

## **Digital Dharma of Audio A/D Converters**

- **Data Conversion**
- **Binary Numbers**
- **The Story of Harry & Claude**
- **Quantization**
- **Successive-Approximation**
- **PCM & PWM**
- **Delta-Sigma Modulation & Noise Shaping**
- **Dither**
- **Life After 16 – A Little Bit Sweeter**

**Dennis Bohn  
Rane Corporation**

**RaneNote 137  
© 1997 Rane Corporation**

### **Introduction**

Among the many definitions for the wonderful word “dharma” is the essential function or nature of a thing. That is what this note is about: the essential function or nature of audio analog-to-digital (A/D) converters. Like everything else in the world, the audio industry has been radically and irrevocably changed by the digital revolution. No one has been spared. Arguments will ensue forever about whether the true nature of the real world is analog or digital; whether the fundamental essence, or *dharma*, of life is continuous (analog) or exists in tiny little chunks (digital). Seek not that answer here. Here we shall but resolve to understand the dharma of audio A/D converters.

## Data Conversion

It is important at the onset of exploring digital audio to understand that once a waveform has been converted into digital format, *nothing can inadvertently occur to change its sonic properties*. While it remains in the digital domain, it is only a series of digital words, representing numbers. Aside from the gross example of having the digital processing actually fail and cause a word to be lost or corrupted into none use, nothing can change the sound of the word. It is just a bunch of “ones” and “zeroes.” There are no “one-halves” or “three-quarters”. The point being that *sonically*, it begins and ends with the conversion process. Nothing is more important to digital audio than data conversion. Everything in-between is just arithmetic and waiting.

That’s why data conversion is really that important. Everything else quite literally is just details. We could go so far as to say that data conversion is the art of digital audio while everything else is the science, in that it is data conversion that ultimately determines whether or not the original sound is preserved (and this comment certainly does not negate the enormous and exacting science involved in truly excellent data conversion.)

Since analog signals continuously vary between an infinite number of states and computers can only handle two, the signals must be converted into *binary digital words* before the computer can work. Each digital word represents the value of the signal at one precise point in time. Today’s common word length is 16-bits or 32-bits. Once converted into digital words, the information may be stored, transmitted, or operated upon within the computer.

In order to properly explore the critical interface between the analog and digital worlds, it is necessary to review a few fundamentals and a little history.

## Binary Numbers

Whenever we speak of “digital,” by inference, we speak of computers (throughout this paper the term “computer” is used to represent any digital-based piece of audio equipment). And computers in their heart of hearts are really quite simple. They only can understand the most basic form of communication or information: yes/no, on/off, open/closed, here/gone – all of which can be symbolically represented by two things – any two things. Two letters, two numbers, two colors, two tones, two temperatures, two charges – it doesn’t matter. Unless *you* have to build something that will recognize these two states – now it matters. So, to keep it simple we choose two numbers: one and zero ... a “1” and a “0.” Officially this is known as binary representation, from Latin *bini* two by two. In mathematics this is a base-2 number system, as opposed to our decimal (from Latin *decima* a tenth part or tithe) number system, which is called base-10 because we use the ten numbers 0-9.

In binary we use only the numbers 0 and 1. “0” is a good symbol for no, off, closed, gone, etc., and “1” is easy to understand as meaning yes, on, open, here, etc. In electronics it is easy to determine whether a circuit is open or closed, conducting or not conducting, has voltage or doesn’t have voltage. Thus the binary number system found use in the very first computer, and nothing has changed today. Computers just got faster and smaller and cheaper, with memory size becoming incomprehensibly large in an incomprehensibly small space.

One problem with using binary numbers is they become big and unwieldy in a hurry. For instance, it takes six digits to express my age in binary, but only two in decimal. But, in binary, we better not call them “digits” since “digits” implies a human finger or toe, of which there are ten, so confusion reigns. To get around that problem John Tukey of Bell Laboratories dubbed the basic unit of information (as defined by Shannon – more on him later) a *binary unit*, or “binary digit” which became abbreviated to “bit.” A bit is the simplest possible message representing one of two states.

So, I’m 6-bits old! Well, not quite. But it takes 6-bits to express my age as 110111. Let’s see how that works. I’m fifty-five years old. So in base-10 symbols that is “55,” which stands for 5-1s plus 5-10s. You may not have ever thought about it, but each digit in our everyday numbers represents an additional power of 10 beginning with 0. That is, the first digit represents the number of 1s ( $10^0$ ), the second digit represents the

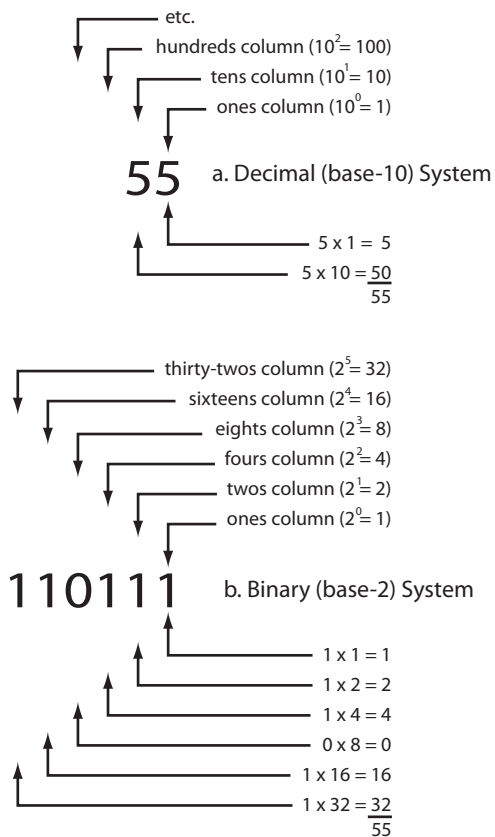


Figure 1. Number Representation Systems

number of 10s ( $10^1$ ), the third digit represents the number of 100s ( $10^2$ ), and so on. We can represent any size number by using this shorthand notation.

Binary number representation is just the same except substituting the powers of 2 for the powers of 10 [any base number system is represented in this manner]. Therefore (moving from right to left) each succeeding bit represents  $2^0 = 1$ ,  $2^1 = 2$ ,  $2^2 = 4$ ,  $2^3 = 8$ ,  $2^4 = 16$ ,  $2^5 = 32$ , etc. Thus, my age breaks down as 1-1, 1-2, 1-4, 0-8, 1-16, and 1-32, represented as “110111,” which is  $32+16+0+4+2+1 = 55$  ...or *double-nickel* to you cool cats. Figure 1 shows the two examples.

Now let’s take a look at how all this came about.

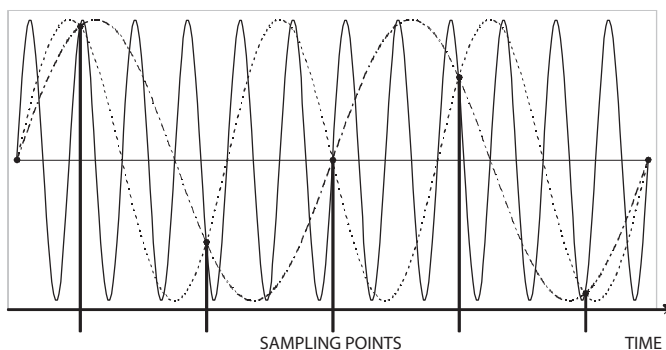


Figure 2. Aliasing Frequencies

## The Story of Harry & Claude

The French mathematician Fourier unknowingly laid the groundwork for A/D conversion in the late 18th century. All data conversion techniques rely on looking at, or *sampling*, the input signal at regular intervals and creating a digital word that represents the value of the analog signal at that precise moment. The fact that we know this works lies with Nyquist.

Harry Nyquist discovered while working at Bell Laboratories in the late ‘20s and wrote a landmark paper<sup>1</sup> describing the criteria for what we know today as sampled data systems. Nyquist taught us that for periodic functions, if you sampled at a rate that was at least twice as fast as the signal of interest, *then no information (data) would be lost upon reconstruction*. And since Fourier had already shown that all alternating signals are made up of nothing more than a sum of harmonically related sine and cosine waves, then audio signals are *periodic functions* and can be sampled without lost of information following Nyquist’s instructions. This became known as the *Nyquist frequency*, which is the highest frequency that may be accurately sampled, and is one-half of the *sampling frequency*. For example, the theoretical Nyquist frequency for the audio CD (compact disc) system is 22.05 kHz, equaling one-half of the standardized sampling frequency of 44.1 kHz.

As powerful as Nyquist’s discoveries were, they were not without their dark side: the biggest being *aliasing* frequencies. Following the Nyquist criteria (as it is now called) guarantees that no information will be lost; it does not, however, guarantee that no information *will be gained*. Although by no means obvious, the act of sampling an analog signal at precise time intervals is an act of *multiplying* the input signal by the sampling pulses. This introduces the possibility of generating “false” signals indistinguishable from the original. In other words, given a set of sampled values, we cannot relate them specifically to one unique signal. As Figure 2 shows, the same set of samples *could* have resulted from any of the three waveforms shown ... and from all possible sum and difference frequencies between the sampling frequency and the one being sampled. All such false waveforms that fit the sample data are called “aliases.” In audio, these frequencies show up mostly as intermodulation distortion products, and they come from the random-like white noise, or any sort of ultrasonic signal present in every electronic system. Solving

<sup>1</sup> Nyquist, Harry, “*Certain topics in Telegraph Transmission Theory*,” published in 1928.

the problem of aliasing frequencies is what improved audio conversion systems to today's level of sophistication. And it was Claude Shannon who pointed the way.

Shannon is recognized as the father of information theory: while a young engineer at Bell Laboratories in 1948, he defined an entirely new field of science. Even before then his genius shined through for, while still a 22-year-old student at MIT he showed in his master's thesis how the algebra invented by the British mathematician George Boole in the mid-1800s, could be applied to electronic circuits. Since that time, *Boolean algebra* has been the rock of digital logic and computer design.<sup>2</sup>

Shannon studied Nyquist's work closely and came up with a deceptively simple addition. He observed (and proved) that if you restrict the input signal's bandwidth to less than one-half the sampling frequency *then no errors due to aliasing are possible*. So bandlimiting your input to no more than one-half the sampling frequency *guarantees no aliasing*. Cool – only it's not possible.

In order to satisfy the Shannon limit (as it is called – Harry gets a “criteria” and Claude gets a “limit”) you must have the proverbial brick-wall, i.e., infinite-slope filter. Well, this isn't going to happen, not in this universe. You cannot guarantee that there is absolutely no signal (or noise) greater than the Nyquist frequency. Fortunately there is a way around this problem. In fact, you go all the way around the problem and look at it from another direction.

If you cannot restrict the input bandwidth so aliasing does not occur, then solve the problem another way: Increase the sampling frequency until the aliasing products that do occur, do so at ultrasonic frequencies, and are effectively dealt with by a simple single-pole filter. This is where the term “oversampling” comes in. For full spectrum audio the minimum sampling frequency must be 40 kHz, giving you a useable theoretical bandwidth of 20 kHz – the limit of normal human hearing. Sampling at anything significantly higher than 40 kHz is termed *oversampling*. In just a few years time, we have seen the audio industry go from the CD system standard of 44.1 kHz, and the pro audio quasi-standard of 48 kHz, to 8-times and 16-times oversampling frequencies of around 350 kHz and 700 kHz respectively. With sampling frequencies this high, aliasing is no longer an issue.

Okay. So audio signals can be changed into digital words (*digitized*) without loss of information, and with no aliasing effects, as long as the sampling frequency is high enough. How is this done?

## Quantization

Quantizing is the process of determining which of the possible values (determined by the number of bits or voltage reference parts) is the closest value to the current sample – i.e., you are assigning a *quantity* to that sample. Quantizing, by definition then, involves deciding between two values and thus always introduces error. How big the error, or how accurate the answer, depends on the number of bits. The more bits, the better the answer. The converter has a reference voltage which is divided up into  $2^n$  parts, where  $n$  is the number of bits. Each part represents the same value. Since you cannot resolve anything smaller than this value, there is error. *There is always error in the conversion process*. This is the *accuracy* issue.

The number of bits determines the converter accuracy. For 8-bits, there are  $2^8 = 256$  possible levels as shown in Figure 3. Since the signal swings positive *and* negative there are 128 levels for each direction. Assuming a  $\pm 5$  V reference<sup>3</sup>, this makes each division, or bit, equal to 39 mV ( $5/128 = .039$ ). Hence, an 8-bit system cannot resolve any change smaller than 39 mV. This means a worst case accuracy error of 0.78%. Table 1 compares the accuracy improvement gained by 16-bit, 20-bit and 24-bit systems along with the reduction in error. (Note: this is not the only way to use the reference voltage. Many schemes exist for coding, but this one nicely illustrates the principles involved.) Each step size (resulting from dividing the reference into

# Bits	# Divisions	Resolution/Div	Max % Error	Max PPM Error
8	$2^7=128$	39 mV	0.78	7812.00
16	$2^{15}=32,768$	153 $\mu$ V	0.003	30.50
20	$2^{19}=524,288$	9.5 $\mu$ V	0.00019	1.90
24	$2^{23}=8,388,608$	0.6 $\mu$ V	0.000012	0.12

Table 1. Quantization Steps For  $\pm 5$  Volts Reference

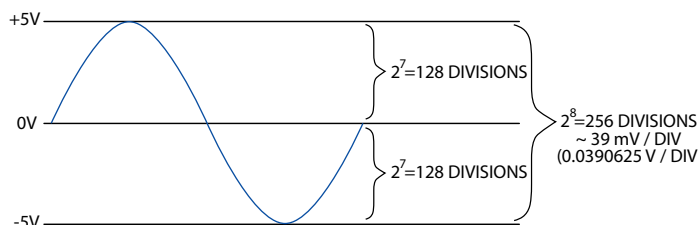


Figure 3. 8-Bit Resolution

<sup>2</sup> See Clive Maxfield's book *Bebob to the Boolean Boogie* (High-Text ISBN 1-878707-22-1, Solana Beach, CA, 1995) for the best treatment.

<sup>3</sup> A single +5 V supply is probably more common today, but this illustrates the point.

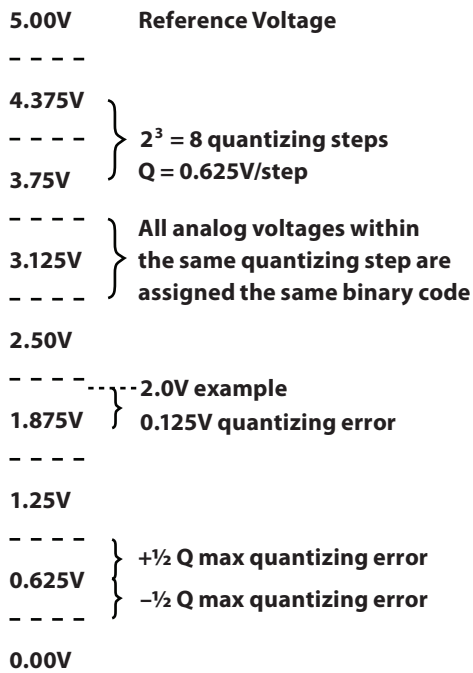


Figure 4. Quantization – 3-Bit, 5V Example

the number of equal parts dictated by the number of bits) is equal and is called a *quantizing step* (also called *quantizing interval* – see Figure 4). Originally this step was termed the *LSB* (least significant bit) since it equals the value of the smallest coded bit, however it is an illogical choice for mathematical treatments and has since be replaced by the more accurate term quantizing step.

The error due to the quantizing process is called *quantizing error* (no definitional stretch here). As shown earlier, each time a sample is taken there is error. Here’s the not obvious part: *the quantizing error can be thought of as an unwanted signal which the quantizing process adds to the perfect original.* An example best illustrates this principle. Let the sampled input value be some arbitrarily chosen value, say, 2 volts. And let this be a 3-bit system with a 5 volt reference. The 3-bits divides the reference into 8 equal parts ( $2^3 = 8$ ) of 0.625 V each, as shown in Figure 4. For the 2 volt input example, the converter must choose between either 1.875 volts or 2.50 volts, and since 2 volts is closer to 1.875 than 2.5, then it is the best fit. This results in a quantizing error of -0.125 volts, i.e., the quantized answer is too small by 0.125 volts. If the input signal had been, say, 2.2 volts, then the quantized answer would have been 2.5 volts and the quantizing error would have been +0.3 volts, i.e., too big by 0.3 volts.

These alternating unwanted signals added by quantizing form a quantized error waveform, that is a kind

of additive broadband noise that is generally uncorrelated with the signal and is called *quantizing noise*. Since the quantizing error is essentially random (i.e. uncorrelated with the input) it can be thought of like *white noise* (noise with equal amounts of all frequencies). This is not quite the same thing as thermal noise, but it is similar. *The energy of this added noise is equally spread over the band from dc to one-half the sampling rate.* This is a most important point and will be returned to when we discuss delta-sigma converters and their use of extreme oversampling.

### Successive Approximation

Successive approximation is one of the earliest and most successful analog-to-digital conversion techniques. Therefore, it is no surprise it became the initial A/D workhorse of the digital audio revolution. Successive approximation paved the way for the delta-sigma techniques to follow.

The heart of any A/D circuit is a comparator. A comparator is an electronic block whose output is determined by comparing the values of its two inputs. If the positive input is larger than the negative input then the output swings positive, and if the negative input exceeds the positive input, the output swings negative. Therefore if a reference voltage is connected to one input and an unknown input signal is applied to the other input, you now have a device that can *compare* and tell you which is larger. Thus a comparator gives you a “high output” (which could be defined to be a “1”) when the input signal exceeds the reference, or a “low output” (which could be defined to be a “0”) when it does not. A comparator is the key ingredient in the successive approximation technique as shown in Figures 5A & 5B.

The name *successive approximation* nicely sums up how the data conversion is done. The circuit evaluates each sample and creates a digital word representing the closest binary value. The process takes the same number of steps as bits available, i.e., a 16-bit system requires 16 steps for *each sample*. The analog sample is successively compared to determine the digital code, beginning with the determination of the biggest (most significant) bit of the code.

The description given in Daniel Sheingold’s *Analog-Digital Conversion Handbook* (see References) offers the best analogy as to how successive approximation works. The process is exactly analogous to a gold miner’s assay scale, or a chemical balance as seen in Figure 5A. This type of scale comes with a set of gradu-

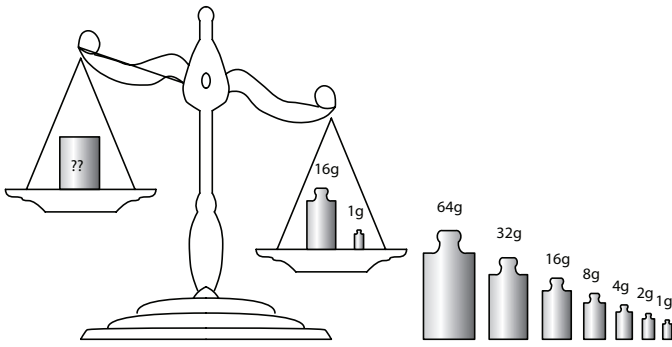


Figure 5A. Successive Approximation Example

In digital terms, we can analyze this example by saying that a “0” was assigned to each weight removed, and a “1” to each weight remaining – in essence creating a digital word equivalent to the unknown sample, with the number of bits equaling the number of weights. And the quantizing error will be no more than  $\frac{1}{2}$  the smallest weight (or  $\frac{1}{2}$  *quantizing step*).

As stated earlier the successive approximation technique must repeat this cycle for each sample. Even with today’s technology, this is a very time consuming process and is still limited to relatively slow sampling rates, but it did get us into the 16-bit, 44.1 kHz digital audio world.

### PCM (Pulse Code Modulation) and PWM (Pulse Width Modulation)

The successive approximation method of data conversion is an example of pulse *code* modulation, or PCM. Three elements are required: sampling, quantizing, and encoding into a fixed length digital word. The reverse process reconstructs the analog signal from the PCM code. The output of a PCM system is a series of digital words, where the word-size is determined by the available bits. For example the output is a series of 8-bit words, or 16-bit words, or 20-bit words, etc., with each word representing the value of one sample.

Pulse *width* modulation, or PWM is quite simple and quite different from PCM. Look at Figure 6. In a typical PWM system, the analog input signal is applied to a comparator whose reference voltage is a triangle-shaped waveform whose repetition rate is the sampling frequency. This simple block forms what is called an *analog modulator*.

A simple way to understand the “modulation” process is to view the output with the input held steady at zero volts. The output forms a 50% duty cycle (50% high, 50% low) square wave. As long as there is no input, the output is a steady square wave. As soon as the

ated weights, each one half the value of the preceding one, such as 1 gram,  $\frac{1}{2}$  gram,  $\frac{1}{4}$  gram,  $\frac{1}{8}$  gram, etc. You compare the unknown sample against these known values by first placing the heaviest weight on the scale. If it tips the scale you remove it; if it does not you leave it and go to the next smaller value. If that value tips the scale you remove it, if it does not you leave it and go to the next lower value, and so on until you reach the smallest weight that tips the scale. (When you get to the last weight, if it does not tip the scale, then you put the next highest weight back on, and that is your best answer.) The sum of all the weights on the scale represents the closest value you can resolve.

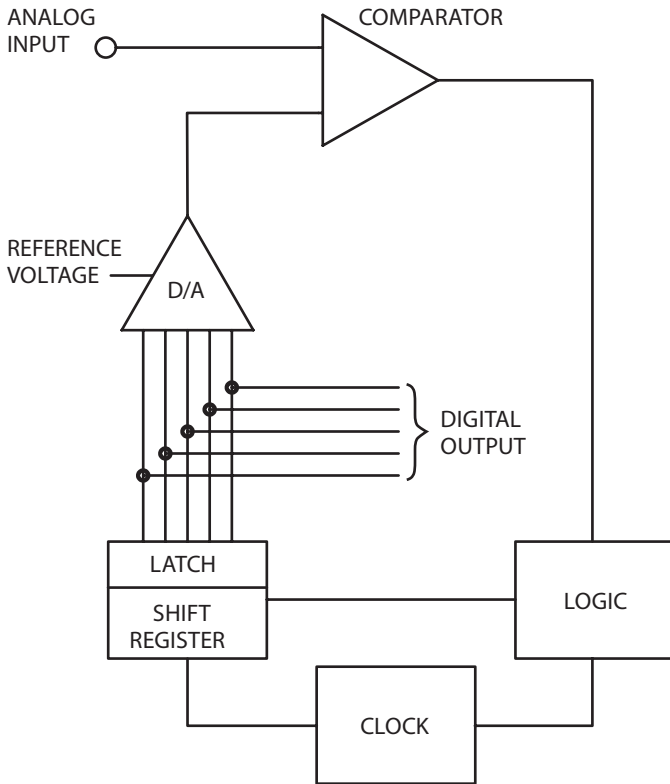


Figure 5B. Successive Approximation A/D Converter

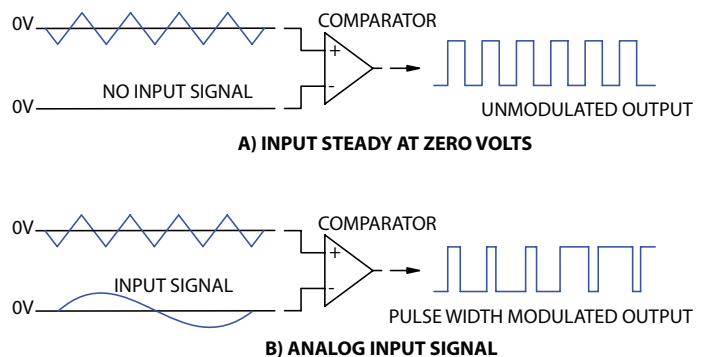


Figure 6. Pulse Width Modulation (PWM)

input is non-zero, the output becomes a pulse-width *modulated* waveform. That is, when the non-zero input is compared against the triangular reference voltage, it varies the length of time the output is either high or low.

For example, say there was a steady DC value applied to the input. For all samples when the value of the triangle is less than the input value, the output stays low, and for all samples when it is greater than the input value, it changes state and remains high. Therefore, if the triangle starts higher than the input value, the output goes high; at the next sample period the triangle has increased in value but is still more than the input, so the output remains high; this continues until the triangle reaches its apex and starts down again; eventually the triangle voltage drops below the input value and the output drops low and stays there until the reference exceeds the input again. *The resulting pulse-width modulated output, when averaged over time, gives the exact input voltage.* For example if the output spends exactly 50% of the time with an output of 5 volts, and 50% of the time at 0 volts, then the *average* output would be exactly 2.5 volts.

This is also an FM, or frequency-modulated system – the varying pulse-width translates into a varying frequency. And it is the core principle of most Class-D switching power amplifiers. The analog input is converted into a variable pulse-width stream used to turn-on the output switching transistors. The analog output voltage is simply the average of the on-times of the positive and negative outputs. Pretty amazing stuff from a simple comparator with a triangle waveform reference.

Another way to look at this, is that this simple device actually *codes a single bit of information*, i.e., a comparator is a *1-bit A/D converter*. PWM is an example of a 1-bit A/D encoding system. And a 1-bit A/D encoder forms the heart of delta-sigma modulation.

## Delta-Sigma Modulation & Noise Shaping

After nearly thirty years, delta-sigma modulation (also sigma-delta<sup>4</sup>) has only recently emerged as the most successful audio A/D converter technology. It waited patiently for the semiconductor industry to develop the technologies necessary to integrate analog and digital circuitry on the same chip. Today's very high-speed "mixed-signal" IC processing allows the total integration of all the circuit elements necessary to create delta-sigma data converters of awesome magnitude<sup>5</sup>.

How the name came about is interesting. Another way to look at the action of the comparator is that the 1-bit information tells the output voltage which direction to go based upon what the input signal is doing. It looks at the input and compares it against its last look (sample) to see if this new sample is bigger or smaller than the last one – that is the information transfer: bigger or smaller, increasing or decreasing. If it is bigger than it tells the output to keep increasing, and if it is smaller it tells the output to stop increasing and start decreasing. It merely reacts to the *change*. Mathematicians use the Greek letter "delta" (symbol  $\Delta$ ) to stand for deviation or small incremental change, which is how this process came to be known as "delta modulation." The "sigma" part came about by the significant improvements made from summing or integrating the signal with the digital output before performing the delta modulation. Here again, mathematicians use the Greek letter "sigma" (symbol  $\Sigma$ ) to stand for summing, so "delta-sigma" became the natural name.

Essentially a delta-sigma converter digitizes the audio signal with a very low resolution (1-bit) A/D converter at a very high sampling rate. It is the oversampling rate and subsequent digital processing that separates this from plain delta modulation (no sigma).

Referring back to the earlier topic of quantizing noise it is possible to calculate the theoretical sine wave signal-to-noise (S/N) ratio (actually the signal-to-*error* ratio, but for our purposes it's close enough to combine) of an A/D converter system knowing only  $n$ , the number of bits. Doing a bit (sorry) of math shows that the value of the added quantizing noise relative to a maximum (full-scale) input equals  $6.02n + 1.76$  dB for a sine wave. For example, a perfect 16-bit system will have a S/N ratio of 98.1 dB, while a 1-bit delta-modulator A/D converter, on the other hand, will have only 7.78 dB!

<sup>4</sup> The name *delta-sigma modulation* was coined by Inose and Yasuda at the University of Tokyo in 1962, but due to a translation misunderstanding, words were interchanged and taken to be *sigma-delta*. Both names are still used, but only *delta-sigma* is actually correct.

<sup>5</sup> Leung, K., et al., "A 120 dB dynamic Range, 96 kHz, Stereo 24-bit Analog-to-Digital Converter," presented at the 102nd Convention of the Audio Engineering Society, Munich, March 22-25, 1997.

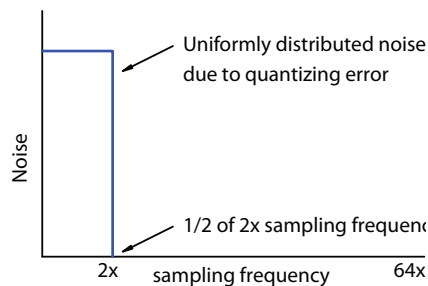
To get an intuitive feel for this, consider that since there is only 1-bit, the amount of quantization error possible is as much as  $\frac{1}{2}$ -bit. That is, since the converter must choose between the only two possibilities of maximum or minimum values, then the error can be as much as half of that. And since this quantization error shows up as added noise, then this reduces the S/N to something on the order of around 2:1 or 6 dB.

One attribute shines true above all others for delta-sigma converters and makes them a superior audio converter: simplicity. The simplicity of 1-bit technology makes the conversion process very fast, and very fast conversions allows use of extreme oversampling. And extreme oversampling pushes the quantizing noise and aliasing artifacts way out to megawiggle-land, where it is easily dealt with by digital filters (typically 64-times oversampling is used, resulting in a sampling frequency on the order of 3 MHz).

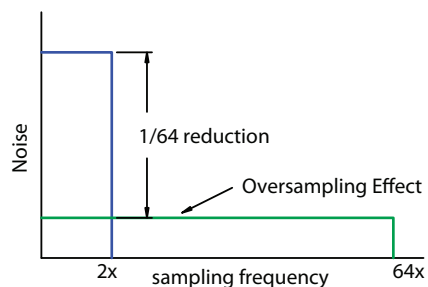
To get a better understanding of how oversampling reduces audible quantization noise, we need to think in terms of noise power. From physics you may remember that power is conserved – i.e., you can *change* it, but you cannot create or destroy it; well, quantization noise power is similar. With oversampling the quantization noise power is spread over a band that is as many times larger as is the rate of oversampling. For example, for 64-times oversampling, the noise power is spread over a band that is 64 times larger, reducing its power density in the audio band by  $\frac{1}{64^{\text{th}}}$ . See Figures 7A-E for example.

Noise shaping helps reduce in-band noise even more. Oversampling pushes out the noise, but it does so uniformly, that is, the spectrum is still flat. Noise shaping changes that. Using very clever complex algorithms and circuit tricks, noise shaping contours the noise so that it is reduced in the audible regions and increased in the inaudible regions. Conservation still holds, the total noise is the same, but the amount of noise present in the audio band is decreased while simultaneously increasing the noise out-of-band – then the digital filter eliminates it. Very slick.

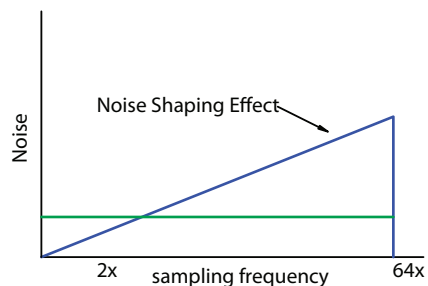
As shown in Figure 8, a delta-sigma modulator consists of three parts: an analog modulator, a digital filter and a decimation circuit. The analog modulator is the 1-bit converter discussed previously with the change of integrating the analog signal before performing the delta modulation. (The integral of the analog signal is encoded rather than the change in the analog signal, as is the case for traditional delta modulation.) Oversampling and noise shaping pushes and contours all the



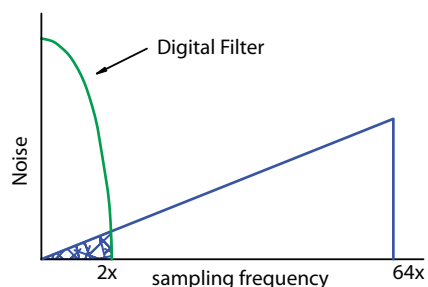
**A. Original Noise Power Distribution**



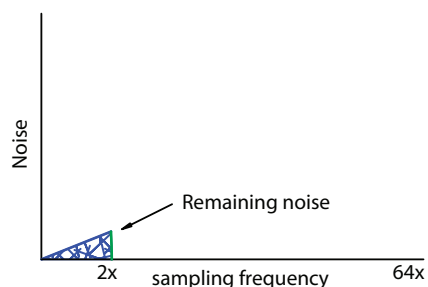
**B. Reduction & Redistribution due to Oversampling**



**C. Noise Shaping "Tilts" Distribution**



**D. Digital Filter Eliminates Out-of-Band Noise**



**E. Remaining Noise Distribution.**

**Figure 7A-E. Noise Power Redistribution & Reduction due to Oversampling, Noise Shaping and Digital Filtering.**

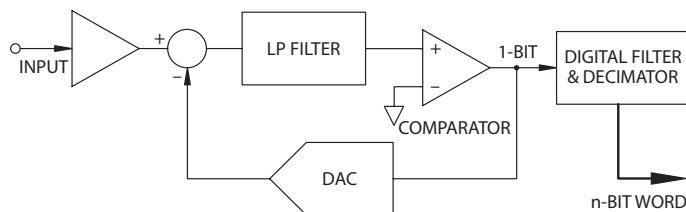


Figure 8. Delta-Sigma A/D Converter

bad stuff (aliasing, quantizing noise, etc.) so the digital filter suppresses it. The decimation circuit, or *decimator*, is the digital circuitry that generates the correct output word length of 16-, 20-, or 24-bits, and restores the desired output sample frequency. It is a digital sample rate reduction filter and is sometimes termed *downsampling* (as opposed to oversampling) since it is here that the sample rate is returned from its 64-times rate to the normal CD rate of 44.1 kHz, or perhaps to 48 kHz, or even 96 kHz, for pro audio applications. The net result is much greater resolution and dynamic range, with increased S/N and far less distortion compared to successive approximation techniques – all at lower costs.

## Dither – Not All Noise Is Bad<sup>6</sup>

Now that oversampling helped get rid of the bad noise, let's add some *good noise* – *dither* noise.

Just what is *dither*? Aside from being a funny sounding word, it is a wonderfully accurate choice for what is being done. The word “dither” comes from a 12th century English term meaning “to tremble.” Today it means to be in a state of indecisive agitation, or to be nervously undecided in acting or doing. Which, if you think about it, is not a bad description of noise.

Dither is one of life's many trade-offs. Here the trade-off is between noise and resolution. Believe it or not, we can introduce dither (a form of noise) and increase our ability to resolve very small values. Values, in fact, smaller than our smallest bit — now that's a good trick. Perhaps you can begin to grasp the concept by making an analogy between dither and anti-lock brakes.<sup>7</sup> Get it?

No? Okay, here's how this analogy works: With regular brakes, if you just stomp on them, you probably create an unsafe skid situation for the car — not a good idea. Instead, if you rapidly tap the brakes, you control the stopping without skidding. We shall call this “dithering the brakes.” What you have done is introduce “noise” (tapping) to an otherwise rigidly binary (*on or off*) function.

So by “tapping” on our analog signal, we can improve our ability to resolve it. By introducing noise, the converter rapidly switches between two quantization levels, rather than picking one or the other, when neither is really correct. Sonically, this comes out as noise, rather than a discrete level with error. Subjectively, what would have been perceived as distortion is now heard as noise.

Lets look at this in more detail. The problem dither helps to solve is that of quantization error caused by the data converter being forced to choose one of two exact levels for each bit it resolves. It cannot choose *between* levels, it must pick one or the other. With 16-bit systems, the digitized waveform for *high frequency, low signal levels* looks very much like a steep staircase with few steps. An examination of the spectral analysis of this waveform reveals lots of nasty sounding distortion products. We can improve this result either by adding

<sup>6</sup>This section is included because of the confusion surrounding the term. However, with the advances made in A/D converter resolution technology, the need for dither in A/D converters has essentially disappeared, making this section more of historical interest. Dither is still necessary for word-length reduction in other digital processing.

<sup>7</sup> Thanks to Bob Moses for this great analogy.

more bits, or by adding dither. Prior to 1997, adding more bits for better resolution was straightforward, but expensive, thereby making dither an inexpensive compromise; today, however, there is less need.

The dither noise is added to the low-level signal before conversion. The mixed noise causes the small signal to jump around, which causes the converter to switch rapidly *between* levels rather than being forced to choose between two fixed values. Now the digitized

waveform still looks like a steep staircase, but each step, instead of being smooth, is comprised of many narrow strips, like vertical blinds. The spectral analysis of this waveform shows almost no distortion products at all, albeit with an increase in the noise content. The dither has caused the distortion products to be pushed out beyond audibility, and replaced with an increase in wideband noise. Figure 9 diagrams this process.

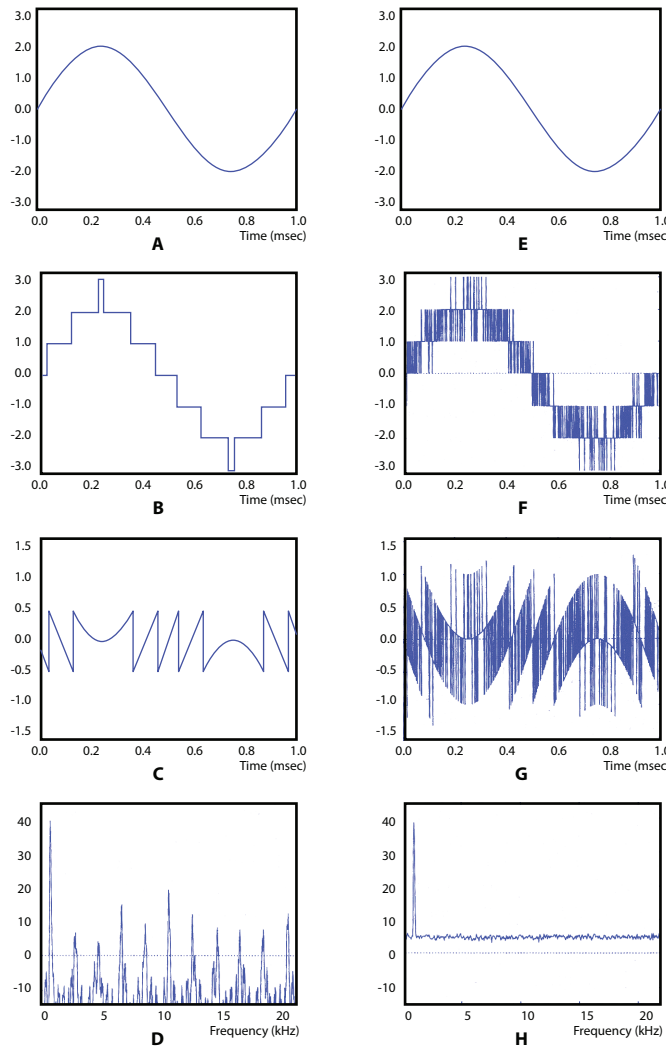


Figure 9. A. Input Signal. B. Output Signal [no dither]. C. Total Error Signal [no dither]. D. Power Spectrum of Output Signal [no dither]. E. Input Signal. F. Output Signal [with dither]. G. Total Error Signal [with dither]. H. Power Spectrum of Output Signal [with dither].<sup>8</sup>

## Life After 16 – A Little Bit Sweeter

Current digital recording standards allow for only 16-bits, yet it is safe to say that for all practical purposes 16-bit technology is history<sup>9</sup>. Everyone who can afford the up-grade is using 20- and 24-bit data converters and *dithering* (vs. truncating) down to 16-bits. Here is what is gained by using 20-bits:

- 24 dB more dynamic range
- *24 dB less residual noise*
- 16:1 reduction in quantization error
- Improved jitter (timing stability) performance

*And if it is 24-bits, add **another 24 dB** to each of the above and make it a **256:1** reduction in quantizing error, with essentially zero jitter!*

As stated in the beginning of this note, with today's technology, analog-to-digital-to-analog conversion is *the* element defining the sound of a piece of equipment, and if it's not done perfectly then everything that follows is compromised.

With 20-bit high-resolution conversion, low signal-level detail is preserved. The improvement in fine detail shows up most noticeably by reducing the quantization errors of low-level signals. Under certain conditions, these coarse data steps can create audio passband harmonics not related to the input signal. Audibility of this quantizing noise is much higher than in normal analog distortion, and is also known as *granulation noise*. 20-bits virtually eliminates granulation noise. Commonly heard examples are musical fades, like reverb tails and cymbal decay. With only 16-bits to work with, they don't so much fade as collapse in noisy chunks.

Where it really matters most is in measuring very small things. It doesn't make much difference when measuring big things. If your ruler measures in whole inch increments and you are measuring something 10 feet long, the most you can be off is ½ inch. Not a big deal. However, if what you're measuring is less than an inch, and your error can be as much as ½ inch, well, now you've got an accuracy problem. This is exactly the problem in digitizing small audio signals. Graduating our audio digital ruler finer and finer means we can accurately resolve smaller and smaller signal levels, allowing us to capture the *musical details*. Getting the exact right answer *does* result in better reproduction of music.

## A/D Converter Measuring Bandwidth Note

Due to the oversampling and noise shaping characteristics of delta-sigma A/D converters, certain measurements must use the appropriate bandwidth or inaccurate answers result. Specifications such as signal-to-noise, dynamic range, and distortion are subject to misleading results if the wrong bandwidth is used. Since noise shaping purposely reduces audible noise by shifting the noise to inaudible higher frequencies, taking measurements over a bandwidth wider than 20 kHz results in answers that do not correlate with the listening experience. Therefore, it is important to set the correct measurement bandwidth to obtain meaningful data.

<sup>9</sup> Historical Footnote: The reason the British divided up the pound into 16 ounces is not as arbitrary as some might suspect, but, rather, was done with great calculation and foresightedness. At the time, you see, technology had advanced to where 4-bit systems were really quite the thing. And, of course, 4-bits allows you to divide things up into 16 different values (since  $2^4 = 16$ ). So one pound was divided up into 16 equal parts called "ounces," for reasons to be explained at another time. Similarly, the roots of a common American money term come from a simple 3-bit system. A 3-bit system allows eight values (since  $2^3 = 8$ ), so if you divide up a dollar into eight parts, each part is, of course, 12.5 cents. Therefore you would call two parts (or *two-bits*, as we Americans say) a "quarter" — obvious.

## References

1. Candy, James C. and Gabor c. Temes, eds. *Oversampling Delta-Sigma Data Converters: Theory, Design, and Simulation* (IEEE Press ISBN 0-87942-285-8, NY, 1992).
2. "Delta Sigma A/D Conversion Technique Overview," *Application Note AN 10* (Crystal Semiconductor Corporation, TX, 1989).
3. Pohlmann, Ken C. *Advanced Digital Audio* (Sams ISBN 0-672-22768-1, IN, 1991).
4. Sheingold, Daniel H., ed. *Analog-Digital Conversion Handbook, 3rd ed.* (Prentice-Hall ISBN 0-13-032848-0, NJ, 1986).
5. "Sigma-Delta ADCs and DACs," *1993 Applications Reference Manual* (Analog Devices, MA, 1993).
6. *The American Heritage Dictionary of the English Language, 3rd ed.* (Houghton Mifflin ISBN 0-395-44895-6, Boston, 1992).
7. Watkinson, John. *The Art of Digital Audio, 2nd ed.* (Focal Press, ISBN 0-240-51320-7, Oxford, England, 1994).
8. <sup>s</sup> Pohlmann, Ken C. *Principles of Digital Audio, 3rd ed.* (McGraw Hill ISBN 0-07-050469-5, NY, 1995). (p.44)