The ABCs of Software Defined Radio

Why Your Next Radio Will Be SDR

Martin Ewing, AA6E

The ABCs of Software Defined Radio

Why Your Next Radio Will Be **SDR**

Martin Ewing, AA6E

Production
Michelle Bloom, WB1ENT
Jodi Morin, KA1JPA
David Pingree,N1NAS

Cover
Sue Fagan, KB1OKW

ISBN: 978-0-87259-632-0

First Edition, First Printing

Dedication

To Eva

Contents

Preface

Over the last decade, we have seen an additional technical revolution in radio communications. Old-timers among us remember the introduction of Single Sideband (SSB) operation, mobile FM, satellites, and now a whole slew of advanced digital modes (PSK31 and the rest). Meanwhile, the basics of how you build a radio had not changed very much since the 1930s and the introduction of the superheterodyne receiver. True, transistors and integrated circuits took the place of vacuum tubes, but the block diagram of a 1980s radio would be understandable to hams of 50 years before.

Now, we have something called *Software Defined Radio* (SDR). In a nutshell, the idea of SDR is to put as much as possible of a transmitter's and receiver's functions into software running on a high-speed digital computer. A receiver can be very simple: an antenna, an analog-to-digital converter chip, and a computer. A transmitter is a computer plus a digital-to-analog converter and a power amplifier. The circuits are straightforward; all the complicated operations are in software. Your grandfather would be puzzled!

My goal in this book is to tell the story of SDR for radio amateurs, but how to do it? Are we going to get into the details of finite impulse response filters, Fourier transforms, and C++? Those are subjects that can help you produce an SDR radio from scratch. But most of us don't even design and build conventional (non-SDR) radios these days! On top of all that, SDR technology is developing rapidly, and the marketplace for SDR in Amateur Radio will be changing quickly, too.

What I will do instead is describe SDR in non-mathematical terms, but in enough detail that the reader can read a radio's specifications and make intelligent choices in the marketplace. I expect you will have a general background in ham radio, electronics, and computers, but not much more. I will present a few subjects in more detail, with references for further reading. When specific SDR products are mentioned, it is to highlight their particular design approaches and not to recommend for or against any particular model.

Our plan for this book goes like this. Chapter 1 defines SDR and lays out the arguments for it, in case we have skeptical readers. Chapter 2 introduces some basic ideas of digital signal processing, while Chapter 3 discusses the

main types of software defined radios that are found in Amateur Radio. Chapter 4 discusses SDR from the computer and software side, while Chapter 5 is about using SDR in real amateur operations. In Chapter 6, we do some prognosticating about what's coming for SDR. It's going to be an SDR world, if it isn't already! For the adventurous, we have an Appendix going into some more detail on one very common type of SDR radio — the quadrature sampling detector. Then finally there is a detailed Glossary that should be a help when you run across terms that aren't familiar.

This book is only a survey of a field that has incredible depth and impact for radio amateurs and the communications industry. We mostly focus on HF transceiver applications for voice, CW, and digital modes. There are big applications in weak signal VHF (SSB, CW, and digital), video and other high-speed communications, and more, but they will wait for another time. If what this book covers prompts you to go further into SDR and digital signal processing (DSP) technologies, that's all to the good! Follow the references.

If any readers already have some background in SDR technology, they may accuse us of "hand waving" explanations that avoid doing it the better way, mathematically. As Professor Einstein is supposed to have said, "explain it as simply as possible, but not more so." That's the goal here.

Many thanks are due for stimulating discussions and support along the way, notably with Ed Hare and the staff of the ARRL Laboratory, Harold Kramer, and Steve Ford.

About the ARRL

The seed for Amateur Radio was planted in the 1890s, when Guglielmo Marconi began his experiments in wireless telegraphy. Soon he was joined by dozens, then hundreds, of others who were enthusiastic about sending and receiving messages through the air—some with a commercial interest, but others solely out of a love for this new communications medium. The United States government began licensing Amateur Radio operators in 1912.

By 1914, there were thousands of Amateur Radio operators—hams—in the United States. Hiram Percy Maxim, a leading Hartford, Connecticut inventor and industrialist, saw the need for an organization to band together this fledgling group of radio experimenters. In May 1914 he founded the American Radio Relay League (ARRL) to meet that need.

Today ARRL, with approximately 155,000 members, is the largest organization of radio amateurs in the United States. The ARRL is a not-for-profit organization that:

- promotes interest in Amateur Radio communications and experimentation
- represents US radio amateurs in legislative matters, and
- maintains fraternalism and a high standard of conduct among Amateur Radio operators.

At ARRL headquarters in the Hartford suburb of Newington, the staff helps serve the needs of members. ARRL is also International Secretariat for the International Amateur Radio Union, which is made up of similar societies in 150 countries around the world.

ARRL publishes the monthly journal *QST*, as well as newsletters and many publications covering all aspects of Amateur Radio. Its headquarters station, W1AW, transmits bulletins of interest to radio amateurs and Morse code practice sessions. The ARRL also coordinates an extensive field organization, which includes volunteers who provide technical information and other support services for radio amateurs as well as communications for public-service activities. In addition, ARRL represents US amateurs with the Federal Communications Commission and other government agencies in the US and abroad.

Membership in ARRL means much more than receiving *QST* each month. In addition to the services already described, ARRL offers membership services on a personal level, such as the Technical Information Service—where members can get answers by phone, email or the ARRL website, to all their technical and operating questions.

Full ARRL membership (available only to licensed radio amateurs) gives you a voice in how the affairs of the organization are governed. ARRL policy is set by a Board of Directors (one from each of 15 Divisions). Each year, one-third of the ARRL Board of Directors stands for election by the full members they represent. The day-to-day operation of ARRL HQ is managed by an Executive Vice President and his staff.

No matter what aspect of Amateur Radio attracts you, ARRL membership is relevant and important. There would be no Amateur Radio as we know it today were it not for the ARRL. We would be happy to welcome you as a member! (An Amateur Radio license is not required for Associate Membership.) For more information about ARRL and answers to any questions you may have about Amateur Radio, write or call:

ARRL—The national association for Amateur Radio
225 Main Street
Newington CT 06111-1494
Voice: 860-594-0200
Fax: 860-594-0259
E-mail: **hq@arrl.org**
Internet: **www.arrl.org/**

Prospective new amateurs call (toll-free):
800-32-NEW HAM (800-326-3942)
You can also contact us via e-mail at **newham@arrl.org**
or check out *ARRLWeb* at **www.arrl.org/**

Chapter One

It's a New World!

This is a book about Software Defined Radio (SDR) for the radio amateur. SDR can be a very technical subject, full of math and algorithms, but we are going to tell the SDR story in a relatively non-technical way, appropriate for many amateurs who need to understand enough about SDR to select and to use this new generation of radio equipment. SDR technology, though it may be hidden inside a traditional box behind a normal-looking front panel, is quickly being incorporated into essentially all advanced radio products.

What is SDR?

Software defined radio is a way of handling many of the functions of ordinary radio receivers and transmitters by converting electrical signals to and from streams of numbers (digits) and processing them in a computer. The computer can be either a highly specialized digital signal processor (DSP) chip or a general purpose personal computer (PC). In any case, the operations are controlled by software (sometimes called *firmware*).

The spectrum of software defined radio products covers a wide range of sizes, shapes and capabilities. Probably most hams think of SDR looking like the FlexRadio products (**Figure 1.1**): a relatively plain box of electronics connected to the antenna at one end and to a general-purpose PC at the other. You control the radio with mouse and keyboard while watching a computer screen.

At another point on the spectrum is a tiny system like the Software Radio Laboratory (SRL) QS1R. **Figure 1.2** shows the single QS1R circuit board outside its case. The receiver comprises a high-speed *Analog-to-Digital Converter* (ADC) and a special *Field Programmable Gate Array* (FPGA) processor and not much

Figure 1.1 — FlexRadio SDR-1000™, introduced in 2003.

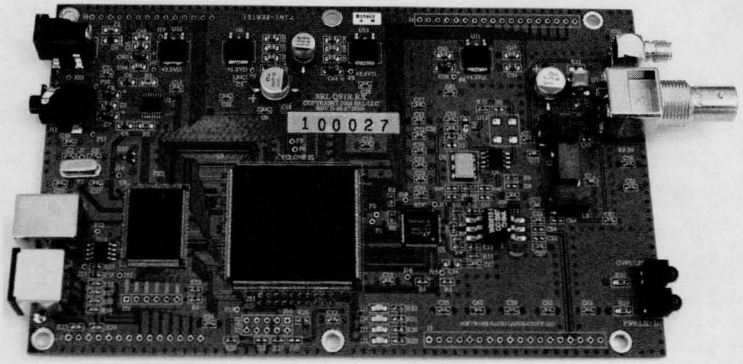


Figure 1.2 — Software Radio Laboratory, QS1R SDR receiver (circuit board).

Figure 1.3 — Kenwood TS-590S Transceiver.

more. The unit attaches to an antenna and to a PC with operating software much like the FlexRadio unit.

The majority of amateur radio products incorporating SDR technology look like the Kenwood TS-590S (**Figure 1.3**) — much the same as earlier amateur transceivers. Internally, analog signals are converted to digital, they are filtered and then detected (demodulated — converted to audio) just as they would be in a separate PC. The new SDR-enhanced transceivers may have a little less flexibility (and visual drama) compared with the Flex or SRL units, but they achieve very high levels of performance in a self-contained package.

The Road to Software Defined Radio

SDR is only the latest big step in radio technology. Amateur radio operators have been involved in many earlier key steps in the evolution of the hobby. Most of us have never operated with the earliest spark transmitters or "coherer" receivers. Vacuum tube technology is still familiar, even if it is slowly dying away. You can still find used tube-based ham rigs, from the earliest CW and AM rigs through to the 1950s and the first several generations of single sideband (SSB) radios. In the 1970s and 80s, transistors took over from tubes, but they still operated in analog mode for most amateur purposes. Personal computers were added at first for logging, control, and other administrative duties. In 1984, *QST* published its first article on Digital Signal Processing. At that time, DSP was mostly limited to audio filtering, because of the limited performance of available processors. But computer speeds advanced according to Moore's Law, which states (roughly) that computer speeds double every 18 months. Today, some commercial SDR receivers are capable of digitizing the entire HF radio spectrum, enabling the ultimate SDR dream — digitizing at or close to the antenna with most other filtering and detection handled by DSP software in the computer.

An increasing number of Amateur Radio products, such as the FlexRadio or SRL products, are built out of SDR components that rely on an external PC. This allows you to build an amateur station the way audiophiles build component stereo systems. You can get great flexibility and performance this way.

Even for radios that are not billed as "SDR," the industry is using SDR methods, because this technology is the most cost-effective way to build high-performance transmitters and receivers. These radios are usually marketed as having "IF DSP" (Intermediate Frequency Digital Signal Processing). The only clues that you're working with SDR might be the availability of continuously adjustable IF band pass, a noise reduction button, and possibly a digital connector for internal firmware updates. Modern "all-in-one" transceivers provide many of the operating features of the component-style SDRs, even if they don't allow you to tinker with their software.

What Will SDR Do For Me?

Whether your radio is all-in-one or component style, SDR technology offers real advantages:

- **Flexibility.** Bugs can be fixed and new features can be added simply by downloading new software. (Software for all-in-one rigs is generally called "firmware" because it is closely associated with the hardware, and updates are supplied by the manufacturer relatively infrequently.)
- **Advanced visualization and tuning.** A panadapter (spectrum) display is often available to let you view what is going on across an amateur band. Selecting a station (with PC-based SDR) is usually a matter of pointing and clicking a blip on the display. Some SDR software allows you to adjust the receive passband by clicking and dragging, just as you might adjust any PC display window.
- **Automatic noise reduction (ANR).** ANR is a special filter that adapts itself to separate your signal (voice or CW) from the ambient noise.
- **Receiver band pass** can be adjusted smoothly from a very narrow setting (100 or 200 Hz) for CW signals in a crowded band up to perhaps 8-15 kHz for FM, AM, or "high fidelity" SSB.
- **Automatic gain control (AGC).** Analog radios have had AGC for a long time, to keep the receiver output volume more or less constant as you tune between strong and weak signals. SDRs have software-based AGC to do the same thing, and also to prevent clipping (overload) of the analog-to-digital converter. Beyond that, the exact operation of the AGC in an SDR can be controlled to offer the best performance in the presence of noise.
- **Do-it-yourself software.** If you have the right skills and experience, you can modify or develop your own SDR software, especially if your SDR is supported by one of the various "open source" software programs. Even if you don't do your own programming, you will benefit by the work of other amateurs who do.

SDR or DSP: What's in a name?

This book is about software defined radio (SDR), but you will often see the term digital signal processing (DSP). In the marketplace, SDR often means a radio that relies on your personal computer for filtering, modulation, and demodulation. The term "DSP radio" is a catch-all used to describe any radio that relies on converting signals to digital form, even if you don't need to attach a separate computer. If you look at SDR and DSP radios in terms of their signal flow and the software methods that are used, they are often the same thing. Radios marketed as "DSP" are generally packaged as standalone systems that look like traditional systems. A recently announced amateur transceiver claims peak DSP power of 2,000 Mflops (million floating point operations per second), which would have made it a supercomputer not very long ago. Despite all that horsepower, the computer is embedded in the box and hidden from the user.

■ **Long-lived radios.** With new software downloads, your hardware investment can be extended through multiple generations as its capability and "personality" is extended.

■ **Community support.** SDR radios, especially the ones that are component-style, often have lively Internet support groups where users, outside developers, and the hardware manufacturers can exchange ideas and get problems solved.

■ **More bang for the buck.** Since so many of the complicated features of SDR systems are in the software, the hardware can be surprisingly simple and inexpensive. For less than $20 it's possible to buy a basic SDR kit that works with your PC and its sound card to deliver remarkably good performance. The FlexRadio Flex-1500™ provides a full transceive capability (5W output) for under $700.

Chapter Two

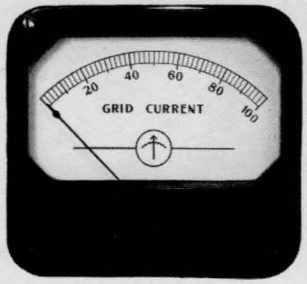
The Meaning of "Digital"

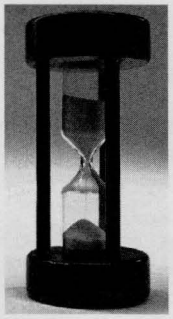
Today, everyone knows a lot about digital consumer products — televisions, cell phones, computers, and many more. If you ask how new digital gear is different from old analog products, you will probably hear about sharper pictures, text and data services, low distortion, and high reliability. An enthusiast might tell you about Moore's Law, which we mentioned in the last chapter: digital things get more powerful or cheaper at a relentless pace. (More precisely, the number of transistors on a given size chip has doubled every 18 months or so for a long time.)

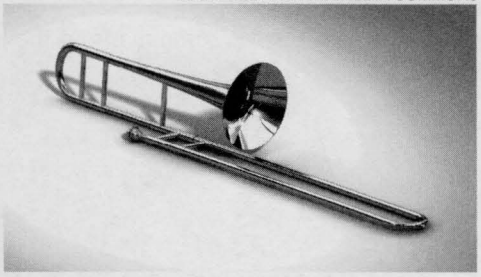
When we talk about software defined radio and the way it is built on digital signal processing, we need to know what "digital" means and how digital systems are different from analog electronics. This is not so much about consumer product features as it is about the nuts and bolts of how the new devices are really different from the technology that Amateur Radio has used for a hundred years.

First, There Was Analog

Let's start with *analog*. The word comes from *analogy* — the way we talk about one idea being like another. An analog quantity, like electrical voltage or current, stands in for some property of the physical world, like the speed of a car, the temperature of the air or, especially for hams, the RF currents in an antenna or the acoustic pressure at a microphone. Analog quantities are usually smoothly varying continuous functions of time. Analog systems don't have to be electronic. A few examples are in **Figure 2.1**.







JOSEF PLCH VIA WIKIMEDIA COMMONS

Figure 2.1 — Some familiar analog devices.

Analog signals can even be processed with analog computers. Some readers may remember the Heath H-1 computer (**Figure 2.2**) that was sold in the 1950s. Before we had practical digital computers, analog computers were used to solve important problems relating to missile guidance, automotive suspensions, chemical plants, and other applications. Analog computers were built from dc-coupled operational amplifiers with various kinds of feedback networks.

Analog electronics has a long history, paralleling the growth of Amateur Radio from the early 1900s through the present day. What were the problems that led to the change to digital methods in so many current systems?

Drift is a big problem. The *gain* of an amplifier (output divided by input) is not constant, but it depends on power supply voltage, operating temperature, component aging, and other factors. Vacuum tubes were a particular problem. Have you ever waited for a radio to warm up and stabilize? (Younger readers may never have had this pleasure!) In real life, the drifting values of components will limit how complicated a circuit you can build that will work reliably. There are ways to reduce drift, like negative feedback and temperature compensation, but they only take you so far.

TECHNIKUM29 MUSEUM

Figure 2.2 — Heath H-1 Analog Computer (modified), about 1956.

Analog designs also struggle with *electrical noise*. Every electronic component has thermal noise due to its temperature.

Active electronic devices like tubes or transistors have additional sources of noise that are produced by the discrete charges (electrons or holes) that they use. All these noise sources are added to the desired signal at every step in an analog system.

Signal fidelity is a another problem with analog systems. Old-timers will remember what it was like to make an early transcontinental telephone call. Voices were faint and tinny and sometimes could not be made out at all. That was the result of many stages of amplifiers and long cables, each of which would add noise and distortion and filter out parts of your voice. You can see the same effect if you make a photocopy of an original page, make a copy of that copy, and so on. It does not take many generations to muddle the text beyond recognition.

The Answer Is Digital

Digital circuits and *digital systems* overcome many of the problems of analog technology by representing data (or signals) as discrete states. (The word *digital* comes from the Latin word for finger. It suggests counting on your fingers — your digits.) The simplest type of digital (logic) device is binary. A binary information "bit" has the value 1 or 0 — in logic terms, "true" or "false." Familiar binary devices include the mechanical on/off switch and relays. While you can do interesting things with relay-like devices, as we used to see in elevator controls or telephone switching centers, digital logic gets really powerful when we use miniature electronic amplifiers — micro transistors — and lots of them — operating in an on-off switching mode.[1]

Digital systems for SDRs, like other kinds of computers, look at data as groups of bits or "words" that are treated as numbers. A single 16-bit word, for example, can be treated as an "unsigned integer" between 0 and 65,535 or as a "signed integer" between –32,768 and +32,767. Larger word sizes of 32 or 64 bits are common, often in a floating point format. (A smaller unit, the byte, is normally 8 bits, but for radio work, data words are usually 16 bits wide or more.)

What is the advantage of turning analog signals into numbers? Because digital processors are built from on-off switching transistors, which are mostly immune to drift, noise, and other analog-style problems, their operation is extremely predictable. They don't make mistakes, unless they are programmed that way! Once your signal enters the digital world, you can transform it and communicate it without any of the degradation of analog systems.

Another great advantage of working with radio signals in the digital arena is that you can exploit the enormous technical progress that has been achieved with general-purpose computer chips. Off-the-shelf consumer PC chips are powerful and inexpensive. Using PC technology is so cost-effective that it's the tool of choice for a wide range of signal-handling jobs. That's going to be a theme of this book.

With all its advantages, digital processing of signals still has some limitations. If you have worked with digital TV, digital voice in Amateur Radio, cell phones,

A Peek at Digital Signal Processing Tools

While this is not a text on DSP mathematics, there are some basic DSP tools that are often mentioned when people talk about DSP and SDR. Here are a couple of thumbnail sketches of important tools we use.

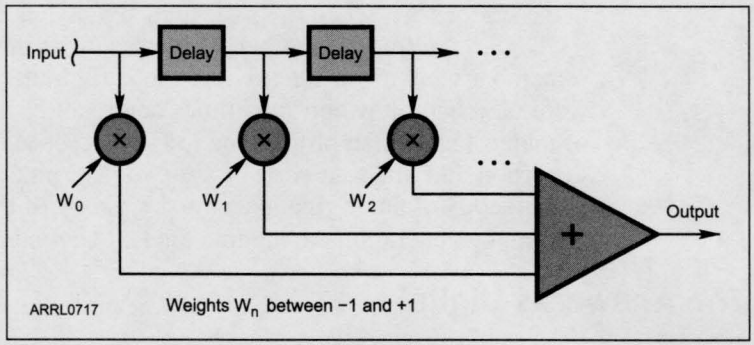


Figure A — FIR filter as tapped delay line.

Finite Impulse Response (FIR) Filter

A *finite impulse response* filter is a simple circuit that is easy to describe and performs well in many SDR applications. In analog terms, you make an FIR by using a tapped delay line, adding the signal that comes from each of the taps with desired "weighting" (gain, positive or negative) for each tap. **Sidebar 2-1 — Figure A** shows the circuit. It is called "finite" because the output only depends on the current input and a certain number of delayed copies of the input.[1] In the digital version, the delay is provided by a digital memory element (like a flip-flop) so that the delay between stages is just one sample time for the input signal. The FIR has some desirable properties, such as a linear phase characteristic (constant delay). In the digital form it is relatively easy to calculate.

Fast Fourier Transform (FFT)

The *Fast Fourier Transform* is a quick way of computing a *Digital Fourier Transform* (DFT), which is a mathematical process that changes a block of N data samples (such as IF voltage samples) into an N-point frequency spectrum (**Sidebar 2-1 — Figure B(a)**.) Each spectrum value is an amplitude (and phase) that tells about the signal in one frequency channel. In analog terms, it is as if the input voltage signal were passed through N adjacent narrow band-pass filters (**Sidebar 2-1 — Figure B(b).**) A simple

etc, you have probably observed the *cliff effect*. The digital signal is transmitted perfectly as long as the signal strength is over a critical value, but once the signal dips below that strength, transmission suddenly stops. By contrast, weaker signals in analog radio modes, such as CW or SSB, grow noisier and noisier, but the fade-out is gradual. You can often copy signals through marginal conditions.

A practical problem with complex digital modes and SDRs is summed up in the phrase "no user-serviceable parts inside." There is a big learning curve to be climbed before an average radio amateur is able to modify or extend the software

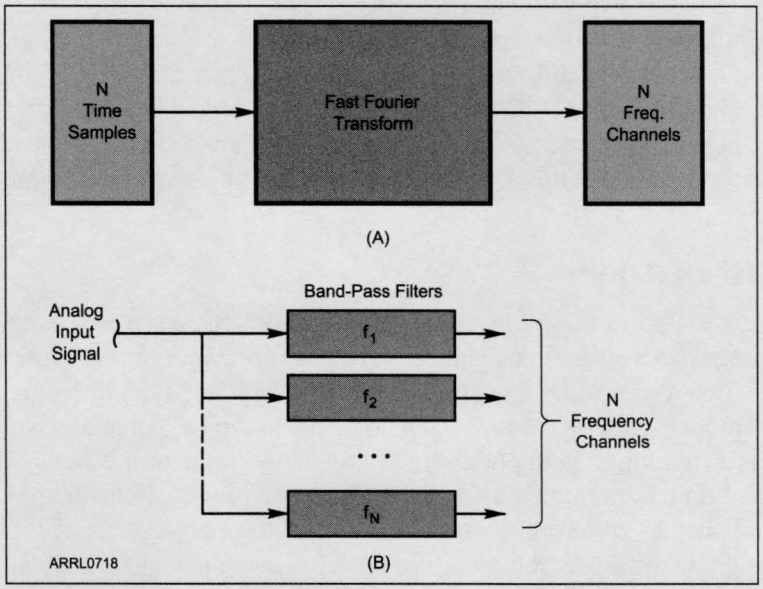


Figure B — Fast Fourier Transform. (A) Transforming a block of time samples to a block of frequency channels. (B) Analog band-pass filter equivalent.

ham application might be to produce a panadapter display of a receiver's IF band pass. FFTs are used for many DSP functions, including filtering, frequency translation, and correlation.

[1]An "Infinite impulse response filter" (IIR) has an output that also depends on previous values of the output. That's what you have if your analog circuit has an LC resonator, for example. If you have an LC tank circuit, the input energy sloshes between the L and the C forever. (Why do you think it is called a tank?) The energy fades away because of losses, but there's always some residual. Digital IIR filters can be very effective, but they can have stability and round-off problems.

that controls digital processors. Besides having to know the programming language and operating system, you need to deal with the sophisticated mathematics used to filter, modulate, and demodulate signals.

Even after you have mastered the math and programming, there is still more trouble ahead if you hope to create a brand-new design. You may need access to advanced test equipment, Printed Circuit Board/Computer-Aided Design (PCB/CAD) layout software, and surface-mount assembly gear. This is not a world for old-school radio tinkerers!

So the world is less kind to home developers than it was, but you do have options. You can bypass many problems by using building blocks such as PCs and PC sound cards, and free, open source software.

One last problem afflicts most recent products incorporating digital processors. Radios that use advanced integrated circuits can be very difficult to service when something fails after several years. Parts are only available for a limited time. Contrast this with the relatively simple job you have in repairing older analog gear.[2]

To Digital and Back Again

A critical part of a successful SDR — or any DSP system — is right at the beginning where an analog signal must be converted into digital form. The analog-to-digital converter (ADC) is usually a single integrated circuit that has some very special properties. The input is an analog signal voltage, but analog signals are always fluctuating. It is important to "sample" the value of that signal at a precise instant of time (determined by a clock signal you provide to the chip). This is accomplished by momentarily closing a switch that causes a capacitor to be charged to the signal voltage, as shown in **Figure 2.3**. Once the data is sampled, ie, after a delay D_1, a process is carried out to effectively measure the voltage of the sample. (There are many ways to do this, depending on the required speed and accuracy.) After the measurement finishes (after delay D_2), the resulting 16-bit data word is placed in a register memory for output.

The data sheets for practical ADC chips are full of complicated specifications, but the most critical items are the *maximum clock rate* (conversion speed) and the *dynamic range*. Dynamic range is closely related to the output word length. An ADC with a 16-bit output word could theoretically have a dynamic range of about 32,000 to 1 in voltage, or about 90 dB in power.[3] A 24-bit ADC could have up to 144 dB dynamic range. Other specifications would tell you how accurately the output tracks the input voltage, how long the sampling switch is closed (the "aperture time"), etc.

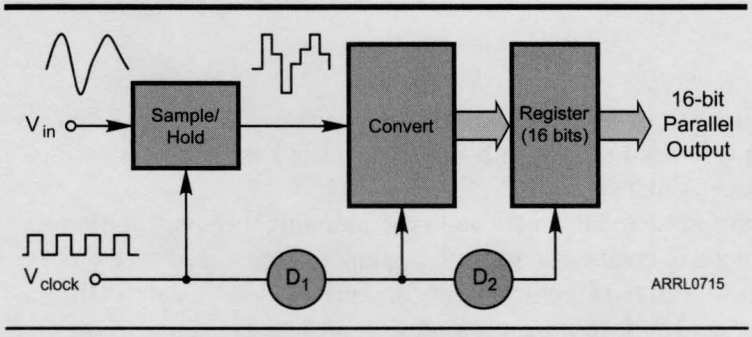


Figure 2.3 — Basic Analog-to-Digital Converter.

The converter's sampling rate is important because it limits the bandwidth that an SDR can handle. Digital signal processing theory shows that if you want to capture all the signal information in a

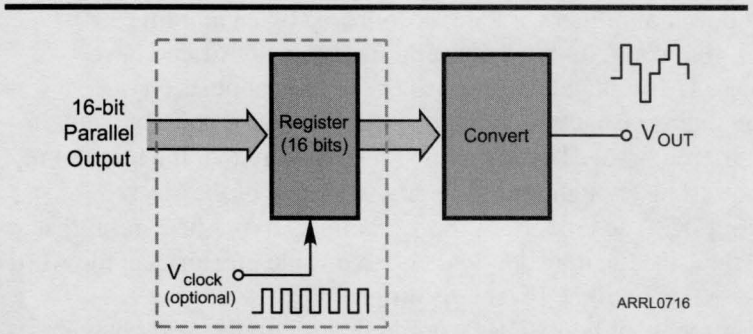


Figure 2.4 — Basic Digital-to-Analog Converter.

certain *band pass B*, (in Hertz), your sample rate needs to be at least 2 B. That's called the *Nyquist sampling rate*.[4] So, for example, the 125 MHz ADC in the SRL QS1R receiver allows all frequencies to be captured below about 62.5 MHz — in other words, all the amateur bands up to 6 meters.

ADC dynamic range is especially important for wide-band receivers like the QS1R. The receiver's input voltage is the sum of all the strong and weak signals over a 62.5 MHz band. You commonly have both very strong and very weak signals in such a broad band. You might want to copy a weak signal that only registers in the very lowest bits of the ADC output, but a strong signal might use all the bits. In fact, the strong signal can cause the ADC to "clip" — to overload — so that the weak signal is obliterated.

Going back to analog from digital data calls for a digital-to-analog converter (DAC). These are generally simpler than ADCs. They are also usually provided as a single integrated circuit. As you see in **Figure 2.4**, a DAC simply accepts a digital data word (16 bits in this case) and converts it to an output voltage as rapidly as possible. For convenience, a clocked input register may accept the parallel input before conversion. Other types of DACs accept a serial input stream, which is internally converted to parallel before conversion. The converter can produce noisy outputs during the conversion, and when the conversion completes there can be sharp steps in the output voltage. The designer will usually follow the ADC with a low-pass filter to remove unwanted high frequencies from the output.

Processing Digital Signals

Once the input signal is converted to a stream of digital samples, it is ready to be transformed in useful ways. For a receiver, we usually want to change radio frequency (RF) or intermediate frequency (IF) to a demodulated audio frequency (AF) signal that we can listen to. Sometimes, what we really want is a decoded text or image stream from an incoming RTTY or SSTV transmission. All that is possible, but to date many SDRs provide only an audio output channel. You can attach further decoders to handle special data or image modes.

Likewise, for a transmitter, the job is to convert an input audio signal (or CW key closures) to a digital sample stream that a DAC can convert to an RF (or IF)

voltage that goes to a power amplifier (or a mixer followed by an amplifier).

All the operations you would find in a conventional, non-SDR radio have their equivalent in SDR. These include mixing, filtering, and modulation or demodulation. Analog radios also use amplifiers to provide signal gain, but gain in a digital system is a different issue. The main thing is to be sure that the minimum signal to be detected is strong enough to register at least a few of the "least significant bits" of the ADC. Then the user (or programmer) has to be careful that the signals don't overflow, that is, they don't add up to a value greater than the word length allows; otherwise there will be severe distortion.

We will get a clearer view of how SDRs work in real life in the next chapter, where we look at specific Amateur Radio products that are using SDR techniques.

Notes

[1]This text was composed on a home computer with a CPU that has around a billion transistors — so they say; I haven't counted them.

[2]I keep a totally analog 1977-vintage transceiver on hand just in case my new radios can't be fixed.

[3]Yes, I promised you there would be no math, but you may recall that the logarithmic power ratio, in decibels (dB) is 10 $\log_{10}$ (power ratio) or 20 $\log_{10}$ (voltage ratio).

[4]The Nyquist rate applies to a single analog-to-digital converter. In the I/Q sampling system (Chapters 3 and 7), two ADCs working at one-half the Nyquist rate can give equivalent results. For example, two input channels of a PC sound card set for a 48 kHz sampling rate can process up to 48 kHz of bandwidth.

Chapter Three

Real-World Software Defined Radio

The term Software Defined Radio (SDR) covers a wide range of possible ways to build radio equipment. While they all use digital signal processing techniques, physically and functionally they can appear rather different to the user.

Current SDR products can be put into three main categories. First there are *sound card* systems in which a receiver has a simple hardware detector producing two *quadrature* audio channels that can be fed to a computer's sound card. As a transmitter, the same sound card can be used to generate channels that are translated to RF in reverse.

A second class of SDR product is the *Audio IF system*. A majority of new radios are designed using a final *intermediate frequency* (usually a third IF) in the high audio range, say 10–30 kHz. Low-cost audio analog-to-digital-converter (ADC) and digital-signal-processing (DSP) chips allow the designer to provide many advanced features, even without an external PC for processing or display. To the casual user, these radios have the look and feel of traditional radios — in terms of front panel controls and indicators.

Finally, there are the *wideband SDR designs*. These come close to fulfilling the SDR vision of minimizing a radio's hardware and maximizing digital processing. These products tend to be small packages containing a high-speed ADC chip and some signal processing logic and digital control. After preliminary processing, the data is transferred to your computer for further computing, display, and control.

Radio by Sound Card

A very effective SDR receiver can be built around a simple detector called the *quadrature sampling detector* (QSD), sometimes known as the Tayloe Detector.[1] This is an analog circuit that mixes an incoming RF signal with a local oscillator (LO) centered in the amateur band (or part of a band) that you want to receive. You get two *baseband* audio signals as output — the so-called *In-Phase (*I*)* and *Quadrature (*Q*)* channels. The I and Q channels are fed to the inputs of a computer's stereo sound card. In the computer, I and Q are digitized. (That's what a sound card does!) Then I and Q data are combined in software so that the full spectrum around the local oscillator (plus or minus the maximum sound card frequency) is displayed, spectrum-analyzer style.

You can make a bare-bones SDR receiver or transceiver using the sound card approach by way of one of the SoftRock kits pioneered by Tony Parks, KB9YIG.[2] **Figure 3.1** shows the author's SoftRock version 6.2, 40-meter receiver board. This, plus a computer, lets you listen in on 40 meters CW!

The complete theory of the operation of the QSD-style radio is beyond the scope of this book, but in the Appendix you will find a more detailed description of how you can view a QSD SDR as an evolution from a very simple conventional direct conversion receiver.

The simple SDR radio of Figure 3.1 is hardly perfect. Sensitivity is limited, and without an input band-pass filter, it is rather easy to overload the radio. Its single crystal-controlled local oscillator limits the receiving band to one segment of one band. But for a parts price of about $15, we should not complain! Later designs in the SoftRock series sport transmit capability and tunable local oscillators.

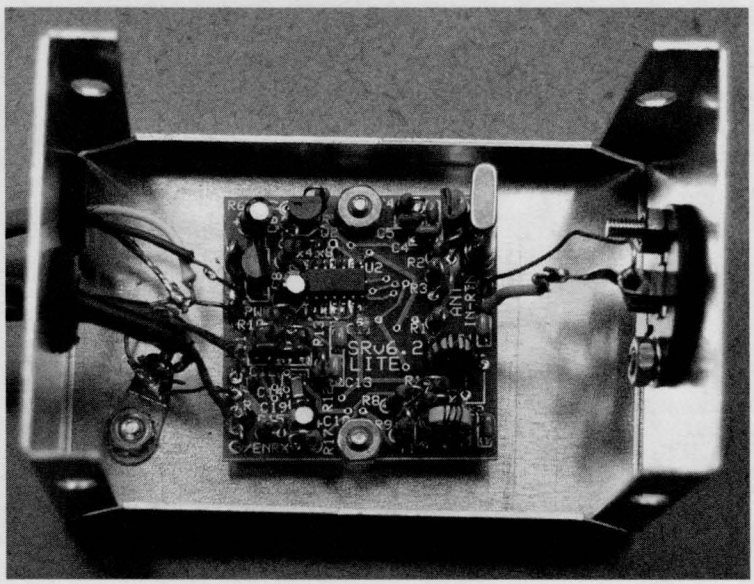


Figure 3.1 — SoftRock version 6.2 40-meter receiver.

Not all radios designed in the sound card style are so humble. In particular, the popular commercial line of SDR transceivers from FlexRadio Systems® are all based on the QSD direct conversion technique. **Figure 3.2** shows the Flex-3000™, a mid-range SDR transceiver. As with

Figure 3.2 — Flex-3000™ Transceiver.

most SDR products, the Flex-3000 hardware is a plain metal box with a switch and a few jacks. At 12.25 inches square, the box will stack nicely with other equipment. As a user, you are on your own to provide a *Windows* PC with its own mouse, keyboard, and monitor, but FlexRadio provides its *PowerSDR*™ software for control and display. (*PowerSDR* will be described further in Chapter 4.)

The Flex-3000 is a full-function 100 W amateur transceiver covering all bands from 160 through 6 meters. SSB, CW, AM, FM, and digital modes are supported. The block diagram (**Figure 3.3**) shows all the hardware blocks required to provide

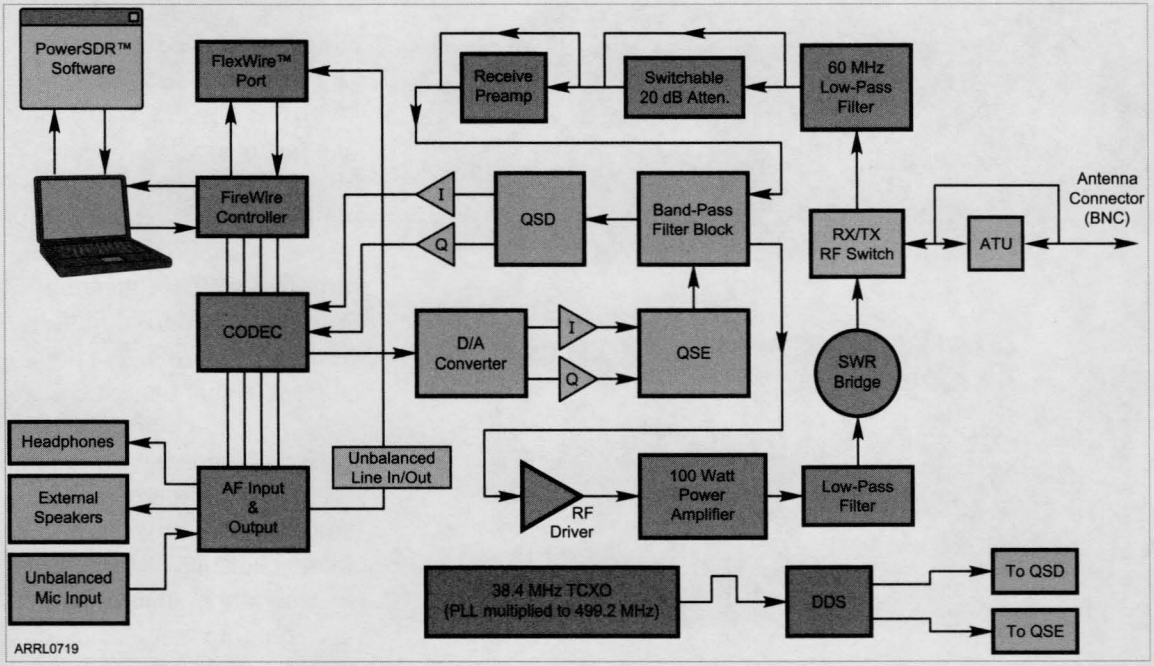


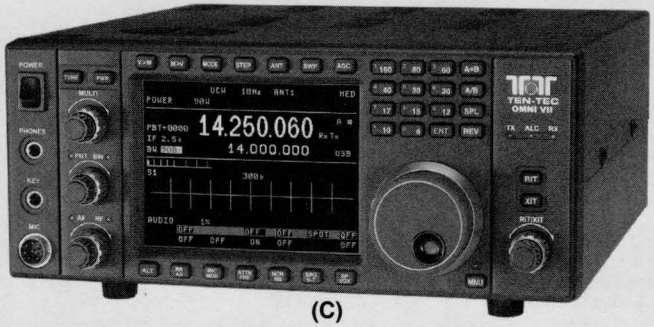
Figure 3.3 — Flex-3000 Block Diagram (adapted from FlexRadio).

these features. At the center, you notice the QSD detector, a big brother to the SoftRock unit described above. For transmitting, the Flex-3000 uses a *quadrature sampling encoder* (QSE). A QSE is basically a QSD run "backwards." It takes I and Q audio channels as input, along with a local oscillator signal, and generates the desired modulated RF as output.

This radio does not actually use a computer's sound card. Instead, it has internal *codec (*coder-decoder*)* and digital-to-analog-converter blocks that are specially adapted for use in the SDR application. The codec processes 24-bit samples at a rate of 96 kHz, providing simultaneous display of up to 96 kHz of an amateur band. Control and data signals are communicated to your PC over a Firewire (IEEE 1394) interface. Some PCs have a Firewire jack built in, but many require an additional Firewire card.

(A)

(B)



(C)

Figure 3.4 — Modern amateur transceivers designed on SDR principles: (A) Icom IC-7600; (B) Yaesu F-950; (C) Ten-Tec Omni-VII.

SDR in the Box: Audio IF Systems — *Embedded SDR*

It is a well-kept secret, but most new HF amateur radio transceivers (**Figure 3.4**) are software defined radios. They are likely to be advertised as "DSP" or "IF DSP" rigs, but the fact is that DSP means these products use the same digital tools we talked about above. What do they do with those tools? They generally implement a kind of SDR. What we mean, in this book at least, is that the critical jobs of shaping the band pass (filtering) and signal detection (demodulation) are done in software (or firmware, if you like). The embedded computer is generally hidden and not user-programmable, and the software/firmware may or may not be easily updated, but still

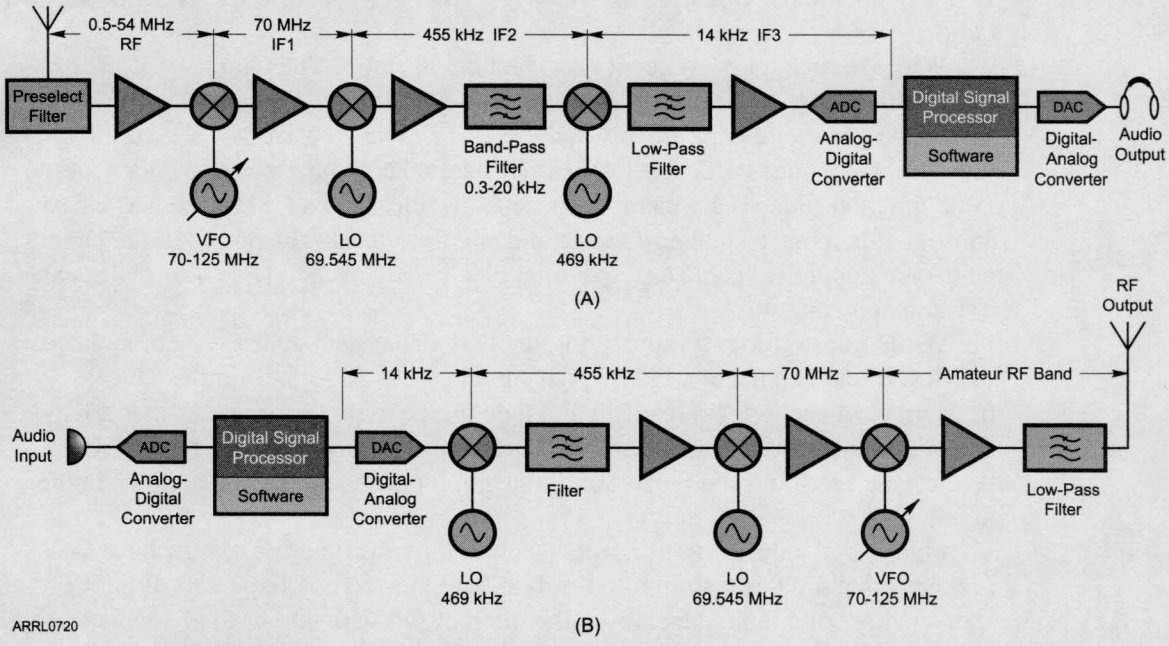


Figure 3.5 — Ten-Tec Omni-VII Transceiver Simplified Block Diagram: (a) Receive Mode; (b) Transmit Mode.

the key functions happen in a software-programmed digital processor.

The simple reason for SDR in these "standard" commercial products is that this is the least expensive way to provide the modern features that users want: flexible band-pass tuning, noise reduction, high-quality audio modulation and detection, etc. For the most part, the manufacturers do not focus on the other potential benefits of SDR methods, such as user programmability, support for new modulation methods, high-level options for computer and remote control interfaces, etc. But let's be happy with the features we do have! SDR methods are offering us remarkably good performance at a modest price point.

How is a mainstream SDR-style rig designed? Let's look at the Ten-Tec Omni VII transceiver, a current mid-range rig. The Omni VII block diagram,[3] **Figure 3.5**, seems conventional on the surface. While receiving, (Figure 3.5a), the signal passes from the antenna, through a pre-selector (analog RF filter), a preamplifier, and then a mixer which up-converts to a 70 MHz IF frequency. The first mixer's local oscillator is a variable frequency oscillator (VFO) that you use to set the rig's frequency. A second mixer converts to the second IF (455 kHz), where one of a set of "roofing filters" is selected. Finally, another mixer converts to the third IF, 14 kHz. The third

IF is the input to the DSP system, which is really a fix-tuned 14 kHz software defined radio.

Why use such a low frequency for the final IF stage? The Omni VII IF bandwidth is at most 20 kHz, which can be accommodated between 4 and 24 kHz, within the range of a good-quality audio ADC, similar to what would be available in a computer sound card. The Omni VII's internal computer performs many of the same functions that your general-purpose PC would do in a QSD system as we saw above, but in a much smaller package and at a lower cost. The output of the Omni's built-in computer is visible as a spectrum display and is audible as a normal speaker or headphone output.

While transmitting, (Figure 3.5b), the DSP subsystem generates a modulated (SSB, CW, etc.) signal at 14 kHz. That signal passes through a similar set of frequency conversions (in reverse) and then goes to a power amplifier that drives the antenna. To save cost, some of the hardware stages (oscillators, the 70 MHz IF, etc.) are used for both receive and transmit modes, but this is not shown in the figure.

Many good amateur transceivers (such as those in Figure 3.4) now have DSP backends that are fairly similar to the Ten-Tec Omni VII. A large part of the "personality" of a radio depends on the internal software, the level of sophistication of its signal processing, and the capability and ease of use of the user interface. From the outside, we (as users) work with controls and displays that are mostly the same as we had in previous non-SDR systems. Some of us like it that way!

Wideband SDR: RF Sampling

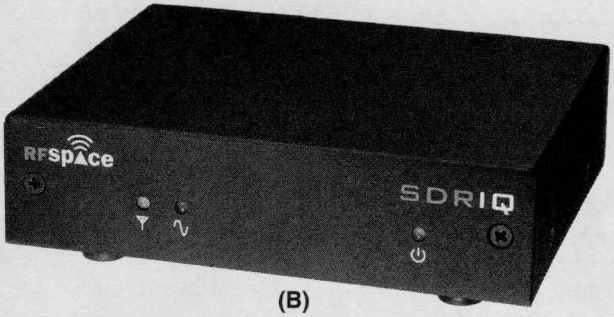
The newest family of SDR products uses high-performance ADC and DAC chips to move as much of the radio design into the computer and digital domain as possible. Three such products are shown in **Figure 3.6**. The hardware is very small — it can be a single board holding just a few advanced integrated circuits.

Let's take a look at the QS1R, Figure 3.6a, from Software Radio Laboratory.[4] The circuit board (Figure 1.2) and the block diagram (**Figure 3.7**) are deceptively simple.

This design is close to the SDR ideal — a little hardware, but a lot of functions in software. The ADC chip produces 16-bit samples every 8 nanoseconds — a rate of 125 million per second. It is fast enough to capture the entire RF spectrum from 0 to 62.5 MHz, according to the Nyquist theorem. A low-pass filter at the input helps to suppress the false "alias" responses that would occur for any inputs above 62.5 MHz.

Because most PCs would have trouble accepting and working on a sustained data flow at such a high rate (250 MB/s), it is necessary to have the first stages of processing on the QS1R board. That's the role of the *field programmable gate*

(A)



(B)

(C)

Figure 3.6 — Fast Sampling SDR hardware. (A) Software Radio Laboratory QS1R; (B) RFspace SDR-IQ; (C) Microtelecom Perseus.

array (FPGA). An FPGA chip contains a set of digital logic elements that can be wired up under software control. With the right program, involving digital mixers and filters, the FPGA cuts down the data rate dramatically. The QS1R's FPGA, with its accompanying software, is able to display up to a 20 MHz band anywhere in the 0 to 62.5 MHz region. An alternate program, *SkimmerServer*,[5] further demonstrates the power of the FPGA. This program is able to decode up to seven separate 192 kHz bands in the HF spectrum, scanning each for CW QSOs. Displays are available showing all CW call signs copied (perhaps thousands) over the seven bands.

Can the wide-band RF sampling approach be turned around for transmitting? In principle, yes; a fast DAC chip, driven by suitable software, can generate an arbitrary signal up to the Nyquist frequency. For example, to go with the QS1R receiver, you might have a DAC that is clocked at 125 MHz that could output signals anywhere from 0 to 62.5 MHz. If your computer is powerful enough (very powerful!), you could generate 20,000 SSB voice channels or over 100,000 CW channels simultaneously. Radio amateurs rarely need more than one transmit channel at a time, of course, so this extreme picture is not a practical one. Furthermore, there is the question of emissions purity. Any DAC will have imperfections like limited resolution (16 bits, for example) and non-linearity (each output step is not exactly proportional to the numerical input). These will cause harmonics in the output. To meet FCC regulations, you will need low-pass filters to ensure that harmonics are adequately suppressed.

For nearly all amateur transmitting applications[6], designs like the *audio IF*

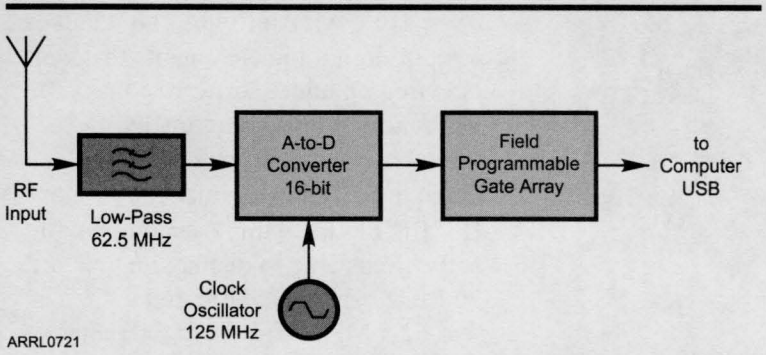


Figure 3.7 — SRL QS1R Receiver Block Diagram.

rigs described above, with signal generation in the low kHz range, are entirely adequate. There is little to be gained by a wideband DAC that directly generates the output frequency in normal HF operations.[7] In any case, the DAC will need to be followed by power amplifier stages and harmonic filtering.

Notes

[1]Gerald Youngblood, AC5OG (now K5SDR), "A Software Defined Radio for the Masses, Part I," *QEX*, July/August, 2001, p 1. Youngblood gives a more detailed analysis of the QSD/QSE methods, which became the basis of FlexRadio products.

[2]SoftRock kits have had limited availability. One source of information is the Yahoo Softrock40 group, at **groups.yahoo.com/group/softrock40**.

[3]If you look at older manufacturers' manuals for HF amateur gear, you find an interesting pattern. About the same time that DSP/SDR methods were introduced, many manuals stopped including detailed block diagrams or schematics. Fortunately, Ten-Tec does provide them. (But the company does not provide computer code listings!)

[4]The QS1R was reviewed by the author in *QST*, September, 2010, pp 41-44. Product details are available at **www.srl-llc.com**.

[5]*SkimmerServer* was developed by Alex Shovkoplyas,VE3NEA, at Afreet Software, Inc. See **www.dxatlas.com**.

[6]We are focusing on CW or voice-like transmissions. Some amateurs are working with broadband data and video at UHF and higher frequencies where broadband DACs may be needed.

[7]Wideband DACs may have interesting applications for wideband data or video modes at UHF or higher frequencies where that operation is permitted.

Chapter Four

Computers and Software for SDR

It's no surprise that the basic feature of SDR is powerful software. The goal of SDR design, after all, is to place as many functions as possible into the software. Why? Because software is cheap, ie, there is practically no cost to a manufacturer or to a user to load a copy of software into an SDR or its accompanying PC. There can be a big one-time cost to develop and test software, of course, but that can be moderated by using freely available open source software components. The other characteristic of software is that it can be modified and updated after the hardware is built or delivered to the user — again, at little cost. Software is more malleable than hardware. It can be adapted for new applications more readily.

Hardware, Software, and Virtual Hardware

We talk about hardware and software (or firmware, or code, or even microcode), as if everybody should know the difference between them. But the difference is not as clear cut as you might think. Yes, hardware is nuts and bolts, transistors, resistors, and all that, while software is a set of instructions. That seems clear enough. But what do we *do* with software? Often we are producing virtual hardware. Think about a modern ham transceiver. While you're scanning the bands you're adjusting the tuning knob and fiddling with the RF gain and volume controls. If you're of a certain age, like your author, you would think you're tuning with a variable capacitor and setting levels with potentiometers (variable resistors). You'd think you're directly changing an electrical component that is part of the receiver circuit. You'd think that S-meter needle is being pushed around by a current in a magnetic coil.

In fact, what you're adjusting with those controls is generally an optical encoder that measures a shaft's angular motion. Along with most of the other buttons and switches on your rig, these encoders are digital inputs to the rig's control software. The S-meter may be a computer graphic. You're not dealing with a traditional radio, it's a virtual radio created by the software.

You could say that modern radio software creates the illusion of a traditional radio. The line between hardware and software is blurred, but the casual user won't care.

Software Categories

When we look at software for SDR and SDR-like applications, we might identify three general categories: *comprehensive*, *embedded*, and *digimode*.

Comprehensive Software

When you ask users of SDR transceivers or receivers about SDR software, you will probably hear about larger comprehensive software products like *PowerSDR* or *WinRAD*, which run in the PCs attached to the SDR front end. The software for an SDR receiver (like the QS1R with *SDRM*ax) naturally concentrates on the receive function. Software for transceivers like *PowerSDR* can be rather more complicated, because of transmit-related functions.

Computer Audio Specifications

I've got a great new personal computer. How do I know what its audio capabilities are? This isn't an easy question to answer — partly because computer makers don't spend a lot of time explaining the fine points of audio to their buyers and partly because as SDR users we don't have a lot in common with folks who just want to listen to an mp3 tune.

Current PCs are likely to incorporate audio *codecs* (built-in sound cards) on the computer's *motherboard* — the main circuit board. The latest codecs have fairly standard interfaces to the computer logic — for example, the Intel High Definition Audio (HD Audio) specification. Somewhat older machines will likely use the Intel Audio Codec 1997 (AC97) standard.

Having a standard tells you the allowed sampling rates (such as 48,000 or 96,000 Hz), sample size (16, 24, or 32 bits), and so on, but your particular HD Audio device doesn't have to support all of those possibilities. (As far as I know, no vendor offers 32-bit sampling.) I looked at four audio devices to get a feel for what is available at different price points.

The main limitations of a sound card ADC for SDR purposes include: maximum sampling rate, frequency response, and dynamic range. The sampling rate and frequency response will limit the

Table 1
Modes and Measured Input Bandwidth of Some Audio Interfaces

Model	*Sample Size*	*Sample Rate*	*Bandwidth*	*Comment*
Edirol FA-66	24 bits	192 kHz	69 kHz	"pro" audio interface
		96	48	
		48	24	
Realtek ACC889	16/24	96	46	integrated on Gigabyte
		48	23	motherboard (HD Audio)
Conexant CX20585	16/24	96	19	on Toshiba L645D laptop
		48	19	
IKey Audio Iconnex	16	48	24	inexpensive USB device

Embedded Software

SDR or DSP software is *embedded* in the hardware of newer amateur transceivers. Their software or firmware is hidden inside a conventional radio enclosure so well that casual users may not even know it is there. It can usually be updated through a computer download, and in some cases (like the Ten-Tec Omni VII) the software can communicate with computer networks directly over an Ethernet port.

Digimode Software

If you look at it the right way, *Digimode* software for PSK31, RTTY, and many other modes (even Morse CW!) is a kind of SDR all by itself. The input is a 3 kHz

---

bandwidth of the SDR receiver — the IF passband. The analog inputs are low-pass filtered to prevent any signals above the Nyquist frequency from "aliasing" and causing spurious responses. (See Chapter 2.)

Alas, for fast sampling rates the real analog input bandwidth is often rather less than the Nyquist rate. To understand why, we should first ask why would anybody want "audio" sampling faster than about 44.1 kHz, the rate used on CD recordings? We know that we can't hear sounds above about 20 kHz at best, and not even that well as we get older. There is simply no meaningful audio signal above 20 kHz.

Sampling faster than the Nyquist rate is called *oversampling*. It is useful in the audio business in order to simplify the anti-aliasing filter design and also to permit an editor to process and combine audio tracks without loss of quality. If the analog bandwidth is less than the Nyquist rate, it is no great loss to an audio recording, as long as you include the 20 kHz where there is a useful signal. For SDR work, more high-frequency response translates to a wider IF passband. Audio gear for home computers is available and relatively inexpensive, but it's not going to be perfect for amateurs who want a high-performance SDR.

The table (at left) gives the results of my bandwidth measurements on four sound cards, or codecs. The bandwidth is measured at the -6 dB point.

The high-end Edirol converter (now out of production) performs very well, but its maximum 192 kHz sampling only records about 70 kHz of audio bandwidth. This is probably as much as we will get from high-end audio recording interfaces, which can be found in the market for $200, give or take.

The Realtek chip, integrated on a computer motherboard, is probably typical of newer computer designs using the HD Audio standard. It does provide 24-bit sampling and 46 kHz bandwidth, and it may come "free" with your desktop computer.

The Conexant codec for the Toshiba laptop looked promising because it offers 96 kHz sampling. That is not useful, however, since the bandwidth is limited to 19 kHz. It does support stereo (I/Q) inputs, so it could be used in a pinch.

The Ikey Audio USB sound card (available for under $40) may be useable, although its 16-bit conversions will limit your dynamic range.

---

wide IF signal — the audio output of an SSB transceiver. The audio is digitized with a normal PC sound card, filtered, and demodulated to provide text on screen or data to download to your hard drive. Eventually, we would like to integrate the digital mode functions with the software that operates the HF transceiver for traditional voice and CW modes.

Computer Audio

SDR software systems for personal computers often use a stereo sound card for input of the I and Q audio channels produced by a QSD-style receiver. This leads to some issues a user needs to consider: What kinds of sound cards are going to work well? What are the system issues? Do I need a second sound card?

Sound Card Configuration

The bare minimum requirement for sound support for I/Q audio input is a dual-channel (stereo) input sound card. Sound capability built into newer computers may be adequate for 24 or 48 kHz IF bandwidths (48 or 96 kHz sampling), at 24 bits. If your computer doesn't have the audio performance you want, there are readily available add-ons. Computer audio is a complicated subject that goes beyond the scope of this book, but we have a little more to say at the sidebar, **Computer Audio Specifications**.

The I and Q audio signals come in, but where will the receiver audio output go? It may be enough to use the same sound card for output to headphones or loudspeakers. (A mono output may be enough, but some operators like to use two outputs for *binaural* reception.) Normal PCs can be set up to handle simultaneous independent input and output audio, but you may need additional capability. For example, if your SDR system is a transceiver, you need both I/Q input and I/Q output channels, plus a microphone input and a speaker/phones output as shown in **Figure 4.1**. Your computer will need to have two or more sound cards, denoted "SC1" and "SC2". Fortunately, recent versions of Microsoft *Windows*

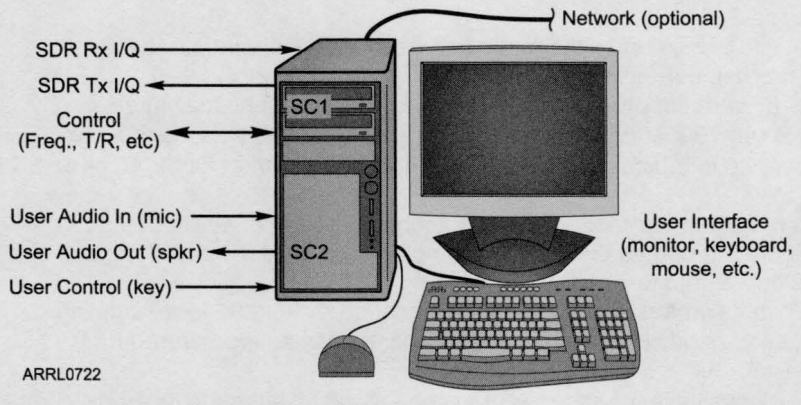


Figure 4.1 — SDR Computer Connections.

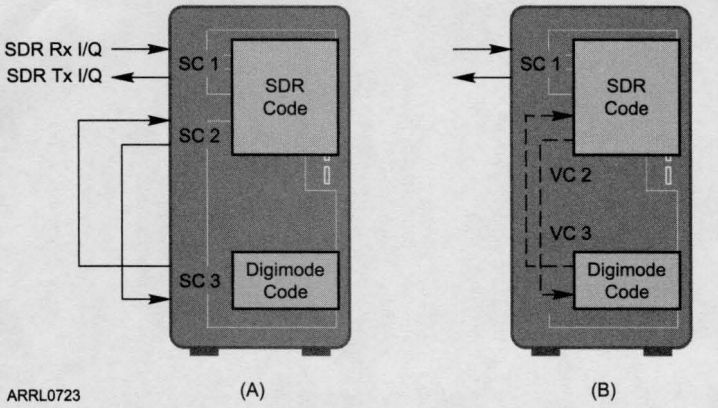


Figure 4.2 — Digimode Connections using (a) external wiring, (b) virtual connections.

and other operating systems make it relatively easy to work with multiple sound cards. Only the audio channels handling I/Q input and output need to be of highest quality; you can economize on the interfaces used for mic and speaker audio.

Supporting Digimodes

You may think you're done when you've wired up the I/Q inputs and outputs, your microphone, and speakers. But what if you'd like to work RTTY, PSK31, or one of the many other digital modes using a popular program like *MixW* or *Fldigi*? The brute force solution would have you add another sound card for the audio input to the digimode package so you can loop the SDR output to the digimode input (**Figure 4.2a**). Or you could install the digimode software on a separate computer with its own sound card there. All this seems pretty complicated and wasteful!

The problem is that with common operating systems it can be tricky to distribute digital sound data among independent programs in the same computer. What we want is simple to describe: route the audio data output of one program to the input of another (Figure 4.2b) through virtual sound cards "VC2" and "VC3." Fortunately, there is a simple solution for *Windows* users — *Virtual Audio Cable* (*VAC*).[1] Some advanced audio systems support flexible signal routing for a variety of operating systems, for example, *Jack*.[2]

Notes

[1] *VAC* by Eugene Muzychenko is available at **software.muzychenko.net/eng/vac.htm**.
[2] *Jack* is available through **jackaudio.org** or through software repositories.

Chapter Five

Using SDR

We've gone over a lot of material in the first several chapters about what software defined radio is, what digital processing is about, and how different SDR radios are put together. Maybe that's all very interesting to you, but you have the strange idea that SDR shouldn't just be about digital integrated circuits, computers, and block diagrams — maybe you'd like to use SDR for ham radio and you want to know what that's like.

Detour through Digimodes

Many amateurs have worked with digital modes — *digimodes*. That's a broad area, including old standbys like radioteletype (RTTY) and Morse code (CW)[1] but also the newer modes like PSK31, MFSK, Olivia, and many more. As we talked about earlier, digimode software really implements a software defined radio working on the SSB audio output of your traditional rig. So if you've used *Digipan*, *MixW*, or *fldigi*, you know how to use an SDR!

If you haven't tried digimodes, I recommend it as the simplest and cheapest way to get into SDR, providing you have a computer with a sound card input and a fairly recent *Windows*-oriented operating system. This describes the great majority of computers hams are likely to have. The only downside, for some of us, is that digimodes are primarily used for keyboard-to-keyboard communications, descending from the classic RTTY teleprinters of the 1940s and 50s. If your typing skills are too limited, this might be a problem. (Some resourceful hams have experimented with speech-to-text software that lets you talk instead of type.)

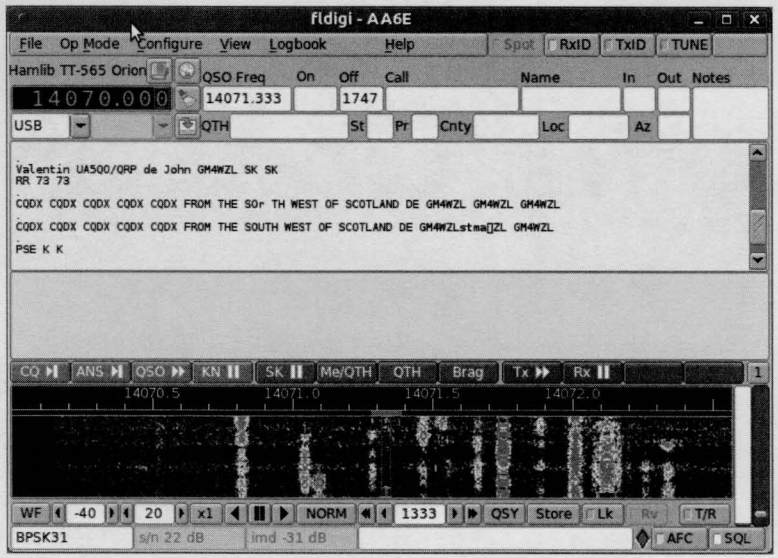


Figure 5.1 — *Fldigi* software screen receiving PSK31.

Using digimode programs is straightforward. First, you need to arrange for the audio connections, usually through an adapter that reduces hum from ground loops and provides for transmit/receive control. You fire up the software, and hopefully you see something like **Figure 5-1**. This is the *fldigi* software viewing the audio output of a Ten-Tec Orion transceiver tuned to 14,070 kHz in USB mode.

For receive, the main action is in the *waterfall* display in the bottom of the image, where the signals in the band 14,070-14,073 kHz flow down the screen. We seem to have 8 or 10 PSK31 signals (each about 100 Hz wide) coming in from Europe this afternoon on 20 meters. With the computer mouse, we can click on any of them. (The indicators at 14,071.33 show where we are tuned right now.) If all is well, the decoded text appears in the top middle part of the window. We have been listening to GM4WZL calling CQ.

The important thing is the way we use the software, because that's very typical of the way SDR programs work. We see the band activity in a waterfall display, we find something of interest and we tune by dragging the cursor with the mouse and clicking, all the while keeping track of band activity on either side of our signal of interest. This is *point and click* tuning.

SDR Receiver Software

Software defined radio receivers like the Perseus, the SDR-IQ, or the SoftRock kits generally let you work with broader bandwidths — 48 kHz and up. Typical receiving modes are CW, SSB, AM, etc — the same as you get on most communications receivers. The *SpectraVue* software that works with the SDR-IQ receiver (and others) is shown in **Figure 5.2**. We see a number of SSB stations on 20 meters across a 48 kHz band. At the top, the spectrum shows up as a simple spectrum plot (power vs. frequency), but just below is another waterfall display that shows the same spectrum but flowing downward (power vs. frequency vs. time). In

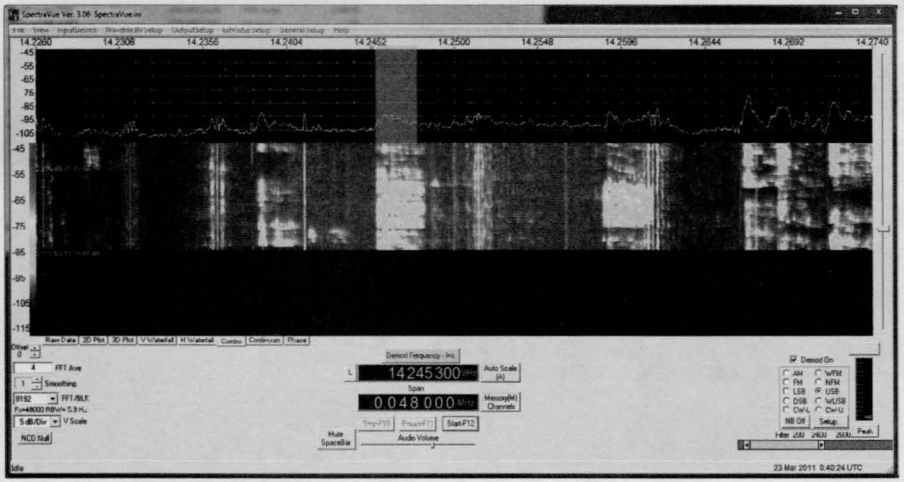


Figure 5.2 — *SpectraVue* software screen.

this case, we have tuned our radio to 14,245.3 kHz, and we are listening to the SSB station highlighted near the center of the display.

There are a number of other software packages for SDR receivers. Some of them are capable of working with a variety of radios that provide I and Q channel output, such as *SpectraVue*, *WinRad*, and *WinRadHD*. Other software may be dedicated to specific hardware systems. For example, the SRL QS1R has its *SDRMAX* series of software, and the Microtelecom Perseus also has its own software.

SDR Transceiver Software

PowerSDR is a software system developed specifically for the FlexRadio series of amateur transceivers. It is an open source development project that can operate with other I/Q radios — for example, the SoftRock receivers. While *PowerSDR* is useful with various receivers, it has many features that are directed to the FlexRadio products, which are complete transceiver systems. The main software window (**Figure 5.3**) shows the complex options available.

Because *PowerSDR* is designed to work with the FlexRadio hardware to produce a competitive amateur transceiver, there are a lot of controls. Some traditional radios have lots of knobs and indicators, while others hide many of the operating choices in a menu selection scheme. *PowerSDR* takes advantage of the large screens usually available on modern personal computers, at least "XGA" (1024 × 768 pixels). Many of the settings that would be put in menus in radios with

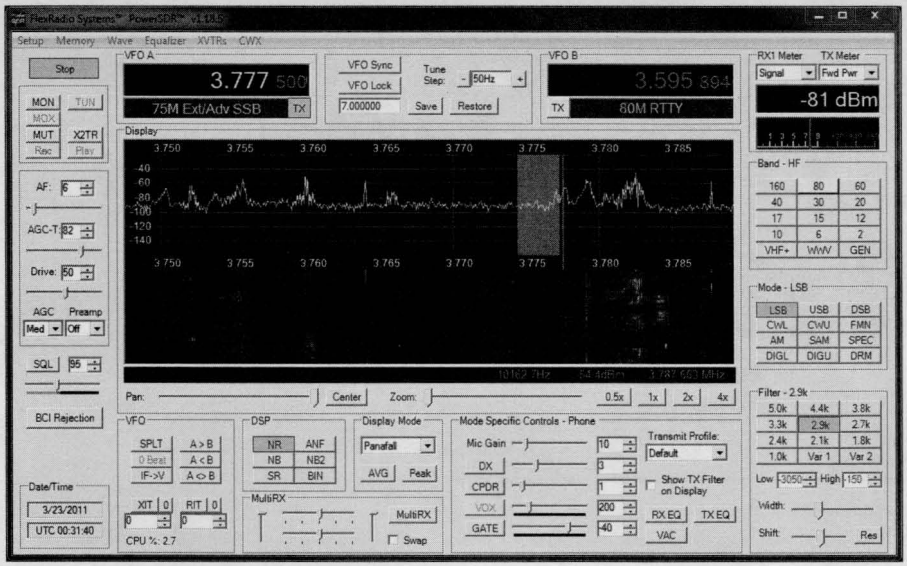


Figure 5.3 — *PowerSDR* software screen.

small screens can be laid out in a format that is easy to read and adjust. Controls are clustered in a logical way, but new users will need some time to master them.

You can change the radio's settings in several ways. Using the computer mouse and buttons, you can point, click, and drag to move the receiver passband, change the passband width, etc. Outside the spectrum display screen, you use your mouse and keyboard to adjust all the parameters like audio and RF gain, operating band, filter width, etc. If you don't care for mouse control, the keyboard can be used for tuning. If you prefer physical knobs, you can add a USB-connected knob to your computer for tuning.

Bells and Whistles?

The interfaces you get with *PowerSDR*, *WinRad*, and most of the other SDR control packages are really powerful and fun to use, but it's interesting to note that there is little that is "SDR" about the interface. That is, you could have many of the same features by attaching a computer to a traditional amateur transceiver. The *Ham Radio Deluxe* (*HRD*)[2] system, developed by Simon Brown, HB9DRV, is an example of what can be done. It integrates digimode decoding (a kind of SDR, as we said before), QSO logging, transceiver control, rotator control, satellite tracking, and more into a suite of programs through which you can operate your entire station. The main element that is not available when using *HRD* is the wide-band

"panoramic" spectrum display that SDR radios normally offer.

The point to make is that SDR is really not about a nice operator interface, although that can be a great side benefit for the amateur. SDR is about a set of signal processing technologies (filtering, modulation, etc.) that allows the radio designer to achieve higher performance and flexibility at lower cost by putting much of the radio in software. As we saw above for the new "DSP" radios, it is possible to embed the SDR technology while keeping the user interface looking quite traditional.

IF Band-Pass Control

Some pre-DSP receivers offered passband tuning (PBT), selectable IF bandwidth, and audio bandwidth adjustments. DSP/SDR radios offer these features as a matter of course, basically at no extra cost, because it's "only" software. With SDR technology, these features are available for the simplest radios, such as the SoftRock kits.

Signal Enhancement

If you sit down for the first time in front of a DSP or SDR radio, you will see some receiver buttons you may not recognize. For example, there is the "DSP" group of buttons for *PowerSDR* in Figure 5-3.

Selecting *Noise Reduction* (NR) helps the desired signal (voice or CW) jump out from the incoherent, undesired noise background. It is a kind of "smart" filter that sometimes can improve intelligibility by adapting itself to your current receiving conditions.

The *Automatic Notch Filter* (ANF) is another smart filter that looks for undesired tones (e.g., carrier signals) and inserts one or more notches in the band pass to eliminate the interference.

Noise Blanking (NB) has been a feature of communications receivers long before the arrival of DSP methods, however DSP blankers are often better than earlier analog circuits. In *PowerSDR,* the NB button will zero out signal samples that exceed a threshold, for example if you're dealing with strong ignition interference. The NB2 button looks for strong samples, but replaces them with the average of samples on either side — a less "violent" alteration of the input signal. Depending on conditions, one option or the other, or both together, may be more effective.

As was hinted in Chapter 1, filter choices like these are a feature of most DSP radios. The more advanced versions allow each option to be "tuned" by setting a number of parameters for each function that can control the thresholds and time constants used by the filters. You can spend many happy hours exploring all the possibilities! In the end, it comes down to your subjective judgment of what helps you copy the other fellow best.

Automatic Gain Control

Signal levels you encounter vary widely, and it is necessary to adjust the receiver gain so that you can tune the band for weaker stations while not being deafened by the strong ones. Maintaining roughly constant receiver volume is the traditional job of automatic gain control (AGC) in communications receivers. When we move into the software defined or DSP world, we need AGC as much as ever, but there are additional concerns.

Analog-to-digital converters have a problem when a signal goes over full scale. The digital output clips, introducing high distortion. (Analog circuits usually behave more gracefully when you overload them.) If the input to the ADC covers a wide band (like the QS1R), we may have to worry about strong signals anywhere in the HF spectrum. If an HF broadcast signal is strong enough that the ADC begins to clip, our weak ham signal will be wiped out. The AGC system has to cut back system gain to be sure that the ADC never overloads. In a wide band receiver this "anti-clip" AGC can work against the "constant volume" AGC function, because a strong signal far away from the station you're interested in can cause your station to be attenuated.

SDR radios may implement AGC in software. Some advanced radios (for example, the Ten-Tec Orion II and Flex radios) offer highly programmable AGC characteristics, where you can adjust thresholds, decay rate, "hang" time, and other parameters to your taste according to operating conditions. Most operators will probably be happy with the usual fast, medium, and slow AGC modes.

Time Delays and CW Issues

It seems like an odd fact, but the simplest and most historic Amateur Radio operating mode — CW — is sometimes troublesome in an SDR transmitter. The problem is that when you send CW, especially with a "bug," or with an electronic keyer, you need immediate audio feedback (sidetone) to be sure you're sending correctly. DSP filters always have a processing delay in the signal path, depending on the sampling rate, the type of filtering needed, and the amount of memory buffering. In some cases, this can amount to tens of milliseconds — enough to confuse high-speed CW operators who are monitoring the signal off the air. The solution is faster processing, smaller buffers, and, if necessary, the use of a monitor tone that is generated from the key contacts directly, and not off the air.

Processing delay is also serious if you wish to operate full break-in CW (QSK). Ideally, in full break-in, you want to be able to listen for signals between the dits and dahs while you are transmitting. This is a challenge, because the inter-element spacing in Morse code at 30 wpm is only 40 ms.[3] To be very effective, the total time to shift from transmit to receive to transmit needs to be much less — below, say, 15 ms. That includes buffering in the receive processing, transmit processing, plus whatever delays there are on the hardware side of the transceiver. Current

SDR transceivers show delays roughly in the 15-70 ms range. Contrast this with traditional QSK rigs that might show transmit-receive changeover delays as short as 5-10 ms. Purists will say that most current SDR-based transceivers can't do true full break-in CW. What is more achievable is "fast semi-break-in," where you can hear between words, or maybe between letters and figures. That's fine for many operators.

User Interfaces and Standards

At this time, there is little standardization of radio interfaces, beyond the phone jacks for audio, USB or other serial ports for digital communication, and a 50-ohm UHF connector for the RF. Software protocols are hardly standardized at all, except (possibly) within a particular vendor's product line. Transceivers still mostly communicate via audio cables and through CAT command interfaces, which traditionally relied on an RS-232 serial data connection to your computer. (New computers have mostly abandoned RS-232 in favor of USB!) Radios are appearing with high-speed connections (USB, Ethernet, or Firewire), sometimes including digital audio, but so far there is little compatibility among them.

If you are a programmer creating an Amateur Radio control application, you have a big problem: there are many different types of radios in use in ham shacks, often with incompatible computer interfaces. You'd like to write code that can be used by anyone, but it is too much work to provide for all possible radios. Fortunately, there is software that is designed to connect your program to many of those rigs. The open source Hamlib project[4] is one example. It creates a "hardware abstraction layer" that makes all rigs look the same to the control program. This method works best when you stick to the "least common denominator" command set that most rigs have — setting the frequency, modulation mode, and a few other items.

It used to be that you could count on radios having a big tuning knob that worked the same way on any rig. Alas, in the SDR world things are more complicated. PC software packages now use a multitude of tricks to let you change frequency — by dragging a cursor, dragging a frequency scale, clicking on digits, typing on the keyboard, etc. If you've learned to use one interface style, you're likely to be stumped if you want to use a different rig. We can hope for some convergence in the future as people agree on what works best in the user interface.

Notes

[1]Old-timers may blanch at calling Morse CW a digital mode, but it is binary (on-off) and it is digital (you use your fingers). Furthermore, if your sending is good, computer software can nicely decode the signal off the air. Die-hards like your author will claim they can copy marginal signals better by ear, but don't listen to them!

[2]*Ham Radio Deluxe* is available at **www.ham-radio-deluxe.com**.

[3]See **en.wikipedia.org/wiki/Morse_code** for more on Morse timing.

[4]See **www.hamlib.org**.

Chapter Six

Coming to a Shack Near You

Software defined radio and digital signal processing have quietly taken over communications technology and Amateur Radio. In earlier chapters, we've seen that several different "flavors" of SDR are now widely available, and in fact it's hard to find any new transceiver products that aren't based on this technology in one way or another. So what does this mean for ham radio in the near future?

"It is difficult to make predictions, especially about the future."[1] What can we say about what might be coming for SDR-related technology in Amateur Radio? Here's a list I made. Yours might be different.

- It will still be possible for QRPers to build great simple rigs with a few transistors, but the Elecraft KX3 and the SDR Cube Transceiver[2] show how SDR methods are being incorporated in ever-smaller self-contained packages. We haven't seen the end of this.
- One of SDR's natural strengths is exploring new modulation methods and message protocols. The explosion in digimodes on the HF bands is an example. My digital mode software supports some 50 modes and submodes for keyboard-to-keyboard or file transmission. Having so many can be overwhelming, but new modes will be coming that open up whole new areas for Amateur Radio.
- We should see more specialized communications modes like digital voice, *WSJT* for meteor scatter and moonbounce, and *WSPR* for propagation beacons — all assisted by SDR.
- Consumer technology often drives the computing business. (Think gaming and HDTV.) We can expect digital video to become more common in amateur work. When will we see the first amateur 3-D video transceiver?
- Competitive activities, like DXing and contesting are also driving Amateur

Radio technology. Band surveillance provided by SDR's wide-band displays helps us see where the action is. *CW Skimmer* and reverse beacon services let us spot desired stations and monitor propagation. Even for the casual operator with basic equipment, these advanced features will be available, expanding what everyone can do with ham radio.

■Station integration will improve. SDR radios should be easier to combine with other computer-based functions like logging, digimode backends, and other station controls such as antenna controls, power amplifiers, etc. There will be improvements, but interface compatibility will continue to be a challenge.

However the future unfolds for Amateur Radio, software defined radio will be a key ingredient. The technology will work its way into nearly all commercial radio products and many ham building projects. With so much computing power (remember the rig with 2,000 Mflops?), the main roadblock may be coordination and integration of radios, computers, and other amateur gear. Getting your radios, computer, antennas, and amplifiers to play well together will be a challenge for a long time to come. There will be some head-scratching, but the job will be fun and educational, and that's what a lot of us are looking for in Amateur Radio.

Notes

[1]This quote has been attributed to all kinds of people, from Yogi Berra to Dan Quayle to Mark Twain. For what it's worth, my vote goes to the physicist Niels Bohr. It's something a physicist would say. See **www.larry.denenberg.com/predictions.html**.

[2]George Heron, N2APB, and Juha Niinikoski, OH2NLT, "SDR Cube Transceiver," *The QRP Quarterly*, Winter, 2011, p 22-29. Also at **www.sdr-cube.com**.

Appendix

How Does a QSD Receiver Work?

If you want more of the flavor of how a modern SDR radio might work, you've come to the right place! In the following pages we will try to figure out the design of a direct-conversion radio based on the Quadrature Sampling Detector (QSD), mentioned in Chapter 3.

Our job is difficult, because I promised we would not use advanced math in this book. Things would be easier if we used complex numbers, exponential functions, and other standard items from the DSP toolkit. Nevertheless, I will give a general picture of how the QSD receivers work. For a more complete story, see the references in the endnotes.[1-5]

Let's start out on the right foot by reviewing how analog direct conversion receivers work, and then go step by step, modifying the design to use SDR principles. Beginning with the simple analog design, we go to a sideband-separating version, a "digital" SDR version, and finally an SDR with a QSD "front end." Why are we looking at direct conversion? (Direct conversion traditionally gets a bad rap, due to its problems with sensitivity, selectivity, and noise.) The reason to look at this design is that with the SDR approach, you can use a surprisingly simple circuit for the QSD and an inexpensive, relatively slow computer sound card (or equivalent ADC chip) for analog-to-digital conversion. We will see that this technique lets you "hear" RF signals within a band of plus or minus about 50 kHz around your local oscillator frequency. A system similar to this is behind the FlexRadio products and many other SDR devices.

For our discussion, let's choose a typical amateur receiver project. Say we want to receive the low 100 kHz of the 20 meter band, 14.000–14.100 MHz.

(Assume for the moment that we have a two-channel sound card with 50 kHz bandwidth.) Let's say there is a CW signal (the one we want to copy) at 14.070 MHz. There is some QRN and QRM throughout the band. In particular, there might be an undesired QRM signal at 14.010. Our receiver wants to use a local oscillator (LO) frequency in the middle of the band, 14.050 MHz, so that the IF is in the ultrasonic audio range (up to nearly 50 kHz) that can be accepted by a good sound card. We can make a diagram of the situation in **Figure A.1**. (Notice the power spectrum or "frequency domain" sketches of our signal. That's often a useful way to view signals. The arrows indicate RF input signal frequencies.) The range of RF frequencies above the LO is called the upper sideband (USB), and the range below is the lower sideband (LSB).

A *mixer* is a device that multiplies one signal by another. The effect is to shift the incoming signals by taking the difference with the LO frequency. The mixer produces sum and difference frequencies. The low-pass filter is there to pass only the difference frequencies.

This receiver is really too simple, and you can see the problem. The signals in the IF are at the difference frequencies, that is, at RF frequency minus the LO frequency — whether positive or negative. The 50 kHz band above the LO frequency is shifted down to 0–50 kHz output band, but the 50 kHz band below the LO is also shifted (and reversed) into the output band. The good (solid) signal that we want is now accompanied in the IF passband by the interfering (dotted) signal. If there was a bad signal at 14.030 MHz, it would land right on top of the good signal, ruining our day! The upper and lower sidebands (either side of the LO) have been folded into the IF. Furthermore the noise (the wavy lines) from both the lower and upper sideband appears in the output. This receiver just has no sideband rejection!

Conventional analog radios generally manage the sideband rejection problem by placing filters ahead of the mixer, but this is impractical when you use a low IF frequency — the sidebands are simply too close to each other for a practical filter to separate them.

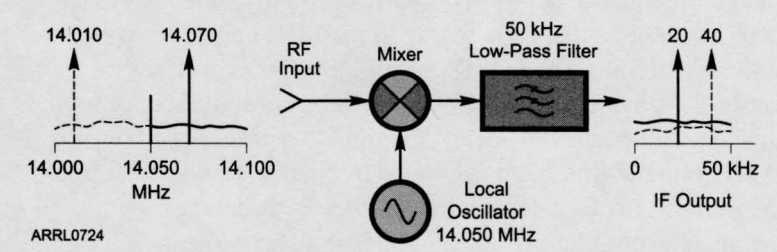


Figure A.1 — Direct conversion without sideband rejection.

It is possible to separate the sidebands and receive a full 100 kHz band in our example receiver. To do that, you need to use the fact that the mixer produces positive frequencies (the upper sideband) and negative frequencies (the lower sideband). The

scheme of Figure A.1 can't distinguish them, but with some more work we can do it.

(Maybe you didn't know there was such a thing as a negative frequency? A *positive frequency* signal is one where the phase increases over time, while the phase of a *negative frequency* signal decreases. The idea of positive and negative frequencies is critical to distinguishing upper and lower sidebands.)

The key to separating sidebands is to change the incoming RF signal into its "in-phase" (I) and "quadrature" (Q) components. (*Quadrature* refers to a quarter of a cycle in phase, or an angle of 90°.) These tell us about the phase of the signal, compared to the local oscillator. Without delving much into mathematics (see the references for that), we can explain I and Q.

If you use a fast oscilloscope to view a strong signal in the RF passband, you will see a sine wave that has a frequency of 14.050 MHz, more or less. If you use the LO to trigger your scope at 14.050 MHz, while you are still viewing the input signal voltage, you will see the pattern sliding to the left or right depending on whether the signal frequency is higher or lower than the LO. (This is a visual way to tell positive from negative frequencies.) We can build on this idea by making two separate IF channels ("I" and "Q") by mixing the RF with two versions of the LO, one phase-shifted by 90° relative to the other. (**Figure A.2**) I and Q serve to capture the signal's phase information. If your input signal was a cosine wave at the LO frequency (0° phase), it would produce a strong (dc) I signal but a zero Q signal. If the frequency was a little higher than the LO, the signal slowly "rotates" out of the I into the Q channel, becoming all Q when the signal has slipped by 90° in phase, then it goes back to all I (but negative) when the slip gets to 180°, and so on. (Electrical engineers call the IF signal a "phasor" that rotates at a rate that is the difference between the RF signal and the LO frequency. The direction of rotation corresponds to upper or lower sideband conversion. The I and Q channels are the "real" and "imaginary" parts of the complex IF signal.) Combining the I and Q IF channels the right way will allow us to get separate IF outputs for the upper and lower sidebands.

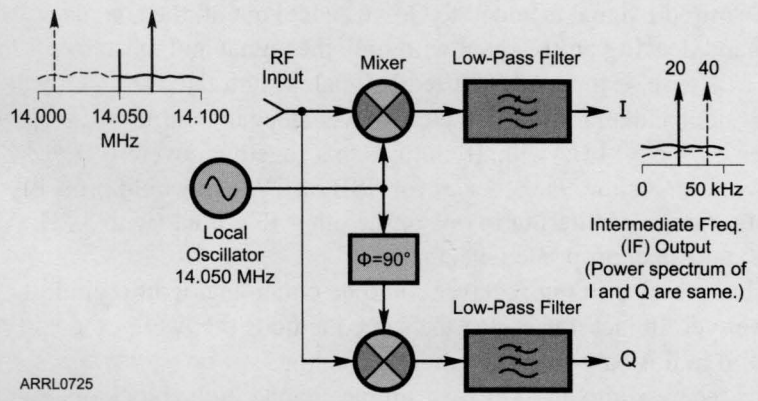


Figure A.2 — Direct conversion with I and Q channels.

Figure A.3 shows

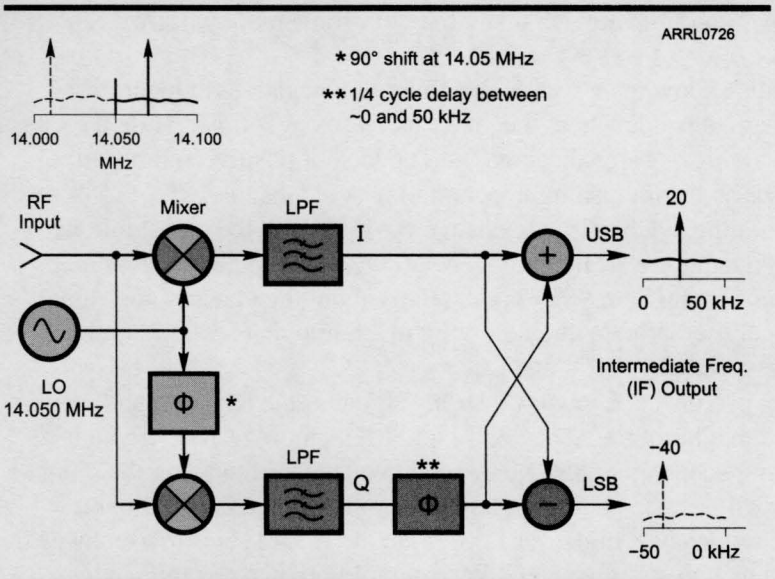


Figure A.3 — Direct conversion receiver, dual sideband output.

how we can make our circuit produce two output signals using the I and Q channels. One output is the upper sideband (corresponding to 14.050–14.100 MHz), and the other is the lower sideband (14.000–14.050 MHz). The key part is the 90° phase shift in the IF. The phase shift in the LO is relatively easy to achieve because it only has to work at one frequency. But the IF passband is very wide in percentage terms, nearly 0–50 kHz, and you need a circuit that shifts 90° at each frequency in the band.[6] This is a problem in an analog circuit, but not so hard in the digital case. A phase shift is the same as a delay, with the delay time equal to one quarter of a cycle of the IF signal. Since a cycle (a period) of a sine wave with frequency f is $1/f$, you can see how implementing a 90° phase shift at very low IF frequencies might be tricky, because the required delay becomes very high. With this design, we are going to have problems around zero frequency — dc — where noise and other problems crop up. (More advanced designs are available that eliminate the zero frequency problem, but that's beyond our scope here.)

When the phase-shifted I signal is added to Q, we cancel out all the signals from the lower sideband, leaving an IF signal with only the signal and noise from the upper sideband. Likewise, if the phase-shifted I signal is subtracted from Q, we cancel out the higher sideband and we have a clean lower sideband output.

This design gives us two 50-kHz-wide IF outputs that together cover 100 kHz of RF spectrum. If we want a complete receiver for SSB or CW, we would probably want to connect another stage of filtering to one or the other IF output (with 3 kHz or smaller band pass) and an appropriate detector.

So far, the signal processing in our receiver could be either analog or digital. It is not an SDR receiver yet. In fact it uses just the same methods that were used early on in amateur sideband radios based on the "phasing method."[7]

We can make this receiver into an SDR by a simple change in the block diagram (**Figure A.4**). Let's digitize the I and Q IF signals, using a high-quality computer

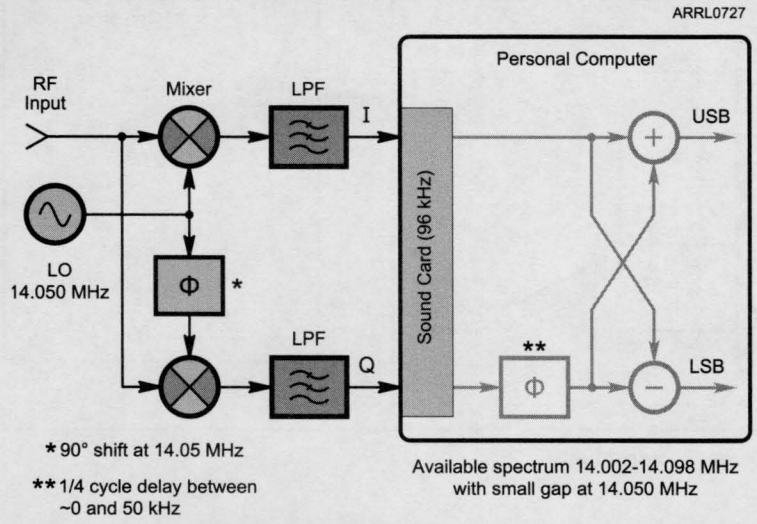


Figure A.4 — Direct conversion receiver, software defined back end.

sound card that will operate at 96 kHz. (That's not quite the 100 kHz we have been discussing, but it should get us close to 96 kHz total band pass, which will be enough.) Once I and Q are digitized, they can be manipulated in software like *PowerSDR* to develop a complete spectrum display. (See Chapter 5.)

One nice feature of the SDR version is that the critical 90° wideband phase shifter can be implemented more easily in software than hardware. A software filter has very predictable response and can usually be made better just by having the CPU work harder (computing more accurately).

The computer approach solves some problems, but possibly makes some new ones. Software defined radios are always critically dependent on the quality of the ADC — the sound card in our case.[8] To get close to our 100 kHz desired total bandwidth, we need to use 96 kHz sampling with about 48 kHz bandwidth. Alas, your consumer-grade sound card may provide only about 20 kHz of bandwidth, limiting your receiver coverage (USB and LSB) to about 40 kHz.

The quadrature sampling detector (QSD) is an elegant way to directly convert the RF signal to I and Q baseband signals. This method (**Figure A.5**), described and patented by Dan Tayloe, N7VE,[9] uses a CMOS switch as a combination sample-and-hold device and mixer, effectively multiplying the signal by two square waves that are offset by 90°, thanks to being derived from a counter that operates at four times the desired LO frequency. The counter's output is decoded and causes CMOS "contact closures" in a sequence *0-1-2-3-0-1-...* that repeats at the 14.050 MHz LO rate. The QSD makes an inexpensive and surprisingly efficient mixer. It is the basis of many designs, including the SoftRock kits and the FlexRadio products.

Now that we have a few ways to generate the USB and LSB IF signals, how do we make a useful receiver? If we are receiving CW (an on-off keyed carrier), we want to convert the signal to a convenient tone (usually between 400 and 1,000 Hz) that will work with speakers, headsets, and our ears. This can readily be done with

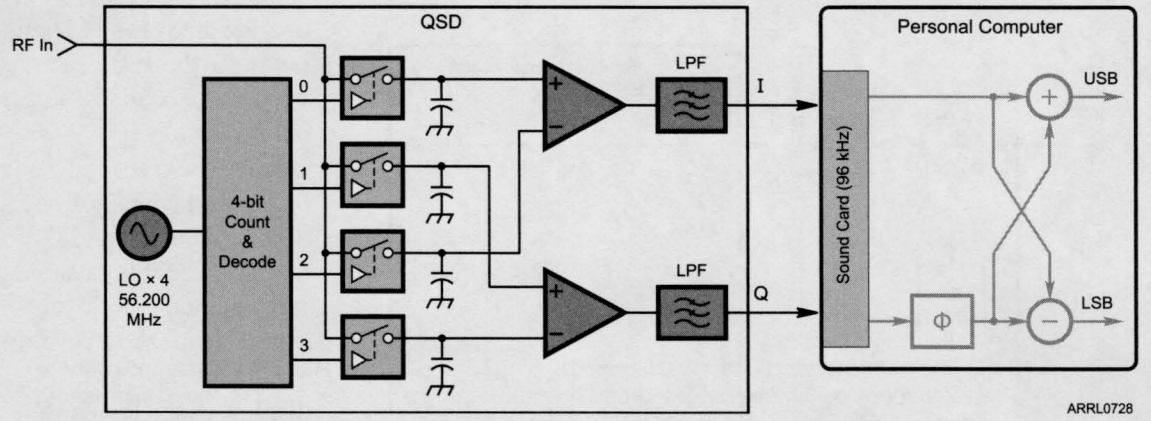


Figure A.5 — Direct conversion receiver, QSD front end.

another stage of frequency conversion in software. We will pick out (tune for) the desired receiver frequency, filter down to a small band pass around that frequency (typically 100–1,000 Hz bandwidth) and convert down to a human-friendly audio tone.

A big advantage of the QSD/SDR scheme, as with most SDR radios, is that we get a large IF band pass (up to about 100 kHz) all available in our computer. For most modes, we just want to listen to a few kHz, but we have a powerful computer, so why not process the whole range of signals and display it? We can use standard FFT software to create a spectrum from the IF and display the result as a continuously updating power spectrum on the computer screen. This gives you a simultaneous view of all signals over the whole IF band — a great operating convenience.

Software defined radios built around the quadrature sampling detector have been proven very effective in amateur service. While they are somewhat limited in bandwidth by computer sound cards, they are a central part of the amateur SDR market.

Notes

[1]D. Thornton Smith, KF6DX/7, "Signals, Samples and Stuff: A DSP Tutorial," Part 1, *QEX* Mar, 1998 p 3ff; Part 2, *QEX*, May, 1998, p 22ff; Part 3, *QEX*, Jul, 1998, p 13ff; Part 4 *QEX*, Sept, 1988, p 19ff. These articles describe many of the techniques that were incorporated in the Ten-Tec Orion transceiver.

[2]Doug Smith, KF6DX, *Digital Signal Processing Technology: Essentials of the Communications Revolution*, American Radio Relay League, Newington, CT, 2003.

[3]Gerald Youngblood, AC5OG, "A Software-Defined Radio for the Masses", Part 1, *QEX*, Jul/Aug, 2002, p 13ff; Part 2, *QEX*, Sept/Oct, 2002, p 10ff; Part 3, *QEX*, Nov/Dec, 2002, p 27ff; Part 4, *QEX*, Mar/Apr, 2003, p 20ff. These papers describe the design that became the FlexRadio Flex-1000 transceiver.

[4]William E. Sabin, WØIYH, and Edgar O. Schoenike, Eds., *HF Radio Systems & Circuits*, Noble Publishing Co., Atlanta, 1998.

[5]H.Ward Silver, NØAX, and Mark J. Wilson, K1RO, Eds., *ARRL Handbook for Radio Communications*, 2010 ed, Chapter 15, "DSP and Software Radio Design" (and other *Handbook* editions). This is a broad but fairly technical survey of DSP and SDR methods.

[6]The transformation of a signal by 90° at all frequencies has the scary name "Hilbert transform."

[7]You can learn more about the phasing method in the *ARRL Handbook for Radio Communications,* 2010 ed., p 13.8ff (and other *Handbook* editions).

[8]See the Sidebar "Computer Audio Specifications" in Chapter 4. Remember, for most people audio is limited to 20 kHz or less, so recordists have little reason to handle higher frequencies. Things are different if you're a dog — or an SDR buff.

[9]U.S. Patent 6,230,000. For details, see **patft.uspto.gov**.

Glossary

Software defined radio and digital signal processing technology come with a large new vocabulary. This glossary defines some of the more arcane terms that may be new to some readers. Some of these will be well-known to many readers (apologies!), but I am trying to cover all the terms that might give trouble to those readers who are new to SDR, DSP, or even new to electronics and Amateur Radio.

Analog — Strictly speaking, an analog quantity is one that represents a signal of interest, such as the electric field of an incoming radio signal. Voltage or current can be an "analog" of something in the real world, like speed or temperature. In practice, analog means a smoothly varying quantity (typically a voltage or current) that conveys a signal in electronic circuits. Analog electronics are built from traditional transistors, resistors, capacitors, etc., usually operating in a linear (non-switched) manner. See **Digital**.

Analog-to-Digital Converter (ADC) — Accepts an analog signal voltage and converts it to a numeric form. An ADC is commonly a single integrated circuit. Basic specifications of an ADC include its maximum conversion rate (megahertz) and its precision (sample size in bits). At this writing, a typical fast ADC chip might produce 16-bit samples at 100 MHz. See **Digital to Analog Converter**.

Binary — A variable that has two states: 1 (on or "true") or 0 (off or "false").

Binaural Reception — "Binaural" signifies using both ears. Hams don't normally use stereo transmission, but binaural processing is useful in the receiver. For example the two ears (through headphones) can be fed different frequency bands using DSP methods, e.g. lower frequencies to the left, higher to the right. A well-tuned CW signal will seem to be in the "center," while noise and QRM signals will mostly be to one side or the other. This helps the brain pick out a weak signal amidst interference. Another binaural technique is to listen to separate receivers on the same frequency, but using different antennas that produce a spatial diversity effect that can also enhance a desired signal.

Break-In (QSK) — A style of Morse/CW operating that allows the receiving station to "break in" to get your attention during your transmission. Break-in requires your station to be able to listen frequently between transmissions. *Full break-in* means that you are able to switch between transmit and receive (T/R) very quickly, typically between the dits and dahs of Morse. *Semi break-in* means that your T/R switching is slower, perhaps between words or short pauses in sending. SDR/DSP rigs may have relatively long T/R switching times, preventing full break-in operation.

CAT Interface (Computer Aided Transceiver) — Originally a Yaesu term, but now a generic reference to a radio transceiver's digital control port. The port is commonly a serial interface that can be adapted for a computer's traditional serial I/O connection (COM port). Some recent CAT interfaces have been implemented over USB and Ethernet.

Central Processing Unit (CPU) — The modern CPU is usually a single integrated circuit that is the heart of a typical computer system, including data registers, arithmetic processors, memory controls, etc.

Digital — A digital quantity is one that is represented by bits (1s or 0s) that may be arranged as "bytes" or "words" (for example, a 16-bit word can represent a number from 0–65,535). Digital circuits are built from switches, gates, memory devices, etc., that normally operate in a non-linear (switched) mode. See **Analog**.

Digital Signal Processing (DSP) — A collection of hardware and software technologies that operate on signals that have been converted to digital (numeric) form. A DSP chip is a special-purpose digital computer optimized for processing digital signals.

Digital-to-Analog Converter (DAC) — Converts a numeric digital input to an analog output voltage. See **Analog-to-Digital Converter**.

Direct Conversion — A direct conversion receiver uses a local oscillator (LO) frequency close to the RF signal frequency. The LO is mixed with the RF, producing an IF frequency near zero or in the audio (baseband) range. See **Superheterodyne**.

Discrete Fourier Transform — See **Fourier Transform**.

Dynamic Range — The ratio of the largest signal that can be handled to the smallest signal that can be detected. SDRs may require a high dynamic range if you want best weak signal sensitivity in the presence of strong signals.

Fast Fourier Transform — See **Fourier Transform**.

Field Programmable Gate Array (FPGA) — An integrated circuit that comprises a large number of gates. Connections among the gates are determined by data that can be loaded into the chip under computer control. FPGAs provide very high processing power for specialized applications like SDR and DSP.

Firewire (IEEE 1394) — A fast serial connection method that is used for high-speed video and audio applications. See **en.wikipedia.org/wiki/Firewire**.

Firmware — The name sometimes given to software that is built into a digital system. Generally, firmware is intended to be updated or reloaded infrequently. Some radio manufacturers provide firmware updates via Internet download on an occasional basis. See **Software**.

Floating Point Arithmetic — A way of representing numbers that may vary widely in magnitude. It is similar to the scientific notation often used to write very big or small numbers. Numbers are represented by a fraction (say between -1 and +1) multiplied by 10 to an exponent (for example, 0.2997×10^9). Modern personal computers support floating point arithmetic in addition to fixed point (integer) arithmetic. For DSP applications, programming with floating point arithmetic is often easier than with integers, because you don't have to worry so much about scaling of numbers, but execution can be somewhat slower. See **en.wikipedia.org/wiki/Floating_point**. The most common standard floating point format is IEEE 754, see **steve.hollasch.net/cgindex/coding/ieeefloat.html**.

Fourier Transform (Discrete Fourier Transform, Fast Fourier Transform) — The *Fourier Transform* is a mathematical process or operation that converts a signal from the time domain to the frequency domain. (The inverse Fourier Transform converts a signal back to its original form.) The *Discrete Fourier Transform* **(DFT)** converts a sequence of (digitized) signal samples between time and frequency domains. The *Fast Fourier Transform* **(FFT)** comprises a family of software algorithms (programming methods) that greatly speed up computing of large Fourier Transforms.

Frequency Domain — An RF or IF signal is said to be in the *frequency domain* when it is described primarily by its amplitude and phase across a range of frequencies. This description is complementary (and equivalent) to a description in the time domain. (Mathematically, the Fourier Transform is used to convert from frequency domain to time domain and back.)

Intermediate Frequency (IF) — A receiver's incoming radio frequency signal is usually converted to an IF frequency that is more convenient for amplification and filtering. The IF can be at either a higher or lower frequency than the input RF. It is common to have multiple IFs in one receiver that can range from VHF (such as 70 MHz) down to "audio" frequencies, such as 14 kHz. See **Superheterodyne**.

Local Oscillator (LO) — A stable oscillator that is used to convert an RF input signal to intermediate frequencies (IF). A variable-frequency LO (VFO) can be used to tune a receiver to a desired frequency or to determine a transmitter's output frequency.

Nyquist Rate, Nyquist Sampling — The *Nyquist sampling rate* is double the highest frequency present in an input signal. If you sample at the Nyquist rate or higher, you have not lost any information about the input. The Nyquist-Shannon theorem says that you can precisely reconstruct the input from the samples.

Oscilloscope — An instrument for viewing the Time Domain behavior of an electrical quantity (voltage), usually as a plot of voltage vs. time. See **Time Domain**.

Power Spectrum (Spectrum) — The Frequency Domain view of a signal: power as a function of frequency, such as displayed by a spectrum analyzer, a "panadapter," or the computer displays of most SDR radios. Since power is the square (or complex squared magnitude) of a signal amplitude, it does not contain phase information.

QRN — The Q signal used by radio amateurs referring to radio frequency noise interference from natural or other non-amateur sources, such as lightning, automotive ignition, etc.

QRM — The Q signal referring to radio frequency interference from other transmitters, usually other amateurs.

Quadrature Sampling Detector (QSD) — A method of direct conversion of an RF input signal to I and Q (in-phase and quadrature) baseband IF streams using analog switching.

Quadrature Sampling Encoder (QSE) — A method of converting synthesized I and Q (in-phase and quadrature) baseband IF streams to an RF output signal. Effectively, this is the Quadrature Sampling Detector "run backwards."

Relay — A relay is a device that causes electrical contacts to close or open when a current flows through a magnetic coil.

Radio Frequency (RF) — The frequency at which amateur communications are desired — a receiver's input frequency or a transmitter's output frequency. In common amateur work, RF ranges from 1.5 MHz–30 MHz and upward.

Software — A software "program," or "application," or "app," is a list of instructions that control a digital processor, such as an SDR system. See **Firmware**.

Software Defined Radio (SDR) — A design technique that leverages the power and flexibility of digital computing to implement radio transmitters and receivers that formerly had to be "hardwired" from electronic components like transistors, inductors, resistors, etc. A "pure" SDR receiver might consist simply of an analog-to-digital converter together with a computer. A pure SDR transmitter might comprise a computer feeding a digital-to-analog converter.

Spectrum Analyzer — An instrument for viewing signals in the Frequency Domain, as a plot of power vs. frequency. Sometimes called a "panadapter" when viewing a receiver's IF band.

Superheterodyne (Heterodyne) — "Heterodyne" is an older term for frequency mixing, where a signal is multiplied by (mixed with) another signal, producing the sum or difference frequencies. A *superheterodyne* receiver, or "superhet," uses a mixer and a local oscillator (LO) to translate the input radio frequencies of interest to a fixed intermediate frequency (IF), for convenient filtering and amplification. Most communications receivers are superhets that may have several mixer and IF amplifiers. See **Direct Conversion**.

Time Domain — An RF or IF signal is said to be in the time domain when it is described primarily by its amplitude and phase at various points in time. See **Frequency Domain**.

INDEX