection to the transceiver's main antenna connector, requiring the RF power output to be reduced to 5 W for most modern transverters.

Table 14.1

Attenuator Component Values

IF Power	10 W	6W	½ – 3 W	< 250 mW
First attenuator	23 dB	20 dB	18 dB	6 dB
R1	50 @ 8W	50 @ 4W	50 @ 2W	50
R2	27 * 4	43 * 4	56 * 4	100
R3	1.5k * 4	1k * 4	560 * 3 or 750 * 4	39
R4	56	62	62	150

Notes:
*4 means four resistors in parallel; * 3 means three resistors in parallel.
See Figure 14.25 for the attenuator schematic.
All resistances are in ohms and all resistors are ¼ W unless noted otherwise.

14.5.3 Basic Transverter Structure

Most radio amateurs now buy commercial transverters and install them as just described. This section is for those who wish to understand the inner workings of their transverters — or perhaps wish to build their own transverters.

The block diagrams of two typical transverter structures are shown in **Figures 14.21** and **14.22**. The transverter in Figure 14.21 has separate receive and transmit paths. A low-level output, typically 100 mW (20 dBm) or less, from the IF radio (labeled IF IN) is mixed with the transceiver's LO. The desired VHF mixing product is selected with the VHF band-pass filter and amplified by one or more low-power amplifier modules or ICs. This architecture requires the IF radio to supply a separate receiver input (typically labeled RX IN) and low-power transmit output (usually referred to as a "transverter output.")

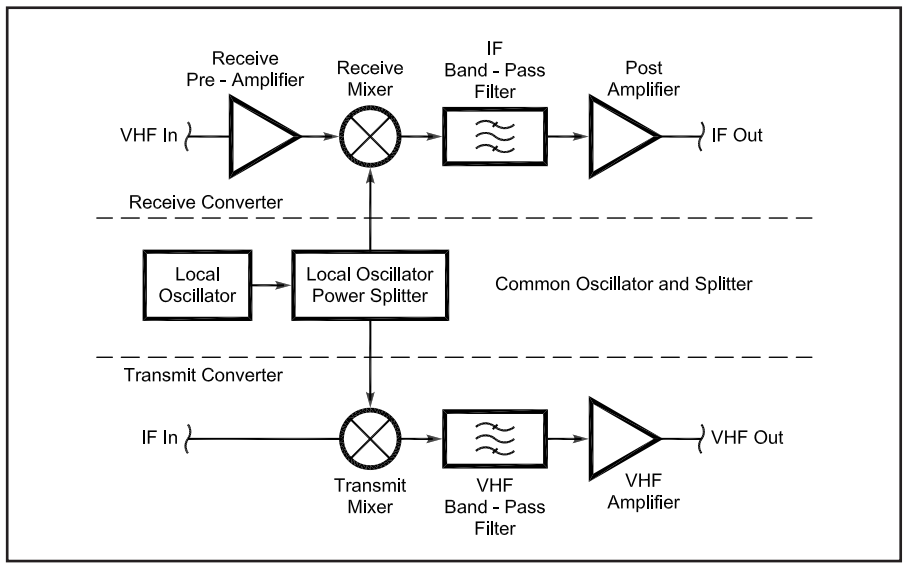


Figure 14.21 — Block diagram for a typical transverter. The upper chain is the receiving converter and the lower chain is the transmitting converter.

In Figure 14.22, the transceiver uses only a single *bilateral* mixer that converts signals flowing in either direction as transmit or receive. This allows the IF transceiver to operate as it normally does, with its internal transmit-receive (TR) switching routing signals. In the case of the 222 MHz transceiver, it is operating on the 24 MHz band. The transceiver output power must be attenuated or turned down to an acceptable level to avoid damaging the transverter's input circuits.

14.5.4 Mixers

The heart of a transverter is a mixer, sometimes called a "frequency mixer" or a "frequency changer," a nonlinear device that combines two signals at different input frequencies to produce new frequencies at the mixer output. These new output frequencies are the sum and difference of the original input frequencies. Mixers are discussed in detail in this book's **Receiving** chapter.

In the 222 MHz transverter of Figure 14.22, the mixer combines input signals with a 198 MHz local oscillator. To transmit, a 24-MHz signal is applied at the IF port from the transceiver; the sum frequency is 222 MHz, which is amplified and sent to the antenna. To receive, 222 MHz signals from the antenna combine with the local oscillator and the difference frequency, 24 MHz, is sent to the IF port on the 12 meter band.

A common choice for a transverter mixer is an inexpensive, double-balanced diode mixer. A high-level version will provide for better dynamic range (see the **Receiving** chapter). A wide dynamic range is important for rejecting out-of-band signals from other services, particularly TV, FM, and commercial/public safety signals, which are often present at the elevated locations common for portable operating.

14.5.5 Preamplifiers

Double-balanced mixers typically have a relatively high noise figure, so it is almost always necessary to use a low-noise preamp ahead of the mixer. The preamp can be a dis-

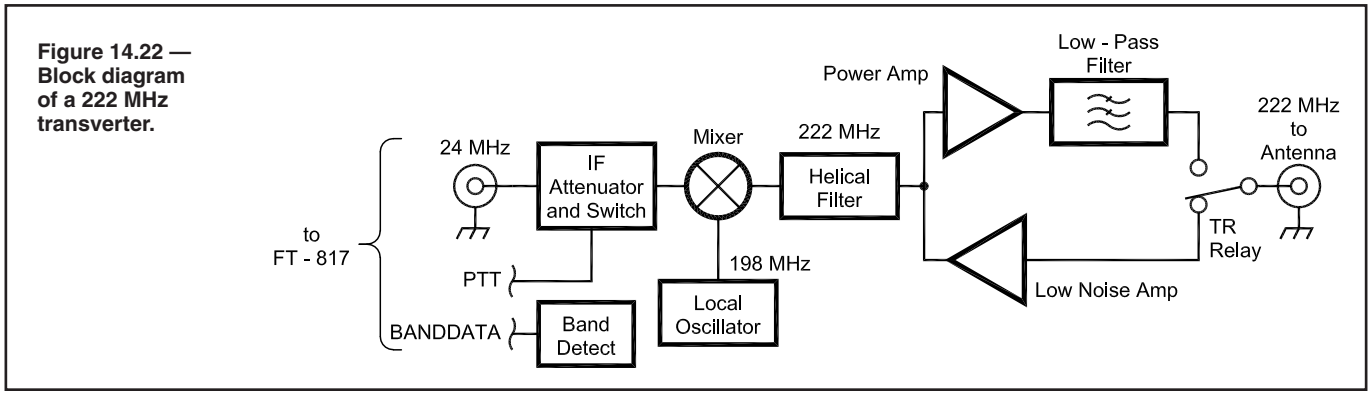


Figure 14.22 — Block diagram of a 222 MHz transverter.

crete semiconductor circuit, or an MMIC RF amplifier can be used. If a preamp is used, a band-pass filter should be used at the input. (See the **Receiving** chapter for more information on preamps, including some discrete component design references.) A wide-band preamp design useful up to 1296 MHz is provided in this book's downloadable supplemental information. (See "Universal MMIC Preamp" by Paul Wade, W1GHZ)

External preamplifiers are often mounted directly at the antenna with power supplied via the main feed line. An RF sensing circuit or a separate control line from the transceiver or transverter is required to route transmit RF around the preamplifier. Some transverters can be mounted at the antenna as well, and may include a preamplifier. Both techniques improve the noise figure (see the **Receiving** chapter) of the receiving chain and also reduce the antenna system cost by allowing less-expensive coax to be used.

14.5.6 Filters

Transverters typically use a filter before and after the mixer. Comb filters, helical filters, and lumped LC filters are all used to provide band-pass responses. A low-pass filter is generally sufficient for the IF input and output, since the IF radio's filters will be used. Another low-pass filter at the output of the transverter power amplifier path reduces harmonics. A band-pass filter at the antenna port can serve the same function and keeps spurious signals from being transmitted on the air. It also helps prevent overload or intermodulation from strong out-of-band signals on receive.

Regardless of filter type, both filters and mixers behave much better when they are terminated in a load impedance that is constant over a wide bandwidth. Filters also behave better when their inputs and outputs are terminated by the impedance for which they were designed. A 3-dB pad made up of chip resistors at filter and mixer inputs and outputs allows each to achieve a reasonable impedance match, which makes both circuits perform more predictably.

Filter designs are available in the **Analog and Digital Filtering** chapter or can be designed using the *ELSIE* software package that is included with this book's downloadable supplemental information. If amplifier modules are used, some models are less stable with capacitor-input filters, so the inductor-input designs are recommended.

Microstrip line and cavity filters are most often used in amateur microwave transverters. Microstrip line filters consist of patterns of printed circuits that are frequency selective. Cavity filters are usually constructed from small plumbing parts that are soldered to circuit boards and then tuned with setscrews. Waveguide filters are also used, most notably at 10 GHz and higher. These consist of a piece of waveguide with metal rods and irises inserted. Tuning is performed with screws.

If portable operation is planned at hilltop sites where commercial or government transmitters are in operation, you should consider using external band-pass filters to reject the extremely strong signals that will be present. Be aware that noise from these transmitters (and possibly intermodulation products) may be quite strong compared to weaker amateur signals.

14.5.7 Local Oscillator

A good local oscillator (LO) is usually the hardest part of any transverter design. It must provide a clean signal, because any undesired frequencies generated by the LO can become a source of "birdies" in the receiver. It must also be stable and have low phase noise. Low phase noise is important, as a noisy local oscillator will raise the noise floor of the receiver, particularly in the presence of strong undesired signals, either out-of-band or in-band. Above 50 MHz, the interesting signals are usually the weak ones, so a low noise floor gets them in the log.

A crystal oscillator is the most obvious LO solution (see the **Oscillators and Synthesizers** chapter). Good crystals have become more expensive and harder to find, except those made for frequencies used in computer hardware — those are cheap and produced in high volume. Oscillator modules are the usual choice, having all of the circuitry optimized for the crystal and temperature stability. Packaged oscillators don't have any provisions for frequency adjustment, but because they are inexpensive, several can be ordered and the module closest to the desired frequency used. In addition, many wide-band SDR transceivers also have transverter outputs that can operate over a wide range of frequencies, allowing more choices for the LO frequency.

The oscillator frequency is usually not high enough to use directly, so it must be multiplied. **Figure 14.23** shows a typical scheme for generating a 10 GHz signal from a VHF crystal oscillator. For this oscillator to be reasonably temperature stable, a temperature stabilized miniature oven is usually used. After warmup, a simple ovenized 200 MHz oscillator stays within a few tenths of a hertz over the time that typical contacts are made. The 200-MHz oscillator is multiplied by approximately 50 to achieve an LO for a 10.386 GHz radio. One hertz of variation in the crystal oscillator will result in about 50 Hz change at 10 GHz, a noticeable change that may cause problems on some digital modes. If the LO is to drive an 80 GHz or 145 GHz radio, the sensitivity to drift is ten times more critical. (See K7MDL's "Multiband Microwave Local Oscillator" paper in the References at the end of this section for an approach to developing a stable signal source that is designed for transverter operation.)

While multiplier circuits can be difficult to design and adjust, the oscillator modules used for digital circuits have a square wave output. These oscillators have very fast rise and fall times, so they are rich in odd harmonics. Separating a third or fifth harmonic with a narrow filter or diplexer and amplifying it with an MMIC produces satisfactory output.

Microwave LOs in commercial and scientific converters are usually phase locked oscillators or PLOs. These items cost over a thousand dollars if purchased new. However, these PLO "bricks" are sometimes available at flea markets and auction sources at low cost. The negatives of using a surplus PLO include the effort to retune them, acquisition of a special crystal, greater weight, and the need for –20 V at 1 A of power that most of them require. The positives include excellent phase noise, great stability after warm up, and low cost. Synthesized LO modules with low phase noise are also available, such as the digiLO from **g5systems.com**. Most of these can also be locked to a GPS-disciplined 10 MHz reference for very accurate and stable frequency control.

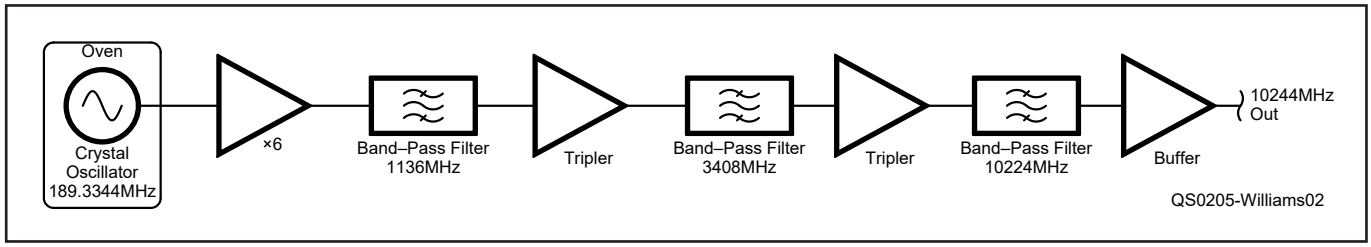


Figure 14.23 — A local oscillator multiplier chain for a 10 GHz transverter.

A high-stability TCXO is highly recommended in the IF radio for use with transverters. For the microwave bands, LO oscillator drift and microphonics can be a problem. It is important to keep the IF radio and the transverter at a constant temperature and avoid vibration. As noted earlier, most modern transverters have a provision for an external frequency reference that will lock the transverter on frequency with an accuracy of plus or minus only a few hertz even at 10 GHz.

14.5.8 Transceiver Power Output Control

Assuming your transceiver can be set to output ½ W or less, perhaps through a dedicated transverter output port, ordinary ¼ W resistors can be used as an input attenuator which may be fixed or variable. Be sure to test your attenuator design before using it between the transceiver and transverter. In the 222 MHz transverter shown in Figure 14.22, two attenuators are used in series, one fixed and one variable. PIN diodes are used to bypass the variable attenuator on receive, but a smaller fixed attenuator is left in the line to protect the mixer in case of a switching failure.

Most modern commercial transverters have built-in attenuators so they can handle the full output power of a 5-W transceiver. If your transmitter outputs 100 W, however, it's up to you to remember to turn the power down every time you switch from straight-through HF or 2 meter operation to transverter operation. If you forget to reduce the power in the heat of battle to try to work a new multiplier, you may not only miss the multiplier but also fry the input circuitry of your transverter and set yourself up for a costly repair.

Another option for power control is to use an external dc voltage to control the transceiver's ALC function. For most transceivers, a negative dc voltage applied to the ALC input can reduce the output power to nearly zero. The ALC input is a high-impedance circuit so little current is required. **Figure 14.24** shows a simple circuit that will work with most transceivers. The Inverting DC-DC Converter project in the downloadable supplemental information for the **Power Sources** chapter will also work and can be connected to the radio's power supply.

14.5.9 Power Amplification

Transverter modules commercially available as kits have an output of 10 to 100 mW (10 to 20 dBm) and usually require additional amplification to be able to fully drive a stand-alone solid-state amplifier to the 10 to 50 W level. RF amplifier modules designed for commercial service are available to amplify the low-level signals to useful power levels.

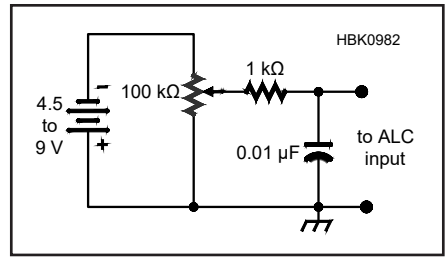


Figure 14.24 — Negative voltage ALC power control circuit. Note the battery polarity and chassis connection. The 0.01 µF capacitor filters out any RF that might be present, and the 1 kΩ resistor limits current into the ALC input.

The power level at the output of a mixer/buffer might be as low as 100 µW (–10 dBm) and must be increased to develop useful output power levels. MMICs are an easy and inexpensive way to raise the power level to 20 – 100 mW (+13 dBm to +20 dBm) required to drive higher power devices or modules. Gain at and above UHF has become much less expensive in recent years due to consumer devices operating at these frequencies.

Hybrid modular amplifiers can be used to easily reach the 10 – 25 W level. While they seem a bit expensive, they are easy to use, and usually the only other components required are a few capacitors and chokes. Higher power modules require more current and heatsinking, increasing the size and weight of the transverter. Amplifier modules made by Mitsubishi /Toshiba (**www.mitsubishielectric.com/semiconductors/products/hf/sirfpowermod/index.html**) are very common in amateur VHF/UHF FM transceivers and can also be used for transverters. Remember that the module must be capable of linear operation for use with SSB and AFSK modulation. Vendors such as RF Parts (**www.rfparts.com**) can supply modules in small quantities. (Also see the RF Power Amplifier Modules listing in the **Component Data and References** chapter.)

14.5.10 TR Switching and Sequencers

Just as in a transceiver, the transmit-receive (TR) functions must be switched at the appropriate time. The simplest approach is to use a relay activated by the transceiver's PTT circuit to switch the input and output circuits. If the timing of the PTT and RF output from the transceiver is well-behaved and some protective attenuation is provided at the receiver connections, this approach may work fine.

However, applying RF from a transmitter before the receive circuits have been disconnected can destroy a sensitive receive preamplifier or mixer. Many commercial transceivers can transmit power immediately upon being activated, before any external circuits can switch from transmit to receive. The unwanted power can burn out mixers and preamps, as well as coaxial relays that are not meant to hot-switch RF power — it doesn't take much power to damage a relay contact while it is switching. Another common problem is that some transceivers output a high-power transient until the ALC loop regains control of the transmitter power. This is addressed by using a sequencer. Several manufacturers such as Kuhne Electronic sell highly reliable sequencers for a modest price to address these problems. If you use one of these and carefully follow the directions, you should be fine.

Another pitfall in the TR switching process is caused by the relay itself. The coil that activates a relay may send a powerful voltage spike back into the transverter when it is de-energized, with devastating consequences for the low-level circuitry. Read the instructions from the transverter manufacturer carefully. If it says to add a protective diode across the relay coil, follow those instructions literally, observing the recommended polarity. (A small bypass capacitor across the diode will keep it from generating harmonics of any RF that is present.) If it doesn't say to take that precaution, consider doing it anyway.

Some approaches involve trying to delay the transceiver output, either by intercepting the PTT line from the microphone or keying devices or by controlling a Transmitter Inhibit input to the transceiver. Both of these approaches work well but need additional connections to the transceiver, are model-specific, and present additional points of possible failure.

A simpler approach is to have a single connection from transceiver to transverter, with PTT voltage added to the IF transceiver output feed line. A microcontroller can then monitor the switching operation. The article "A Smart Fool-resistant Conditional Sequencer" by Paul Wade, W1GHZ (see this book's downloadable supplemental information for the complete article) describes the Arduino-based controller (an Arduino Pro Trinket) suitable for adaptation to many transverter circuits. For example, an RF sensor can detect any transmitted power and switches the input circuit to a safe state to absorb unwanted power until the sequencer finishes all switching.

In **Figure 14.25**, the PIN diode switching circuit is controlled by the signal lines TXE (Transmit Enable) and RXD (Receive Disable). Both controlling signals for the PIN switch are asserted High. To receive, both are de-asserted. RXD is asserted, but not TXE, for the SAFE state, and then TXE is asserted along with RXD to transmit. TXE should never be asserted without RXD, a combination that would enable both transmitter and receiver simultaneously.

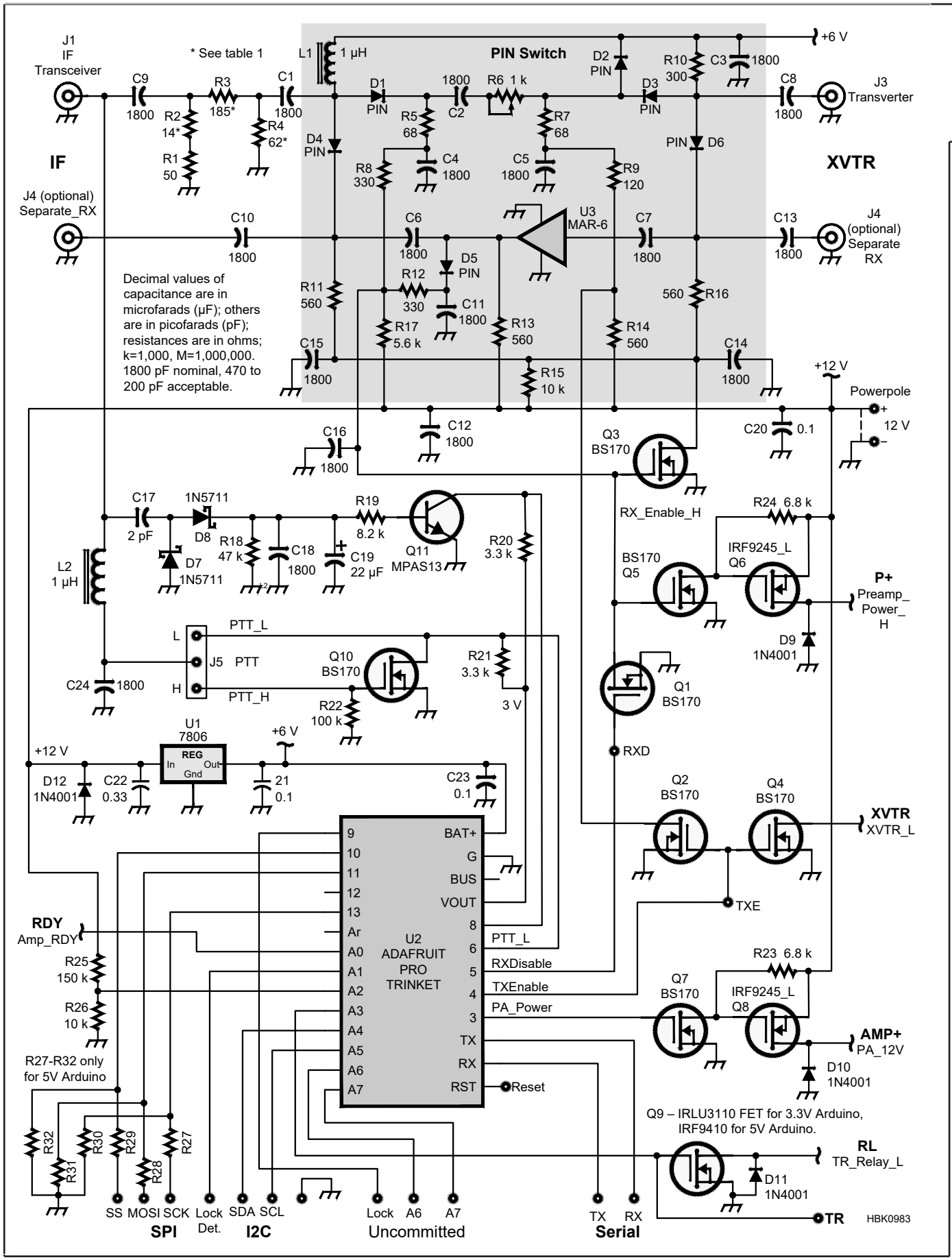

J1
IF
Transceiver
* See table 1
L1 1 µH
PIN Switch
+6 V
J3
Transverter
IF
XVTR
J4 (optional)
Separate_RX
J4
(optional)
Separate
RX
U3
MAR-6
Decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k=1,000, M=1,000,000. 1800 pF nominal, 470 to 200 pF acceptable.
+12 V
Powerpole
12 V
RX_Enable_H
BS170
IRF9245_L
P+
Preamp_
Power_
H
PTT_L
PTT_H
J5 PTT
3 V
RXD
XVTR
XVTR_L
TXE
RDY
Amp_RDY
RXDisable
TXEnable
PA_Power
AMP+
PA_12V
R27-R32 only
for 5V Arduino
Q9 – IRLU3110 FET for 3.3V Arduino,
IRF9410 for 5V Arduino.
RL
TR_Relay_L
TR
HBK0983
U1
7806
REG
In
Out
Gnd
U2
ADAFRUIT
PRO
TRINKET
BAT+
G
BUS
VOUT
TX
RX
RST
Reset
SS MOSI SCK
SPI
Lock
Det.
SDA SCL
I2C
Lock A6 A7
Uncommitted
TX RX
Serial

Figure 14.25 — A processor-controlled sequencer. The PIN switch circuitry at the top is controlled by the Arduino processor through the TXE (Transmit Enable) and RXD (Receive Disable) control lines.

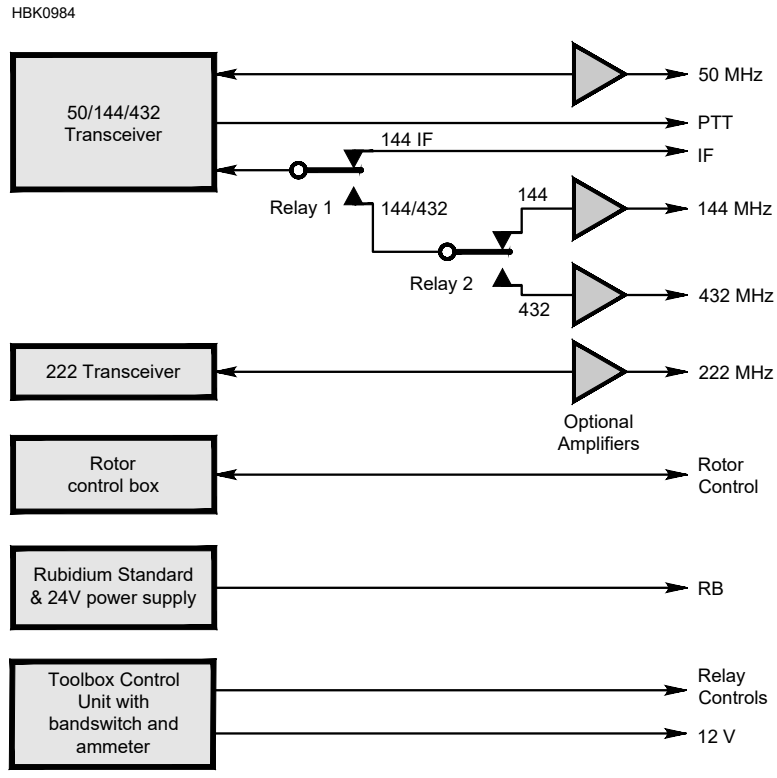


Figure 14.26 — Control unit for multiband transverter system by N6NB. The switch enclosure goes inside the toolbox or other microwave equipment box. It switches the IF, PTT line, and rubidium standard reference signal through the six bands from 902 MHz to 10 GHz.

14.5.11 Transverter Systems

Building a separate transverter for each band has been the norm for decades, but recent technological advances are making multiband transverters possible. For example, Down East Microwave is in the final pre-production stages (as of November 2019) of producing a six-band transverter covering 902, 1296, 2304, 3456, 5760, and 10368 MHz requiring only a single 144-MHz IF radio. Wideband SDR-based radios are also becoming available, moving transceiver performance to the UHF and microwave bands, with transverter opportunities extending into the millimeter-wave spectrum.

Another approach is taken by Wayne Overbeck, N6NB, in creating his "Ten-Band Toolbox Transverter" system designed for roving in VHF+ contests. (The full article describing the transverter system is available online; see the References list at the end of this section.) The system consists of two packages: an IF radio and control unit which goes inside the vehicle (see **Figure 14.26**) and a transverter package designed for mounting on top of the vehicle, **Figures 14.27** and **14.28**. The transverter package is integrated with antennas so that the whole package is easy to install and remove.

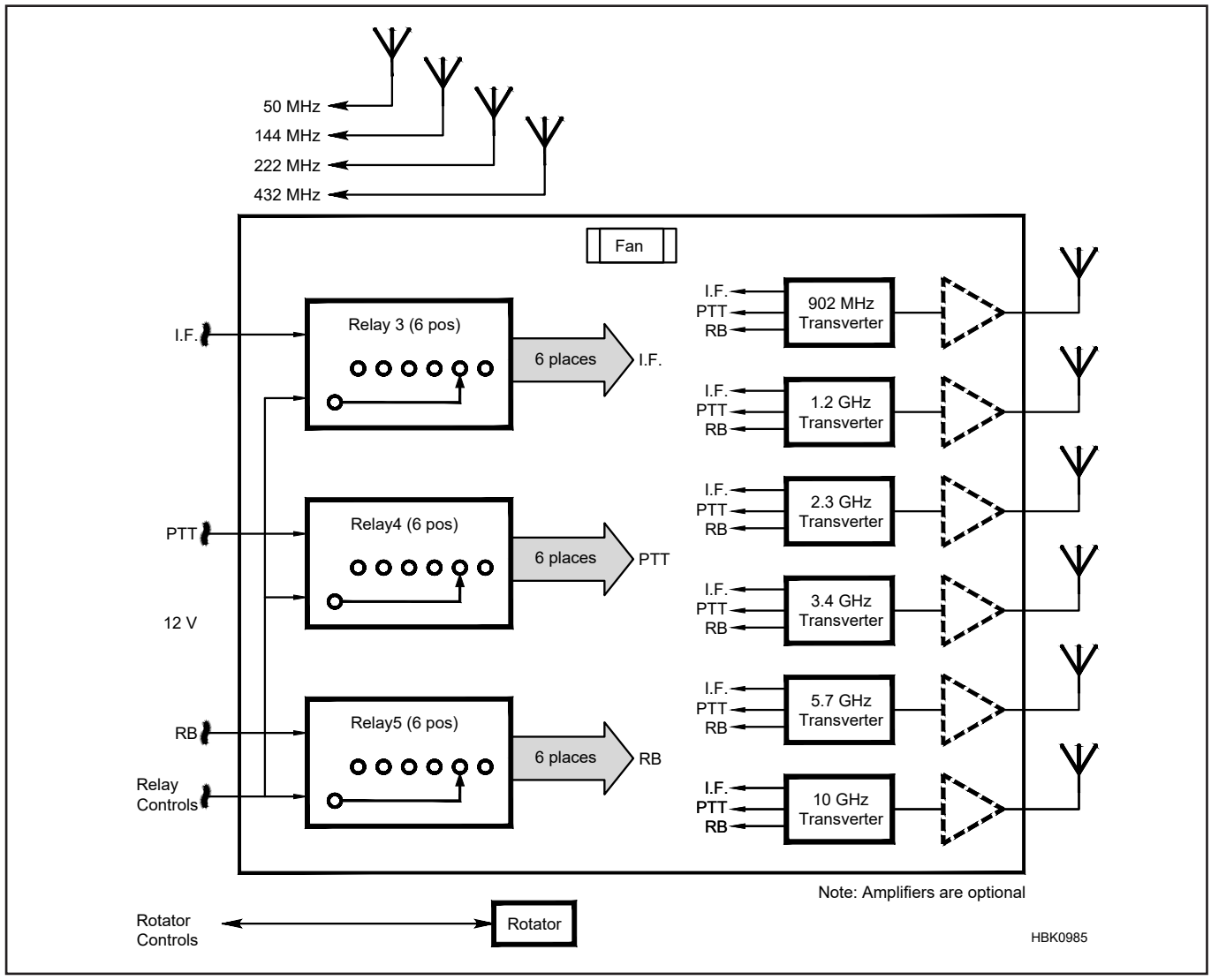


Figure 14.27 — Block diagram of integrated transverter package.

This package can be used as a fixed or mobile/rover station and covers all bands from 50 MHz to 10 GHz with a single transceiver and a group of transverters. A six-position or eight-position rotary switch is mounted in a small box to serve as a band switch. One set of its contacts is used to switch the PTT control line (the TR switching line) between transverters. Another set of contacts switches a dc voltage (usually either 12 V or 28 V) to control a six-position SMA relay (available on the surplus market) that routes the 144 MHz IF signals between transverters. The result is a multiband station with rapid single-knob band switching.

If additional switch positions are available, the relay can also be configured to transfer the IF radio's output directly to an antenna for straight-through operation on 144 or 432 MHz. Sometimes a 222 MHz transverter is also connected to this package, although that often requires switching the IF rig from 144 MHz to 28 MHz, the usual IF for a 222 MHz transverter. That's why many VHF operators have a separate transceiver and transverter for 222 MHz.

Figure 14.28 — Seven transverter packages designed to be mounted on the roof of the vehicle with antennas attached.

14.5.12 Transverter References

2200 and 630 Meter Transmit Converters and Amplifiers

njdtechnologies.net/monitor-sensors-630-meter-and-2200-meter-linear-transverters

Wilson, M., K1RO, "Monitor Sensors TVTR1 630-Meter Transverter," *QST*, July 2018, pp. 44 –48.

Wilson, M., K1RO, "Monitor Sensors TVTR2 2200-Meter Transverter," *QST*, Oct. 2019, pp. 47 – 49.

www.monitorsensors.com/ham-radio

www.n1bug.com/lfmf

Articles and Papers

Kostro, S., N2CEI, "Multiband VHF/UHF and Microwave Transverters, Rev C," Down East Microwave, Inc., **www.pnwvhfs.org/conference/2018/pdf/Steve-N2CEI-Multiband-Transverters- RevC.pdf**.

Lau, Z., KH6CP/W1VT, "Home-Brewing a 10-GHz SSB/CW Transverter," *QST*, Part 1, May 1993; Part 2, June 1993.

Lau, Z., KH6CP/W1VT, "RF" column, *QEX*.

Lewis, M., K7MDL, "Multiband Microwave Local Oscillator," **pnwvhfs.org/conference/2019/pdf/Multiband-LO- Project-by-Mike-K7MDL.pdf**.

Lewis, M., K7MDL, "Multi-Band VHF+ Local Oscillator, Decoder, and Remote Antenna Switch," *2019 Central States VHF Conference Proceedings*.

Overbeck, W., N6NB, "10-Band Toolbox Stations for Roving," **n6nb.com/toolbox.htm**.

Overbeck, W., N6NB, "Small 10-Band Rover Stations that a Septuagenarian Can Lift and Roll Aboard a Plane," **n6nb.com/small_10_band_rovers_with_up_to_seven_transverters.pdf**.

w1ghz.org, W1GHZ Microwave page.

Wade, P., W1GHZ, "432 MHz Transverter for an SDR," *2019 Central States VHF Conference Proceedings*.

Wade, P., W1GHZ, "2304 and 5760 GHz Transverters," *2016 Microwave Update Proceedings*, pp. 104 – 106 and 110 – 122.

www.downeastmicrowave.com, DEMI Knowledge Base.