An Introduction to Digital Signal Processing

QST July 1991, pp. 35-37

An Introduction to Digital Signal Processing

DSP makes headlines even in an age where *digital* seems to be applied to just about everything electronic—and now it's part of Amateur Radio. Here's a look at what DSP is and can do.

By Bruce S. Hale, KB1MW/7
2238 168th Ave NE
Bellevue, WA 98008

Digital signal processing, or DSP, is the hottest thing in ham conversations these days. Reminds you of the first time you heard about packet radio, right? Everyone was talking about it, and you felt like *you* were the only one who didn't understand it. Later on, you found out that just about everyone was as mystified (at first) as you were. Now, you've probably got some packet equipment in your station, and you can't imagine how you got along without it.

Get ready for the same thing with DSP. You're just hearing about the first new DSP ham gear, but before too long, DSP systems will be just about everywhere. There's another parallel, too—just as packet was in use in commercial communications before it hit the ham scene, DSP has been around military and commercial electronics for quite a while.

What Does It Mean?

So what *is* DSP anyway, and what's it going to do for you? DSP is *digital signal processing*. You know what signals are—that's the easy part. And signal processing—any time a signal passes through some electronic or mechanical system and gets changed (by design), the signal is processed (accidental changes are usually called *interference* or *noise*). Filters are a good example of simple signal-processing systems: Signals go in one end of a filter and comes out the other end changed, or processed.

What about the *digital* part? These days, just about everything electronic seems to be billed as digital (or "digital ready" or "digital compatible"). What's it all mean? Simply put, modern electronic systems work with two kinds of signals: analog and digital. Analog signals are what we're all used to. An analog signal is continuously variable, like the waveform at A in Fig 1. Digital signals, on the other hand, are made up of a stream of single values; one value for each small interval, like the signal at C in Fig 1. We can convert an analog signal into a digital signal by *sampling* the analog signal at regular intervals (Fig 1B).

How often do we have to take these samples? Well, if we don't take them often enough, we can't get an accurate digital representation of the analog signal. If our digital representation of the analog signal isn't accurate, we can't accurately reconstruct the analog signal from our digital data. It turns out that we have to take the samples at a frequency *at least* twice the highest frequency contained in the signal. This means that if we're sampling an audio signal with a 20-kHz upper limit, our sampling rate must be at least 40 kHz. (The 44.1-kHz sampling rate in a digital CD [*compact disc*] audio system allows for a little overhead). The device that does the sampling is called an *analog-to-digital converter (ADC)*.

So a digital signal is just a list of numbers that represent the value of the signal at each sampling interval. What can we do with this long list of numbers? We can reconstruct the original analog signal with a *digital-to-analog converter (DAC)*. But what if we change the digital data before we send it to the DAC? Here's where computers come in (you knew you'd run into computers in here somewhere, right?), because computers are great at working with long lists of numbers. We can use a computer to change the numbers, and create a new, *processed* signal.

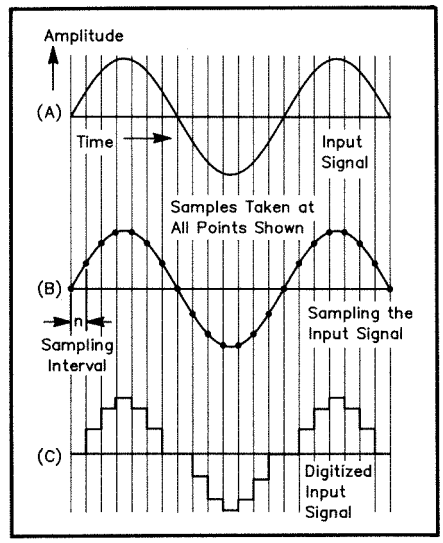

Fig 1—*Digitizing* an analog signal (A) involves *sampling* that signal at regular intervals (B)—at a rate fast enough to ensure a reasonably accurate digital map of that signal (C). Digitization changes analog signals to a form DSP systems can handle.

In a nutshell, that's digital signal processing. An analog-to-digital converter (ADC) converts the continuously variable analog signal into a discrete digital signal, the digital data is processed (changed in some useful way) and then a digital-to-analog converter (DAC) converts the digital signal back into an analog signal. (Of course, DSP can also be applied to signals that start out life in digital form; such a system needs no ADC. And a DAC isn't necessary if the processed data need not be converted to analog form.)

All we have to do to implement a digital signal processing system is come up with an equation that expresses the signal we'd like in terms of the signal we've got. A *digital averager* can serve as a simple DSP example. In a four-sample averager, the output corresponding to any sampling interval consists of the average of the input at that interval and the last three input values. Fig 2 shows a system like this. Averaging the samples tends to smooth out rapid data fluctuations. (For example, averaging can smooth a large noise spike by averaging it with the three surrounding values). Slower data deviations pass through the averager unaffected. In effect, the averager serves as a low-pass filter.

Other DSP systems can be used to implement high-pass filters, audio mixers, equalizers, compressors and expanders. (The list goes on, but you get the idea.) It's even possible to transform *time-domain* data, where the value of each sample represents the signal level at a certain time, into *frequency-domain* data, where the sample value represents the signal level at a certain frequency. Such a process allows a DSP system to control a signal's spec-

Let's Play Filter

Digital signal processing is neat, but what *useful* job could it do in, say, a garden-variety ham transceiver? The answer—"any signal-processing job describable with numbers"—still doesn't let us get a handle on what DSP can do. We're not digital engineers; we need a concrete example. Here's one:

Your new MF/HF transceiver has opened your ears to shortwave listening—so much so that you want to improve how the radio receives AM-bandwidth signals. For a bit under a hundred bucks, you buy a made-for-the-job, plug-in *analog* signal processor that, once installed, helps the radio select shortwave signals you want from signals you don't. The processor comes in a small, silver box—a box labeled *crystal filter*.

No kidding, your radio's intermediate-frequency (IF) filters serve as analog signal processors that change radio signals fed into them into a form more useful to you. IF filters do this by passing more energy at some frequencies (frequencies in the filter *passband*) than others (frequencies in the filter *stopband*). If we can describe that job with numbers, we can tell a DSP system to do the same job.

And so we will. We'll just assume that our IF filter is so stupid it needs to be *told* how to behave! Fig A can help us issue the orders in coordinate form: The graph's vertical scale talks signal attenuation in decibels (dB) relative to the filter's minimum attenuation. The horizontal scale shows IF-signal frequency in kilohertz (kHz) relative to the filter's center frequency.

Order 1: "Filter, assign signals from −1.5 to +1.5 kHz of your center frequency the attenuation value of 0." Signals at these frequency values go into and come out of the filter undiminished. (Sure, a real IF filter exhibits some *insertion loss* even in its passband, but I won't tell this filter if you won't.)

Order 2: "Filter, assign signals +2.5 and −2.5 kHz away from your center frequency the attenuation value of 2." The filter knocks signals at these spacings from center down by 2 dB. So far, so good.

Order 3: "Filter, assign signals +3 and −3 kHz away from your center frequency the attenuation value of 6." Cool! Our filter exhibits a −6-dB bandwidth of 6 kHz—pretty much what standard accessory AM filters do these days.

Order 4: "Filter, assign signals +3.5 and −3.5 kHz away from your center frequency the attenuation value of 16."

Order 5: "Filter, assign signals +4 and −4 kHz away from your center frequency the attenuation value of 30." Now we're cooking—the filter cuts signals 4 kHz away from center by 30 dB—by five S units, if, as many hams do, you define an S unit as 6 dB.

Order 6: "Filter, assign signals +4.5 and −4.5 kHz away from your center frequency the attenuation value of 42."

Order 7: "Filter, assign signals at and greater than +5 and −5 kHz away from your center frequency the attenuation value of 60." *Yes!* Our filter exhibits a −60-to-−6-dB shape factor of 1.67—quite good for cutting 5-kHz heterodynes during shortwave-broadcast reception.

Of course, we've vastly shortened the command list necessary to *actually* tell our filter how to implement the smooth passband curve shown in Fig A. We've defined only 14 points and a short segment of that curve, but geometry tells us that a line consists of an infinite number of points. Actually completing that job—or commanding the filter how to connect and smooth our data-point dots in a valid and useful way—*could* be done by issuing more orders. Doing so would run us into a *QST* page-space problem, so we'll consider our filter fully trained for now.

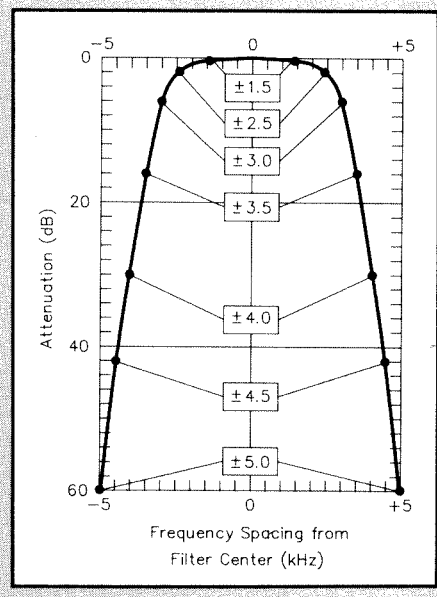

Fig A—Plot an IF filter's attenuation-versus-frequency response in terms of frequency-versus-attenuation coordinates, and you're well on the way to understanding how DSP can synthesize almost any filter response imaginable.

IF-filter action *can* be defined in terms of numbers: The filter grabs signals at certain frequencies, changes their amplitude (as appropriate) relative to their frequency, and spits out the result—high-speed and real-time. (Yes, a real filter delays signals somewhat—sometimes to a degree that varies with frequency—but *this* filter won't do what I don't tell it to.)

No doubt about it, digital signal processing can do the same job, hardware and software allowing. We'd just provide our DSP-system computer with a list of the commands necessary to manipulate signal amplitude as a function of input frequency. And think of the possibilities! Merely by providing our DSP system with the right attenuation-versus-frequency commands, we could synthesize, say, a 20-Hz-wide, 60-dB-deep notch 1.73 kHz above the filter's center frequency. We could move the passband and/or notch; change their width; round, square or slope their sides; add or subtract notches—*merely by issuing new commands.* We could even have the system sense, track and reject steady carriers—data is data, remember, and all we need to do is tell our DSP system how to manipulate that data into a form more useful to us. Yes, the prospects of DSP are exciting!

In the realm of filter synthesis, DSP is a dream come true. Doing things of a nature that *only* DSP can touch—recovering signals that seem totally obscured by noise; synthesizing and recovering signals modulated by *any combination* of amplitude, frequency and phase values; voice synthesis, recognition and recording; sensing, shaping and using data patterns where utter randomness seems to hold sway, DSP can perform miracles. Ongoing hardware and software advances allow DSP to work more and more of that magic in real time at increasingly higher frequencies. Some of that DSP muscle may already be at work in a ham station near you.

—*David Newkirk, WJ1Z*

trum, or to extract certain frequency components (like frequency-shift-keyed [FSK] data) from background noise.

It's Not New

Digital signal processing is not new. Every time you look at an image from the *Voyager* spacecraft, you're looking at the results of DSP. The picture you see arrived from the spacecraft via radio as a stream of digital data. After recording the data and transforming it into an image with computers, image-processing engineers at NASA's Jet Propulsion Laboratory (JPL) cleaned it up with DSP, perhaps combining several images—again with processing—to make a single final image, like the ones you've seen on television. So essential is DSP to this process that you probably wouldn't even recognize the original image as having anything in common with the

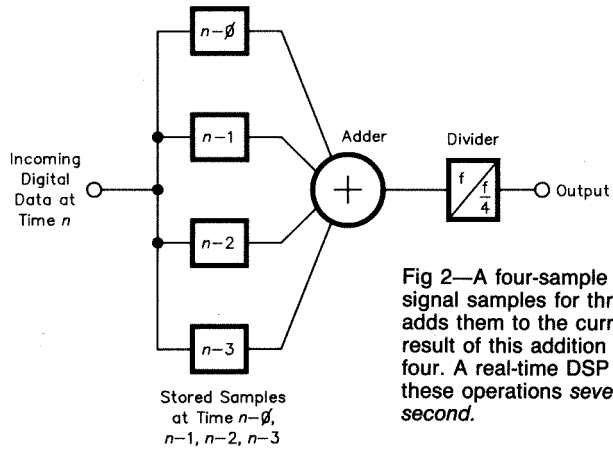

Fig 2—A four-sample averager stores signal samples for three intervals and then adds them to the current-sample data. The result of this addition is then divided by four. A real-time DSP system must perform these operations *several thousand times a second.*

finished picture! And this process didn't start with *Voyager*: Almost every picture transmitted by spacecraft since the beginning of the space program arrived on earth as some form of digital data. The engineers receiving these signals processed them into useful images using DSP techniques.

If digital signal processing isn't new, why are so many of us so excited about it now? The *speed* of modern DSP hardware, for one thing. DSP requires a *lot* of computing power. Until recently, processing a single spacecraft image could take hours, or even days. Today's DSP ICs can do such processing *in real time*. This means that you can put a signal into one end of a DSP system based on these new ICs and immediately get the processed signal out the other end, instead of waiting around while the computer swallows the signal, thinks it over and does the processing. These new chips can be used to do signal processing previously possible only with analog hardware—at radio frequencies, for example.

The big deal about this is that using an averaging filter real-time in a radio system means that we need to do the averaging *several thousand times each second* if we want to keep up with the data. Even though the calculations involved are simple and repetitive, an ordinary computer just can't keep up. To solve this problem, engineers design dedicated DSP ICs that do one job blindingly fast. Because dedicated ICs have a single purpose in life, they can be optimized to do simple, repetitive calculations at very high data-transmission rates. A typical DSP IC can perform 25 to 33 *million* calculations in one second!

Because digital processing takes place inside a computer—a highly specialized computer—it all happens under the control of a program. Where analog engineers work with (for instance) resistors and capacitors, DSP engineers work with mathematics and computer programs. This is another advantage DSP offers over analog signal processing: If we want to change the way our DSP system handles signals, all we need to do is change its program. If the program exists in an EPROM, we just pull out the old EPROM and pop in a new one (or reprogram the old). Presto, we've got a different filter or a better modem. This allows creative programmers to add new features to existing hardware—even after you've installed that hardware in your home or ham station.

It's All in the Technique

All this may sound simple, but constructing a real-time DSP system *isn't* simple. True to its title, this article provides only the most basic introduction to DSP. Just the mathematics involved in DSP can fill several textbooks and many hours of engineering-course work! Control programs for dedicated DSP ICs can be fairly difficult to write and hard to debug.

Digital signal processors *are* available in microcomputers today, however. The NeXT computer contains a DSP chip, and plug-in DSP boards are available for other microcomputers. You can expect to see more computers with built-in DSP systems as high-powered multimedia (combined sound and motion video) computers hit the market.

Digital signals are everywhere—people are using digital fax transceivers, listening to digital music systems (using compact discs and digital audio tape), and all kinds of communications are routed in digital form. That brings up yet another benefit of DSP: to a DSP system, data is data. It makes no difference whether the data came from a fax image, a telephone conversation or a spacecraft speeding through the solar system. All of this data consists merely of streams of numbers; *all a DSP system does is change the numbers.* The key to DSP's rising importance is *how*, and how fast, it can change numbers in a useful way.

. . . and DSP Can Be Fun

Does the USS *Enterprise* work 20-meter packet? In late 1989, digital signal processing helped decode mysterious movie audio with a strangely familiar ring. *Gateway*, the ARRL packet-radio newsletter (now part of *QEX*), reported it this way on January 5, 1990:

Packet Radio Stars in Trek Flick

Several months ago, Harold Price, NK6K, challenged me to demodulate what he thought might be HF packets in the motion picture *Star Trek IV*. During the scene where Scotty is valiantly trying to beam both Chekov and Uhura back from the USS *Enterprise*, Scotty is having a hard time hearing them. One of the sources of interference appeared to be HF packet radio.

Always being one to rise to a challenge, I took on the job of doing some fancy DSP footwork. Almost from the first, I was certain that it must be an HF packet because my very first demodulation attempt revealed flags before the start of a frame and end of frame. I knew it was HDLC of some variety.

Several things impeded the effort, including Scotty's voice on top of the packets and some SSB nearly on top of the signal. All of this had to be filtered out. I spent an hour of time on the Cray-2 at work and used the fanciest FSK demodulator I could write, and I finally had the noisy baseband signal plotted on paper in front of me. I did my best to get an integral number of samples per baud as the signal was very noisy, and though the bits could be made out by eye, I could tell that it was going to take another hour of Cray-2 time to get the clock recovered and to make good bit decisions. In a couple of places, HDLC showed me what were clearly bit errors, and these could be done by eye as well.

After the filtering and building a demodulator for the badly mistuned signal (it was almost 900 Hz below "normal"), I took the bits to Phil Karn, KA9Q, and he decoded the NRZI data, proving beyond a shadow of a doubt that it was indeed an HF Amateur Radio packet. It was WA8ZCN-ø sending an RR for NR3 on 20 meters. I got Bill Harrigill, WA8ZCN, on the phone and he agrees that it was probably him. Thanks, Harold, for the challenge, and Phil for the help.—*Bob McGwier, N4HY, via CompuServe's HamNet*