Contents

Chapter 8 — Online Content

- SDR Simplified — Fourier Transforms by Ray Mack, W5IFS
- SDR Simplified — Fundamentals of Sampling by Ray Mack, W5IFS
- SDR Simplified — More on Sampling by Ray Mack, W5IFS
- “Radio Mathematics” — see online content for the Electrical Fundamentals chapter

Chapter 8
DSP and SDR Fundamentals

This chapter was updated by Doug Grant, K1DG, from material originally created by Alan Bloom, N1AL. It explores the fundamentals of digital signal processing (DSP) and software defined radio (SDR). Material is also taken from *QEX* "SDR: Simplified" articles by Ray Mack, W5IFS. Key to DSP and SDR, analog-digital conversion and types of converters are covered, as well.

DSP technology has progressed to the point where it is the dominant technology in our radio equipment. DSP has largely replaced analog hardware circuits with digital processors and software, offering amateurs flexibility and features only dreamed of in the past.

Software defined radio has displaced the superheterodyne architecture that was dominant since its introduction in the 1920s. SDR is only possible through DSP techniques implemented on the advanced platforms that have become available.

This chapter begins with the fundamentals of DSP and extends them to SDR design. Other chapters will cover DSP methods in more detail for implementing oscillators, modulation, filters, and the functions associated with receiving and transmitting.

For the fullest understanding of this chapter, the reader should have a basic familiarity of the topics covered in the **Radio Fundamentals** chapter as well as some high-school trigonometry.

References to math tutorials are provided in **Radio Mathematics** which is part of this book's online content along with additional background and support materials, including a glossary.

8.1 Introduction to DSP

Digital signal processing (DSP) has been around a long time. The essential theory was developed by mathematicians such as Newton, Gauss, and Fourier in the 17th, 18th, and 19th centuries. It was not until the latter half of the 20th century, however, that digital computers became available that could do the calculations fast enough to process signals in real time. Today DSP is important in many fields, such as seismology, acoustics, radar, medical imaging, nuclear engineering, audio and video processing, as well as voice and data communications.

In all those systems, the idea is to process a digitized signal so as to extract information from it or to control its characteristics in some way. For example, an EKG monitor in a hospital extracts the essential characteristics of the signal from the patient's heart for display on a screen. A digital communications receiver uses DSP to filter and demodulate the received RF signal before sending it to the speaker, headphones, or waterfall/band-scope display. In some systems, the signal to be processed may have more than one dimension. An example is image data, which requires two-dimensional processing. Similarly, the controller for an electrically-steerable antenna array uses multi-dimensional DSP techniques to determine the amplitude and phase of the RF signal in each of the antenna elements. A CT scanner analyzes X-ray data in three dimensions to determine the internal structures of a human body.

A typical DSP system is conceptually very simple. It consists of only three sections, as illustrated in **Figure 8.1**. An *analog-to-digital converter* (ADC, or A/D) at the input converts an analog signal into a series of digital numbers that represent snapshots of the signal at a series of equally spaced sample times. The digital signal processor does some kind of calculations on that digital signal to generate a new stream of numbers at its output. A *digital-to-analog converter* (DAC, or D/A) then converts those numbers back into analog form.

Some DSP systems may not have all three components. For example, a DSP-based audio-frequency generator does not need an ADC. Similarly, there is no need for a DAC in a measurement system that monitors some sensor output, processes the signal, and stores the result in a computer file or displays it on a digital readout.

The term "DSP" is normally understood to imply processing that occurs in real time, at least in some sense. For example, an RF or microwave signal analyzer might include a DSP coprocessor that processes chunks of sampled data in batch mode for display a fraction of a second later.

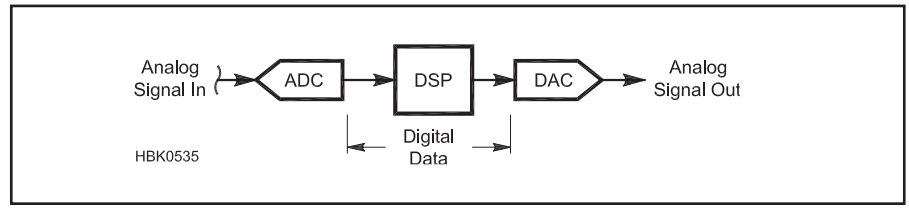


Figure 8.1 — A generic DSP system.

8.2 Introduction to SDR

The concept of a *software-defined radio* (SDR) has been around for a long time, but became popular in the 1990s. By then, DSP and data converter technology had developed to the point that it was possible to implement almost all the signal-processing functions of a transceiver using inexpensive programmable digital hardware. The frequency, bandwidth, modulation, filtering, and other characteristics can be changed under software control, rather than being fixed by the hardware design as in a conventional analog radio. Adding a new modulation type or a new improved filter design is a simple matter of downloading new software.

Compared to analog radios, SDR has some major advantages. In analog radios, passive components such as capacitors, resistors, and even inductors are subject to changes in value due to temperature drift and aging. They also have loose initial value tolerances and often require adjustments or calibration. Likewise, active components such as amplifiers and mixers also are subject to variations in performance. If the function of these components can be replaced with software, most of these problems disappear.

SDR is appealing to regulatory bodies such as the FCC because it makes possible a communications system called *cognitive radio* in which multiple radio services can share the same frequency spectrum.[1] Each node in a wireless network is programmed to dynamically change its transmission or reception characteristics to avoid interference to or from other users. In this way, services that in the past enjoyed fixed frequency allocations but that only use their channels a small percentage of the time can share their spectrum with other wireless users with minimal interference.

There has been much, sometimes heated, discussion about the precise definition of a software-defined radio (SDR). Most feel that, at minimum, an SDR must implement in software at least some of the functions that have traditionally been done in hardware. Others feel that a radio doesn't count as an SDR unless nearly all the signal-processing functions, from the RF input to the audio output (for the receiver) and from the microphone ADC (analog-to-digital converter, or A/D) to the power amplifier input (for the transmitter), are done in software. Others add the requirement that the software must be re-configurable by downloading new code, preferably open-source. For our purposes we will use a rather loose definition and consider any signal-processing function done in software to fall under the general category of SDR.

Some SDRs use a personal computer to do the computational work and external hardware to convert the transmitted and received RF signals to lower-frequency signals that the computer's audio interface can handle. Some SDRs avoid the use of the PC's sound card by including their own audio *codec* (short for "coder-decoder," a chip that includes both A/D and D/A converter functions) and using analog audio for the user interface. Some SDRs transfer the downconverted (and possibly filtered) data to the PC via a USB port. Modern PCs provide a lot of computational power for the money and are getting cheaper and more powerful all the time. They also come with a large color display, a keyboard and mouse for easy data entry and navigation, a large memory and hard disk, which allows running logging programs and other software while simultaneously doing the signal processing required by the SDR.

Smaller, even less expensive, computing platforms such as Arduino, Raspberry Pi, Beaglebone, Red Pitaya and many others have become available. While they often lack the peripherals and user interfaces of a complete PC, some of them have sufficient computational power to be useful for SDR experimentation.

Some SDRs have almost no knobs or buttons on the box and none of the traditional features such as frequency display, meters, and so on. In such radios, all control functions are done on an external PC with control software provided by the manufacturer. The user interface is the PC keyboard, monitor, and mouse. Other SDRs look more like conventional analog radios and don't need an external PC for control. While the signal processing is done with one or more embedded DSPs, the user interface consists of knobs and pushbuttons which many users prefer to a mouse and keyboard and pull-down menus. Some newer SDRs use a touchscreen for the user interface to emulate the buttons.

Either method offers all the important advantages of applying DSP techniques to signal processing. The channel filter can have a much better *shape factor* (the ratio between the width of the passband and the frequency difference of the stopband edges). FIR filters are linear phase and have less ringing than analog filters of the same bandwidth and shape factor. Once the signal is in the digital domain all the fancy digital signal processing algorithms can be applied such as automatic notch filters, adaptive channel equalization, noise reduction, noise blanking, and feedforward automatic gain control. Correcting bugs, improving performance or adding new features is as simple as downloading new software.

Following an overview of SDR systems, the details of the various blocks used in such systems, including analog-digital conversion, are then explored. The chapter concludes with a discussion of the basic theory of discrete-time and digital signals, with emphasis on topics relevant to radio communications.

8.2.1 SDR Architecture Options

The transition between analog and digital signals can occur at any of several places in the signal chain between the antenna and the user interface. This choice is an important factor in determining the overall architecture of the SDR. This section presents several block-diagram-level concepts for software-defined radio and compares architecture options.

DSP AT AUDIO FREQUENCIES

The initial use of DSP by amateur radio operators was to implement audio-frequency filtering. By the early 1990s, converter costs were reasonable and the processing power required was available in relatively inexpensive microprocessors with enhanced math function capability. The flexibility of digital filters (see the **Analog and Digital Filtering** chapter) and the ability to alter bandwidth and perform noise reduction resulted in quick adoption of DSP techniques in amateur equipment.

In 1992, Dave Hershberger, W9GR, designed an audio-frequency DSP filter based on the TMS320C10, one of the earliest practical DSP chips available.[2] This filter was an external standalone unit that plugged into the headphone jack of a receiver and included filters with various bandwidths, an automatic multi-frequency notch filter, and an adaptive noise filter.

The advantage of DSP at audio frequencies is that it can be easily added to an unmodified analog radio as in **Figure 8.2**. Many amateurs use a similar approach to implement digital modulation modes, using a PC and software as the outboard DSP processor. The software produces the required transmission wave-

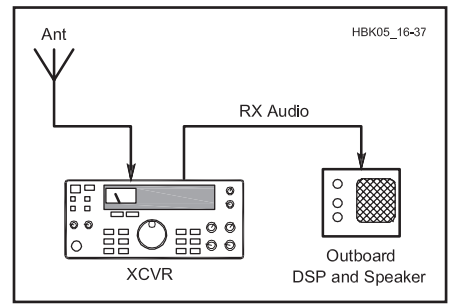


Figure 8.2 — An outboard DSP audio processor.

forms (and demodulates the received signal), using the PC's audio input and output connected to the audio input and output of a conventional SSB transceiver.

DOWNCONVERSION TO ANALOG BASEBAND

A related technique is to *downconvert* a slice of the radio spectrum to baseband audio using a technique similar to the direct-conversion receivers popular with simple low-power CW transceivers. (See the **Receiving** chapter.) This idea was pioneered by Gerald Youngblood, AC5OG (now K5SDR), with the SDR-1000 transceiver, which he described in a series of *QEX* articles in 2002 and 2003.[3] The receiver block diagram is shown in **Figure 8.3**. It uses a unique I/Q demodulator designed by Dan Tayloe, N7VE, to convert the RF frequency directly to baseband I ("In-phase") and Q ("Quadrature") signals.[4] (I/Q modulation and quadrature signals are discussed in the **Modulation** chapter.)

The baseband signals are fed to the stereo input of a PC's sound card, represented by the low-pass filters and analog-to-digital converters (A/D, or ADC) in the figure. Software in the PC does all the signal processing and demodulation, ultimately producing audio for the operator to hear or a display of decoded bits in the form of text on the screen. The transmitter is the same block diagram in reverse, with an I/Q modulator converting the I/Q signal from the sound card up to the RF frequency where it is filtered and amplified to the final power level.

The sound card method manages to achieve reasonable performance with simple, inexpensive hardware and resources already available in the PC. Once the A/D converters in the sound card have digitized the signal, the DSP capability of the PC can do amazing things with it. In addition to implementing conventional transceiver functions such as several types of detector, variable-bandwidth filters, software AGC, an S-meter and speech compression, the software can include some extra features such as an automatic notch filter, noise reduction, panadapter and waterfall spectrum displays and decoding of signals such as RTTY, PSK, and WSJT modes.

The simple hardware of the SDR-1000 does impose some performance limitations. Because of imperfections in the analog downconverter, unwanted-sideband rejection is not perfect. This is called "image rejection" in the SDR-1000 literature. On the panadapter display, strong signals show up weakly on the opposite side of the display, equally-spaced from the center. DC offset in the analog circuitry causes a spurious signal to appear at the center of the bandwidth. To prevent an unwanted tone from appearing in the audio output, the software demodulator is tuned slightly off frequency, but that means interference at the image frequency can cause problems because of the imperfect image rejection. The dynamic range depends on the sound card performance as well as the RF hardware. Some newer SDRs include an integrated audio codec or other type of A/D and D/A converters optimized for the application so that the PC's sound card is not needed.

DIGITIZING AT IF

Another option for implementing software-defined radios involves performing the analog-digital conversion at an intermediate frequency. **Figure 8.4** shows such a design. In the receiver, placing the A/D converter after a crystal IF filter improves the *blocking dynamic range* (BDR) for interfering signals that fall outside the crystal filter bandwidth. BDR is the ratio, expressed in dB, between the noise level (normally assuming a 500 Hz bandwidth) and an interfering signal strong enough to cause 1 dB gain reduction of the desired signal. (See the **Receiving** chapter.) As shown, the downconversion to I/Q format still uses lower-speed A/D converters, but

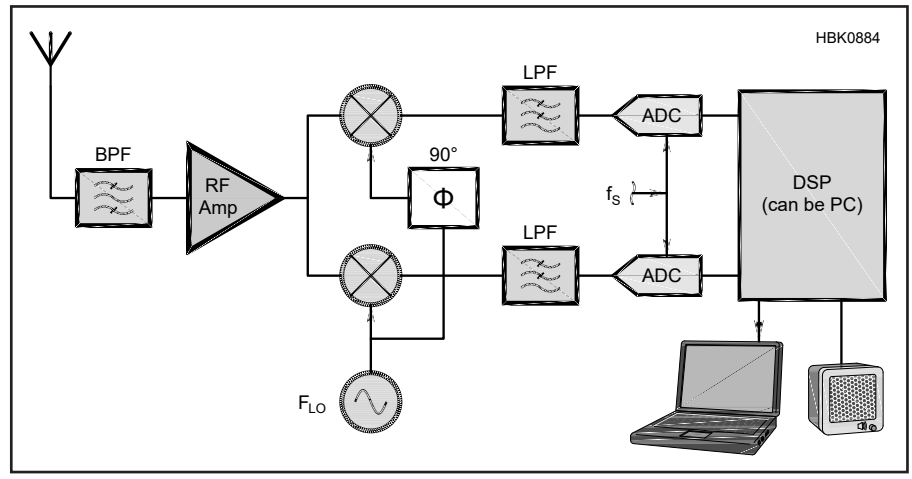


Figure 8.3 — Direct-to-baseband SDR receiver architecture.

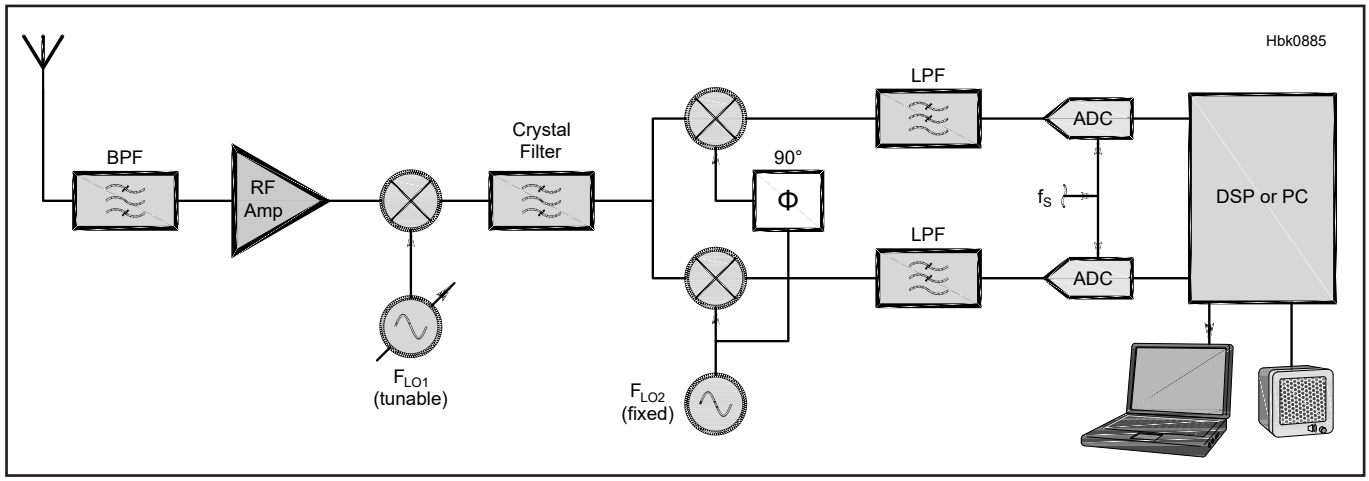


Figure 8.4 — Hybrid superhet/DSP SDR receiver architecture.

often the signal is actually at a low IF, say, 15 kHz or so. This allows an SSB-bandwidth signal to be contained within the 20 kHz bandwidth of a typical audio codec and avoids errors due to dc offsets in the signal path. With careful design, a receiver with such an architecture can achieve 140 dB or more of BDR (if there are no other limiting factors such as LO phase noise). The third-order dynamic range is similar to that achieved with a conventional analog architecture since the circuitry up to the crystal filter, including amplifiers and mixer(s) is the same. The Elecraft K3 and Icom IC-7851 transceivers used this architecture and achieved very high receiver performance.

An advantage of the IF-based approach compared to directly sampling the RF frequency is that the ADC does not have to run at such a high sample rate. In fact, because the crystal filter acts as a high-performance, narrow-bandwidth anti-aliasing filter, *undersampling* is possible if the A/D converter has sufficient sampling bandwidth (ADCs intended for audio applications generally do not). With bandwidths of a few kHz or less, sample rates in the tens of kHz can be used even though the center frequency of the IF signal is much higher, so long as the ADC's sample-and-hold circuit has sufficient bandwidth.

Another example of an IF-sampling SDR is the family of "SDR dongles" based on chips designed for satellite TV applications such as the RTL2832U manufactured by Realtek. Several different manufacturers produce such devices. The RTL2832U IC includes a pair of 8-bit A/D converters operating at a 28.8 Msps sample rate and a programmable DSP system including digital low-pass filters, demodulation, decoding, and other functions. The outputs of the A/D converters and the outputs of the digital processor are accessible via the chip's integrated USB port. The RTL2832U device is usually preceded by a either a superheterodyne (superhet) or quadrature zero-IF tuner. With a superhet receiver, only one of the on-chip ADCs is used, and the RTL2832U does the quadrature downconversion digitally. Bandwidth-limited IFs up to 30+ MHz can be undersampled. With a quadrature zero-IF downconverting tuner, both ADCs are used. You can find a more detailed description of implementation options using this IC at **www.pa3fwm.nl/technotes/tn20.html**. A typical design (of many available), developed by RTL-SDR.com is shown in **Figure 8.5C**.

Some systems, especially in VHF/UHF/microwave applications such as cellular base stations which deal with signals that are tens of MHz wide and in the GHz frequency range, often use the approach shown in **Figure 8.6**. Here, a high-speed, high-bandwidth ADC digitizes an IF typically in the hundreds of

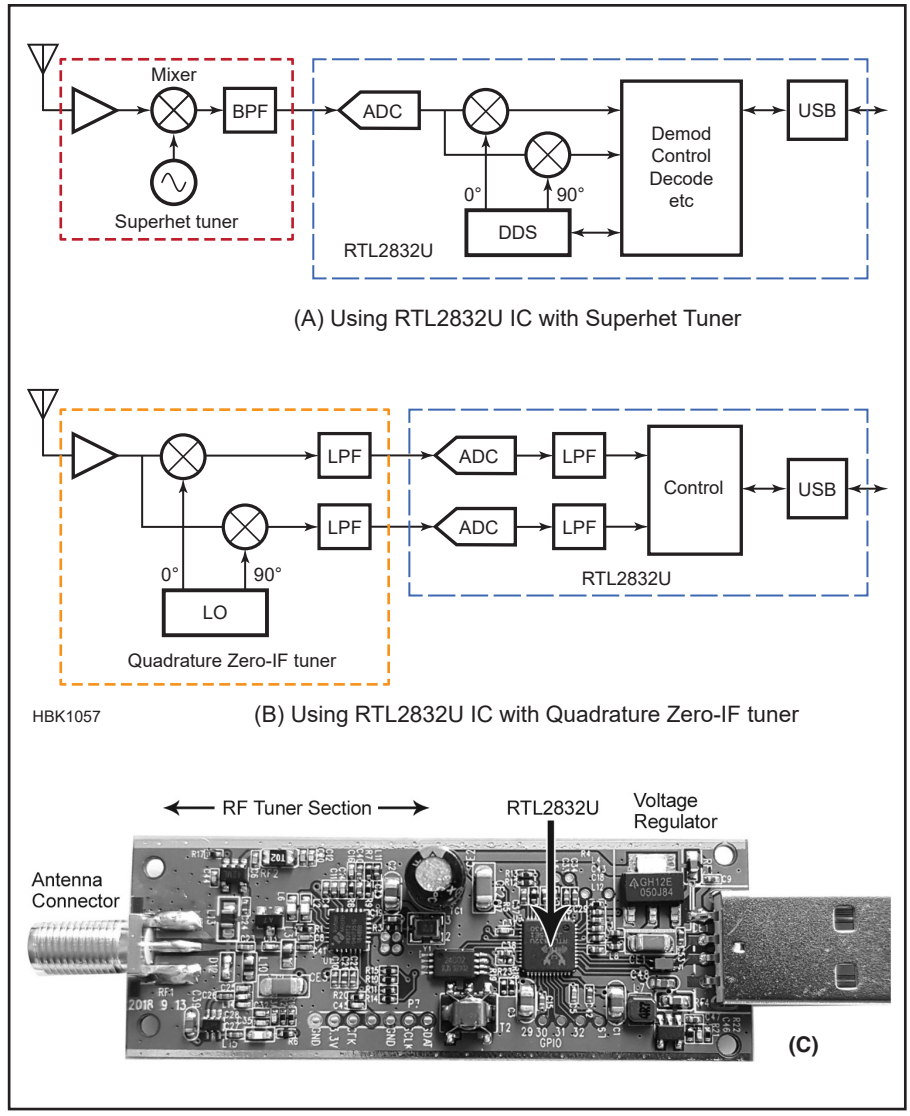


Figure 8.5 — Low cost SDR receiver using RTL2832U IC with a superhet tuner (A) and quadrature zero-IF tuner (B). An RTL-SDR dongle (C) plugs into a PC USB port. (photo courtesy of RTL-SDR.com)

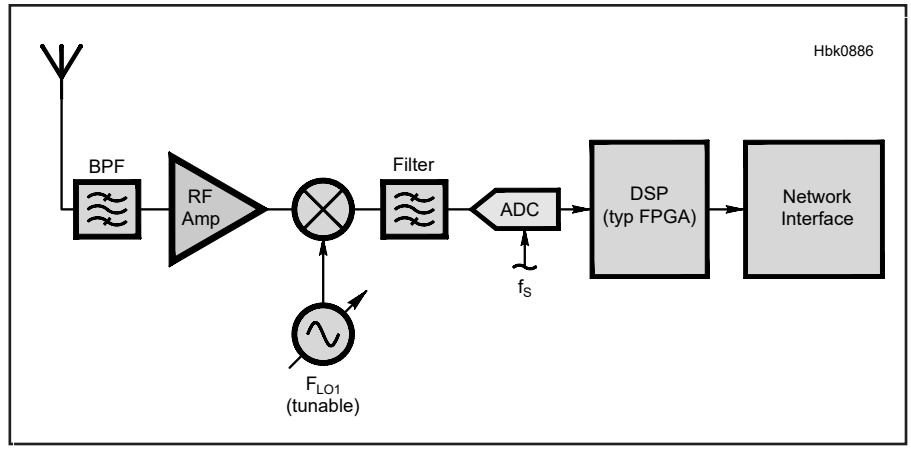


Figure 8.6 — High-IF sampling SDR receiver architecture.

MHz. The output data rate of such a converter is too high to be handled by a PC or even a low-cost programmable DSP microprocessor. The conversion from the original signal to baseband I/Q components, and then the demodulation to data streams for the network is done in dedicated digital hardware blocks.

DIRECT RF DIGITIZING

The ultimate SDR architecture is to convert between the analog and digital domains right at the frequency to be transmitted or received or convert a wide range of frequencies and do all filtering in the digital domain. The receive path of such a design is shown in **Figure 8.7**. In this receiver, the only remaining analog components in the signal chain are a wideband anti-aliasing filter similar to a preselector and an amplifier to improve the noise figure of the ADC if necessary. The FlexRadio Flex-6700, Elecraft K4, and Icom IC-7610 are examples of transceivers that use this basic architecture. The local oscillator, mixer, IF filters, AGC, demodulators and other circuitry are all replaced by digital hardware and software. The digital/software implementations of these functions are perfectly stable with time and temperature, and need no adjustments. The K4 DSP FPGA actually delivers three outputs: a wideband path for the panadapter display, and two 48-kHz IF outputs (for "dual-watch" capability) where the final high-performance filtering and demodulation occur.

HYBRID SDR/ANALOG RECEIVERS

It has only been recently that low-cost high-speed ADCs have become available with specifications good enough to allow reasonable performance in an RF-sampling communications receiver. Today it is possible to achieve blocking dynamic range (BDR) of 130 dB. Hybrid SDR/analog receivers can provide even higher performance.

The highest performance receivers currently available for HF applications use both analog and DSP processing, with some amplification and filtering and analog downconversion to an intermediate frequency (and sometimes two downconversions) so that additional selectivity can be applied before the signal reaches the ADC. This approach is used in the Elecraft K3 and Icom IC-7851 transceivers.

Figure 8.8 shows two examples of hybrid receiver chains. The Yaesu FTDX101 series transceivers use two receive signal chains. One is a direct RF sampler which is used only to provide a fast-responding digital band scope display. The other path which ultimately delivers audio to the user is a hybrid design, with a 9 MHz IF and optional narrow crystal filters, offering very high receiver performance.

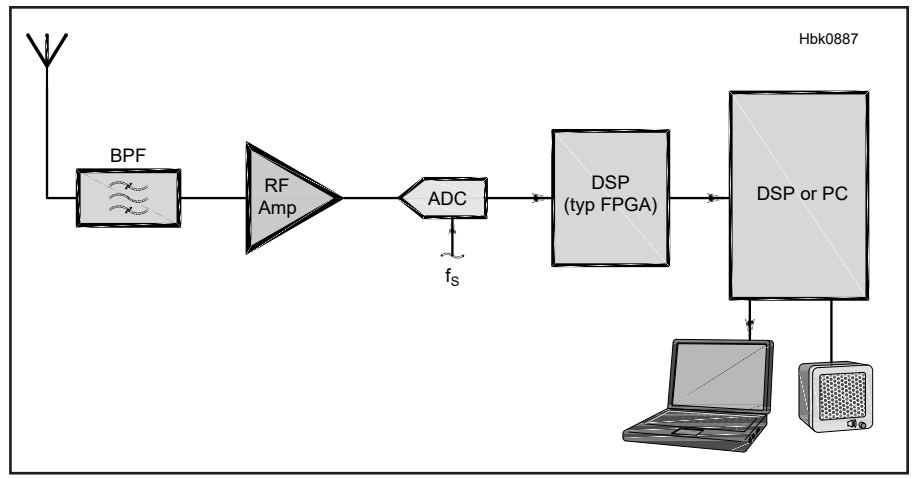


Figure 8.7 — Direct RF-sampling DSP SDR receiver architecture.

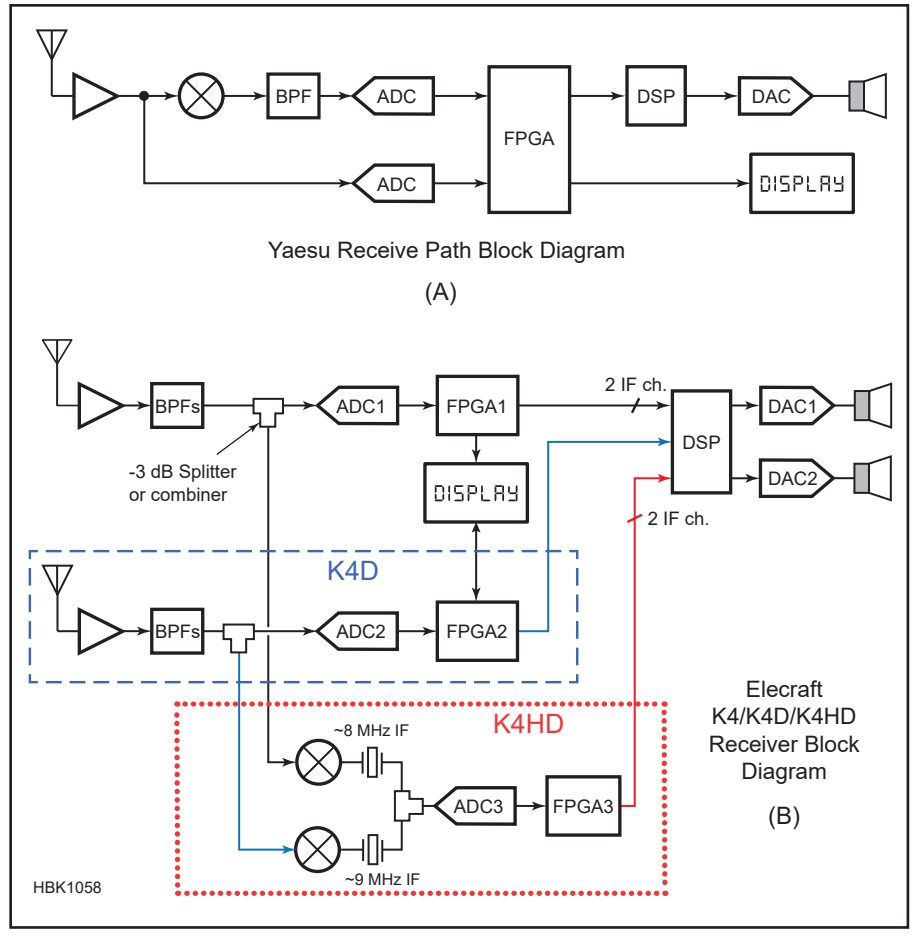


Figure 8.8 — Receiver architectures of current high-performance transceivers — the Yaesu FTDX101 series (A) and Elecraft K4 (B). K4D and K4HD refer to model numbers, see text. (Figure 8.8B courtesy of Elecraft)

The Elecraft K4 platform has three versions. The K4D adds a second direct-sampling receiver to the basic K4 to provide diversity or dual-receive capability. The K4HD, intended for use in the most demanding environments, adds a two-channel *hybrid* signal chain to the K4D. Each channel includes an analog downconversion, one channel to an 8 MHz IF and one to a 9 MHz IF. These two analog IF signals are filtered by a bank of narrowband crystal filters, then combined and digitized by one A/D converter. The FPGA following this ADC delivers two 48 kHz digital IF signals to the DSP, which does the final filtering and demodulation as in the base model.

It is worth noting here that while huge BDR numbers can be measured in the laboratory, the performance achieved in a real-world environment with a receiver connected to an antenna is quite different. Often local noise sources raise the noise floor such that the receiver's full BDR cannot actually be utilized and other specifications become more important. In the real world, RF-sampling SDRs can often provide performance indistinguishable to conventional all-analog and hybrid receivers.

Third-order dynamic range (3IMD_DR or IP_3) is not a meaningful specification for this type of radio because it is based on the behavior of analog circuits which degrade gradually as the signal level increases. Calculation of 3IMD DR assumes that distortion products increase 3 dB for each 1 dB increase in signal level, which is not always true for an ADC. The level of the distortion products in an ADC tends to be more-or-less independent of signal level until the signal peak exceeds the ADC's full-scale input, at which point the distortion increases dramatically. It is important to read the data sheet carefully and note the test conditions for the distortion measurements.

There are definite advantages to sampling at RF. For one thing, it saves a lot of analog circuitry. Even though a high-speed ADC is more expensive than an audio converter, the radio may be end up being cheaper to build because of the reduced component count and fewer adjustments. Performance is improved in some areas. For example, image rejection is no longer a worry, as long as the anti-aliasing filter is doing its job. (See the **Receiving** chapter.) The dynamic range of an SDR theoretically does not depend on signal spacing — close-in dynamic range is often better than with a conventional architecture that uses a wide IF filter. With no crystal filters in the signal chain, the entire system has a completely linear phase response, which can improve the quality of both analog and digital signals after demodulation.

The biggest challenge with RF sampling is what to do with the torrent of high-speed data coming out of the receiver's ADC and how to generate transmit data fast enough to keep up with the DAC sample rate. To cover 0 to 54 MHz without aliasing requires a sample rate of at least 120 or 130 MHz, and commercial products typically operate the ADC at sample rates well over 200 MHz. That is much faster than a typical microprocessor or programmable DSP can handle. The local oscillator, mixer, and decimator or interpolator must be implemented in digital hardware so that the DSP can send and receive data at a more-reasonable sample rate. *Digital downconverters* (DDC) perform those functions and output a lower-sample-rate digital I/Q signal to the DSP. Stand-alone DDC ICs were available in the past, but the function is now usually integrated with the A/D converter. It is also possible to implement a DDC in a *field-programmable gate array* or FPGA. (See the **Transceiver Design Topics** chapter.) *Digital upconverters* (DUC) do the same conversion in reverse for the transmitter and are available integrated with the D/A converter or can be implemented in an FPGA. Some commercial integrated DDC/DAC products even include the capability to encode several digital modulation formats such as GMSK, QPSK and $\pi/4$ DQPSK. In an attempt to simplify the interface to the digital domain, many high-speed converters now use a standardized serial interface specification called JESD204B, capable of handling up to 12Gb/s. Code to implement this interface on the digital FPGA is readily available.

Some designers have been successful in repurposing a graphics processor (GPU) for this application, and some GPU manufacturers now offer FFT libraries to assist in the design process.

8.2.2 Advantages and Limitations of DSP and SDR

Digital signal processing has the reputation of being more complicated than the analog circuitry that it replaces. In reality, once the analog signal has been converted into the digital domain, complicated functions can be implemented in software much more simply than would be possible with analog components. For example, the traditional "phasing" method of generating an SSB signal without an expensive crystal filter requires various mixers, oscillators, filters and a wide-band audio-frequency phase-shift network built with a network of high-precision resistors and capacitors. To implement the same function in a DSP system requires adding one additional subroutine to the software program — and no additional hardware.

Many features that are straightforward with DSP techniques are difficult or impractical to implement with analog circuitry. A few examples drawn just from the communications field are imageless mixing, noise reduction, OFDM (orthogonal frequency division multiplexing) modulation and adaptive channel equalization. Digital signals can have much more dynamic range than analog signals, limited only by the number of bits used to represent the signal. For example, it is easy to add an extra 20 or 30 dB of headroom to the intermediate signal processing stages to ensure that there is no measurable degradation of the signal in a filter, for example. It would be difficult or impossible to add that much dynamic range with analog circuitry. Replacing analog circuitry with software algorithms eliminates the problems of nonlinearity and drift of component values with time and temperature. The programmable nature of most DSP systems means you can make the equivalent of circuit modifications without having to unsolder any components.

Despite its many advantages, we don't mean to imply that DSP is best in all situations. High-power and very high-frequency signals are still the domain of analog circuitry. Where simplicity and low power consumption are primary goals, a DSP solution may not be the best choice. For example, a simple CW receiver that draws a few milliamps from the power supply can be built with two or three analog ICs and a handful of discrete components. We are still a long way from that kind of low power radio using SDR.

In many high-performance systems, the performance of the analog-to-digital converter and digital-to-analog converter are the limiting factors. That is why, even with the latest generation of affordable ADC technology, it is still possible to obtain better blocking dynamic range in an HF receiver using a hybrid analog-digital system rather than going all-digital by routing the RF input directly to an ADC. This may change as A/D converter technology continues to evolve and performance rises.

8.3 Analog-Digital Conversion

Analog-to-digital (and digital-to-analog) conversion consists of taking data in one form, such as digital binary data or an analog ac RF waveform, and creating an equivalent representation of it in the opposite domain. Converters that create a digital representation of analog voltages or currents are called *analog-to-digital converters* (ADC), *analog/digital converters, A/D converters* or *A-to-D converters*. Similarly, converters that create analog voltages or currents from digital quantities are called *digital-to-analog converters* (DAC), *digital/analog converters, D/A converters* or *D-to-A converters*. The word "conversion" in this first section on the properties of converting information between the analog and digital domains will apply equally to analog-to-digital or digital-to-analog conversion.

Converters are typically implemented as integrated circuits that include all of the necessary interfaces and sub-systems to perform the entire conversion process. In some cases, the converter may be integrated on the same chip as other analog functions (such as buffer amplifiers, reference voltages, etc.) or digital functions (such as digital filters). Schematic symbols for ADCs and DACs are shown in **Figure 8.9**.

This section defines the basic elements of analog-digital conversion along with an overview of different types of converters and their key specifications and behaviors. The following section discusses the use of converters for DSP and SDR functions which are the most demanding application in radio.

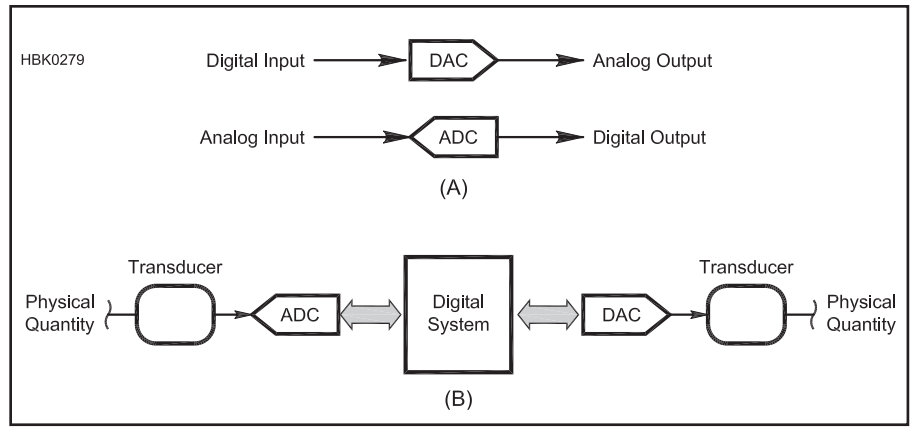


Figure 8.9 — Schematic symbols (A) for digital-to-analog converters (DAC) and analog-to-digital converters (ADC). The general block diagram of a system (B) that digitizes an analog signal, operates on it as digital data, then converts it back to analog form.

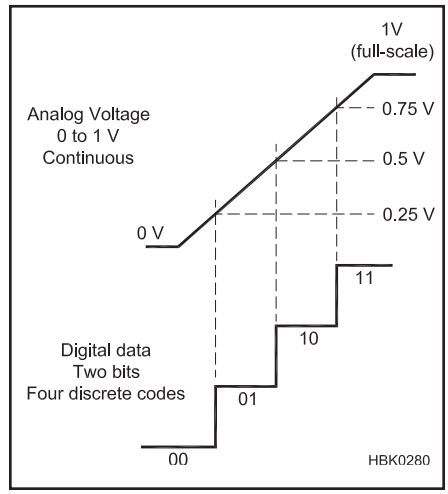


Figure 8.10 — The analog voltage varies continuously between 0 and 1 V, but the two-bit digital system only has four values to represent the analog voltage, so representation of the analog voltage is coarse.

8.3.1 Basic Conversion Metrics

Figure 8.10 shows two different representations of the same physical phenomenon; an analog voltage changing from 0 to 1 V. In the analog world, the voltage is continuous and can be represented by any real number between 0 and 1. In the digital world, the number of possible values that can represent any phenomenon is limited by the number of bits contained in each value.

In Figure 8.10, there are only four two-bit digital values 00, 01, 10, and 11, each corresponding to the analog voltage being within a specific range of voltages. If the analog voltage is anywhere in the range 0 to 0.25 V, the digital value representing the analog voltage will be 00, no matter whether the voltage is 0.0001 or 0.24999 V. The range 0.25 to 0.5 V is represented by the digital value 01, and so forth.

The process of converting a continuous range of possible values to a limited number of discrete values is called *digitization* and each discrete value is called a *code* or a *quantization code*. If the code is a binary number, the number of possible codes that can represent an analog quantity is 2^N, where N is the number of bits in the code. A two-bit number can have four codes as shown in Figure 8.10, a four-bit number can have sixteen codes, an eight-bit number 256 codes, and so forth. Assuming that the smallest change in code values is one bit, that value is called the *least significant bit* (LSB) regardless of its position in the format used to represent digital numbers.

Binary-coded-decimal (BCD) is a code in which groups of four bits represent individual decimal values of 0 – 9. In the *hexadecimal* code, groups of four bits represent decimal values of 0 – 15 and are represented by the digits 0 – 9 followed by the letters A through F. Other types of codes that may be encountered include *Gray code* and *octal*.

RESOLUTION AND RANGE

The *resolution* or *step size* of the conversion is the smallest change in the analog value that the conversion can represent. The *range* of the conversion is the total span of analog values that the conversion can process. The maximum value in the range is called the *full-scale* (F.S.) value. In Figure 8.10, the conversion range is 1 V. The resolution of the conversion is

$$resolution = \frac{range}{2^N}$$

In Figure 8.10, the conversion resolution is ¼ × 1 V = 0.25 V in the figure. If each code had four bits instead, it would have a resolution of ½⁴ × 1 V = 0.0625 V. Conversion range does not necessarily have zero as one end point. For example, a conversion range of 5 V may span 0 to 5 V, –5 to 0 V, –2.5 to +2.5 V, and so on.

Analog-digital conversion can have a range that is *unipolar* or *bipolar*. Unipolar means a conversion range that is entirely positive (or negative), usually referring to voltage. Bipolar means the range can take on both positive and negative values.

Because each code represents a range of possible analog values, the limited number of available codes creates *quantization error*. This is the maximum variation in analog values that can be represented by the same code. In Figure 8.10, any value from 0.25 through 0.50 V could be represented by the same code: 01. The quantization error in this case is 0.25 V.

Resolution can also be defined by the number of bits in the conversion. The higher the number of bits, the smaller the resolution as demonstrated above. Since many converters have variable ranges set by external components or voltages, referring to percent resolution or as a number of bits is preferred. The conversions between percent resolution and number of bits are as follows:

$$\%\,\text{resolution} = \frac{1}{2^N} \times 100\%$$

and

$$N = \frac{\log\left(\frac{100\%}{\%\,resolution}\right)}{\log 2}$$

Quantization error can also be specified as a number of least significant bits (LSB) where each bit is equivalent to the conversion's resolution.

ACCURACY

The number of bits of an A/D converter's resolution does not equate to the accuracy of the converter. A companion to resolution, *accuracy* refers to the ability of the converter to either assign the correct code to an analog value or create the true analog value from a specific code. As with resolution, it is most convenient to refer to accuracy as either a percentage of full scale or in bits. *Full-scale error* is the maximum deviation of the code's value or the analog quantity's value as a percentage of the full scale value. If a converter's accuracy is given as 0.02% F.S. and the conversion range is 5 V, the conversion can be in error by as much as 0.02% × 5 V = 1 mV from the correct or expected value. Most A/D converters include provision for user adjustment to calibrate the converter's full-scale range. Some A/D converters include a reference voltage source internal to the device, while others require an external reference. External references can often provide higher accuracy and stability over temperature than internal references. *Offset* has the same meaning in conversion as it does in analog electronics — a consistent shift in the value of the conversion from the ideal value. It can also be adjusted if necessary by the user.

Linearity error represents the maximum deviation of the code transition points from the ideal code transition points after adjusting for the full scale and errors. This is also called *integral nonlinearity* (INL). In the converter of Figure 8.10, ideal step transitions occur at 0.25 V intervals. If the linearity error for the conversion was given as 0.05% F.S., any actual step size could be in error by as much as 0.05% × 5 V = 2.5 mV due to the transition to the next step being at the wrong point. *Differential nonlinearity* is a measure of how much any two adjacent step sizes deviate from the ideal step size. Errors can be represented as a number of bits, usually assumed to be least significant bits, or LSB, with one bit representing the same range as the conversion resolution. A typical A/D converter may specify its INL or DNL error as ± 0.5 LSB (least-significant bits).

CONVERSION RATE AND BANDWIDTH

Another important parameter of the conversion is the *conversion rate* or its reciprocal, *conversion speed*. A digital code that represents an analog value at a specific time is called a *sample*, so conversion rate, f_S, is specified in *samples per second* (sps) and conversion speed as some period of time per sample, such as 1 msec. Because of the mechanics by which conversion is performed, conversion speed can also be specified as a number of cycles of clock signal used by the digital system performing the conversion. Conversion rate then depends on the frequency of the clock.

According to the *Nyquist Sampling Theorem*, in order to accurately represent the input signal, a conversion must occur at a rate at least twice the bandwidth of the analog signal. This minimum rate is the *Nyquist rate* and the maximum frequency allowed in the analog signal is the *Nyquist frequency*. In this way, the converter *bandwidth* is limited to one-half the conversion rate.

Referring to the process of converting analog signals to digital samples, if a lower rate is used, called *undersampling*, false signals called *aliases* will be created in the digital representation of the input signal at frequencies related to the difference between the Nyquist sampling rate and f_S. This is called *aliasing*. Sampling faster than the Nyquist rate is called *oversampling* and can be used to advantage by following the converter with ah digital filter.

Because conversions occur at some maximum rate, there is always the possibility of signals greater than the Nyquist frequency being present in an analog signal undergoing conversion or that is being created from digital values. These signals would result in aliases and must be removed by *band-limiting filters* that remove them prior to conversion. The mechanics of the sampling process are discussed later in this and following chapters as they apply to specific functions.

DISTORTION AND NOISE

Distortion and noise in a conversion are characterized by several parameters all related to linearity and accuracy. THD+N (Total Harmonic Distortion + Noise) is a measure of how much distortion and noise is introduced by the conversion. THD+N can be specified in percent or in dB. Smaller values are better. SINAD (Signal to Noise and Distortion Ratio) is related to THD+N, generally specified along with a desired signal level to show what signal level is required to achieve a certain level of SINAD or the highest signal level at which a certain level of SINAD can be maintained. (See the Analog Devices application note MT-003 by Kester in the Bibliography for further information about SINAD and other noise metrics.)

ADC OVERLOAD

When *overloaded*, A/D converters behave similarly to amplifiers driven into saturation or clipping. Consider the output of an ADC for various inputs as shown in **Table 8.1**. (This example assumes a 16-bit resolution and offset binary code). "–FS" and "+FS" are the negative and positive Full-Scale input voltage levels of the ADC.

This table shows the ADC cannot distinguish between an input signal exactly at the top (or bottom) of its range and a signal that exceeds the input range. In fact, the ADC cannot tell if the input range has been slightly exceeded or grossly exceeded. This is an example of *hard clipping*.

Fortunately, most ADCs have an output signal to indicate the input range has been exceeded and the converter is in *overload* or *overflow/underflow* mode. This allows the system designer to take appropriate action — usually reducing the gain of an amplifier or increasing the attenuation of any stages ahead of the ADC.

In designing an SDR system, careful attention must be given to the peak signal reaching the A/D converter. Long periods of overload can create artifacts in the DSP after the converter. RF signal levels are usually expressed

Table 8.1
Output of an ADC for Various Inputs

Input voltage	*Output code (binary)*	*Output code (decimal)*
At or below -FS	0000 0000 0000 0000	0
Just above -FS	0000 0000 0000 0001	1
Middle of range	1000 0000 0000 0000	32768
Just below +FS	1111 1111 1111 1110	65534
At +FS-1 LSB	1111 1111 1111 1111	65535
At or above +FS	1111 1111 1111 1111	65535

in terms of power, rather than voltage, and often a signal is assumed to be a single sine wave. In a wideband receiver, the input signal is really the sum of many uncorrelated signals in the passband of the system, most of which are not even close to sinusoids.

In the **Radio Fundamentals** chapter, it is explained that ac waveforms are best characterized by their RMS (root-mean-square) values, and that a sine wave has an RMS value of 0.707 times the peak value. That means the peak value of the ac signal is 1.414 times the RMS value. This ratio is the *crest factor*. In terms of voltage, the 1.414 ratio is equal to 20 log (1.414), which is 3.01 dB. In terms of power, the 1.414 voltage ratio is a power ratio of $(1.414)^2$, which is 10 log (2), which is also 3.01 dB.

Other waveforms have different crest factors. A triangle wave has a crest factor of 1.732, or 4.77 dB. Signals that include multiple individual signals within the passband of the A/D converter tend to look almost like noise, which has a high crest factor (true Gaussian noise has an infinite crest factor!). In addition, some environments have locally-generated noise from sources such as power-line leakage or arcing, electric fences, or nearby switch-mode power supplies, not to mention atmospheric noise. Any of these can cause an A/D converter to overload.

Typical high-speed A/D converters are high-input-impedance devices and require a dc bias or offset on the input waveform when operated from a single-polarity power source. The input signal from preceding stages is usually transformer-coupled to the A/D converter input, biased from a signal generated by the A/D converter. Converter manufacturers usually give excellent guidance on how to couple the signal into the converter for optimal performance. Most ADCs utilize an input voltage range on the order of 2 V_{P-P}. It is good practice to plan for several dB of overhead in the signal level reaching the A/D converter. Allowing for 10 dB of headroom means that the expected signal levels should only use about ⅓ of the A/D converter's input voltage range.

8.3.2 Analog-to-Digital Converters

There are a number of methods by which the conversion from an analog quantity to a set of digital samples can be performed. Each method or *architecture* has its strong points — simplicity, speed, resolution, accuracy — all affect the decision of which method to use for a particular application. In order to pick the right type of ADC, it is important to decide which of these criteria most strongly affect the performance of your application.

The following sections present ADC architectures used in amateur radio from the fastest to progressively slower conversion. For a more complete review of ADC architectures, see Kester's article "Which ADC Architecture Is Right for Your Application?" listed in the Reference section.

FLASH CONVERTER

The simplest type of ADC is the *flash converter*, shown in **Figure 8.11**. It continually generates a digital representation of the analog signal at its input. The flash converter uses an array of comparators that compare the amplitude of the input signal to a set of reference voltages. There is one reference voltage for each step.

The outputs of the comparator array represent a digital value in which each bit indicates whether the input signal is greater (1) or less than (0) the reference voltage for that comparator. A digital logic *priority encoder* then converts the array of bits into a digital output code. Each successive conversion is available as quickly as the comparators can respond and the priority encoder can create the output code. Flash converters are generally used for applications in which high speed is more important than bits of resolution.

Flash converters are the fastest of all ADCs (conversion speeds can be in the nanosecond

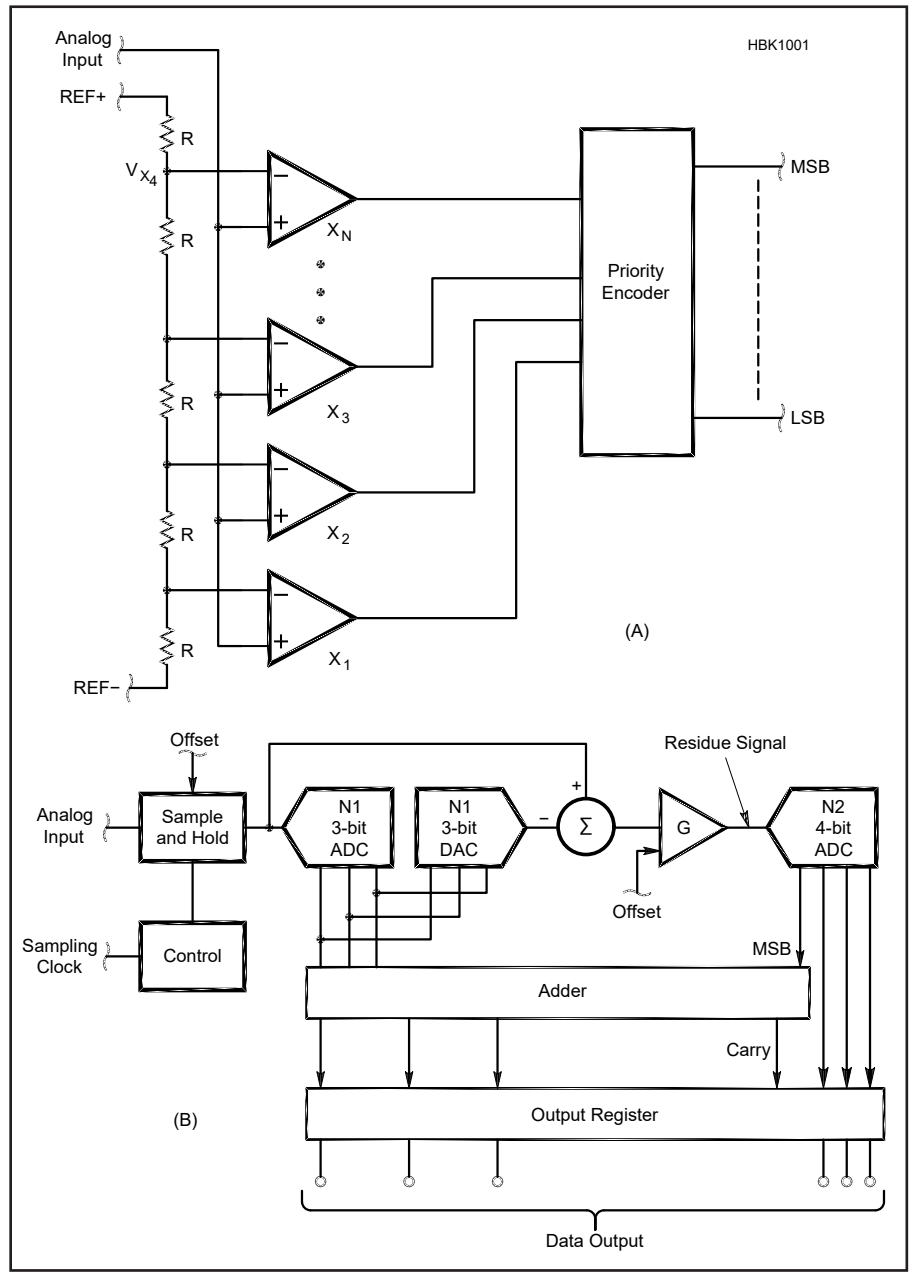


Figure 8.11 — The comparators in a flash converter (A) switch continuously to follow the input signal. The encoder converts the array of comparator outputs to a single digital word. Flash converters (N1 and N2) form the basis of the pipelined converter (B) that is used at the very high sample rates needed for SDR applications.

range, equivalent to sampling rates in the gigasamples/second) but do not have high resolution because of the number of comparators and reference voltages required. For example, an 8-bit flash converter requires 255 comparators, while a 12-bit version would require 4095 comparators.

One specialized variant of the flash converter is the *bar-graph display driver*. Such devices accept an analog input and deliver output signals capable of directly driving an LED display rather than a binary digital output word. These devices are commonly used to replace meters as front panel displays for signals that represent relatively slow-varying parameters such as transmitter output power. The LM3914 from Texas Instruments is one such device.

PIPELINED CONVERTER

A *pipelined converter* uses two or more stages of flash converters as shown in Figure 8.11B. The output of the first stage flash converter (N1) is changed back to an analog signal by a DAC and the difference between the DAC output and input signal is called the *residue signal*. The residue signal is amplified then digitized by a second flash converter (N2). The digital outputs of the two flash converters are combined by an adder and a register to create the overall converter output word. The digital outputs of multiple stages can be combined to yield more bits than a single-stage flash converter would produce. Pipelined converters are discussed in additional detail in the section Data Converters for SDR and DSP later in this chapter.

SUCCESSIVE-APPROXIMATION CONVERTER

The *successive-approximation* A/D converter is one of the most widely-used types of converters. As shown in **Figure 8.12**, it uses a single comparator and DAC (digital-to-analog converter) to arrive at the value of the input voltage by comparing it to successive analog values generated by the DAC. This type of converter offers a good compromise of conversion speed and resolution.

The control logic begins a conversion by setting the output of the DAC to ½ of the conversion range. If the DAC output is greater than the analog input value, the output of the comparator is 0 and the most significant bit of the digital value is set to 0. If the DAC output is less than the analog input, the bit is set to 1. The DAC output then either increases or decreases by ¼ of the range, depending on whether the value of the first comparison was 1 or 0. At this point the input voltage is being compared to either ¼ of the scale or ¾ of the range. One test is made for each bit in digital output code and the result accumulated in a storage register, called the Successive-Approximation Register, which is why such converters are often called "SAR" converters. The process is then repeated, forming a series of approximations, until a test has been made for all bits in the code.

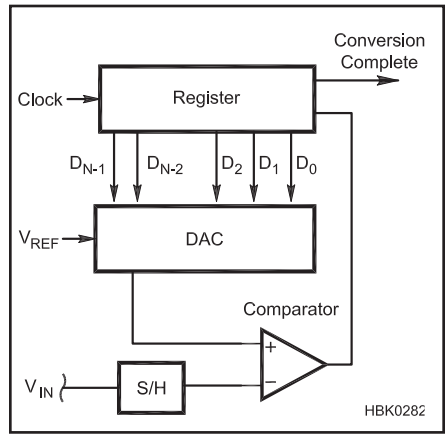


Figure 8.12 — The successive-approximation converter creates a digital word as it varies the DAC signal in order to keep the comparator's noninverting terminal close to the input voltage. A sample-and-hold circuit (S/H) holds the input signal steady while the measurement is being made.

While the digital circuitry to implement the SAR converter is more complex, it is less expensive to build and calibrate than the array of comparators and precision resistors of the flash converter, especially for higher resolutions. Each conversion also takes a known and fixed number of clock cycles. SAR A/D converters are used for speeds up to a few Msps. They are often used with an analog multiplexer in front so that multiple signals in a system can be measured relatively quickly, rather than using a separate converter for every signal.

DUAL-SLOPE INTEGRATING CONVERTERS

The *dual-slope integrating ADC* is shown in **Figure 8.13**. It makes a conversion by integrating the input signal by charging a capac-

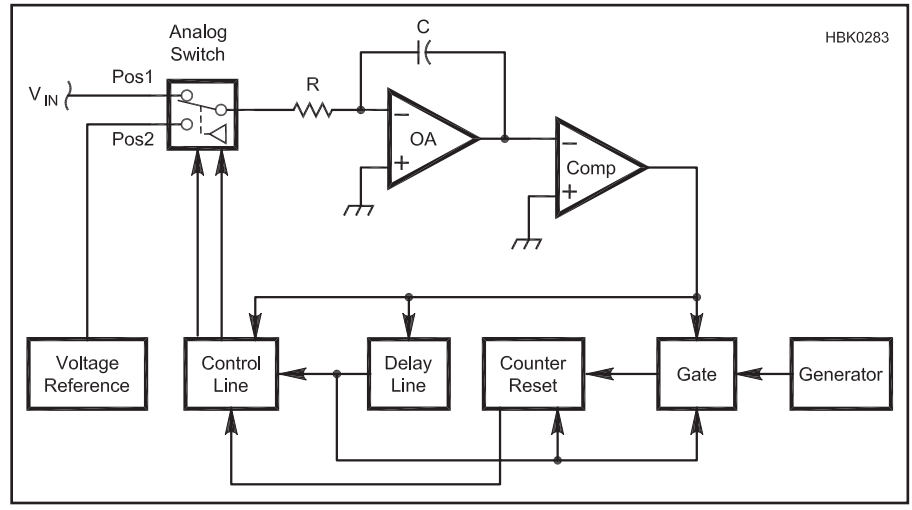


Figure 8.13 — Dual-slope integrating converter. By using a constant-current source to continually charge a capacitor to a known reference voltage then discharge it, the resulting frequency is directly proportional to the resistor value.

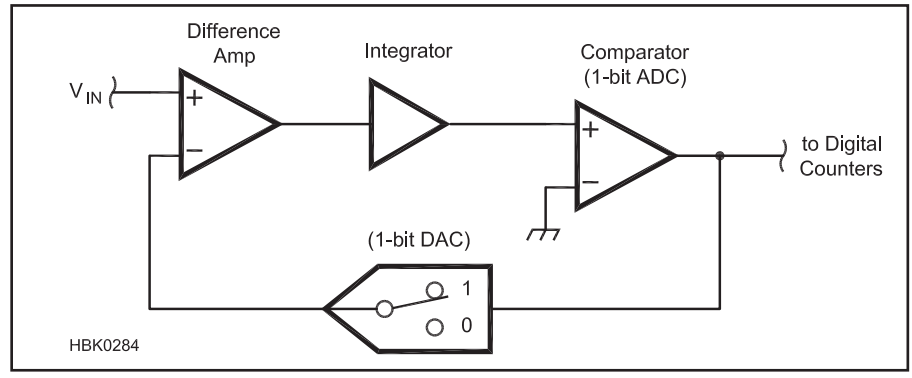


Figure 8.14 — Delta-encoded converter. The 1-bit DAC is operated in such a way that the bit stream out of the comparator represents the value of the input voltage.

itor for a fixed period of time then measuring the time it takes for the capacitor to discharge back to its starting value. The integration period is often set to reject an interfering signal by integrating it over an exact number of cycles. For example, setting the integration period to 100 milliseconds results in 6 full cycles of a 60 Hz sinusoidal interferer and 5 full cycles of a 50 Hz interferer.

Dual-slope ADCs are low-cost and relatively immune to temperature variations. Due to the slow speed of the conversion these converters are generally only used in test instruments such as multimeters or for measurement of slowly-varying dc signals.

DELTA-ENCODED CONVERTERS

Instead of charging and discharging a capacitor from 0 V to the level of the input signal, and then back to 0 V, the *delta-encoded ADC* in **Figure 8.14** continually compares the output of a DAC to the input signal using a comparator. Whenever the signal changes, the DAC is adjusted until its output is equal to the input signal. Digital counter circuits keep track of the DAC value and generate the digital output code.

SIGMA-DELTA CONVERTERS

The *sigma-delta converter* also uses a DAC and a comparator in a feedback loop to generate a digital signal as shown in **Figure 8.15**. An integrator stores the sum of the input signal and the DAC output. (This is "sigma" or sum in the converter's name.) The comparator output indicates whether the integrator output is above or below the reference voltage and that signal is used to adjust the DAC's output so that the integrator output stays close to the reference voltage. (This is the "delta" in the name.) The stream of 0s and 1s from the comparator forms a high-speed digital bit stream that is digitally-filtered to form the output code. Sigma-delta converters are used where high resolution (16 to 24 bits) is required at sampling rates in the ksps range, including very slowly-varying signals and even audio signals.

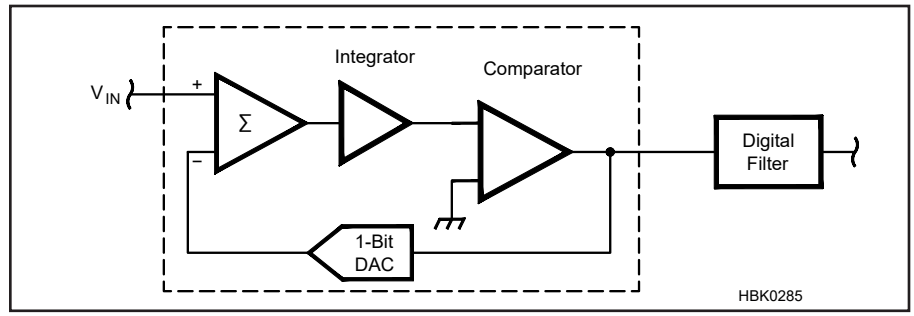


Figure 8.15 — Sigma-delta converter. Similar to the delta-encoded converter (Figure 8.14), the converter runs much faster than the output samples and uses a digital filter to derive the actual output value.

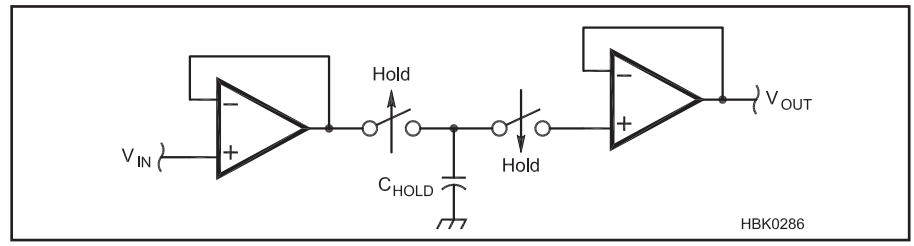


Figure 8.16 — Sample-and-hold (S/H). An input buffer isolates the sampled voltage from the input signal by charging the capacitor C_{HOLD} to that voltage with the input switch closed and the output switch open. When a measurement is being taken, the input switch is open to prevent the input signal from changing the capacitor voltage, and the output switch is closed so that the output buffer can generate a steady voltage at its output.

8.3.3 Analog-to-Digital Converter Subsystems

SAMPLE-AND-HOLD

ADCs that use a sequence of operations to create the digital output code must have a means of holding the input signal steady while the measurements are being made. This function is performed by the circuit of **Figure 8.16**. A high input-impedance buffer drives the external storage capacitor, C_{HOLD}, so that its voltage is the same as the input signal. Another high input-impedance buffer is used to provide a replica of the voltage on C_{HOLD} to the conversion circuitry.

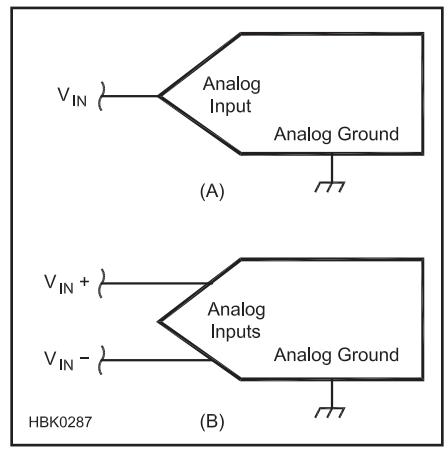


Figure 8.17 — Single-ended ADC inputs have a single active line and a ground or return line (A). Single-ended ADC input are often susceptible to noise and common mode signals or any kind of disturbance on their ground rails. In (B), the differential inputs used help the circuit "ignore" offsets and shifts in the input signal.

When a conversion is started, a digital control signal opens the input switch, closes the output switch, and the capacitor's voltage is measured by the converter. It is important that the capacitor used for C_{HOLD} have low *leakage* so that while the measurement is being made, the voltage stays constant for the length of time required to complete the conversion process. This is of particular important in high-precision conversion.

SINGLE-ENDED AND DIFFERENTIAL INPUTS

The input of most ADCs is *single-ended*, in which the input signal is measured between the input pin and a common ground. Shown in **Figure 8.17A**, this is acceptable for most applications, but if the voltage to be measured is small or is the difference between two non-zero voltages, an ADC with *differential inputs* should be used as in Figure 8.17B. Differential inputs are also useful when measuring current as the voltage across a small resistor in series with the current. In that case, neither side of the resistor is likely to be at ground, so a differential input is very useful. Differential inputs also help avoid the issue of noise contamination as discussed below.

INPUT BUFFERING AND FILTERING

The input impedance of most ADCs is high enough that the source of the input signal is unaffected. However, to protect the ADC input and reduce loading on the input source, an external buffer stage can be used. **Figure 8.18** shows a typical buffer arrangement with clamping diodes to protect against electrostatic discharge (ESD) and an RC-filter to

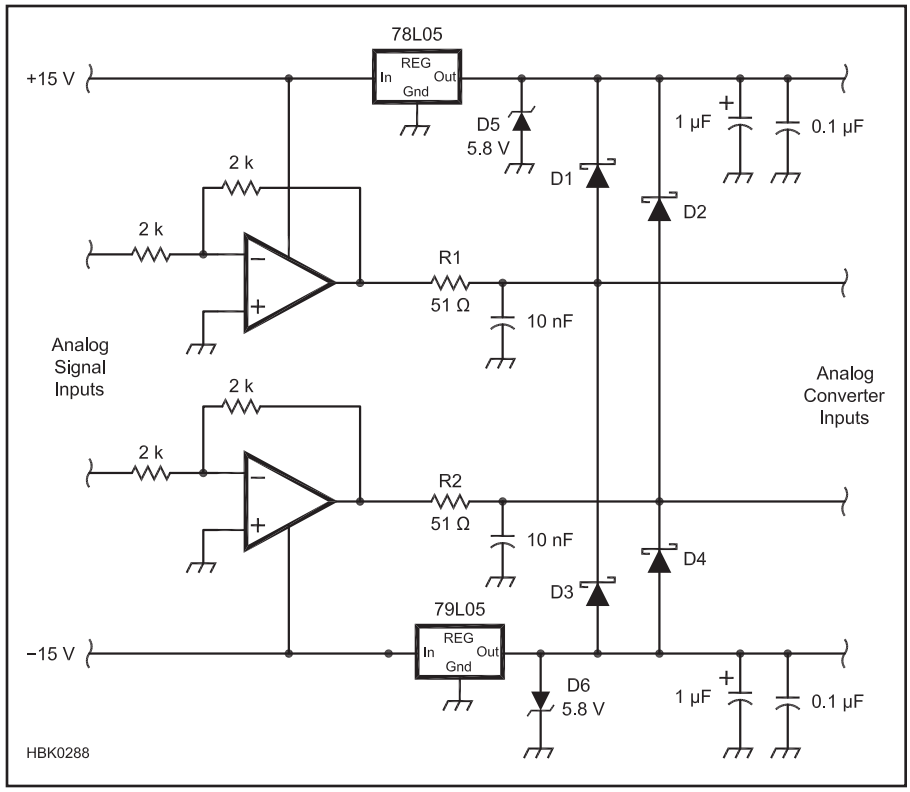


Figure 8.18 — Typical ADC input buffer-filter circuit. Unity-gain voltage followers help isolate the ADC from the input source. RC filters following the buffers act as band-limiting filters to prevent aliasing. Zener diodes are used to clamp the transient voltage and route the energy of transient into the power supply system.

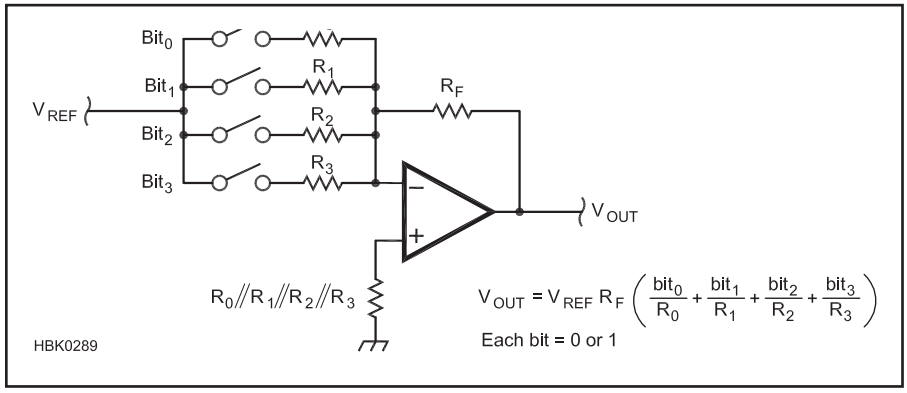


Figure 8.19 — Summing DAC. The output voltage is the inverted, weighted sum of the inputs to each summing resistor at the input. Digital data at the input controls the current into the summing resistors and thus, the output voltage.

prevent RF signals or noise from affecting the input signal. In addition, to attenuate higher-frequency signals that might cause aliases, the input filters can also act as band-limiting filters.

ANALOG AND DIGITAL "GROUND"

By definition, ADCs straddle the analog and digital domain. In principle, the signals remain separate and isolated from each other. In practice, however, voltages and currents from the analog and digital circuitry can be mixed together. This can result in the contamination of an analog signal with components of digital signals, and rarely, vice versa. Mostly, this is a problem when trying to measure small voltages in the presence of large power or RF signals.

The usual problem is that currents from high-speed digital circuitry find their way into analog signal paths and create transients and other artifacts that affect the measurement of the analog signal. Thus, it is important to have separate current paths for the two types of signals. The manufacturer of the converter will provide guidance for the proper use of the converter either in the device's data sheet or as application notes. Look for separate pins on the converter, such as "AGND" (Analog Ground) or "DGND" (Digital Ground) that indicate how the two types of signal return paths should be connected.

8.3.4 Digital-to-Analog Converters (DAC)

Converting a digital value to an analog quantity is considerably simpler than the reverse, but there are several issues primarily associated with DACs that affect the selection of a particular converter.

As each new digital value is converted to analog, the output of the DAC makes an abrupt *step change*. Even if very small, the response of the DAC output does not respond perfectly or instantaneously. *Settling time* is the amount of time required for the DAC's output to stabilize within a certain amount of the final value. It is specified by the manufacturer and can be degraded if the load connected to the DAC is too heavy or if it is highly reactive. Settling time is critical in control applications and also sets the limit on update sampling rate.

Monotonicity is another aspect of characterizing the DAC's accuracy. A DAC is *monotonic* if increasing the digital input value causes the output of the DAC to increase with every step. Because of errors in the internal conversion circuitry, it is possible for there to be some steps that are too small or too large, leading to output values that seem "out of order." These are usually quite small, but if used in a precision control application where the DAC is part of a feedback loop, monotonicity is important because the control system can go unstable.

SUMMING DAC

A *summing DAC*, shown in **Figure 8.19**, is a summing amplifier with all of the inputs connected to a single reference voltage through switches. The digital value to be converted controls which switches are closed. The larger the digital value, the more switches are closed. Higher current causes the summing amplifier's output voltage to be higher, as well.

The input resistors are *binary-weighted* so that the summing network resistors representing the more significant bit values inject more current into the op amp's summing junction. Each resistor differs from its neighboring resistors in the amount of current it injects into the summing node by a factor of two, recreating the effect of each digital bit in the output voltage. At high resolutions, this becomes a problem because of the wide spread in resistor values — a 12-bit DAC

would require a spread of 2048 between the largest and smallest resistor values. Summing DACs are generally only available with low resolution for that reason.

The *current output DAC* functions identically to the summing DAC, but does not have an op amp to convert current in the digitally-controlled resistor network to voltage. It consists only of the resistor network, so an external current-to-voltage circuit (discussed in the previous section on op amps) is required to change the current to a voltage. In some applications, the conversion to voltage is not required or it is already provided by some other circuit.

R-2R LADDER DAC

The summing and current output DACs both used binary weighted resistors to convert the binary digital value into the analog output. The practical limitation of this design is the large difference in value between the smallest and largest resistor. For example, in a 12-bit DAC, the smallest and largest resistors differ by a factor of 2048 (which is 2^{12-1}). This can be difficult to fabricate in an IC since it is difficult to match a wide range of resistors values closely enough to maintain the desired binary-weighted relationship with sufficient accuracy. For DACs with resolutions of 8 bits or more, the R-2R ladder DAC of **Figure 8.20** is a better design. R-2R DACs can operate in either voltage mode or current mode.

The problems of manufacturing are greatly reduced when resistances are fairly close in value. By using the *R-2R ladder* shown in the figure, the same method of varying current injected into an op amp circuit's summing junction can be accomplished with resistors of only two values, R and 2R. In fact, since the op amp feedback resistor is also one of the IC resistors, the absolute value of the resistance R is unimportant, as long as the ratio of R:2R is maintained. This simplifies manufacturing greatly and is an example of IC design being based on ratios instead of absolute values. For this reason, most DACs use the R-2R ladder design and the performance differences lie mostly in their speed and accuracy.

Similar ladder networks can be constructed using capacitors. Such networks are used on many SAR-type A/D converters because in some fabrication processes, capacitors are easier to make than resistors.

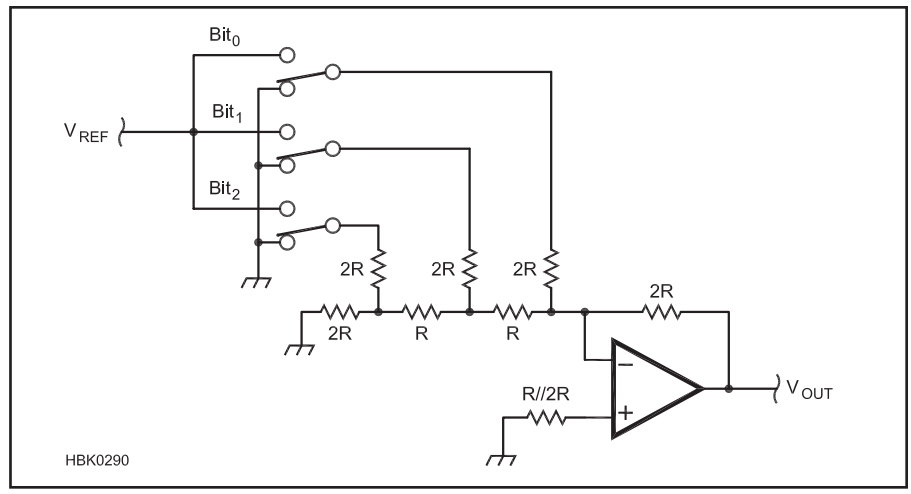


Figure 8.20 — R-2R Ladder DAC. This is the most common form of DAC because all of the resistor values are similar, making it easier to manufacture. The similarity in resistor values also means that there will be less variation of the comparator with temperature and other effects that affect all resistors similarly.

DIGITAL POTENTIOMETERS

Since D/A converters are commonly used to provide digitally-controlled adjustments, a special class of D/A converters has evolved, specifically intended for use as calibration/adjustment devices. These are known as "digital potentiometers." They differ from traditional D/A converters in that they appear as three-terminal variable resistors of known value, and often include non-volatile control registers. They are specified in terms of resistance value, number of steps, and number of devices per package.

8.3.5 Choosing a Converter

From the point of view of performance, choosing a converter, either an ADC or a DAC, comes down to resolution, accuracy and speed. Begin by determining the percent resolution or the dynamic range of the converter. Use the equations in the preceding section to determine the number of bits the converter must have. Select from converters with the next highest number of bits. For example, if you determine that you need 7 bits of resolution, use an 8-bit converter.

Next, consider accuracy. If the converter is needed for test instrumentation, you'll need to perform an *error budget* on the instrument's conversion processes, include errors in the analog circuitry. Once you have calculated percent errors, you can determine the requirements for FS error, offset error, and nonlinearities. If an ADC is going to be used for receiving applications, the spurious-free dynamic range may be more important than high DC precision.

The remaining performance criterion is the speed and rate at which the converter can operate. Conversions should be able to be made at a minimum of twice the bandwidth of the signal you wish to reproduce. If the converter will be running near its maximum rate, be sure that the associated digital interface, supporting circuitry, and software can support the required data rates, too!

Having established the conversion performance requirements, the next step is to consider cost, amount of associated circuitry, power requirements, and so forth. For example, a self-contained ADC is easier to use and takes up less PC board space, but may not be as accurate as one that allows the designer to use an external voltage reference to set the conversion range. Other considerations, such as the nature of the required digital interface, as discussed in the next section, can also affect the selection of the converter.

8.4 Data Converters for SDR and DSP

In this section we will focus more closely on converters for use with DSP and SDR. The first requirement when selecting a DAC or ADC is that it be able to handle the required *sample rate*. For communications-quality voice (300 – 3000 Hz), a sample rate on the order of 8000 samples per second (8 ksps) is adequate and has been used in the public switched telephone network for decades. For CD-quality music (20 – 20,000 Hz), the standard sample rate is 44.1 ksps, while 48 ksps and even 96 or 192 ksps are used for newer audio formats and applications such as professional studio equipment.

In an SDR system intended to use microphone inputs and speaker or headphone outputs, there is a need to convert between analog and digital at audio rates. Fortunately, combination A/D and D/A chips and software packages called *codecs* (short for coder-decoder) are readily available to do this job. Both "audio codecs," with sample rates consistent with high-fidelity audio up to 20 kHz or so, and "voice codecs," intended for voice-quality audio up to 4 or 8 kHz) are available. The switched telephone network originally used 8-bit logarithmic codecs to digitize and reconstruct voiceband audio signals. Logarithmic coding was used to compress the dynamic range of a voice signal to reduce the number of bits to be transported.

Today, linear-coded devices have become preferable since DSP algorithms work on linear-coded data and often perform more efficient compression algorithms to reduce the number of bits needed to transport digitized voice. Since audio codecs are used in PC and other consumer systems they are very low cost (or are free if the PC is part of the SDR system). They have more than adequate performance for SDR applications dealing with relatively narrowband signals, such as CW, SSB, AM, and digital signals such as RTTY, PACTOR, and WSJT modes that use bandwidths comparable to SSB.

Processing wideband RF signals requires data converters with sample rates in the mega-samples per second (Msps) range. Fortunately, manufacturers of digital communications infrastructure equipment for the cellular industry have similar needs, and manufacturers of data converter ICs have developed products that are produced in reasonable volume (which lowers the cost) and can be repurposed for amateur SDRs.

The resolution of a data converter expressed as the number of bits in the data words gives an approximation of the converter's signal-to-noise ratio. For example, an 8-bit ADC can only represent the sampled analog signal as one of $2^8 = 256$ possible numbers. The smallest signal that it can resolve is therefore 1/256 of full scale. In terms of signal-to-noise ratio, the rule of thumb is 6 dB per bit, so an ideal 8-bit converter should have a dynamic range of about 48 dB.

The actual formula is slightly different. In an ideal, error-free ADC, the *quantization error* is up to ±1/2 of one least-significant bit (LSB) of the digital word, or ±1/512 of full scale with 8-bit resolution. Similarly, a DAC can only generate the analog signal to within ±1/2 LSB of the desired value. These unavoidable errors are indistinguishable from noise in a signal-processing system.

It can be shown mathematically that a series of uniformly-distributed random numbers between +0.5 LSB and –0.5 LSB has an RMS value of

$$\frac{LSB}{\sqrt{12}}$$

This represents the quantization noise of an A/D.

The **Radio Fundamentals** chapter shows that a sinusoidal signal of 2 V peak-to-peak (1 V peak) has an RMS voltage equal to 0.707 V, or $\sqrt{2}$ / 2 V. In the case of a full-scale sinusoidal signal applied to an ADC of N bits (2^N LSBs), the RMS signal voltage can be written as $\left(2^{N-1}LSB\right)/\sqrt{2}$.

Combining that information results in the following equation for the signal-to-noise ratio in decibels for an ideal data converter word of width N bits:

$$SNR = 20\log\left(\frac{\frac{2^{N-1}LSB}{\sqrt{2}}}{\frac{LSB}{\sqrt{12}}}\right)$$

which simplifies to

SNR = (6.02N + 1.76) dB

The extra 1.76 dB arises from the fact that noise has a lower RMS value than a sine wave for the same peak-to-peak amplitude. Therefore, with an ideal 8-bit converter, SNR = 49.9 dB. A perfect 16-bit ADC would achieve a 98.1 dB signal-to-noise ratio. Of course, real-world devices are never perfect so actual performance is always somewhat less. Internally-generated noise degrades the SNR, and errors in the linearity of the converter's transfer function give rise to distortion in the form of spurious spectral components and harmonics. (See also "Clocking the RF ADC: Should you worry about jitter or phase noise?" a Texas Instruments application note available for download at **www.ti.com/lit/an/slyt705/slyt705.pdf**.)

Distortion and Noise

Distortion and noise in a conversion are characterized by several parameters all related to linearity and accuracy. THD+N (Total Harmonic Distortion + Noise) is a measure of how much distortion and noise is introduced by the conversion. THD+N can be specified in percent or in dB. Smaller values are better. SINAD (Signal to Noise and Distortion Ratio) is related to THD+N, generally specified along with a desired signal level to show what signal level is required to achieve a certain level of SINAD or the highest signal level at which a certain level of SINAD can be maintained. (See the Analog Devices application note MT-003 by Kester in the Bibliography for further information about SINAD and other noise metrics.)

8.4.1 Using Audio ADCs for SDR

A/D and D/A converters used in audio systems and PC "sound cards" are often repurposed for use in hybrid SDRs. While they may offer as many as 24 bits of resolution, in reality audio A/D converters only provide 110 to 120 dB of SNR and "dynamic range," and the best "24-bit" audio D/A converters deliver 128 dB. According to the formula above, a 24-bit converter should have 1.76 + (6.02 × 24), or 146.2 dB. What happened to the remaining 20 to 40 dB?

Part of the reason that audio converters use 24-bit data words is that the recording and playback formats for many systems (such as Blu-Ray Disc and studio equipment) are designed to accommodate 24-bit data words, even if the lower-order bits are essentially meaningless. For reference, a least-significant bit for a typical 24-bit audio converter with a 5 V p-p signal range is about 100 nanovolts (0.1 microvolts) and it is difficult to keep noise that low in the real world.

Furthermore, the testing of A/D converters intended for use in audio systems is different from how we might specify them for radio applications. For example, the "dynamic range" specification in digital audio systems is tested by measuring the THD+N (Total Harmonic Distortion plus Noise) for a 1 kHz input signal at a level 60 dB below full-scale, converting the number to a positive value, then adding 60. For example, an audio D/A converter that delivers THD+N of –51 dB under this test condition can be specified as having a dynamic range of 111 dB. While this is not the same way that communications systems define dynamic range, the goal of both audio and radio systems is to evolve

towards higher dynamic range, so the performance generally heads in the right direction in both fields.

SIGMA-DELTA CONVERTERS

The noise spectrum of an A/D converter occupies the bandwidth from dc to one-half the sampling frequency, and is more-or-less uniformly spread out over that range. Thus, in any smaller slice of the output spectrum, the noise is much lower compared to the full-scale output of the converter. Thus, if we have an A/D converter sampling at a very high rate relative to the bandwidth of interest, and then add a digital filter after the A/D converter that eliminates the noise components outside the spectrum of interest, we can in principle increase the SNR in the desired bandwidth. An A/D conversion system that operates this way is said to be *oversampling*.

Oversampling by a factor of N improves the SNR in a given bandwidth by the square root of N since the number of samples of the desired signal is multiplied by N, and the noise, which is uncorrelated, increases as the square root of N. This can be simplified to 3 dB SNR improvement for every doubling of the sample rate. Thus, a 4× oversampling rate is equal to adding another bit of resolution to a converter.

Audio converters are an example of oversampled converters, and usually use a technique known as *sigma-delta modulation*. The sigma-delta converter introduced earlier uses a DAC and a comparator in a feedback loop to generate a digital signal. An integrator stores the sum of the input signal and the DAC output. This is "sigma" or sum in the converter's name. The comparator is in essence a one-bit A/D converter, and its output indicates whether the integrator output is above or below the reference voltage. That signal is used to adjust the DAC's output so that the integrator output stays close to the reference voltage. This is the "delta" in the name. The stream of 0s and 1s from the comparator forms a high-speed digital bit stream that is digitally-filtered to form the output code, at a much lower output sample rate.

Most ADCs and DACs used in audio systems use an extreme form of oversampling, where the internal converter may oversample by a rate of 128 or 256 times the desired output rate. For example, the comparator of an audio A/D intended for a 48 ksps output rate may be clocked at 256 × 48 ksps, or 12.288 MHz. In addition, sigma-delta audio converters use a technique called noise shaping to push most of the quantization noise to frequencies above the audio band, thereby reducing it in the audio spectrum. Noise shaping is accomplished by adding more integrators in the feedback loop of the simple sigma-delta modulator to form a "higher-order loop" with a low-pass response for the signal, but a high-pass response for the noise. The digital low-pass filter then removes the noise and produces output data words of higher width at the desired sample rate. The high input sampling rate of the converter also relaxes the requirements for external band-limiting (anti-aliasing) filters on the input signal. **Figure 8.21** illustrates the relationship between loop order, sample rate, and improvements in signal-to-noise ratio.

Since audio A/D converters are the audio input device on many PC-based systems, they are quite inexpensive. The Left- and Right-channel stereo inputs can be repurposed and renamed "I" and "Q" for use in SDR applications. Even if the converter specifications are not exactly ideal they provide adequate performance for a hybrid SDR.

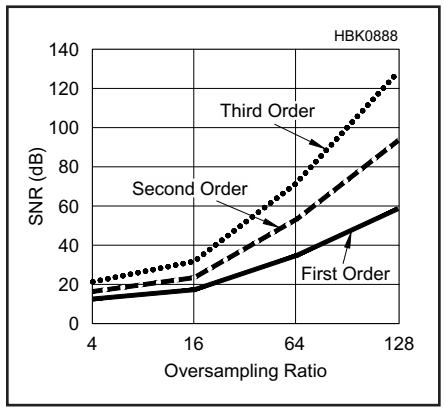


Figure 8.21 — The benefits of noise-shaping as the order of the sigma-delta converter feedback loop is increased.

8.4.2 High-Speed ADCs for SDR

Commercial applications such as cellular base stations have driven the need for high-speed converters capable of sampling RF or high-IF signals directly. These have enabled the development of amateur radio HF transceivers using direct RF sampling. In addition to the usual specifications of linearity, etc., most manufacturers of high-speed data converters intended for use in wideband SDR systems now include a specification for *spurious-free dynamic range* (SFDR). This is the ratio, normally expressed in dB, between a (usually) full-scale sine wave and the worst-case spurious signal. It is a useful figure of merit for comparing such converters. (High-speed SDR hardware from several manufacturers and vendors is addressed in the column "SDR: Simplified" by Ray Mack, W5IFS, in the January/February 2013 issue of *QEX*.)

PIPELINED ADC ARCHITECTURE

Most high-speed converters intended for SDR applications use a *pipelined* architecture. This type of A/D converter consists of multiple stages, each of which includes a sample-and-hold function, low-resolution A/D and D/A converter, and a gain stage. The first stage

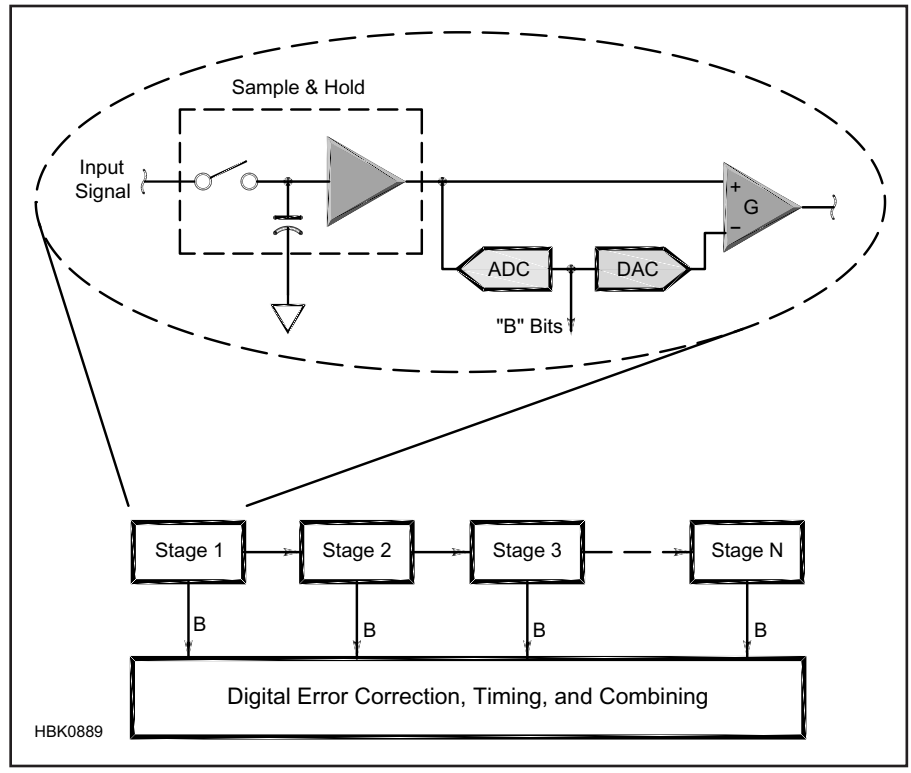


Figure 8.22 — Block diagram of pipelined high-speed A/D converter.

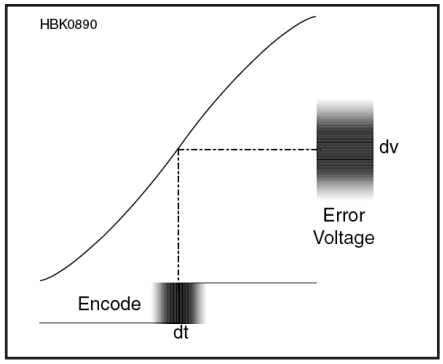


Figure 8.23 — RMS jitter vs. RMS noise effect of sampling uncertainty on signal-to-noise. [from Figure 1 of Analog Devices Application Note AN-501, available at www.analog.com, used with permission]

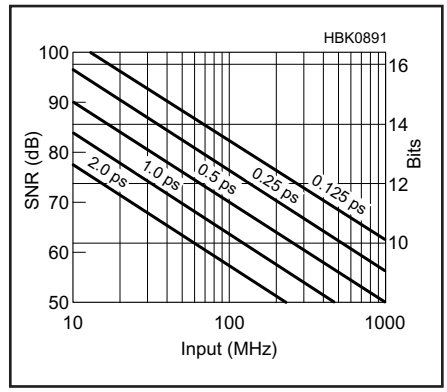


Figure 8.24 — Signal-to-noise ratio due to aperture jitter. [from Figure 1 of Analog Devices Application Note AN-501, available at www.analog.com, used with permission]

performs a coarse quantization of the input signal, of say, "B" bits, where B is typically 1 to 4 in commercial products. The resultant data word is applied to a D/A converter, and the output analog voltage is subtracted from the original input signal, producing an error signal. That error signal is amplified and passed to another, usually identical, converter stage with enough overlap to compensate for any errors in the previous conversion. Many stages can be connected together, with the last stage only including an A/D converter. The digital outputs from each stage are combined and corrected, as well as time-aligned, to produce a higher-resolution output. **Figure 8.22** illustrates the basic pipelined structure.

Since each stage is typically a low-resolution flash converter, pipelined converters are capable of quite high resolution — up to 16 bits — and sample rates in the hundreds of Msps, even into the Gsps (gigasamples per second) range. There is a delay before the first sample emerges, of course, since an N-stage pipeline converter must perform N conversions before the first valid output is available. However, each successive sample is produced at the full sample rate.

Some pipelined converters also include some basic digital down conversion on chip, with some degree of programmability.

The cost of an A/D or D/A converter is generally proportional to both speed and resolution, so it is important to avoid over specifying these components. Later in this chapter we will discuss how to determine the required sample rate and resolution for a given application.

While sample rate, resolution, and SFDR are the principal selection criteria for data converters in a DSP system, other parameters such as signal-to-noise ratio, harmonic and intermodulation distortion, full-power bandwidth, and aperture delay jitter can also affect system performance. Of course, basic specifications such as power requirements, interface type (serial or parallel), and cost also determine a device's suitability for a particular application. As with any electronic component, it is very important to read and fully understand the data sheet.

Most manufacturers of high-performance converters also include recommendations on support circuitry to be used with their devices to extract the best possible performance. For example, some ADCs require buffer amplifiers to drive the input, or external voltage references to set the full scale range. Many converters, especially high-speed and high-resolution devices are very sensitive to circuit layout, power supply decoupling and grounding. Most manufacturers provide evaluation boards to allow testing the converter IC on the bench, and the layout files can be copied into the final system design.

CLOCKING HIGH-SPEED CONVERTERS

ADCs generally require some kind of clock function to set the sample rate. Any cycle-to-cycle variation in the sample timing ("jitter") causes an error. It can be shown that in RF-sampling converters, sample-clock jitter is analogous to phase noise in the local oscillators of analog radios, and can degrade the performance of the converter substantially. **Figure 8.23** shows how this comes about.

As A/D and D/A converters achieve higher and higher sample rates and signal bandwidths, the clocking becomes critical. Most logic families produce output waveforms that have more timing jitter than can be tolerated.

The degradation of a high-frequency sampled signal due to sample jitter can be calculated and plotted. In **Figure 8.24** you can see that a 100 MHz signal sampled with a 1ps jitter sample clock (typical of ACT-series logic gates) will have just over 60 dB of signal-to-noise ratio, about equal to 10-bit performance, even if the A/D converter is capable of 16 bits! Improving the sample clock jitter to 0.125 ps using a specialized clock-distribution IC, improves the system performance to nearly 14 bits.

8.5 Digital Signal Processors

The term *digital signal processor* (DSP) is commonly understood to mean a special-purpose microprocessor with an architecture that has been optimized for signal processing rather than data processing. And indeed, in many systems the box labeled "DSP" in Figure 8.1 is such a device. A microprocessor has the advantage of flexibility because it can easily be re-programmed. Even with a single program, it can perform many completely different tasks at different points in the code. On-chip hardware resources such as multipliers and other computational units are used efficiently because they are shared among various processes.

That is also the Achilles' heel of programmable DSPs. Any hardware resource that is shared among various processes can be used by only one process at a time. That can create bottlenecks that limit the maximum computation speed. Some DSP chips include multiple computational units or multiple *cores* (basically multiple copies of the entire processor) that can be used in parallel to speed up processing.

8.5.1 Microprocessor-type DSP ICs

Programming a DSP IC is relatively easy. C compilers are available for most devices, so you don't have to learn assembly language. Typically you include a connector on your circuit board into which is plugged an *in-circuit programmer* (ICP), which is connected to a PC via a serial or USB cable. The software is written and compiled on the PC and then downloaded to the DSP. The same hardware often also includes an *in-circuit debugger*

(ICD) so that the program can be debugged on the actual circuitry used in the design. The combination of the editor, compiler, programmer, debugger, simulator and related software is called an *integrated development environment* (IDE).

Until recently, you had to use an *in-circuit emulator* (ICE), which is a device that plugs into the circuit board in place of the microprocessor. The ICE provides sophisticated debugging tools that function while the emulator runs the user's software on the target device at full speed. Nowadays, however, it is more common to use the ICD function that is built into many DSP chips and which provides most of the functions of a full-fledged ICE. It is much cheaper and does not require using a socket for the microprocessor chip.

The architecture of a digital signal processor shares some similarities to that of a general-purpose microprocessor but also differs in important respects. For example, DSPs generally don't spend much of their lives handling large computer files, so they tend to have a smaller memory address space than processors intended to be used in computers. On the other hand, the memory they do have is often built into the DSP chip itself to improve speed and to reduce pin count by eliminating the external address and data bus.

Most microprocessors use the traditional *Von Neumann architecture* in which the program and data are stored in the same memory space. However, most DSPs use a *Harvard architecture*, which means that data and program are stored in separate memories. That speeds up the processor because it can be reading the next program instruction at the same time as it is reading or writing data in response to the previous instruction. Some DSPs have two data memories so they can read and/or write two data words at the same time. Most devices actually use a modified Harvard architecture by providing some (typically slower and less convenient) method for the processor to read and write data to program memory.

Probably the key difference between general-purpose and digital-signal processors is in the computational core, often called the *arithmetic logic unit* (ALU). The ALU in a traditional microprocessor only performs integer addition, subtraction and bitwise logic operations such as AND, OR, one-bit shifting and so on. More-complicated calculations, such as multiplication, division and operations with floating-point numbers, are done in software routines that exercise the simple resources of the ALU multiple times to generate the more-complicated results.

In contrast, a DSP has special hardware to perform many of these operations much faster. For example, the *multiplier-accumulator* (MAC) multiplies two numbers and adds (accumulates) the product with the previous results in a single step. Many common DSP algorithms involve the sum of a large number of products, so nearly all DSPs include this function. **Figure 8.25** is a simplified block diagram of the dsPIC series from Microchip. Its architecture is basically that of a general-purpose microcontroller to which has been added a DSP engine, which includes a MAC, a barrel shifter and other DSP features. It uses a modified Harvard architecture with two data memories that can be simultaneously accessed.

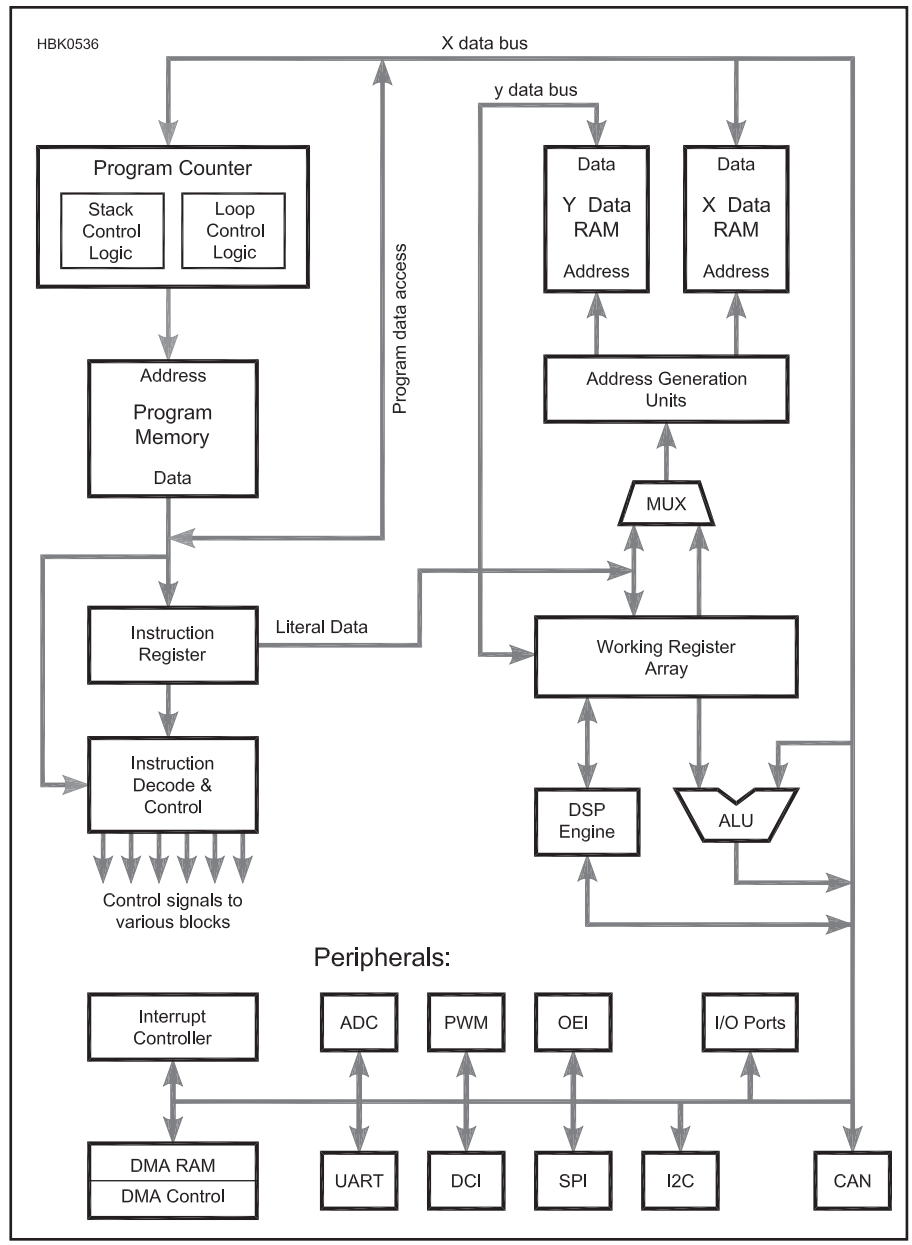


Figure 8.25 — Simplified block diagram of a dsPIC processor.

8.5.2 Fixed-Point versus Floating-Point

DSP microprocessors are sometimes differentiated on their ability to handle fixed-point (integer) or *floating-point* numbers.

A floating-point number is the binary equivalent of scientific notation. Recall that the decimal integer 123000 is expressed as 1.23×10^5 in scientific notation. It is common practice to place the decimal point after the first non-zero digit and indicate how many digits the decimal point must be moved by the *exponent* of ten, 5 in this case. The 1.23 part is called the *mantissa*. In a computer,

base-2 binary numbers are used in place of the base-10 decimal numbers used in scientific notation. The *binary point* (equivalent to the decimal point in a decimal number) is assumed to be to the left of the first non-zero bit. For example the binary number 00110100 when converted to a 16-bit floating point number would have an 11-bit mantissa of 11010000000 (with the binary point assumed to be to the left of the first "1") and a 5-bit exponent of 00110 (decimal +6).

A floating point number can represent a signal with much more dynamic range than an integer number with the same number of bits. For example, a 16-bit fixed-point signed integer can vary from –32768 to +32767. The difference between the smallest (1) and largest signal that can be represented is 20 log(65535) = 96 dB. If the 16 bits are divided into an 11-bit mantissa and 5-bit exponent, the available range is 20 log(2048) = 66 dB from the mantissa and 20 log(2^{32}) = 193 dB from the exponent for a total of 259 dB. The disadvantage is that the mantissa has less resolution, potentially increasing noise and distortion. Normally floating-point numbers are at least 32 bits wide to mitigate that effect.

Some DSPs can process floating-point numbers directly in hardware. Fixed-point DSPs can also handle floating-point numbers, but it must be done in software and that slows the computations down. The additional dynamic range afforded by floating-point processing is normally not needed for radio communications signals. The dynamic range of radio signals can usually be handled by the 16-bit data words used by most fixed-point DSPs. Using integer arithmetic and a fixed-point processor saves the additional cost of a floating-point processor or the additional computational overhead of floating-point calculations on a fixed-point device. However, it requires careful attention to detail on the part of the programmer to make sure the signal can never exceed the maximum integer value or get so weak that the signal-to-noise ratio is degraded. If cost or computation time is not an issue, it is much easier to program in floating point since dynamic range issues can be ignored for most computations.

The term *pipeline* refers to the ability of a microprocessor to perform portions of several instructions at the same time. The sequence of operations required to perform an instruction is broken down into steps. Since each step is performed by a different chunk of hardware, different chunks can be working on different instructions at the same time. Most DSPs have at least a simple form of pipelining in which the next instruction is being fetched while the previous instruction is being executed. Some DSPs can do a multiply-accumulate while the next two multiplicands are being read from memory and the previous accumulated result is being stored so that the entire operation can occur in a single clock cycle. MACs per second is a common figure of merit for measuring DSP speed. For conventional microprocessors, a more common figure of merit is millions of instructions per second (MIPS) or floating-point operations per second (FLOPS).

Many DSPs have a sophisticated address generation unit that can automatically increment one or more data memory pointers so that repetitive calculations can step through memory without the processor having to calculate the addresses. *Zero-overhead looping* is the ability to automatically jump the address pointer back to the beginning of the array when it reaches the end. That saves several microprocessor instructions per loop that normally would be required to check the current address and jump when it reaches a predetermined value.

While most DSPs do not include a full hardware divider, some do include special instructions and hardware to speed up division calculations. A *barrel shifter* is another common DSP feature. It allows shifting a data word a specified number of bits in a single clock cycle. *Direct memory access* (DMA) refers to special hardware that can automatically transfer data between memory and various peripheral devices or ports without processor overhead.

8.5.3 DSP in Embedded Systems

An *embedded system* is a device that is not a "computer" but nevertheless has a microprocessor or DSP chip embedded somewhere in its circuitry. Examples are microwave ovens, automobiles, mobile telephones and software-defined radios. DSPs intended for embedded systems often include a wide array of on-chip peripherals such as various kinds of timers, multiple hardware interrupts, serial ports of various types, a real-time clock, pulse-width modulators, optical encoder interfaces, A/D and D/A converters and lots of general-purpose digital I/O pins. Some DSPs have architectures that are well-suited for general-purpose control applications as well as digital signal processing.

Table 8.2 lists some manufacturers of DSP chips targeted to embedded systems. It should be mentioned that microprocessors made by Intel and AMD and intended for use as CPUs in personal computers also include extensive DSP capability. However, they are optimized for data-processing applications under an operating system such as Windows or Linux, and are not very efficient at signal-processing applications.

Table 8.2

Manufacturers of DSP Microprocessors for Communications Applications

Company	*URL*
Analog Devices	**www.analog.com**
Cirrus Logic	**www.cirrus.com**
Microchip	**www.microchip.com**
Texas Instruments	**www.ti.com**

8.5.4 Typical DSP Processors

When selecting a DSP device for a new design, often the available development environment is more important than the characteristics of the device itself. Microchip's dsPIC family of DSPs was chosen for the examples in this chapter because their integrated development environment is extensive and easy to use and the IDE software is available for free download from their website.[5] The processor instruction set is a superset of the PIC24 family of general-purpose microcontrollers, with which many hams are already familiar. The company offers a line of low-cost evaluation boards and starter kits as well as an inexpensive in-circuit debugger, the ICD 3. The free IDE software includes a simulator that can run dsPIC software on a PC (at a much slower rate, of course), so that you can experiment with DSP algorithms before buying any hardware.

The Microchip DSP family is limited to 100 million instructions per second. In a system with, say, a 100 kHz sample rate, 1000 instructions per sample are available which should be plenty if the calculations are not too complex. However if the sample rate is 1 MHz, then you get only 100 instructions per sample, which likely would be insufficient.

If more horsepower is required, you'll need to select a processor from a different manufacturer. Look for one with a well-integrated suite of development software that is powerful and easy to use. Also check out the cost and availability of development hardware such as evaluation kits, programmers and debuggers. Once those requirements are met, then you can move on to selecting a specific device with the performance and features required for your application. It can be helpful at the beginning of a project to first write some of the key software routines and test them on a simulator to estimate execution times, in order to determine how powerful a processor is needed.

When estimating execution time, don't forget to include the effect of interrupts. Most DSP systems require real-time response and make extensive use of interrupts to ensure that certain events happen at the correct times. Although this is hidden from the programmer's view when programming in C, the interrupt service routines contain quite a bit of overhead each time they are called (to save the processor state when responding and to

recall the state just before returning from the interrupt). Sometimes an interrupt may be called more often than you expect, which can eat up processor cycles and so increase the execution time of other unrelated routines.

In the past, may embedded systems were written in assembly language so save memory and increase processing speed. Many early microprocessors and DSPs did not have enough memory to support a high-level language. Today, most processors have sufficient memory and processing speed to support a C kernel and library without difficulty. For anything but the simplest of programs, it is not only faster and easier to develop software in C but it is easier to support and maintain as well, especially if people other than the original programmer might become involved. Far more people know the C programming language than any particular processor's assembly language. It is true that the version of C used on a DSP chip is usually modified from standard ANSI C to support specific hardware features, but it would still be far easier to learn for a programmer familiar with writing C code on a PC or on a different DSP.

A common technique is first to write the entire application in C. Then, if execution time is not acceptable, analyze the system to determine in which software routines the bottlenecks are occurring. You can then rewrite those routines in assembly language. Having an already-working version written in C (even if too slow) can be helpful in testing and troubleshooting the equivalent assembly language.

8.5.5 DSP Without a Dedicated Processor

One way to speed up processing is to move all or part of the computations from the programmable DSP to an *application-specific integrated circuit* (ASIC), which has an architecture that has been optimized to perform some specific DSP function.

You could also design your own application-specific circuitry using a PC board full of discrete logic devices. Nowadays, however, it is more common to do that with a *programmable-logic device* (PLD). This is an IC that includes many general-purpose logic elements, but the connections between the elements are undefined when the device is manufactured. The user defines those connections by programming the device to perform whatever function is required. PLDs come in a wide variety of types, described by an alphabet soup of acronyms.

Programmable-array logic (PAL), *programmable logic array* (PLA), and *generic array logic* (GAL) devices are relatively simple arrays of AND gates, OR gates, inverters and latches. They are often used as "glue logic" to replace the miscellaneous discrete logic ICs that would otherwise be used to interface various larger digital devices on a circuit board. They are sometimes grouped under the general category of *small PLD* (SPLD). A *complex PLD* (CPLD) is similar but bigger, often consisting of an array of PALs with programmable interconnections between them.

A *field-programmable gate array* (FPGA) is bigger yet, with up to millions of gates per device. An FPGA includes an array of *complex logic blocks* (CLB), each of which includes some programmable logic, often implemented with a RAM *look-up table* (LUT), and output registers. *Input/output blocks* (IOB) also contain registers and can be configured as input, output, or bi-directional interfaces to the IC pins. The interconnections between blocks are much more flexible and complicated than in CPLDs. Some FPGAs also include higher-level circuit blocks such as general-purpose RAM, dozens or hundreds of hardware multipliers, and even entire on-chip microprocessors.

Some of the more inexpensive PLDs are *one-time programmable* (OTP), meaning you must throw the old device away if you want to change the programming. Other devices are re-programmable or even *in-circuit programmable* (ICP) which allows changing the internal circuit configuration after the device has been soldered onto the PC board, typically under the control of an on-board microprocessor. That offers the best of both worlds, with speed nearly as fast as an ASIC but retaining many of the benefits of the reprogrammability of a microprocessor-type DSP. Most large FPGAs store their programming in *volatile memory*, which is RAM that must be reloaded every time power is applied, typically by a ROM located on the same circuit board. Some FPGAs have programmable ROM on-chip.

Programming a PLD is quite different from programming a microprocessor. A microprocessor performs its operations sequentially — only one operation can be performed at a time. Writing a PLD program is more like designing a circuit. Different parts of the circuit can be doing different things at the same time. Special *hardware-description languages* (HDL) have been devised for programming the more complicated parts such as ASICs and FPGAs. The two most common industry-standard HDLs are Verilog and VHDL. (The arguments about which is "best" approach the fervor of the Windows vs Linux wars!) There is also a version of the C++ programming language called SystemC that includes a series of libraries that extend the language to include HDL functions. It is popular with some designers because it allows simulation and hardware description using the same software tool.

Despite the speed advantage of FPGAs, most amateurs use microprocessor-type devices for their DSP designs, supplemented with off-the-shelf ASICs where necessary. The primary reason is that the design process for an FPGA is quite complicated, involving obtaining and learning to use several sophisticated software tools. The steps involved in programming an FPGA are:

1. Simulate the design at a high abstraction level to prove the algorithms.
2. Generate the HDL code, either manually or using some tool.
3. Simulate and test the HDL program.
4. Synthesize the gate-level netlist.
5. Verify the netlist.
6. Perform a timing analysis.
7. Modify the design if necessary to meet timing constraints.
8. "Place and route" the chip design.
9. Program and test the part.

Many of the software tools needed to perform those steps are quite expensive, although some manufacturers do offer free proprietary software for their own devices. Some principal manufacturers are listed in **Table 8.3**.

8.5.6 Using Graphics Processors for DSP

As PCs have become used for graphics-intensive applications such as games, developers quickly found out that the microprocessor chip in the PC was unable to provide enough computing horsepower to do complex graphics functions like 3D rendering, shading, etc. This gave rise to a new breed of specialized processors called Graphics Processing Units (GPU). These processors consist of simple computational blocks that

Table 8.3
PLD Manufacturers

Company	*Devices*	*URL*
Achronix	FPGA	**www.achronix.com**
Atmel	SPLD, CPLD, PGA, ASIC	**www.microchip.com**
Intel	CPLD, FPGA, ASIC	**www.intel.com**
Lattice Semiconductor	SPLD, CPLD, FPGA	**www.latticesemi.com**
Microsemi	FPGA	**www.microsemi.com**
Xilinx	CPLD, FPGA	**www.xilinx.com**

are capable of only a few functions like multiplication and addition. However, unlike a typical PC CPU with a handful of cores to speed up processing, GPUs have hundreds or even thousands of cores! In addition, they are optimized for handling large arrays of data. This makes them well suited for DSP functions such as Fast Fourier Transforms.

Fortunately, some manufacturers of GPUs have recognized that their products are useful for applications other than graphics, and are now providing libraries and programming support for using their products in communications systems such as SDR. GPU manufacturer Nvidia, for example, offers the CUDA parallel computing platform and programming model with the cuFFT library.

The inexpensive Linux-based Raspberry Pi computer board has become very popular among amateurs and experimenters. The BCM283x main processor is a system-on-chip device manufactured by Broadcom (now owned by Avago) that contains both a CPU and a GPU. At 700MHz to 1.2 GHz, the CPU is too underpowered for many of the DSP algorithms used in SDR. However, the graphics processor included in the device can be programmed to do some of the FFT algorithms, and offers a 10x speed improvement.[6]

8.6 Digital (Discrete-time) Signals

Digital signals differ from analog signals in two ways. One is that they are digitized in time, a process called *sampling*. The other is that they are digitized in amplitude, a process called *quantization*. Sampling and quantization affect the digitized signal in different ways so the following sections will consider their effects separately.

8.6.1 Sampling — Digitization in Time

Sampling is the process of measuring a signal at discrete points in time and recording the measured values. An example from history is recording the number of sunspots. If an observer goes out at noon every day and writes down the number of observed sunspots, then that data can be used to plot sunspot number versus time. In this case, we say the *sample rate* is one sample per day. The data can then be analyzed in various ways to determine short and long-term trends. After recording only a few months of data it will quickly become apparent that sunspot number has a marked periodicity — the numbers tend to repeat every 27 days (which happens to be the rotation rate of the sun as seen from earth).

What if, instead of taking a reading once a day, the readings were taken only once per month? With a 30-day sample period, the 27-day periodicity would likely be impossible to see. Clearly, the sample rate must be at least some minimum value to accurately represent the measured signal. Based on earlier work by Harry Nyquist, Claude Shannon proved in 1948 that in order to sample a signal without loss of information, the sample rate must be greater than the *Nyquist rate*, which is two times the bandwidth of the signal. In other words, the bandwidth must be less than the *Nyquist frequency*, which is one-half the sample rate. This is known as the *Nyquist sampling criterion*.

That simple rule has some profound implications. If all the frequency components of a signal are contained within a bandwidth of B Hz, then sampling at a rate greater than 2B samples per second is sufficient to represent the signal with 100% accuracy and with no loss of information. It is theoretically possible to convert the samples back to an analog signal that is exactly identical to the original.

Of course, a real-world digital system measures those samples with only a finite number of bits of resolution, with consequences that we will investigate in the section on quantization that follows. In addition, sampling theory assumes that there is absolutely no signal energy outside the specified bandwidth; in other words the stopband attenuation is infinity dB. Any residual signal in the stop-band shows up as distortion or noise in the sampled signal.

To simplify the discussion, let's think about sampling a signal of a single frequency (a sine wave). **Figure 8.26** illustrates what happens if the sample rate is too low. As shown, the sample rate is approximately ⅞ the sine-wave frequency. You can see that the sampled signal has a period about 8 times greater than the period of the sine wave, or ⅛ the frequency. The samples are the same as if the analog signal had been a sine wave of ⅛ the actual frequency.

That is an example of a general principle. Sampling a signal produces an output with components at frequencies equal to the difference between the actual frequency of the analog signal and the sample rate. If the sam-

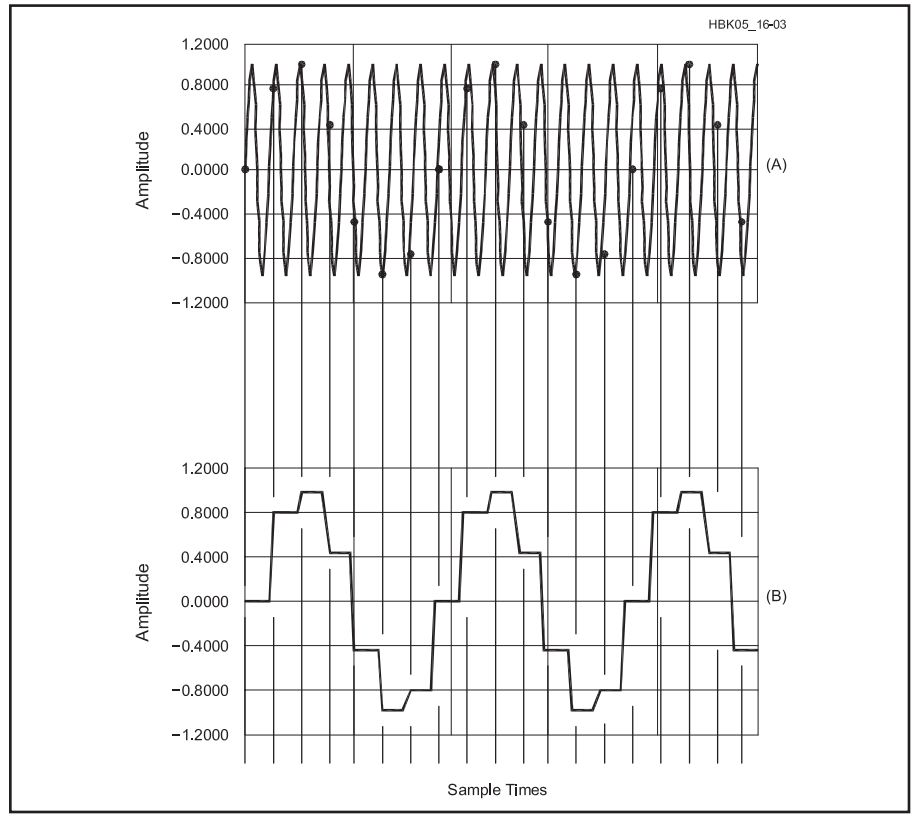


Figure 8.26 — Undersampled sine wave (A). Samples aliased to a lower frequency (B).

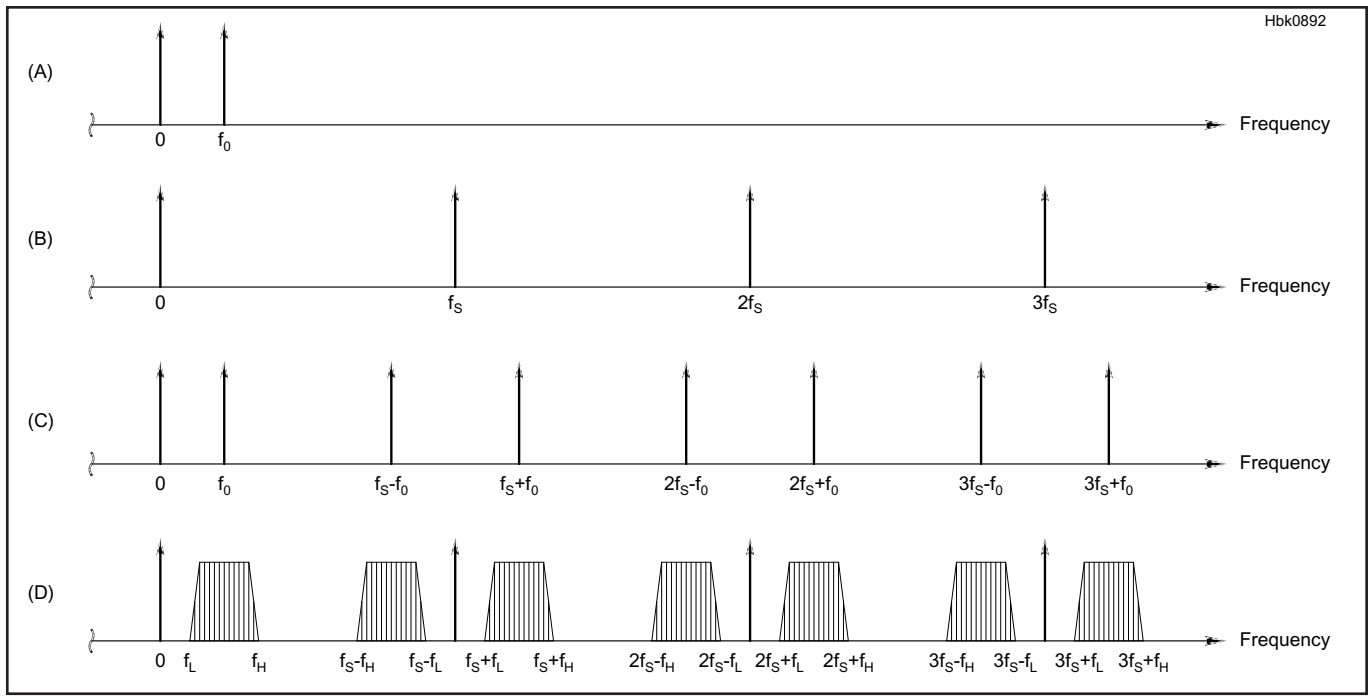


Figure 8.27 — Frequency relationships in a sampled-data system; spectrum of an analog sine wave (A). The spectrum of the sampling function, including all harmonics (B). The spectrum of the sampled sine wave (C). The spectrum of a sampled band of frequencies between f_L and f_H (D).

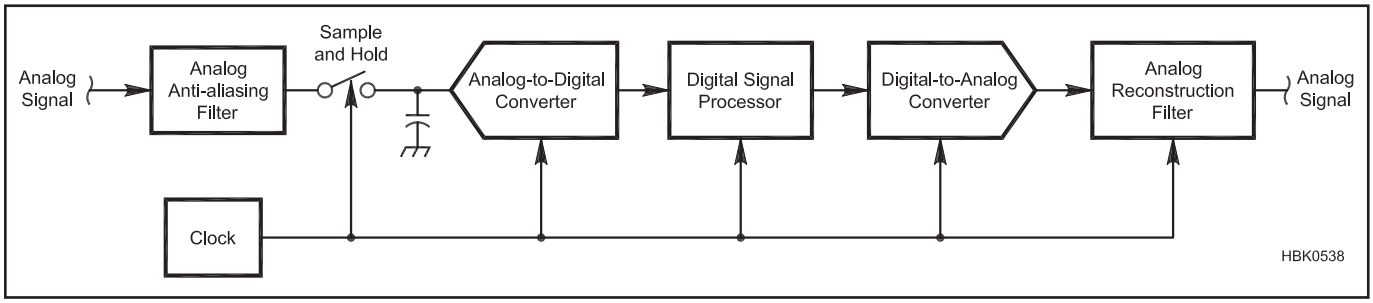


Figure 8.28 — A more complete block diagram of a DSP system.

pling rate is too low (or the input signal too high in frequency) the signal at ($f_s - f_{sig}$) is indistinguishable from an in-band signal, which is why the new signal is called an *alias*. In the above example, the alias frequency f_o is

$$f_o = f_{sig} - f_s = (1 - 7/8) f_{sig} = (1/8) f_{sig}$$

where f_{sig} is the frequency of the signal before sampling and f_s is the sample rate.

The sampling process also creates aliases from any input signals around harmonics of the sampling frequency. **Figure 8.27** shows the frequency-domain representation of the signal relationships in a sampled-data system. Figure 8.27A shows the spectrum of a single sine wave input signal, and Figure 8.27B shows the spectrum of the sampling clock and its harmonics. You can look at Figure 8.27C two ways. First, you can consider all the individual frequencies as the output signals generated by sampling the input signal at f_o. On the other hand, those are also the input signal frequencies that alias to f_o, calculated from the equation

$$f_{sig} = |f_o - N f_s|$$

where N is the harmonic number. Experienced RF engineers will recognize this phenomenon as being very similar to image-generation in the mixing process of a heterodyne receiver. The sampled signal (equivalent to the mixer output) contains the sum and difference frequencies of the input signal and all the harmonics of the sample frequency. Figure 8.27D shows the more general case of sampling a band of signals between f_L and f_H. Note the inversion of the aliased spectrum that extends below fs. If f_H is higher than $f_S/2$, the higher frequencies in the band will overlap the original band in the output spectrum and create artifacts.

An *anti-aliasing filter* before the sampler prevents unwanted signals from creating artifacts in the sampled data output, as shown in **Figure 8.28**. For a baseband signal (one that extends to zero Hz), the anti-aliasing filter is a low-pass filter whose stopband extends from the Nyquist frequency to infinity. Of course, practical filters do not transition instantaneously from the passband to the stopband, so the bandwidth of the passband must be somewhat less than half the sample rate.

Figure 8.26 showed each sample being held at a constant value for the duration of one sample period. However, sampling theory actually assumes that the sample is only valid at the instant the signal is sampled; it is zero

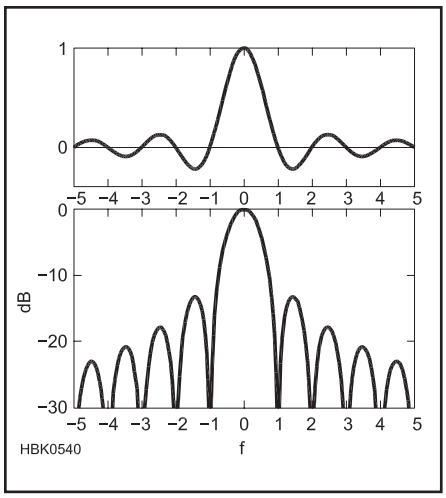


Figure 8.29 — The sinc function, where the horizontal axis is frequency normalized to the sample rate. At the bottom is the same function in decibels.

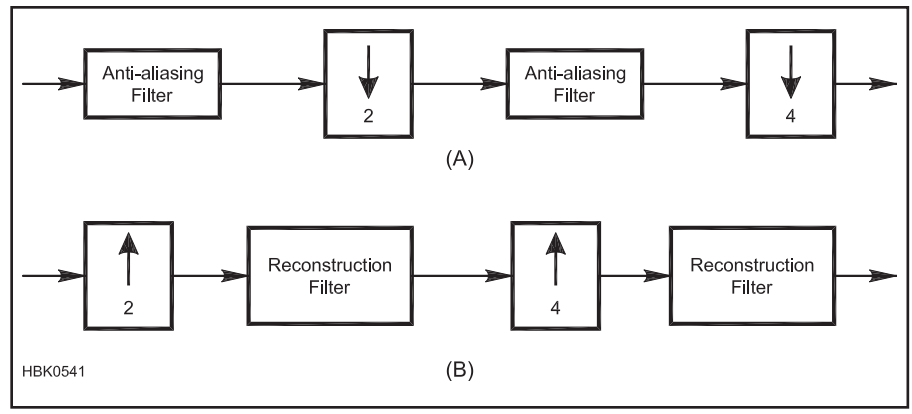


Figure 8.30 — Decimation (A) and interpolation (B). The arrow's direction indicates decimation (down) or interpolation (up) and the number is the factor.

or undefined at all other times. A series of such infinitely-narrow impulses has harmonics all the way to infinite frequency. Each harmonic has the same amplitude and is modulated by the signal being sampled. See Figure 8.27D. When a digitized signal is converted back to analog form, unwanted harmonics must be filtered out by a *reconstruction filter* as shown in Figure 8.28. This is similar to the anti-aliasing filter used at the input in that its bandwidth should be no greater than one-half the sample rate. It is a low-pass filter for a baseband signal and a band-pass filter for an undersampled signal.

Most DACs actually do hold each sample value for the entire sample period. This is called *zero-order hold* and results in a frequency response in the shape of a sinc function

$$\mathrm{sinc}(f) = \frac{\sin(\pi f)}{\pi f}$$

where f is normalized to the sample rate, f = frequency / sample rate.

The graph of the sinc function in **Figure 8.29** shows both positive and negative frequencies for reasons explained in the Analytic Signals section. Note that the logarithmic frequency response has notches at the sample rate and all of its harmonics. If the signal bandwidth is much less than the Nyquist frequency, then most of the signal at the harmonics falls near the notch frequencies, easing the task of the reconstruction filter. If the signal bandwidth is small enough (sample rate is high enough), the harmonics are almost completely notched out and a reconstruction filter may not even be required.

The sin(πf)/πf frequency response also affects the passband. For example if the passband extends to f_S / 4 (f = 0.25), then the response is

$$20 \log \frac{\sin(\pi \cdot 0.25)}{\pi \cdot 0.25} = -0.9 \text{ dB}$$

at the top edge of the passband. At the Nyquist frequency, (f = 0.5), the error is 3.9 dB. If the signal bandwidth is a large proportion of the Nyquist frequency, then some kind of digital or analog compensation filter may be required to correct for the high-frequency rolloff. The interpolation filters used in digital audio systems have this correction included in the frequency response.

8.6.2 Decimation and Interpolation

The term *decimation* simply means reducing the sample rate. For example to decimate by two, simply eliminate every second sample. That works fine as long as the signal bandwidth satisfies the Nyquist criterion at the lower, output sample rate. If the analog anti-aliasing filter is not narrow enough, then a digital anti-aliasing filter in the DSP can be used to reduce the bandwidth to the necessary value. This must be done *before* decimation to satisfy the Nyquist criterion.

If you need to decimate by a large amount, then the digital anti-aliasing filter must have a very small bandwidth compared to the sample rate. As we will see later, a digital filter with a small bandwidth is computationally intensive. For this reason, large decimation factors are normally accomplished in multiple steps, as shown in **Figure 8.30A**. The first decimation is by a small factor, typically 2, so that the first anti-alias filter can be as simple as possible. The second decimation stage then does not have to decimate by such a large factor, simplifying its task. In addition, since it is running at only half the input sample rate it has more time to do its calculations. Generally it is most efficient to decimate by the smallest factor in the first stage, a larger factor in the second, and the largest factors in the third and any subsequent stages. The larger the total decimation factor, the greater the number of stages is appropriate but more than three stages is uncommon.

Interpolation means increasing the sample rate. One way to do that is simply to insert additional zero-value samples, a process called *zero-stuffing*. For example, to interpolate by a factor of three, insert two zero-value samples after each input sample. That works, but may not give the results you expect. Recall that a sampled signal has additional copies of the baseband signal at all harmonics of the sample rate. All of those harmonics remain in the resampled signal, even though the sample rate is now higher. To eliminate them, the signal must be filtered after interpolation. After filtering, there is signal only at baseband and around the harmonics of the interpolated (higher-frequency) sample rate. It's as if the analog signal had been sampled at the higher rate to begin with, which relaxes the requirements on the reconstruction filter.

Just as with decimation, interpolation by a large factor is best done in stages, as shown in Figure 8.30B. In this case, the stage running at the lowest sample rate (again the first stage) is the one with the lowest interpolation factor.

Zero-stuffing followed by filtering is not the only way to interpolate. Really what you are trying to do is to fill in between the lower-rate samples with additional samples that "connect the dots" in as smooth a manner as possible. It can be shown that that is mathematically equivalent to zero-stuffing and filtering. For example, if instead of inserting zero-value samples you instead simply repeat the last input sample, you have a situation similar to the zero-order hold of a DAC output. It is equivalent to zero-stuffing followed by a low-pass filter with a frequency response of sin(πf)/πf. If you do a straight-line interpolation between input samples (a "first-order" interpolation), it turns out that it is equivalent

to a low pass filter with a frequency response of $[\sin(\pi f)/\pi f]^2$, which has a sharper cutoff and better stop-band rejection than a zero-order interpolation. Higher-order interpolations have smoother responses in the time domain which translate to better filter responses in the frequency domain.

So far we have only covered decimation and interpolation by integer factors. It is also possible to change the sample rate by a non-integer factor, which is called *resampling* or *multi-rate* conversion. For example, if you want to increase the sample rate by a factor of 4/3, simply interpolate by 4 and then decimate by 3. That method can become impractical for some resample ratios. For example, to convert an audio file recorded from a computer sound card at 48 kHz to the 44.1 kHz required by a compact disc, the resample ratio is 44,100 / 48,000 = 147 / 160. After interpolation by 147, the 44.1 kHz input file is sampled at 6.4827 MHz, which would result in excessive processing overhead.

In addition, the interpolation/decimation method only works for resample ratios that are rational numbers (the ratio of two integers). To resample by an irrational number, a different method is required. The technique is as follows. For each output sample, first determine the two nearest input samples. Calculate the coefficients of the Nth-order equation that describes the trajectory between the two input samples. Knowing the trajectory between the input samples and the output sample's relative position between them, the value of the output sample can be calculated from the equation.

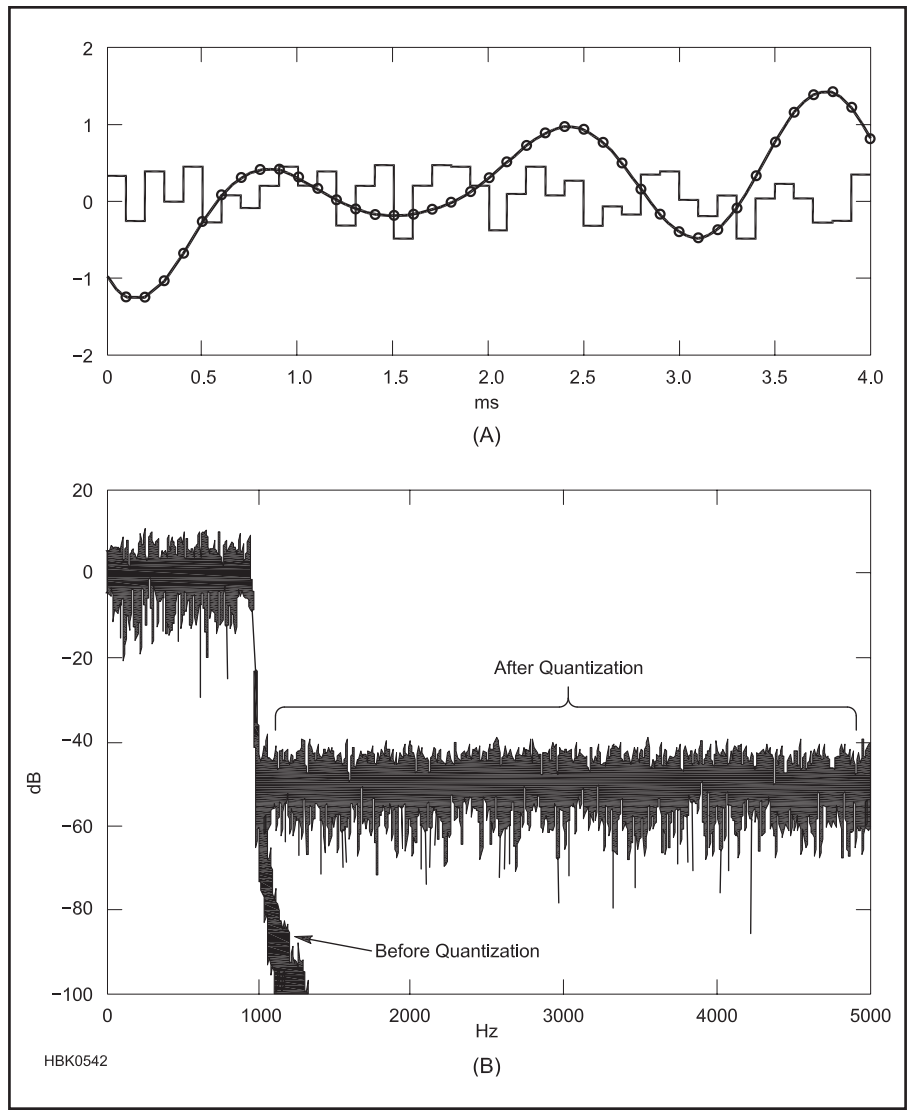


Figure 8.31 — Quantization error of a random noise signal that has been band-limited to 1 kHz to simulate an audio signal. (A) The sampler resolution is 8 bits and the sample rate is 10 kHz. Sample values are indicated by circles. Also shown is the quantization error, in units of LSB. Below is the frequency spectrum of the signal before and after quantization. (B).

8.6.3 Quantization — Digitization in Amplitude

While sampling (digitization in time) theoretically causes no loss of signal information, quantization (digitization in amplitude) always does. For example, an 8-bit signed number can represent a signal as a value from –128 to +127. For each sample, the A/D converter assigns whichever number in that range is closest to the analog signal at that instant. If a particular sample has a value of 10, there is no way to tell if the original signal was 9.5, 10.5 or somewhere in between. That information has been lost forever.

QUANTIZATION NOISE

When quantizing a complex signal such as speech, this error shows up as noise, called *quantization noise* as discussed in section 8.2.2. The error is random — it is equally likely to be anywhere in the range of –½ to +½ of a single step of the ADC. In **Figure 8.31** the smooth waveform is a band-limited analog noise signal plotted over a 4 millisecond period. The vertical axis of Figure 8.31A indicates the quantization error for each sample in LSB, represented by the staircase-like waveform. The spectrum of the signal is shown in Figure 8.30B, both before quantization and after quantization. While the original signal has the frequency components above 1 kHz reduced by analog filtering, the digitized version has considerable noise in the 1 to 5 kHz band.

One critical point that is sometimes overlooked is that quantization noise is spread over the entire bandwidth from zero Hz to the sample rate. If you are digitizing a 3 kHz audio channel with a 48 ksps sampler, only a fraction of the noise power is within the channel. For that reason, the effective signal-to-noise ratio depends not only on the number of bits but also the sample rate, f_s, and the signal bandwidth, B:

$$SNR_{eff} = SNR + 10\log\left(\frac{f_s}{2B}\right) \text{ dB}$$

The reason for the factor of two in the denominator is that the bandwidth of a positive-frequency scalar signal should be compared to the Nyquist bandwidth, $f_s/2$. When filtering a complex signal (one that contains I and Q parts), the 2B in the denominator should be replaced by B.

When choosing an A/D converter don't forget that the effective SNR depends on the sample rate. As an example, let's compare an Analog Devices AD9235 12-bit, 65 Msps ADC to an AD7653, which is a 16-bit 1 Msps ADC from the same manufacturer. Assume a 10 kHz signal bandwidth.

An ideal 12-bit ADC has a SNR of 1.76 + 6.02 × 12 = 74.0 dB. The AD9235's performance is not far from the ideal; its SNR is specified at 70.5 dB at its 65 Msps maximum sample rate. It costs about $12.50. In a 10 kHz bandwidth, the effective SNR is 70.5 + 10 log (65,000/20) = 105.6 dB.

An ideal 16-bit ADC has an SNR of 98.1 dB. The AD7653 is specified at 86 dB and costs about $15. The effective SNR is 86 + 10log(1000/20) = 93 dB.

So the 12-bit ADC with 70.5 dB SNR is actually 12.6 dB better than the 16-bit device with 86 dB SNR in this application! Even an ideal 16-bit, 1 Msps ADC would only have an effective SNR of 98.1 + 10log(1000/20) = 103.1 dB, still worse than the actual performance of the AD9235 when measured with the same bandwidth. Note that to actually realize 105.6 dB of dynamic range the signal from the ADC would need to be digitally filtered to a 10 kHz bandwidth while increasing the bits of data resolution from 12 to at least 18.

Oversampling is the name given to the technique of using a higher-than-necessary sample rate in order to achieve an improved S/N ratio. Don't forget that when the high-sample-rate signal is decimated the data words must have enough bits to support the higher dynamic range at the lower sample rate. As a rule of thumb, the quantization noise should be at least 10 dB less than the signal noise in order not to significantly degrade the SNR. In the AD9235 example, assuming a 100 kHz output sample rate, about 18 bits would be required: 1.76 + 6.02 × 18 + 10log(100/20) = 117.1 dB, which is 11.5 dB better than the 105.6 dB dynamic range of the ADC in a 10 kHz bandwidth.

Most ADCs and DACs used in high-fidelity audio systems use an extreme form of oversampling, where the internal converter may oversample by a rate of 128 or 256 times, but with very low resolution (in some cases just a 1-bit ADC!). In addition, such converters use a technique called noise shaping to push most of the quantization noise to frequencies near the sample rate, and reduce it in the audio spectrum. The noise is then removed in the decimation filter.

Although quantization error manifests itself as noise when digitizing a complex nonperiodic signal, it can show up as discrete spurious frequencies when digitizing a periodic signal. **Figure 8.32** illustrates a 1 kHz sine wave sampled with 8-bit resolution at a 9.5 kHz rate. On average there are 9.5 samples per cycle of the sine wave so that the sampling error repeats every second cycle. That 500-Hz periodicity in the error signal causes a spurious signal at 500 Hz and harmonics. As the signal frequency is changed, the spurs move around in a complicated manner that depends on the ratio of sample rate to signal frequency. In real-world ADCs, nonlinearities in the transfer function can also create spurious signals that vary unpredictably as a function of the signal amplitude, especially at low signal levels.

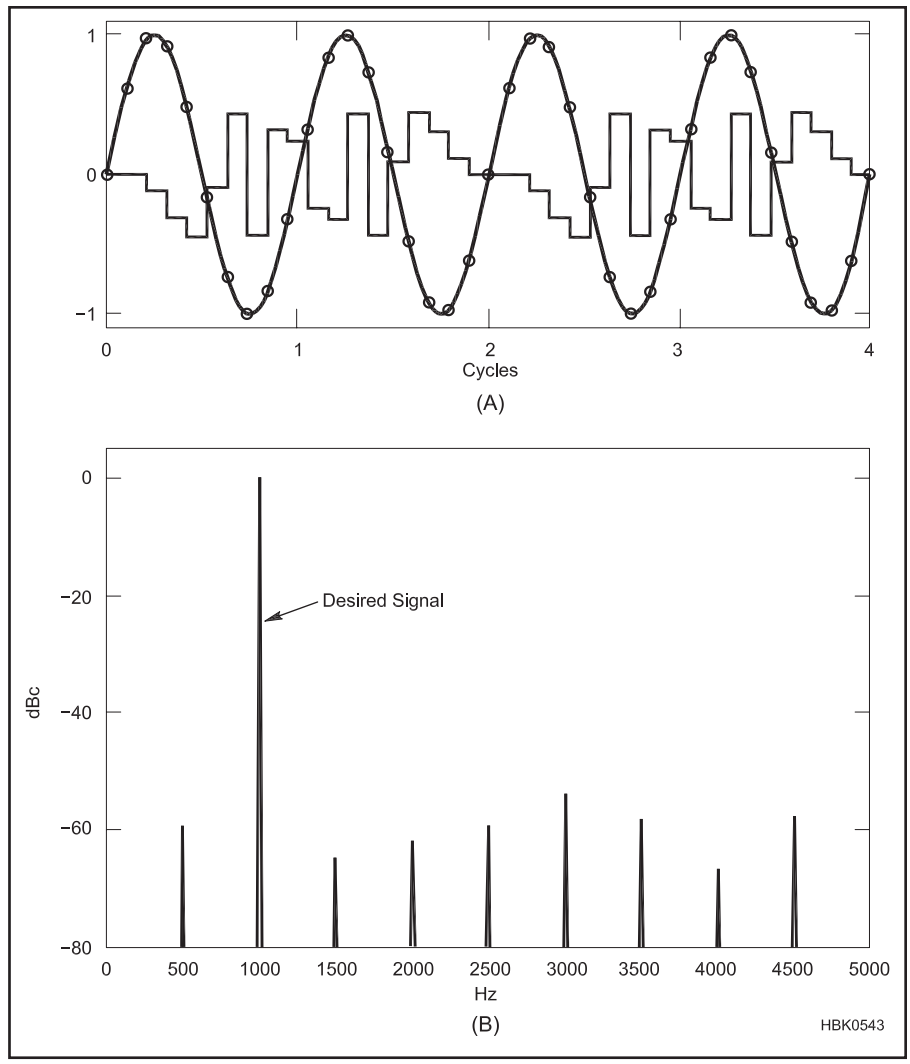


Figure 8.32 — Quantization error of a 1 kHz sine wave sampled at 9.5 kHz with 8-bit resolution (A). Sample values are indicated by circles. Also shown is the quantization error, in units of LSB. Below is the frequency spectrum, showing the spurious frequencies caused by the quantization (B).

In many applications, broadband noise is preferable to spurious signals on discrete frequencies. The solution is to add *dithering*. Essentially this involves adding a small amount of noise, typically on the order of an LSB or two, in order to randomize the quantization error. Some ADCs have dithering capability built in to improve the SFDR, even though it does degrade the SNR slightly. Dithering is also useful in cases where the input signal to an ADC is smaller than one LSB. Even though the signal would be well above the noise level after narrow-band filtering, it cannot be detected if the ADC input is always below one LSB. In many systems there is already sufficient noise from input amplifiers, other signals in the passband, as well as from the ADC itself, to cause natural dithering.

8.7 The Fourier Transform

The *Fourier transform* is the software equivalent of a hardware spectrum analyzer. It takes in a signal in the time domain and outputs a set of data (that can also be thought of as a signal) in the frequency domain that shows the spectral content of the input signal. The Fourier transform works on both periodic and non-periodic signals, but since the periodic case is easier to explain we will start with that. (Joseph Fourier was a French mathematician and physicist who introduced the transformation technique bearing his name in 1822. His contributions to physics, mathematics, and engineering were wide-ranging and at the heart of many technologies used today.)

A *periodic* signal is one that repeats every τ seconds, where τ is the period. That means that the signal can consist only of frequencies whose sinusoidal waveforms have an integer number of cycles in τ seconds. In other words, the signal is made up of sinusoids that are at the frequency $1/\tau$ and its harmonics. Fourier's insight was that you can determine if a frequency is present by multiplying the waveform by a sinusoid of that frequency and integrating the result. The result of the integration yields the amplitude of that harmonic. If the integration yields zero, then that frequency is not present.

To see how that works, look at **Figure 8.33**. For the purpose of discussion, assume the signal to be tested consists of a single tone at the second harmonic as shown at (A). The first test frequency is the fundamental, shown at (B). When you multiply the two together you get the waveform at (C). Integrating that signal gives its average value, which is zero. However if you multiply the test signal by a sine wave at the second harmonic (D), the resulting waveform (E) has a large dc offset so the integration yields a large non-zero value. It turns out that all harmonics other than the second yield a zero result. That is, the second harmonic is *orthogonal* to all the others.

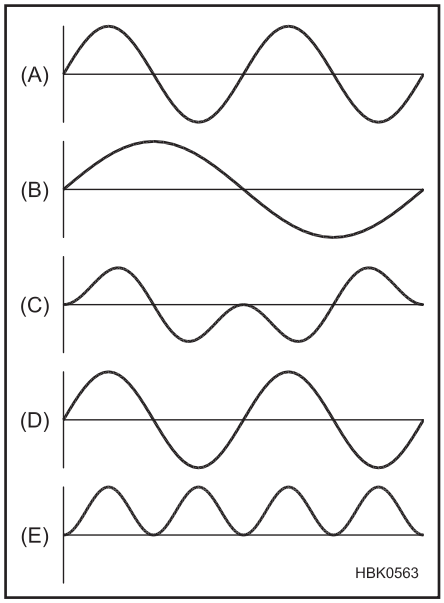


Figure 8.33 — Signal to be tested for frequency content (A). Fundamental test frequency (B). Product of signal and fundamental (C). Second harmonic test frequency (D). Product of signal and second harmonic (E).

If the test waveform included more than one frequency, each of those frequencies would yield a non-zero result when tested with the equivalent-frequency sine wave. The presence of additional frequencies does not disturb the tests for other frequencies since they are all orthogonal with each other.

You may have noticed that this method only works if the test sine wave is in phase with the one in the signal. If they are 90° out of phase, the integration yields zero. The Fourier transform therefore multiplies the signal by both a sine wave and a cosine wave at each frequency. The results of the two tests then yield both the amplitude and phase of that frequency component of the signal using the equations

$$A = \sqrt{a^2 + b^2}$$

and

$$\varphi = \arctan\left(\frac{b}{a}\right)$$

where A is the amplitude, φ is the phase, a is the cosine amplitude and b is the sine amplitude.

If one period of the signal contains, say, 256 samples, then testing a single frequency requires multiplying the signal by the sine wave and by the cosine wave 256 times and adding the results 256 times as well, for a total of 512 multiplications and additions. (This is referred to as a *multiply-accumulate* for the processing instructions that are required.) There are 128 frequencies that must be tested, since the 128th harmonic is at the Nyquist frequency. The total number of calculations is therefore 512 x 128 = 256^2 multiplications and additions. That is a general result. For any sample size, n, calculating the digital Fourier transform requires n^2 multiply-accumulates.

An excellent visual representation of Fourier's discovery is available online at **blog.matthen.com/post/42112703604/the-smooth-motion-of-rotating-circles-can-be-used**. The fundamental and harmonics are shown as a series of rotating vectors. Each vector traces out a circle, rotating at its assigned frequency and phase. When vectors are added together the result is the desired waveform!

8.7.1 The Fast Fourier Transform (FFT)

The number of calculations grows rapidly with sample size. Calculating the Fourier transform on 1024 samples requires over a million multiply-accumulate cycles. However, you may notice that there is some redundancy in the calculations. When testing the second harmonic, for example, each of the two cycles of the test sine wave is identical. It would be possible to pre-add signal data from the first and second halves of the sequence and then just multiply once by a single cycle of the test sine wave. Also, the first quarter cycle of a sine wave is just a mirror-image of the second quarter cycle and the first half is just the negative of the second half.

In 1965, J. W. Cooley and John W. Tukey published an algorithm that takes advantage of all the symmetries inherent in the Fourier transform to speed up the calculations. The *Cooley-Tukey algorithm*, usually just called the *fast Fourier transform* (FFT), makes the number of calculations proportional to $n\log_2(n)$ instead of n^2. For a 1024-point FFT, the calculation time is proportional to $1024\log_2(1024) = 10{,}240$ instead of $1024^2 = 1{,}048{,}576$, more than a 100-times improvement.

You'll notice that sample sizes are usually a power of two, such as $2^7 = 128$, $2^8 = 256$ and $2^9 = 512$. That is because the FFT algorithm is most efficient with sequences of such sizes. The algorithm uses a process called *radix-2 decimation in time*, that is, it first breaks the data into two chunks of equal size, then breaks each of those chunks into two still-smaller chunks of equal size, and so on. It is possible to squeeze even a little more efficiency out of the algorithm with a *radix-4* FFT which is based on decimation by four instead of by two. That is why you often see sample sizes that are powers of four, such as $4^3 = 64$, $4^4 = 256$ and $4^5 = 1024$. Other variations on the algorithm include decimation in frequency rather than time, mixed-radix FFTs that use different decimation factors at different stages in the calculation, and in-place calculation that puts the results into the same storage buffer as the input data, saving memory. The latter method causes the order of the output data to be scrambled by bit-reversing the address words. For example, address 01010000 becomes 00001010.

The heart of most FFT algorithms is a data flow pattern called a *butterfly*. **Figure 8.34** shows a two element butterfly. The input con-

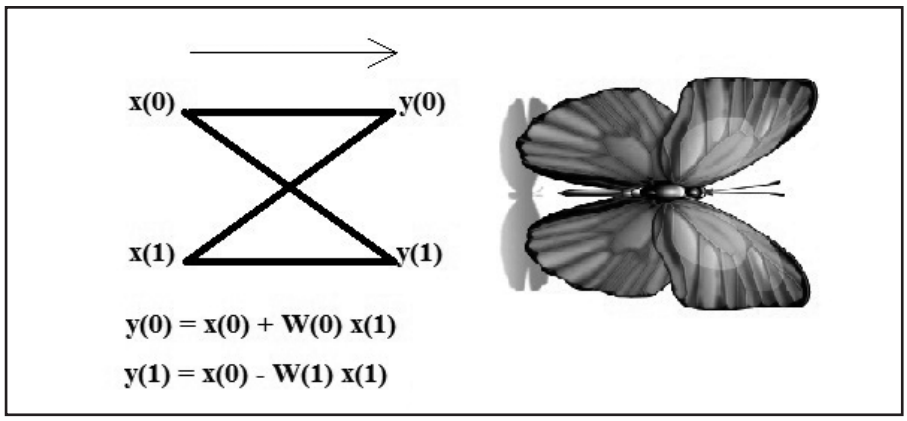


Figure 8.34 — A two-element "butterfly" signal flow diagram. The crossing of the lines makes the diagram look like stylized butterfly wings.

tains two elements x(0) and x(1), and the output has two elements y(0) and y(1). The equations for the butterfly are:

$$y(0) = x(0) + x(1)\, W^k$$

$$y(1) = x(1) - x(1)\, W^k$$

where W^k is the sum of the sine and cosine harmonics cos(2k) + jsin(2k).

Figure 8.35 shows the data flow diagram of the scrambled in/natural out (the output is ordered from lowest to highest element). The number in the circle at each stage indicates the value of k for W^k. Notice that the first stage of calculation performs four 2×2 butterfly operations, the second is two 4×4 butterfly operations, and the final is one 8×8 butterfly. Each stage has fewer butterfly operations so it is called a *decimation in time* algorithm.

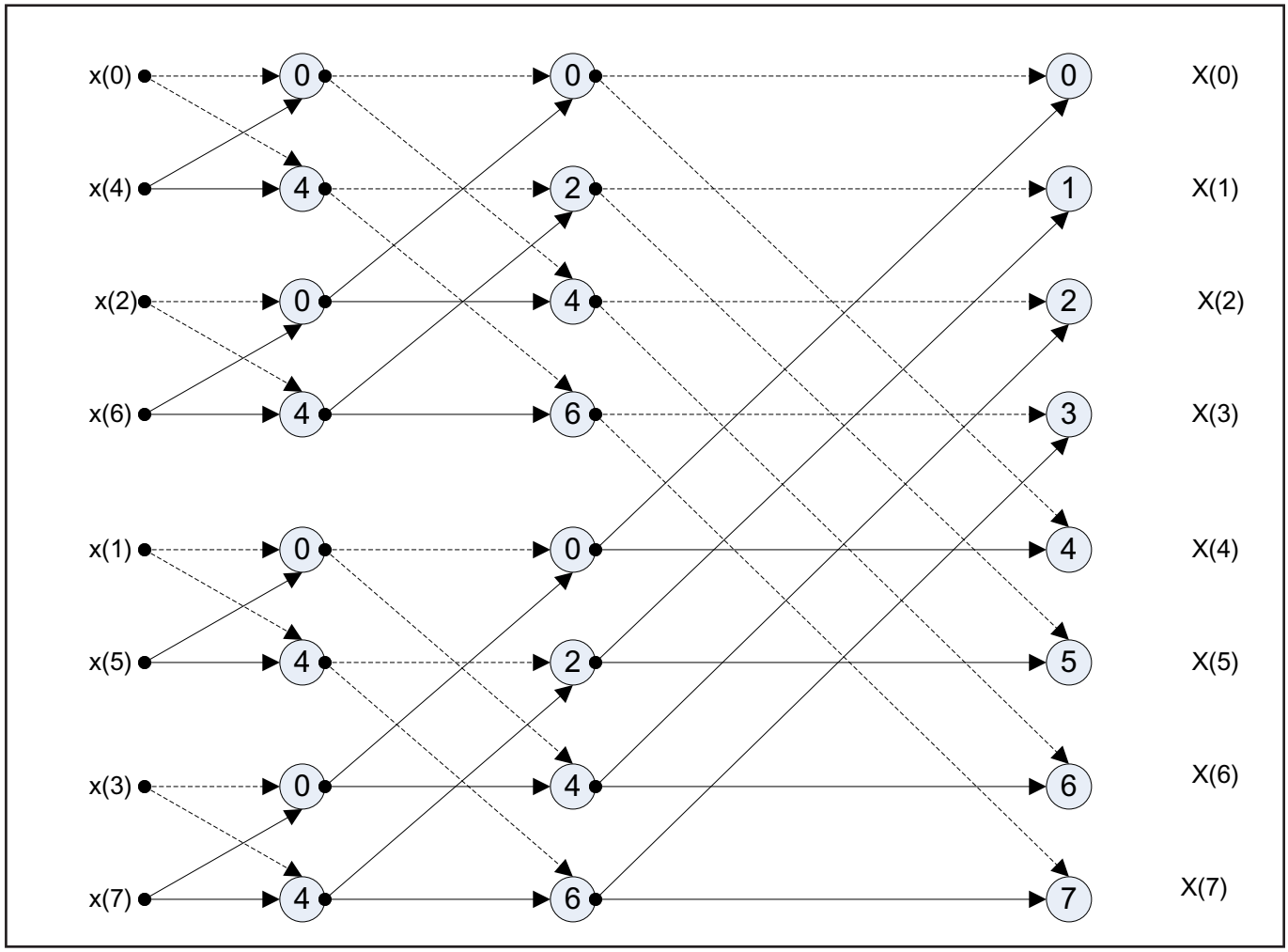
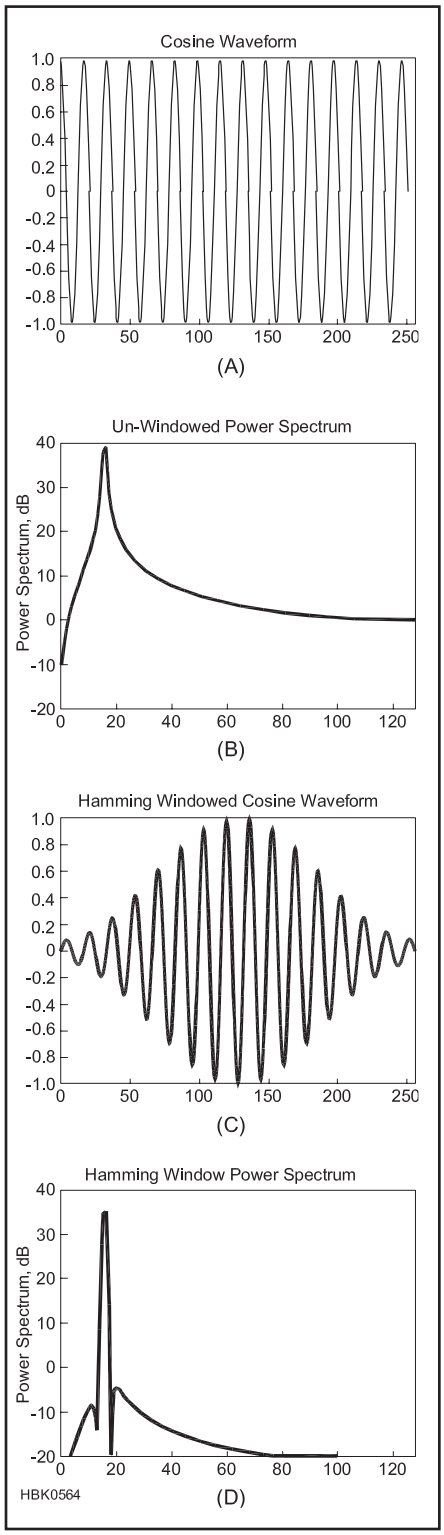


Figure 8.35 — A complete flow diagram for an 8-element FFT calculation. Each solid line represents a direct contribution of the element to the left. Each dashed line represents a contribution that is multiplied by the complex operator W^k. Each number in a circle represents the k value used for the multiplication. This example requires just 24 complex multiplication operations instead of a full 256 calculations.

8.7.2 Non-periodic Signals

So far we have assumed that the signal to be transformed is periodic, so that there is an integer number of cycles of each sine wave harmonic in the sequence. With a non-periodic signal, that is not necessarily so. The various frequencies in the signal are not exact harmonics of 1/ and are no longer orthogonal to the test frequencies. The result is *spectral leakage*; a single frequency in the signal may give a non-zero result when tested at a number of different harmonic frequencies. In **Figure 8.36**, (B) illustrates the FFT of a single sine wave at a non-harmonic frequency. You can see that the spurious response extends quite far from the actual frequency.

Those far-out spurious responses are primarily caused by the abrupt termination of the signal at the edges of the sequence. The spectrum can be cleaned up considerably by tapering the edges with a window, in the manner of digital filter coefficients as described in the **Analog and Digital Filtering** chapter. In fact, the same window functions are used for both. Figure 8.36C illustrates the result of applying a Hamming window to the signal in (A) and the resulting improved spectrum is shown at (D). Just as with FIR filters, different windows excel in different areas. Windows with a gradual transition to zero at the edges do a better job of suppressing spurious responses but smear adjacent spectral lines, analogous to using a wider resolution bandwidth in an analog spectrum analyzer. Windows with a wider center section and a more abrupt transition to zero at the edges have less smearing but more spurious responses.



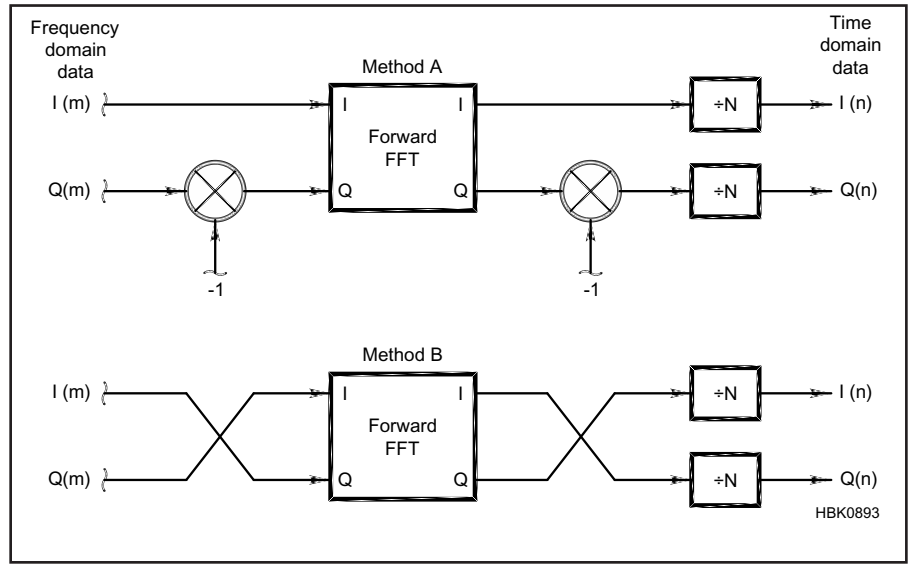
◀**Figure 8.36 — Illustrating the use of windowing to minimize spectral leakage, the figures show (A) a cosine waveform not at a harmonic frequency, (B) the resulting unwindowed power spectrum, (C) the same cosine waveform with a Hamming window, and (D) the much narrower power spectrum of the windowed waveform.**

8.7.3 The Inverse Fourier Transform (IFT)

It is also possible to run the Fourier transform "in reverse," changing frequency-domain data back into time-domain data. The IFT takes a collection of sine wave amplitude and phases and adds them together to create a waveform just as the "forward" Fourier transform changes the waveform into the collection of sine waves.

Just as clever ways of performing the Fourier transform led to the FFT, similarities in the calculations for the FFT and IFT allow the same calculations that perform the "forward" FFT to be used to perform the IFT. **Figure 8.37** shows two methods of implementing an IFT using the FFT algorithm on the I and Q channel data. Both methods have as their input the frequency-domain data, I(m) and Q(m), which represent the amplitude and phase of sinusoidal components (harmonics). The output is a series of time-domain samples, I(n) and Q(n), which represent the waveform. The divide-by-N functions are scaling factors required by the calculations. Advanced methods are even able to perform the FFT and IFT simultaneously on different streams of data, allowing one specialized set of hardware to do double duty.

Transceivers don't generate transmittable signals using the IFT which is included in this discussion for completeness. Transmit chains usually perform I/Q modulation with digital data on the I and Q channel carriers, filter the output (either with analog or digital filters), and amplify it conventionally to useful levels. Nevertheless, the elegance of using the very same algorithm to convert between the time and frequency domains illustrate the deep mathematical connection between time and frequency.



▲**Figure 8.37 — Two methods of performing the Inverse Fourier Transform (IFT) using the "forward" FFT algorithm.**

8.8 References and Bibliography

References

1. FCC Report and Order FCC 01-264 authorized software-defined radios in the commercial service in 2001. Search **www.fcc.gov** for the document.
2. Hershberger, D., W9GR, "Low-Cost Digital Signal Processing for the Radio Amateur," *QST*, Sep. 1992, pp. 43 – 51.
3. Youngblood, G., AC5OG, "A Software-Defined Radio for the Masses," *QEX*; Part 1, July/Aug. 2002; Part 2, Sep./Oct. 2002; Part 3, Nov./Dec. 2002; Part 4, Mar./Apr. 2003.
4. Tayloe, D., N7VE, "Notes on 'Ideal' Commutating Mixers (Nov./Dec. 1999)" Letters to the Editor, *QEX*, Mar. 2001, p. 61.
5. The Microchip IDE software can be downloaded free at **www.microchip.com**.
6. For more details on using the Raspberry Pi GPU for SDR applications, see: **www.raspberrypi.org/blog/accelerating-fourier-transforms-using-the-gpu**
7. Blanchard, R., W6UYG, "Sugar-Coated Single Sideband," *QST*, Oct. 1952, pp. 38 – 39, 128.
8. Weaver, Jr, D. K., "A Third Method of Generation and Detection of Single-Sideband Signals" *Proc. IRE*, Dec. 1956.
9. Wright, Jr., H., W1PNB, "The Third Method of S.S.B.," *QST*, Sep. 1957, pp. 11 – 15.

Bibliography

DSP SOFTWARE TOOLS

The following books are referenced in the DSP files in the online content.

Alkin, O., *PC-DSP* (Prentice Hall, 1990).

Kamas, A. and Lee, E., *Digital Signal Processing Experiments* (Prentice Hall, 1989).

Momentum Data Systems, Inc., *QEDesign*, Costa Mesa, CA, 1990.

Stearns, S. D. and David, R. A., *Signal Processing Algorithms in FORTRAN and C* (Prentice Hall, 1993).

TEXTBOOKS

Hayward, W., Campbell, R., and Larkin, B., *Experimental Methods in RF Design* (ARRL, 2009). See Chapter 10 "DSP Components" and Chapter 11 "DSP Applications in Communications."

Lathi, B.P., *Signal Processing and Linear Systems* (Oxford University Press, 1998).

Lyons, R., *Understanding Digital Signal Processing* 3rd Edition (Pearson, 2012).

Madisetti, V. K. and Williams, D. B., Ed., *The Digital Signal Processing Handbook* (CRC Press, 1998).

Mar, A., Ed., *Digital Signal Processing Applications Using the ADSP-2100 Family, Vol. I* (Prentice Hall, 1990). While oriented to the ADSP-2100, there is much good general information on DSP algorithms. Both volume I and II are available for free download on the Analog Devices website, **www.analog.com**.

Parks, T. W. and Burrus, C.S., *Digital Filter Design* (John Wiley and Sons, 1987).

Sabin, W. E. and Schoenike, E. O., Ed., *HF Radio Systems and Circuits* 2nd Edition (Noble Publishing Corp, 1998).

Widrow, B. and Stearns, S. D., *Adaptive Signal Processing* (Prentice Hall, 1985).

Zverev, A. I., *Handbook of Filter Synthesis* (John Wiley and Sons, 1967).

ARTICLES AND PAPERS

Ahlstrom, J., N2ADR, "An All-Digital SSB Exciter for HF," *QEX*, May/Jun 2008, pp. 3 – 10.

Ahlstrom, J., N2ADR, "An All-Digital Transceiver for HF," *QEX*, Jan./Feb. 2011, pp. 3 – 8.

Åsbrink, L., SM5BSZ, "Linrad: New Possibilities for the Communications Experimenter," *QEX*, Part 1, Nov./Dec. 2002; Part 2, Jan./Feb. 2003; Part 3 May/June 2003.

Bloom, J., KE3Z, "Measuring SINAD Using DSP," *QEX*, June 1993, pp. 9 – 18.

Bloom, J., KE3Z, "Negative Frequencies and Complex Signals," *QEX*, Sep. 1994, pp. 22 – 27.

Bloom, J., KE3Z, "Correlation of Sampled Signals," *QEX*, Feb. 1996, pp. 24 – 28.

Brannon, B., "Basics of Digital Receiver Design," *QEX*, Sep./Oct. 1999, pp. 36 – 44.

Cahn, H., WB2CPU, "Direct Digital Synthesis — An Intuitive Introduction," *QST*, Aug. 1994, pp. 30 – 32.

Cowling, S., WA2DFI, "Hands-On SDR," *QEX*, columns beginning Sep. 2014.

Danzer, P., N1II, "A Practical Demonstration of Fourier Series," *QST*, Apr. 2007, p. 37.

de Boer, P. T., PA3FWN, "How RTL-SDR dongles work," **www.pa3fwm.nl/technotes/tn20.html**

Dobranski, L., VA3LGD, "The Need for Applications Programming Interfaces (APIs) in Amateur Radio," *QEX*, Jan./Feb. 1999, pp. 19 – 21.

Gradijan, S., WB5KIA, "Build a Super Transceiver — Software for Software Controllable Radios," *QEX*, Sep./Oct. 2004, pp. 30 – 34.

Hightower, M., KF6SJ, "Simple SDR Receiver," *QEX*, Mar./Apr. 2012, pp. 3 – 8.

Hill, C., W5BAA, "SDR2GO: A DSP Odyssey," *QEX*, Mar./Apr. 2011, pp. 36 – 43.

Kester, W., "SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor," Analog Devices, MT-003, **www.analog.com/media/en/training-seminars/tutorials/MT-003.pdf.**

Kester, W., "Which ADC Architecture Is Right for Your Application?," Analog Devices, **www.analog.com/media/en/analog-dialogue/volume-39/number-2/articles/the-right-adc-architecture.pdf.**

Mack, R., W5IFS, "SDR: Simplified," *QEX*, columns beginning Nov. 2009.

Martin, K., "Complex Signal Processing is NOT — Complex," **www.mikrocontroller.net/attachment/151612/complex_signals.pdf.**

Nickels, R., W9RAN, "Cheap and Easy SDR," *QST*, Jan. 2013, pp. 30 – 35.

Scarlett, J., KD7O, "A High-Performance Digital Transceiver Design," *QEX*, Part 1, July/Aug. 2002; Part 2, Mar./Apr. 2003.
Smith, D., KF6DX, "Signals, Samples and Stuff: A DSP Tutorial, Parts 1 – 4," *QEX*, Mar./Apr. 1998 to Sep./Oct. 1998.
Stephensen, J., KD6OZH, "Software Defined Radios for Digital Communications," *QEX*, Nov./Dec. 2004, pp. 23 – 35. [Open-source platform]
Veatch, J., WA2EUJ, "The DSP-610 Transceiver," *QST*, Aug. 2012, pp. 30 – 32.
Ward, R., WA4ZZU, "Basic Digital Filters," *QEX*, Aug. 1993, pp. 7 – 8.
Youngblood, G., AC5OG, "A Software-Defined Radio for the Masses," *QEX*; Part 1, July/Aug. 2002; Part 2, Sep./Oct. 2002; Part 3, Nov./Dec. 2002; Part 4, Mar./Apr. 2003.

SDR AND DSP RECEIVER TESTING

Hakanson, E., "Understanding SDRs and their RF Test Requirements," Anritsu Application Note.
MacLeod, J.R. et al., "Software Defined Radio Receiver Test-Bed," IEEE Vehicular Tech Conf, Fall 2001, VTS 54th, Vol. 3, pp. 565 – 1569.
Sierra, R, Rhode & Schwarz, "Challenges In Testing Software Defined Radios," SDR Forum Sandiego Workshop, 2007.
Sirmans, D. and Urell, B., "Digital Receiver Test Results," Next Generation Weather Radar Program.
"Testing and Troubleshooting Digital RF Communications Receiver Designs," Agilent Technologies Application Note 1314. **www.keysight.com/us/en/assets/7018-06706/application-notes/5968-3579.pdf**