Audio precision® TECHNOTE

Application and Technical Support for Audio Precision APWIN Users

TN-23

Jitter Theory

By Julian Dunn

Introduction

Digital audio systems are unlike analog audio systems in two fundamental respects:

- The signal, in its analog state a continuously variable voltage or current, is represented digitally by a limited number of discrete numerical values.
- These numerical values represent the signal only at specific points in time, or *sampling instants,* rather than continuously at every moment in time.

Sampling instants are determined by various devices. The most common are the *analog-to-digital converter* (ADC) and the *digital-to-analog converter* (DAC) which interface between the digital and analog representations of the same signal. These devices will often have a *sample clock* to control their *sampling rate* or *sampling frequency*.

Sampling instants can also be determined by a *sample rate converter* (SRC) that uses numerical processes to convert a digital signal at one sampling frequency to a digital signal at another sampling frequency. An SRC might not have a physical sample clock at all, but in the numerical process of regenerating signal samples to correspond with new sampling instants is considered to use a *virtual sample clock*.

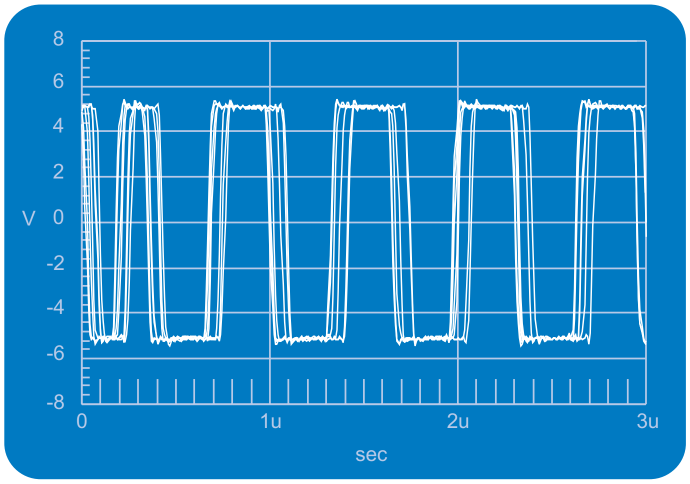


Figure 1. Jittered AES3 waveform

Digital audio is often thought to be immune to the many plagues of analog recording and transmission: distortion, line noise, tape hiss, flutter, crosstalk; and if not immune, digital audio is certainly highly resistant to most of these maladies. But when practicalities such as oscillator instability, cable losses or noise pickup do intrude, they often affect the digital signal in the time domain as *jitter*.

This jitter can be on the interface carrying the digital signal. Interface jitter can result in data errors or loss of lock, which represent fault conditions; it can also be coupled into equipment to produce jitter in a sample clock, the effect being a (normally) subtle reduction in the accuracy of the sampling process.

What Is Jitter?

Jitter is the variation in the time of an event—such as a regular clock signal—*from nominal.*

For example, the jitter on a regular clock signal is the difference between the actual pulse transition times of the real clock and the transition times that would have occurred had the clock been ideal, that is to say, perfectly regular.

Against this nominal reference, the zero-crossing transitions of many of the pulses in a jittered data stream are seen to vary in time from the ideal clock timing. Expressed another way, jitter is phase modulation of the digital interface signal.

The jitter component can be extracted from the clock or digital interface signal to be analyzed as a signal in its own right. Among the more useful ways of characterizing jitter is by examining its frequency spectrum and identifying the significant frequency components of the jitter itself.

Measuring Jitter

When very little jitter is present, the pulse transitions are moved back or forth by only small measures of time. When the jitter is increased, the transitions move across a larger range of times.

Jitter amplitude, then, is a measure of time displacement and is expressed in units of time, either as fractions of a second or *unit intervals*. For those new to jitter measurement, this can lead to some disconcerting graph labels, with time on the vertical axis versus time on the horizontal axis, for example.

Jitter frequency is the rate at which this phase-shifting is taking place. Like other noise or interference signals, the jitter modulation signal can be a pure and regular sine wave, a complex waveform or have a completely random character.

The Unit Interval

The *unit interval* (UI) is a measure of time that scales with the interface data rate, and is often a convenient term for interface jitter discussions. The UI is defined as the *shortest nominal time interval in the coding scheme.* For an AES3 signal at a 48 kHz frame rate, there are 32 bits per subframe and 64 bits per frame, giving a nominal 128 pulses per frame in the channel after bi-phase mark encoding is applied. So, in this case:

$$1\,\text{UI} / (128 \quad 48000) = 163\,\text{ns}$$

The UI is used for several of the jitter specifications in AES3[1] (the Audio Engineering Society's standard for interfacing two-channel linear digital audio), with the result that the specifications scale appropriately with the data and frame rate. As an example, the dimensions in UI for 96 kHz frame rates are exactly half the size, in seconds, as the dimensions in UI for frame rates of 48 kHz. This scaling matches the scaling of the capabilities and requirements of the receivers and transmitters on the interface.

Note: Some specifications in data transmission define the unit interval as the duration of one bit transmission. This produces results incompatible with the AES3 specification and is not used here.

How Can You See Jitter?

Jitter on a digital signal can be observed as pulse transitions that occur slightly before or after the transitions of an ideal clock. Any meaningful measurement, then, must involve a comparison between the jittered signal and an ideal clock.

In practice, there are often no ideal clocks to compare with, and real jitter measurements must be *self-referenced*—made relative to the signal itself.

The simplest and most misleading self-referenced technique is "looking at the waveform on an oscilloscope," triggering the oscilloscope on the jittered signal as shown in Figure 2. Unfortunately, you will get deceptive results that depend on the interval between the oscilloscope trigger and the transition being examined, and also on the frequency spectrum of the jitter. Rather than jitter, this technique

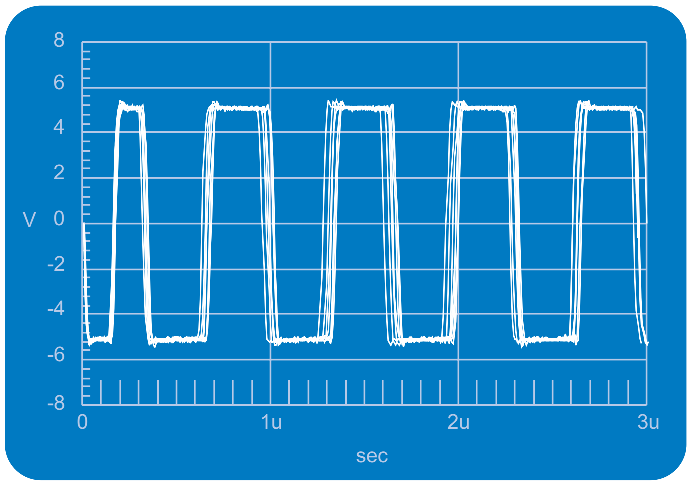


Figure 2. Interval variations on an oscilloscope. Not a valid way to view jitter.

displays *interval variations.* There is a relationship between the two, but at some frequencies jitter will not be shown at all, and at others the jitter amplitude will appear doubled. In particular, this approach is very insensitive to low-frequency jitter.

Instead, the ideal clock can be simulated by phase-locking a relatively low-jitter oscillator to the jittered signal or real clock, using a phase-locked loop (PLL). (A sidebar on phase-locked loop characteristics is on Page 4.) This self-referencing technique will have a high-pass characteristic with a corner frequency that is related to the PLL corner frequency. The PLL provides an ideal clock signal useful as an oscilloscope external trigger or as a reference signal in dual-trace oscilloscope viewing, for example.

If an oscilloscope is triggered by the PLL reference clock and the scope time base is set to the duration of about one UI, a great many sequential pulses will be shown at once, all stacked on top of one another due to the persistence of the screen phosphors. This distinctive display is called an *eye pattern*, a version of which is shown in Figure 3. The opening in an eye pattern is narrowed by the time spread of the pulse transitions. A narrow eye, then, indicates jitter.

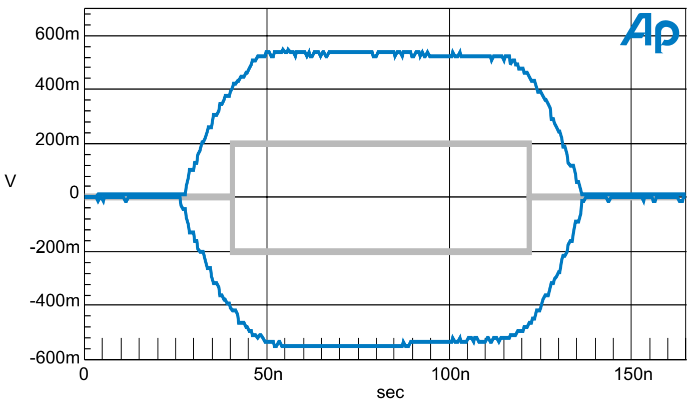


Figure 3. APWIN DSP eye pattern. The blue line is the eye formed by the interface signal; the grey box represents the opening which satisfies the minimum input characteristics specified in AES3.

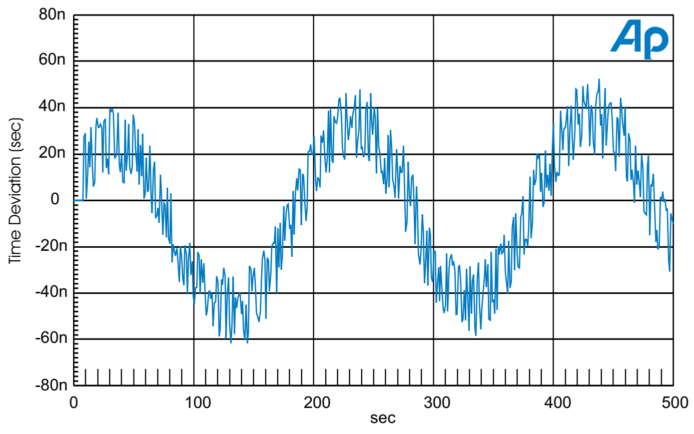


Figure 4. 5 kHz jitter vs. time

Using digital signal processing (DSP) techniques, a DSP analyzer can approximate the ideal clock reference by calculating the clock timing based on an averaging of the incoming signal. The DSP analyzer can then capture the signal (and its jitter) very accurately. From this data the analyzer can display the variation in timing and amplitude of the pulse stream as an eye pattern as in Figure 3; show the jitter waveform in the time domain as in Figure 4, or, using FFT spectrum analysis, plot the jitter in the frequency domain, as in Figure 5.

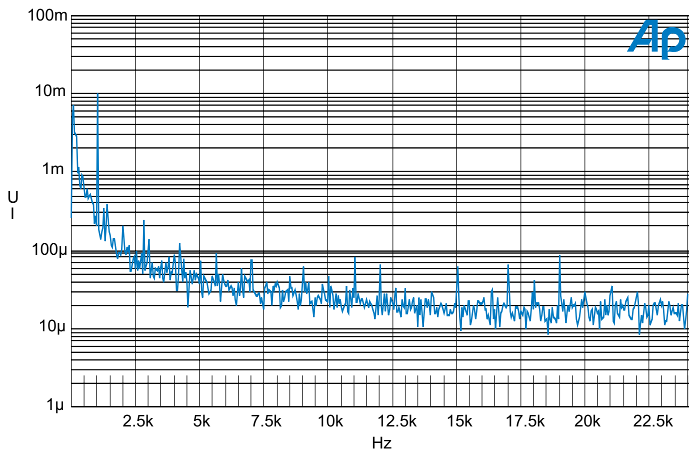


Figure 5. FFT spectrum analysis of jitter signal

Jitter in Sampling Processes

Jitter can affect a digital audio signal in two broad realms: in the sampling process, and in the digital interface.

Sampling jitter is the term given to errors in the timing of the sampling processes of an ADC, a DAC or an SRC. Larger amounts of sampling jitter may cause an audible degradation to the signal. Sampling jitter is discussed in detail beginning on Page 8.

Jitter in the Interface: Data Recovery

Quite apart from the gradual degradation that can result from jitter on sampling clocks, jitter is also an important characteristic to be controlled for reliable data communications. Jitter in digital audio interface signals should be kept within the range that can be tolerated by the data receiver; otherwise, the data may be corrupted. These levels are typically orders of magnitude larger than the jitter levels that would cause concern in sampling clocks. *Interface jitter* is discussed in detail beginning on this page.

Jitter in Clock Recovery for Synchronization

In many digital audio applications it is important for the signals to be stored, transmitted, or processed together. This requires that the signals be time-aligned. In other applications it is important that the audio sample rate exactly matches a multiple of another rate, such as a video frame rate, so that the video and digital audio signals may be encoded, stored or transmitted together. The action of controlling timing in this way is called *clock synchronization*.

When a clock is synchronized from an external “sync” source, jitter can be coupled from the sampling jitter of the sync source clock. It can also be introduced in the sync interface. Fortunately, it is possible to filter out sync jitter while maintaining the underlying synchronization. The resulting system imposes the characteristics of a low-pass filter on the jitter, resulting in jitter attenuation above the filter corner frequency.

When sample timing is derived from an external synchronization source in this way, the jitter attenuation properties of the sync systems become important for the quality of the audio signal. There are other circumstances where this is not so important.

Digital Interface Jitter

Interface jitter occurs as digital signals are passed from one device to another, where jitter can be introduced, amplified, accumulated and attenuated, depending on the characteristics of the devices in the signal chain. Jitter in data transmitters and receivers, line losses in cabling, and noise and other spurious signals can all cause jitter and degrade the interface signal.

The AES3 digital audio interface format[1] now has specifications for jitter. (The consumer version of the interface, which is described in IEC60958-3:2000[2] also has jitter specifications.) This specification was drawn up to resolve problems that would occur when units that conformed to the interface specification were interconnected and yet the interface did not work reliably.[3]

Intrinsic Jitter

If a unit is either free-running or synchronized with a relatively jitter-free signal, then any output jitter measured at the transmitter is due to the device itself. This is referred to as *intrinsic jitter*.

The level of intrinsic jitter is mainly determined by two characteristics: the phase noise of the oscillator in the clock circuit and, for an externally synchronized device, the characteristics of the clock recovery PLL.

Phase-Locked Loop Characteristics

A mechanical flywheel will slowly follow gradual speed changes but will largely ignore short-term fluctuations. This behavior is similar to that of a phase-locked loop (PLL). The lighter the flywheel the more rapidly it will follow changes and the "cut-off" or corner frequency is higher. The corner frequency of a PLL is determined by its feedback, or loop gain. This feedback falls with frequency, both as a result of the characteristics of the loop filter and from the integration of frequency into phase that is taking place before the phase detector output. At the corner frequency the gain around the loop is unity.

For jitter spectral components below the corner frequency, the negative feedback means that the PLL output will closely follow the PLL input, and the phase noise of the oscillator is attenuated. Above the corner frequency the feedback falls. This means that the jitter of the PLL output will be determined increasingly by the phase noise of the oscillator and less by the input jitter. A key element in the design of a transmitter or receiver PLL is this compromise between intrinsic jitter and jitter attenuation.

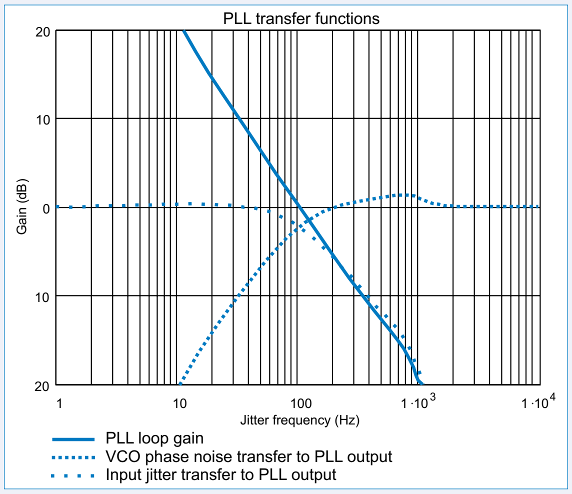


Figure 6. Phase lock loop transfer functions.

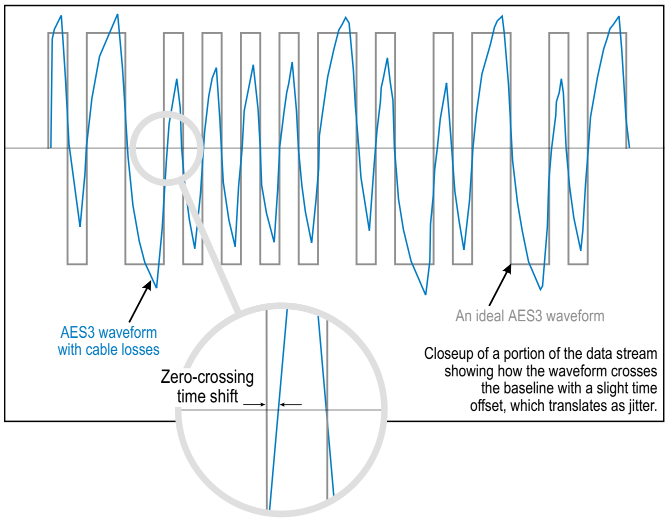
For example, consider the quartz clock oscillator in a CD player. Since it is free-running, any jitter at the output is due to the phase noise of the oscillator plus any digital logic delay jitter. Quartz oscillators have low phase noise and the high speed logic devices have very little delay jitter, so the jitter is low—often less than 1 ps rms for jitter frequencies above 700 Hz.

A device designed to lock to external signals with a range of sampling frequencies may have a voltage controlled oscillator (VCO) as a clock. VCOs generally have much higher phase noise than a quartz oscillator; free-running VCOs typically have levels of intrinsic jitter of more than 1 ns rms above 700 Hz. However, in a clock-recovery application the VCO would be within a phase-locked loop (see the sidebar above) in order to synchronize with the external reference, and the intrinsic jitter of the oscillator would be attenuated by the PLL.

Intrinsic jitter often must be measured in situations where there is no low-jitter reference available, and so the measurements are self-referenced by locking a PLL to the clock signal recovered from the data stream. The characteristics of this PLL will determine the low-frequency cut-off point of the measurement. AES3 specifies a standard response for this measurement with a 3 dB corner frequency of 700 Hz.

The intrinsic jitter levels in AES3 are specified as a peak measurement, rather than rms. This is because the authors were concerned with the maximum excursion of timing deviations—as it is these that would produce data errors.

Cable-Induced Jitter



AES3 waveform with cable losses
An ideal AES3 waveform
Zero-crossing time shift
Closeup of a portion of the data stream showing how the waveform crosses the baseline with a slight time offset, which translates as jitter.

Figure 7. AES3 ideal waveform with cable-affected waveform overlaid

The other source of jitter on the digital interface is as a result of the non-ideal nature of the interconnection. Resistance in the cable or inconsistent impedance can cause high frequency losses which result in a smearing of the pulse transitions, as shown in Figure 7.

This would not be a serious problem if the effect were the same on every transition. That would just result in a

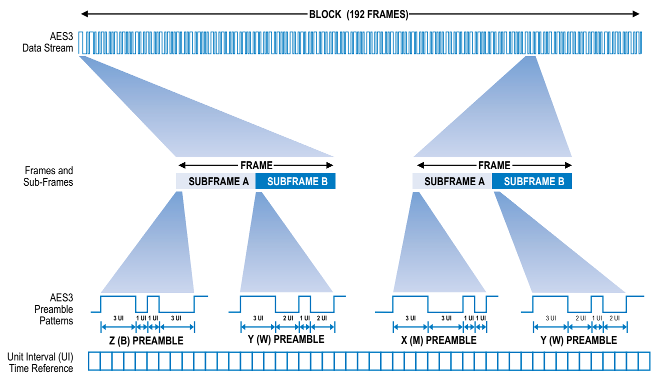


Figure 8. AES3 data pattern. Note that the Y preambles are identical in every frame

Intersymbol Interference

Figure 9 shows five AES3 interface signals, each with a different data pattern in the first three bits. The data is encoded by the bi-phase mark encoding scheme (also called Manchester code or FM code), which has a transition between every bit symbol and also a transition in the middle of the symbol if it is "1," but not if it is "0." The black signal represents "1-1-1," the grey is "1-1-0," the blue "1-0-0," the pale blue "0-1-0" and the dotted blue is "0-0-0."

The figure also shows the signals as they may look after transmission down a long length of cable. These cable-affected signals were generated using the Audio Precision System Two cable simulation, and the five results have been overlaid on each other. The losses in a real cable would affect the signals in this manner, rolling off the high frequencies and reshaping the pulses with slower rise and fall times.

In each case the data shown were immediately preceded by the Y preamble, the preamble which begins the B subframe. (See Figure 8.) This preamble is a fixed pattern which lasts for 5 bit periods (10 unit intervals, or UI). A consequence of this is that the traces coming into the left-hand side of the cable simulator plot are at almost exactly the same voltage, since they have all followed the same path for a while. (The preamble is nominally 8 UI long, but the last part of the preceding bit and the first part of the following bit period are fixed to the pattern, resulting in a fixed pattern that is 10 UI long.)

The black, grey and blue traces have a transition starting at 1465 ns (9 UI) from the subframe start because they have an initial "1" in their data. The pale blue and dotted blue traces start with an initial "0" so they do not yet show a transition. All five traces then change direction at 1628 ns (10 UI) corresponding with the end of the first bit symbol. (The frame rate of this signal is 48 kHz, so 1 UI is 162.8 ns.)

The markers "a" and "b" indicate that the times of the zero-crossings from those transitions are 1705 ns and 1745 ns. The earlier transitions are those which have a "1" value in the first bit and the later transitions those which have a "0."

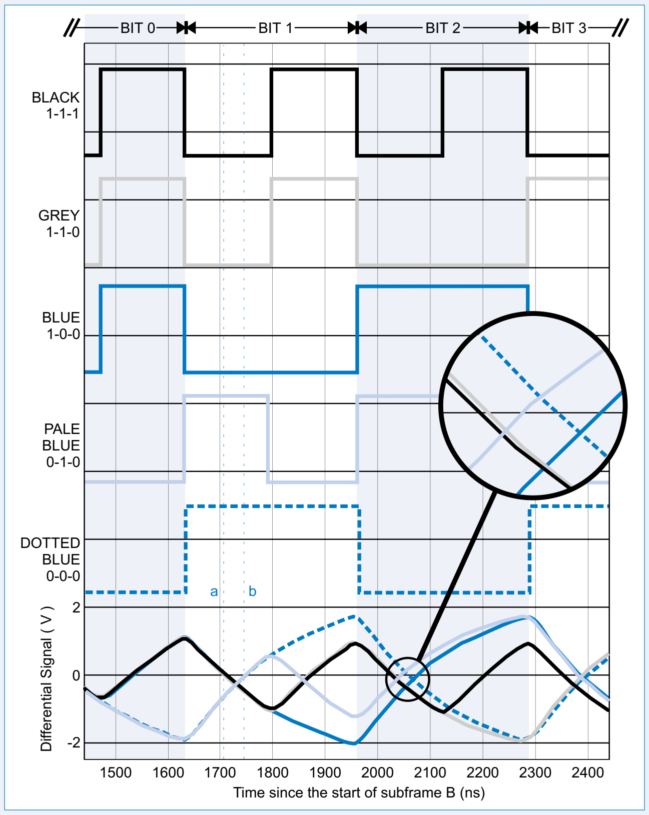


Figure 9. AES3 Intersymbol Interference

As a result of the high-frequency losses in the cable simulation the transition time is quite slow, so the zero crossings are about 100 ns after the inflections that indicate the start of the transitions. This interaction between the value of the first data symbol and the timing of the start of the second data symbol is called intersymbol interference.

This interference is more complex after the second bit symbol (about 2050 ns from the start of the subframe, also shown in the magnified view). Here there are four different zero-crossing times corresponding to the four possible bit patterns of the first two bits in the subframe. Most of the timing difference is due to the value of the second bit, but in addition there is a smaller difference relating to the state of the first bit.

small static delay to the signal which could be ignored. However, that would only be the case the pulse stream were perfectly regular—a string of embedded ones or zeros, for example. But real pulse streams consist of bit patterns which are changing from moment to moment, and in the presence of cable losses these give rise to *intersymbol interference*. The proximity and width of data pulses effectively shift the baseline for their neighbors, and with the longer rise and fall times in the cable, the transitions are moved from their ideal zero crossings.

As the AES3 interface uses the same signal to carry both clock and data, it is possible to induce jitter on the clock as a result of the data modulation. This means that care should be taken about mechanisms for interference between the data and the timing of the clock. The smearing of the waveform as a result of cable losses is one such mechanism. See Figure 9 and the *Intersymbol Interference* sidebar.

Data Jitter

Data jitter is a term used to describe the jitter of the transitions in the parts of the AES3 waveform modulated by the data. This form of jitter is often an indicator of intersymbol interference.

Figure 9 in the Intersymbol Interference sidebar illustrates this mechanism inducing data jitter of about 50 ns peak-to-peak in some of the transitions. Data jitter can also be produced by circuit asymmetries where a delay may vary between positive-going and negative-going transitions.

Preamble Jitter

Preamble jitter is a term used to describe the jitter on the transitions in AES3 preambles. The preambles are a set of static patterns which are used to identify the start of the digital audio subframes and blocks. (See Figure 8.) The Y preamble at the start of the second (B) subframe is a completely regular fixed pattern. This unchanging preamble can be used to make jitter measurements that are not sensitive to intersymbol interference, and are therefore a better indicator of either jitter at the transmitter device or noise-induced jitter, rather than jitter due to data modulation.

Interfering-Noise-Induced Jitter

If the pulse transitions were not sloped by the cable losses, the rise and fall times of the pulses would be so short that their zero crossings would be relatively unaffected by any added noise. However, the long transition times induced by cable losses allow noise and other spurious signals to "ride" the transitions, resulting in a shift of the zero crossing points of the pulses.

For example, noise on the signal can vary the time at which a transition is detected. The sensitivity to this noise depends on the speed of the transition, which, in turn, depends on the cable losses. This is illustrated in Figure 10.

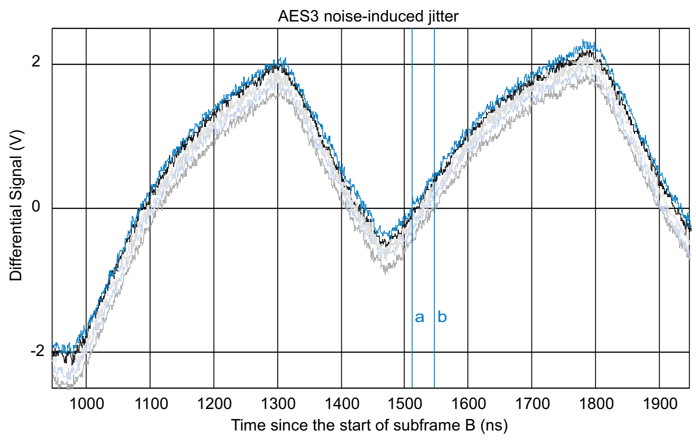


Figure 10. AES3 noise-induced jitter

The five traces on Figure 10 are all of the same part of the B subframe Y preamble. (As mentioned before, this static preamble pattern is chosen because it is not sensitive to data jitter, making the noise-induced jitter mechanism more obvious.) The two markers, "a" and "b" show the range of timings for the zero crossing resulting from the third transition. Their separation is 31 ns. In this example, the noise producing this variation is a low-frequency sine wave of about 300 mV. This type of interference might be induced by coupling from a power line.

The amount of jitter introduced by noise on the cable is directly related to the slope at the zero crossing, as voltage is related to time by that slope. With fast transitions any interfering noise will not produce much jitter: the voltage deviation will cause a smaller time deviation.

Note: In this example a long cable was simulated by the Audio Precision System Two Cascade. However, this level of jitter would be reduced by several orders of magnitude for a short interconnection.

Notice that the direction of the time deviation is related to the direction of the transition. For a transition shifted up by noise the rising transition will be early and the falling transition will be late; for a transition shifted down the opposite is true. Unlike data jitter from intersymbol interference, this form of jitter is more apparent to devices that recover a clock from a particular edge in the preamble pattern. That edge will only have one polarity and so the timing deviation of successive edges will sum together.

However, for systems using many of the edges in the subframe, transitions will be almost evenly matched in both directions and the cancellation will reduce the coupling of low frequency noise-induced jitter into the recovered clock. For noise at high frequencies successive deviations will not correlate and so cancellation will not occur.

Jitter Tolerance

An AES3 digital audio receiver should be able to decode interface signals that have jitter that is small compared with the length of the pulses that it has to decode. As the jitter level is increased the receiver will start to decode the signal incorrectly and then will fail to decode the signal—occasionally muting or sometimes losing "lock" altogether. The maximum level of jitter before the receiver starts to produce errors is called the *jitter tolerance* of the device.

As the PLL characteristics sidebar showed, a clock-recovery PLL has a low-pass characteristic analogous to a mechanical flywheel: it responds or "tracks" to changes slower than the rate of the corner frequency, and it filters out changes that are faster.

Jitter tolerance, then, is independent of frequency for jitter above the corner frequency of the receiver, but as the rate of change of the timing (the jitter frequency) is reduced, the receiver is increasingly able to follow the changes. This means that at lower jitter rates the receiver will be able to track increasing amounts of jitter, and so jitter tolerance rises.

For jitter frequencies close to the corner frequency it is possible—as a result of a poorly damped design—that the jitter tolerance is significantly reduced. This occurs because the resonance in the receiver is causing the match between the deviation of the incoming data transition timing and the receiver's estimation of the data transition timing to actually be worse than if the receiver was not tracking the jitter at all.

The AES3 interface specification defines a *jitter tolerance template*, shown in Figure 11. The tolerance is defined in UI. The line on the graph represents a lower limit for receiver jitter tolerance to sinusoidal jitter of the frequency

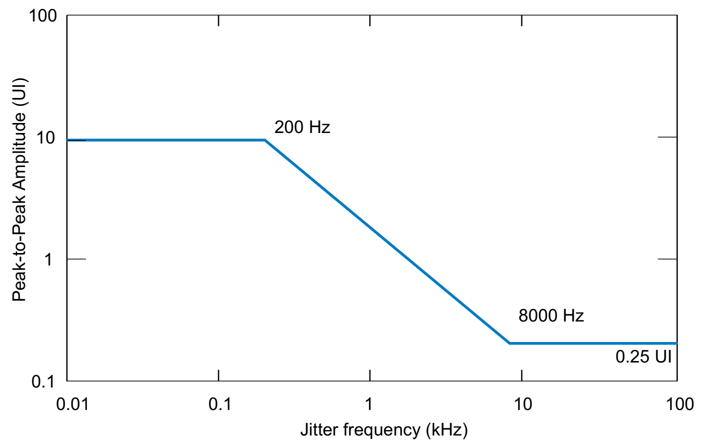


Figure 11. AES3 jitter tolerance template

shown on the X axis. Note that this template implies that receivers should have a corner frequency above about 8 kHz. This means that the receiver PLL will not be able to attenuate jitter below that frequency; instead, it will track the jitter and pass it on. A second PLL with a lower corner frequency must be used if significant jitter attenuation is required.

The Jitter Transfer Function and Jitter Gain

For a device that is synchronized to another clock (such as a digital input, a word clock, or a video sync reference) jitter on the external source could be passed through to the output. The jitter on the output is then a combination of this transferred jitter and the intrinsic jitter of the device.

Although the relation between input and output jitter can be very complex, it is still useful to model the transfer as a simple linear process. The *jitter transfer function* is a measure of the relation between input and output jitter, or *jitter gain*, versus jitter frequency.

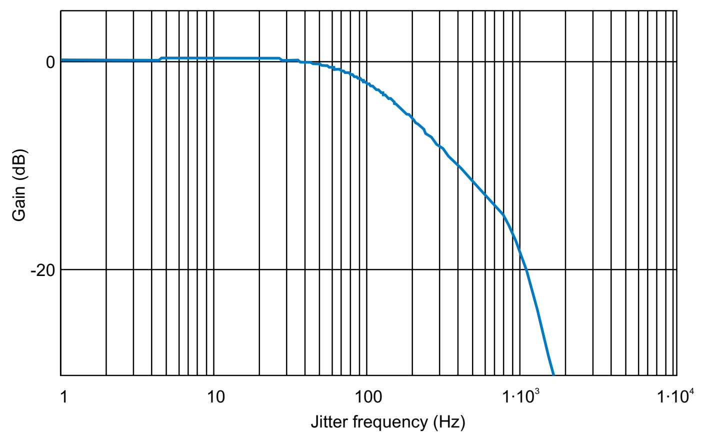


Figure 12. Jitter transfer function

Figure 12 shows the calculated jitter transfer function produced by a PLL with a corner frequency of 100 Hz. Notice that below the corner frequency the jitter gain is about 0 dB. Above the corner frequency the PLL attenuates the jitter—initially with a slope of 6 dB per octave. This design has a second-order loop filter with a corner at 1 kHz which results in an 18 dB per octave slope above that frequency.

Notice that below the PLL corner, the gain reaches a peak of about 0.5 dB. It is usual for there to be a certain amount of gain just below the corner frequency; this is called *jitter peaking* and it is a consequence of the phase characteristic of the feedback loop in the PLL.

The AES3 standard sets an upper limit of +2 dB for jitter gain.

Non-Linear Jitter Behavior

The linear jitter transfer analysis does not account for non-linear relations between input and output jitter. Phase detectors can often have a “dead” spot where they are not sensitive to small phase deviations. As a result, the PLL output will drift until the phase detector becomes active and applies a correction. This drift will cycle back and forth, producing jitter.

Another non-linear jitter mechanism is the aliasing of high-frequency jitter as it is sampled by a lower-frequency mechanism within the PLL. For example, a 48 kHz frame-rate AES3 signal with a jitter component at 47 kHz could be used to generate an internal clock signal at a 48 kHz rate in order to lock a PLL. This 47 kHz signal would alias to the much lower frequency of 1 kHz where it might not be attenuated. When measuring a jitter transfer function this behavior would make it appear that the gain rises to maxima at multiples of the frame rate.

Jitter Accumulation

In a short chain of digital audio devices, with each device locked to the previous one, there are several contributions to the jitter at the end of the chain. Each device will add its own intrinsic jitter, and each interconnecting cable will make some contribution with cable-induced jitter. There will also be some jitter gain or loss at each stage.

This process has been called *jitter accumulation*. The effect varies with the individual device jitter characteristics and the data patterns at each stage, but in some circumstances and with some “pathological” signals the jitter mechanisms could all combine in an unfortunate manner.

In a chain of devices with clock recovery systems having similar characteristics a pathological signal will have the same effect at each stage. As Table 1 shows, this can lead to a very large amount of jitter accumulation after only a few similar stages.

For the purposes of this calculation we are looking at jitter at frequencies below the jitter transfer function corner frequencies of all the devices, so jitter attenuation does not occur. Assume—for simplicity—that all the devices contribute the same amount of jitter, *J*, at each stage (this is lumping cable-induced and intrinsic jitter together). Also assume that each device also amplifies the jitter from the previous stage by the same gain—bearing in mind that gain is only possible for jitter near the peak in the jitter transfer function.

Table 1 lists the total output jitter produced at the end of three chains of stages, as a multiple of *J*:

Jitter Gain per Device	Total Jitter *(J)* after 3 Stages	Total Jitter *(J)* after 4 Stages	Total Jitter *(J)* after 5 Stages
0 dB (ideal)	3 *J*	4 *J*	5 *J*
1 dB	3.8 *J*	5.4 *J*	7.1 *J*
3 dB	6.2 *J*	10.2 *J*	15.8 *J*
6 dB	13.9 *J*	29.8 *J*	61.4 *J*

Table 1. Jitter accumulation

This shows that with a gain of 0 dB at each stage the output jitter is simply a sum of the jitter produced at each stage. (These jitter levels are peak values so they will add). Remember that this happens at frequencies *below* the corner frequency; at higher frequencies the input jitter will be attenuated, so the final output jitter will grow more slowly.

The gains of greater than 0 dB show the effect of jitter transfer function peaking. If peaking is present it will only occur near to the PLL corner frequency. Where the jitter is wide-band only a small proportion of it will be amplified and the peaking will have little effect. However, there are mechanisms that can concentrate the jitter in the region of the peak.

First, AES3 data-jitter can have narrow spectral components. With low-level audio signals, for example, the jitter will become coherent with the polarity of the signal. This occurs because for signals close to zero, the more significant bits within the data word change together as an extension of the sign bit. If the interface audio signal is a low-level tone at one frequency, then the cable-induced jitter will tend towards a square wave at that frequency. Occasionally, a spectral peak could coincide with the peak in the jitter transfer function.

In a chain of devices using clock recovery systems with similar characteristics, this signal will have the same effect at each stage. The figure of 6 dB in the table reflects levels of peaking found in equipment that had been designed before this problem was widely understood. As the table shows, this can lead to a very large amount of jitter accumulation after only a few similar stages.

The normal symptom of a pathological level of jitter accumulation is for the equipment towards the end of the chain to very occasionally lose data, or even lock. Unfortunately, the circumstances are such that it is difficult to reproduce when the maintenance engineer is called.

The AES3 specification, since 1997, has two clauses that are intended to address potential jitter accumulation problems. The primary statement specifies that all devices should have a sinusoidal jitter gain of less than 2 dB at any frequency.

In addition, there is a standard jitter attenuation specification that should be met by devices claiming to attenuate interface jitter. This requires attenuation of at least 6 dB above 1 kHz. This frequency is much lower than the jitter tolerance template corner frequency, so these devices need a transmit clock which is separate from the data recovery clock that determines the jitter tolerance.

Sampling Jitter

Sampling jitter is the variation in the timing an audio signal through jitter in an analog to digital (ADC), digital to analog (DAC), or asynchronous sample rate converter (ASRC). In the former two cases this can often be associated with an observable *sample* clock signal but in an ASRC it may be a totally numerical process, as the samples of a signal are regenerated to correspond with new sampling instants: in that case the sample clock is a *virtual sample clock*.

There are many circumstances where a sample clock has to be derived from an external source. In the domestic environment this could be a digital audio recorder or a digital surround processor where the DAC sample clock is derived from the digital input data stream. For professional applications there are also devices with DACs, applications where the sample clocks of ADCs need to be derived from an external sync or where a digital stream needs to be resynchronized to a different reference using an ASRC.

Often this external source will have jitter that can be observed, measured and commented on. However, that is not sampling jitter. The external source might make a contribution to the sample clock jitter but that contribution depends on the characteristics of the clock recovery circuit (or numerical algorithm) between the external source connection and the actual sample clock. This will have intrinsic jitter, jitter attenuation, and non-linearities in its behavior.

Sampling Jitter and the External Clock

There are many circumstances in which a sample clock must be derived from an external source. In a digital audio recorder or a digital surround processor, for example, the sample clock controlling the DAC is extracted from the input data stream. In other applications the sample clock of an ADC might need to be locked to an external sync signal, or a digital data stream might need to be resynchronized to a different clock reference using an *asynchronous sample rate converter* (ASRC).

This external clock source may well have jitter, but that, by definition, is *not* sampling jitter. The external source might make a contribution to the sample clock jitter, but that contribution depends on the characteristics of the clock recovery circuit (or numerical algorithm) between the external source connection and the actual (or virtual) sample clock. This will have intrinsic jitter, jitter attenuation, and jitter non-linearities in its behavior.

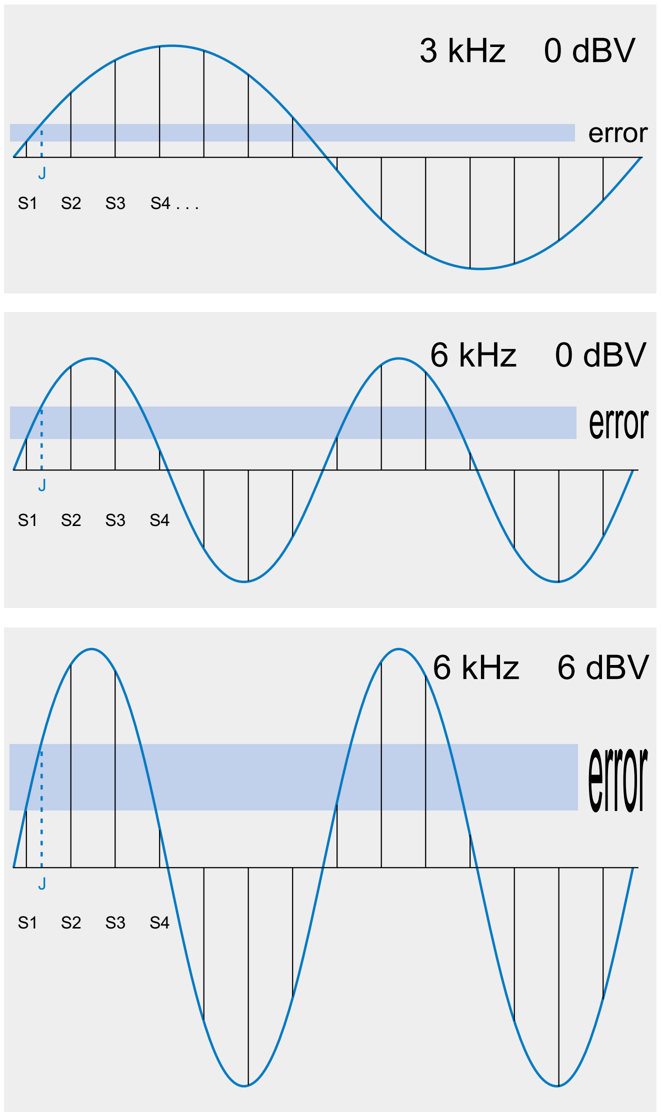


Figure 13. In these examples the sampling rate is constant, but the sampled signal is varied in frequency and amplitude. Note how the amplitude value error for a jittered sample instant (J) increases with signal rate of change.

Time-Domain Model

First, we will look at sampling jitter in the time domain.

The effect of a sample being converted at the wrong time can be considered simply in terms of the amplitude error introduced. Any signal that is not DC will change over time, and a wrong sampling instant will produce a wrong amplitude value. As you can see in Figure 13, the amplitude error is proportional to the rate of change, or slope, of the audio signal, which is greatest for high-level high-frequency signals.

Figure 14 illustrates the effect of random sampling jitter on a pure tone. The tone is shown as having an amplitude of 2 V rms and a frequency of 1 kHz. The error signal is calculated using random Gaussian jitter of amplitude 10 ns rms, and the simulation that produced this graph calculates the error of each sample at a sampling frequency

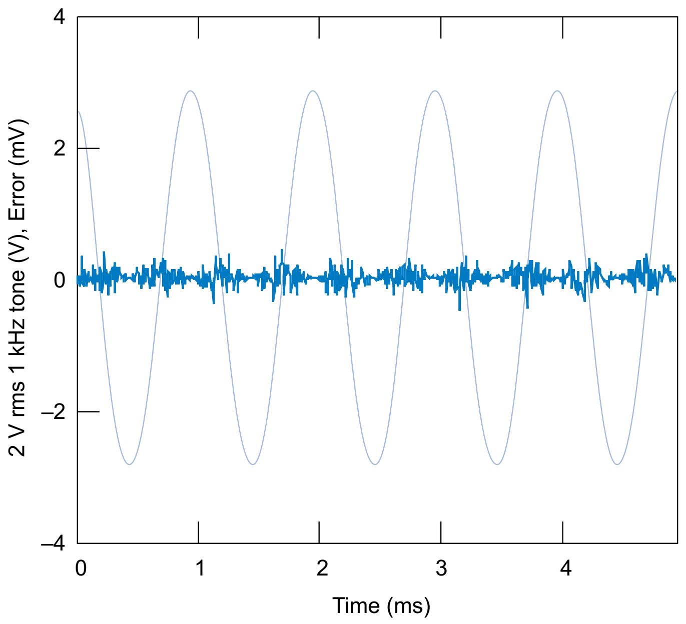


Figure 14. Sampling jitter on a 1 kHz tone. The blue line is the signal; the red line is the error introduced by the jitter, shown on a scale enlarged 1000 times

of 176.4 kHz, which represents a 4X oversampled DAC in a CD player.

Notice how the error signal and the tone intermodulate. The error is the product of the slope of the tone and the jitter; as a result there are minima in the error at the peaks of the tone where the slope is flat.

The root-mean-square (rms) error computed by the simulation is 124 μV rms, or –84 dB relative to the tone. Assuming that this error is spread fairly evenly throughout the 88.2 kHz bandwidth represented by the sampling frequency of 176.4 kHz, we can estimate that measured over the nominal audio band to 20 kHz, the noise level would be 60 μV rms. This is 90.5 dB below the level of the tone.

This method of analyzing the effect of jitter can be used to make an estimate of the acceptable level of jitter of any given form. It can be simplified to calculate the level of jitter that, if applied to a "worst-case" signal, would produce an error of amplitude equal to the quantization interval. For example, a worst-case full-scale 20 kHz sine wave in a 16-bit system would have a maximum slope of:

$$2 \times \pi \times F \times A = 4.1\,\text{LSB} / \text{ns}$$

$$\textit{where}$$

$$F = 20\ \text{kHz, the tone frequency}$$

$$A = 2^{15} = 32768\,\text{LSB, the tone amplitude (peak).}$$

From this one might conclude that the jitter level should be no more than 244 ps peak, but that limit is fairly arbitrary—there is nothing special about an error of 1 LSB amplitude—and has little relation to the audibility of the error, which will be related to the spectral content of the error.

Frequency-Domain Model

Another method of looking at the effect of jitter is to consider it as a modulation process, and analyze it in terms of frequency components. It can be shown mathematically that a simple relationship exists between a jitter spectral component, an audio signal spectral component and the resulting jitter modulation product.

If a signal is sampled with errors in the sampling instants, the effect is to modulate the signal in time. This is expressed mathematically in (1). The output signal *υ* (*t*) is a time-displaced version of the input signal, *υ*(*t*), and the variation in the displacement is the jitter.

$$\upsilon\ (t) = \upsilon(t - \Delta t). \qquad (1)$$

The effect of this can be analyzed by considering sinusoidal jitter of frequency ω_j and peak-to-peak amplitude *J*.

$$\Delta t = j(t) = \frac{J}{2} \times \sin\left(\omega_j t\right). \qquad (2)$$

The input signal may be a sine wave.

$$\upsilon(t) = A\cos\left(\omega_i t\right). \qquad (3)$$

These equations can be combined and rearranged to:

$$\upsilon\ (t) = A\cos\left(\omega_\iota\, t\right)\cos\ \frac{J\omega_i}{2}\sin\left(\omega_j t\right)\div$$

$$+ A\sin\left(\omega_\iota\, t\right)\sin\ \frac{J\omega_i}{2}\sin\left(\omega_j t\right)\div. \qquad (4)$$

Jitter amplitude (typically less than 10 ns) is generally much smaller than the signal period (typically greater than 40,000 ns). The product of small jitter modulation levels is itself very small, and for such cases we can make the following small-angle approximations:

$$\cos\ \frac{J\omega_i}{2}\sin\left(\omega_j t\right)\div\quad 1 \qquad (5)$$

and

$$\sin\ \frac{J\omega_i}{2}\sin\left(\omega_j t\right)\div\quad \frac{J\omega_i}{2}\sin\left(\omega_j t\right). \qquad (6)$$

Using these (4) becomes

$$\upsilon\ (t) = A\cos\left(\omega_i t\right) + A\frac{J\omega_i}{4}\cos\left(\left(\omega_i - \omega_j\right)t\right)$$

$$- A\frac{J\omega_i}{4}\cos\left(\left(\omega_i + \omega_j\right)t\right). \qquad (7)$$

The output signal has the input signal with two other components at frequencies offset from the input signal frequency by the jitter frequency, and their amplitude is related to the product of jitter amplitude and signal frequency. This result can be used when estimating the potential audibility of jitter modulation products.

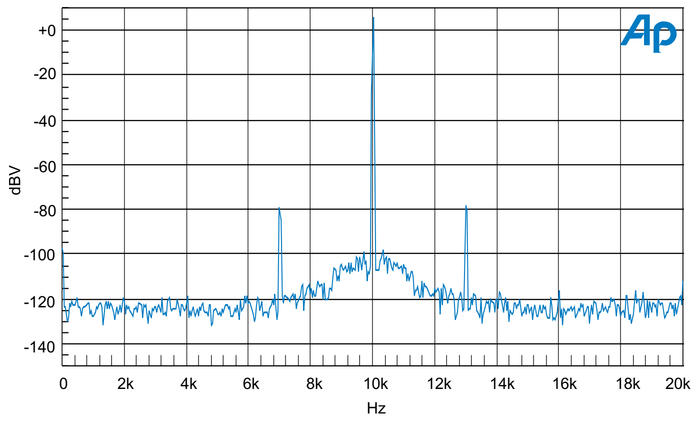


Figure 15. Jitter-modulated sidebands.

Figure 15 illustrates this effect on a real signal. The input signal is at 10 kHz and the jitter modulation is at 3 kHz. The two components at 3 kHz offset from the input signal are the upper and lower jitter modulation sidebands. (In this figure there are also “skirts” to the spectrum closer to the 10 kHz component. These are due to some low-frequency noise-like jitter in the system).

The ratio of signal to each ‘single’ sideband, in dB, is:

$$R_{ssb} = 20\log_{10}\ \frac{J\omega_i}{4}\div \mathrm{dB}. \qquad (8)$$

This result is for sinusoidal jitter components. Using Fourier analysis, more complex waveforms can be broken down into sinusoidal components and the formula can be applied.

For convenience, the formula can be modified by summing the levels in both sidebands to give a total error, and using rms jitter levels, J_n, in nanoseconds and frequency, f_i, in kHz:

$$R_{dsb} = 20\log_{10}\left(J_n f_i\right) - 104\ \mathrm{dB}. \qquad (9)$$

Influence of ADC/DAC Architecture

The effect of jitter on converters can be more complex than just the time modulation of the audio signal as discussed above. Other signals (for example, ultrasonic noise created in a noise-shaping low-bit converter) can be sampled with the desired audio signal; in some case another modulation process could taking place as well.

Oversampling Converters

An *oversampling converter* is one that is processing samples at much more than the minimum rate required by the bandwidth of the system. This oversampling rate can typically be from 2X to 256X. The higher rates also use *noise shaping*, an important technique which can help provide low-cost solutions to high-resolution conversion. (Noise shaping can produce a separate side effect that is discussed later.)

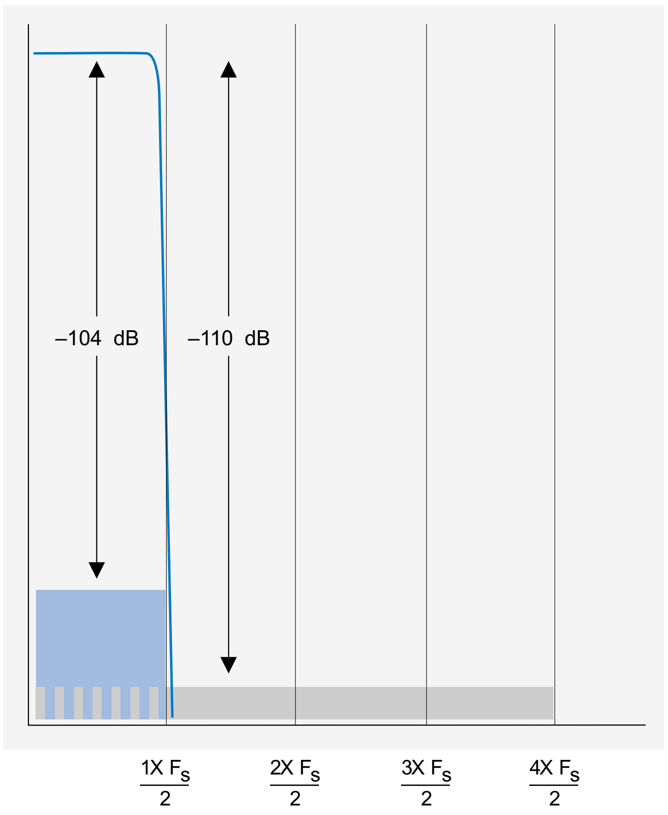


Figure 16. The blue square represents the error created by spectrally flat jitter in a 1X converter. The grey rectangle represents the jitter error in a 4X converter, which contains the same power spread over a wider spectrum. The jitter power in the audio passband has been reduced.

Since the jitter bandwidth in a sample clock can extend to half the sampling frequency of the converter, the jitter in an oversampled converter will be spread over a wider spectrum than the jitter in a non-oversampled converter. The error caused by jitter modulation is related to the jitter spectrum, so the error signal from an oversampled converter is also spread across a wider spectrum.

To illustrate this: consider a 1 kHz signal being sampled with 1 ns of spectrally flat, noise-like jitter. By calculation, this will produce a total error 104 dB below the signal. This total error figure remains the same regardless of the sample rate of the converter.

As you can see in Figure 16, in a 4X oversampled DAC this error signal will be spread over four times the frequency range compared with a 1X converter. For audio purposes, of course, we limit our interest to the 20 Hz to 20 kHz bandwidth, and a measurement made over that range contains only one-quarter of the power of the full spectrum of error noise. One-quarter the power implies one-half the voltage, resulting in an error 6 dB lower than that for the non-oversampled converter.

Jitter sources, however, are normally not spectrally flat. Jitter is usually dominated by lower-frequency components, due both to the typical phase noise spectrum of oscillators and to the low-pass jitter filtering common in clock recovery circuits. Oversampling will not reduce the impact of this lower-frequency jitter.

Jitter-Induced Tones from Noise-Shaped Converters

I have observed an interesting artifact. Very low-level tones were observed on the output of a noise-shaped one-bit converter. This was a DAC that did not use any of the above techniques to reduce sampling jitter sensitivity—just a low-jitter quartz crystal VCO to make sure that any jitter on the sample clock was at a very low level. These tones appeared to be related to the modulation of the VCO control voltage. However, they went away when a higher speed and lower distortion op-amp was used in the post filter. I concluded that the effect was of non-linearity (in the lower performance op-amps) on the jittered ultrasonic noise that demodulated the jitter. (As the modulation would be similar to phase modulation, rather than to amplitude modulation, this requires some asymmetry between the upper and lower sidebands. The jitter modulation produces upper and lower sidebands which have opposite phase. If they had the same amplitude, then on demodulation they would cancel.)

Normally jitter does not produce tones in the absence of signal. Modulation sidebands may be tonal if the jitter is tonal—but they do have to be sidebands of a modulated signal.

Noise-Shaping and One-Bit Converters

For high rates of oversampling it is possible to reduce the number of bits while shaping the resultant quantization noise out of the audio band. This technique has many advantages, but it *does* generate ultrasonic noise. The level of this noise is related to the quantization interval. For a one-bit converter the total noise is close to the full-scale level of the converter.

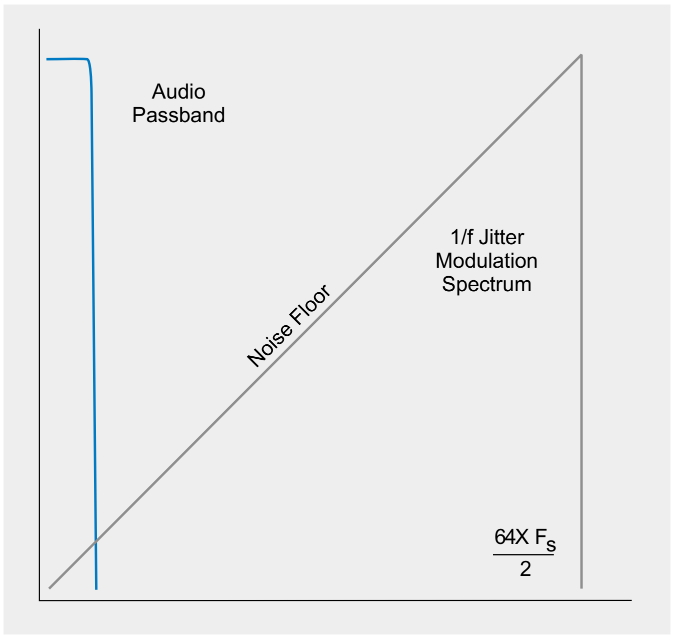


Figure 17. Ultrasonic jitter modulation products.

The action of sampling jitter on this ultrasonic noise produces modulation products, just as it does on audio signals. These modulation products can fall in the audio band, and as the ultrasonic noise is present even when the audio signal is at a low level, there will be no benefit from masking. The effect is to raise the noise floor and so reduce the dynamic range of the converter.

To illustrate the scale of the problem: consider the ultrasonic noise produced in a 64X oversampled 48 kHz delta-sigma converter. (ADC or DAC—it does not matter in this example.) The ultrasonic noise is in a band starting above the audio band and going on to half the sample rate, 1.5 MHz. For 1 ns rms jitter the formula would imply modulation effects at levels of the order of:

$$20\log_{10}\left(1\,\mathrm{ns}\times 1000\,\mathrm{kHz}\right)-104=-44\ \mathrm{dB}.$$

Since this noise is spread over the band to 1.5 MHz, the level within the audio band would be less than this. With spectrally flat jitter it would be about 20 dB less, but it is likely that there would be at least a $1/f$ characteristic to the jitter spectrum, so the reduction may be increased to 40 dB. This leaves a noise floor of about –84 dB FS.

Reducing Jitter Sensitivity in Delta-Sigma Converters

Most of the commercially available integrated delta-sigma converter devices do not have anything like this sensitivity to jitter. How is that?

We find that it is possible to largely eliminate the effect of jitter modulation of high level ultrasonic quantization noise by *filtering the noise out before it is sampled*. Several techniques are available:

Switched-Capacitor Filters

Sampling or re-sampling occurs at the interface between the sampled signal domain and the continuous-time signal domain, which is not always the same as the interface between the digital and analog domains. A switched-capacitor filter operates on analog signals in the sampled signal domain.

In a delta-sigma ADC, the ultrasonic quantization noise is used in the delta-sigma modulator for feedback. This noise can be kept in the sampled signal domain if the analog filters within the modulator are implemented as switched-capacitor filters.

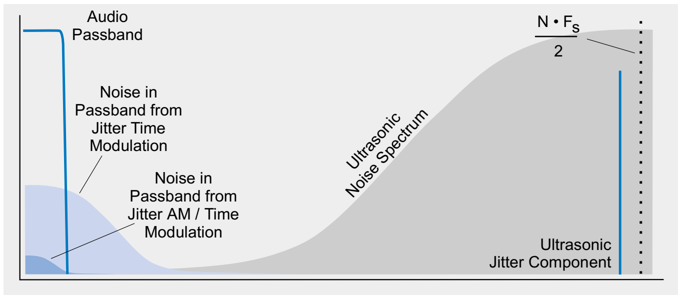


Figure 18. Spectrum of amplitude + time modulation products of ultrasonic jitter with ultrasonic noise.

In a delta-sigma DAC, the ultrasonic noise on the DAC output can also be attenuated with a switched-capacitor filter.[4]

Multi-Bit Noise-Shaped Converters

Since integrated switched-capacitor filters have to be quite large to have a good noise performance, an alternate solution is finding favor. By increasing the number of levels in the quantizer, the quantization noise in the modulator is reduced. This reduces the ultrasonic noise by the same proportion. The Analog Devices AD1855, for example, has a 64-level modulator and does not use a switched-capacitor filter.[5]

Jitter-Induced Amplitude Modulation

There is another solution to high sensitivity to jitter due to the modulation of the ultrasonic quantizer noise. It is the combination of the jitter-induced time modulation with jitter-induced amplitude modulation.

Ordinarily, a DAC has a *current* or *voltage* output which is independent of sampling frequency. However, a DAC that uses a quantum of *charge* with every sample, rather than current or voltage, will have a current output that *scales with sampling frequency*. In this circumstance sampling jitter produces an amplitude modulation effect, which combines with the pure jitter time modulation in an advantageous manner.

With this kind of DAC, sampling jitter produces an amplitude modulation effect with the following output:[6]

$$\begin{aligned}\upsilon(t)=A\cos\left(\omega_i t\right)-A\frac{J\omega_i}{4}\cos\left(\left(\omega_i-\omega_j\right)t\right)\\ -A\frac{J\omega_i}{4}\cos\left(\left(\omega_i+\omega_j\right)t\right).\end{aligned}\qquad(10)$$

This amplitude modulation combines with the pure jitter modulation to produce the following:

$$\begin{aligned}\upsilon(t)=A\cos\left(\omega_i t\right)+A\frac{J\left(\omega_i-\omega_j\right)}{4}\cos\left(\left(\omega_i-\omega_j\right)t\right)\\ -A\frac{J\left(\omega_i+\omega_j\right)}{4}\cos\left(\left(\omega_i+\omega_j\right)t\right).\end{aligned}\qquad(11)$$

The sidebands for this combination now scale with the sideband frequencies, $\omega_i-\omega_j$ and $\omega_i+\omega_j$, rather than the modulated frequency, ω_i. Where ω_i is ultrasonic (and the sideband offset ω_j is very large) this reduces the impact of the jitter on any sideband modulated down into the audio band in approximate proportion to the ratio between the ultrasonic noise frequency component and the audio band frequency.

Where the signal under consideration is at high level and high frequency (a component of the ultrasonic noise) and the sideband offset is very large (due to an ultrasonic jitter signal), sidebands can be modulated down to much lower frequencies. This technique reduces the impact of the jitter modulating the ultrasonic noise down into the audio band in approximate proportion to the oversampling ratio, e.g. 256:1.

Sampling Jitter in Rate Converters

Sample rate converters (SRCs) are used to convert a signal from one sample rate to another. The conversion involves interpolating between the sample points on the input stream to generate values for the new sample points.

Where the two sample rates have an exact integer relationship the new sample points can be determined with no error. In that case it is possible to do the conversion with no sampling jitter, but the input and output streams need to be synchronized. A 44.1 kHz to 96 kHz sample rate conversion, for example, can be done using the mathematical relation of 320/147. The timing of the output stream can be determined from the input stream so that every 147 input samples corresponds with 320 output samples. The interpolation filter coefficients can be computed based on this exact relation. This sort of SRC is called a *synchronous* sample rate converter (SSRC).

Often the output sample frequency cannot be locked to the input. Additionally, some equipment is designed to retain the flexibility to cope with an arbitrary relationship between input and output timing. In these cases the conversion is more complex and includes an algorithm that tries to track the relation between the input and output samples based on their actual time of arrival. This sort of SRC is called an *asynchronous* sample rate converter (ASRC).

Virtual Timing Resolution

The algorithm used to estimate timing relations in an ASRC takes as an input the timing of the sample clock of one of the streams, and measures that with a higher rate clock that is synchronous with the other stream. The jitter in this measurement is determined by the resolution of the measurement clock.

For example, when converting from 48 kHz to 96 kHz the measurement clock may be working at 256 X 96 kHz. This resolution of 40 ns is the amplitude of the time quantization jitter being fed into the time-tracking algorithm.

This potential source of sampling jitter can have strong spectral components: if the 48 kHz clock is 5 ppm low and the 256 X 96 kHz rate is 6 ppm low, then the 40 ns time quantization jitter will be in the form of a sawtooth at a rate of about 25 Hz (256 X 96 kHz X 1 ppm).

Virtual Jitter Attenuation Characteristic

The ASRC timing estimation algorithm will have a jitter attenuation characteristic that can be modeled as a low-pass filter with a corner frequency. However, as this is a numerical process, if the device has enough mathematical resolution the filter corner frequency can be set very low. This means that an ASRC can have a high level of jitter attenuation.

As integrated ASRCs become less expensive, they are seen as a low-cost solution for the effective elimination of sampling jitter for DACs. The output sampling frequency can be fixed to a low-jitter, free-running crystal oscillator and the incoming data stream can be converted to that sampling frequency in the ASRC. A measurement of the clock at the DAC may reveal the low jitter of the crystal oscillator.

However, the re-sampling process within the ASRC needs to be considered as well. As the ASRC jitter is purely a deviation in the numerical value generated by the timing estimation algorithm, it cannot be measured directly. However, it can be evaluated by examining the effect on a high-frequency, high-level digital tone signal passing through the device.

Sampling Jitter Transfer Function

It is often convenient to assess the jitter performance of a device, be it an ADC, DAC or an ASRC, through its transfer function; that is, the effect it has on an audio signal. For the ASRC it may be the only method available.

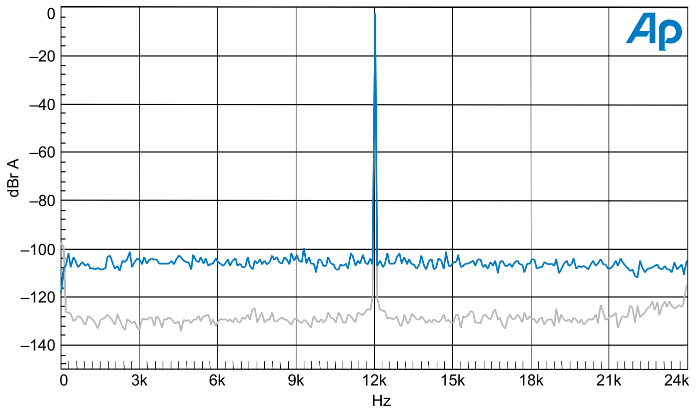


Figure 19. FFT of DAC stimulated with a 12 kHz tone at –3 dB FS, with (blue trace) and without (grey trace) 9.8 ns peak-to-peak wideband jitter.

Figure 19 shows the frequency spectrum of a DAC stimulated with a 12 kHz tone at –3 dB FS. The digital input signal to the DAC is used to recover the sample clock, and the effect of the wideband jitter on that input is to raise the noise floor evenly throughout the band.

Jitter attenuation does not have a flat response. Since the jitter stimulation is flat and the modulation effect is also flat, one can conclude that in this case there is no jitter attenuation.

The System Two analog analyzer reports that the 22 kHz unweighted noise is 96 dB below the tone without the jitter stimulus and 80 dB below with the jitter. From this we can calculate that the jitter producing modulation of up to +10 kHz and –12 kHz offsets is 1.32 ns rms, or 12 ps/√Hz.

More accurate spot frequency measurements can be made using a sinusoidal jitter stimulus. Figure 20 illustrates

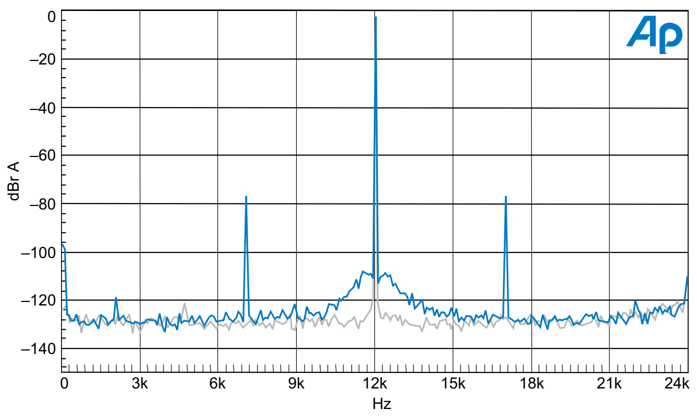


Figure 20. FFT of DAC stimulated with 12 kHz tone, with (blue trace) and without (grey trace) 3.5 ns rms 5 kHz sine wave jitter.

this. The error signal—dominated by the sidebands—is 71.4 dB below the 12 kHz tone. By calculation we can see that this corresponds with sampling jitter of 3.5 ns—the same level as the jitter applied to the interface. This indicates that at 5 kHz there is no jitter attenuation between the applied stimulus jitter on the interface and the sampling clock on the DAC. (The skirts around the 12 kHz tone are probably low-frequency noise in the jitter generation mechanism.)

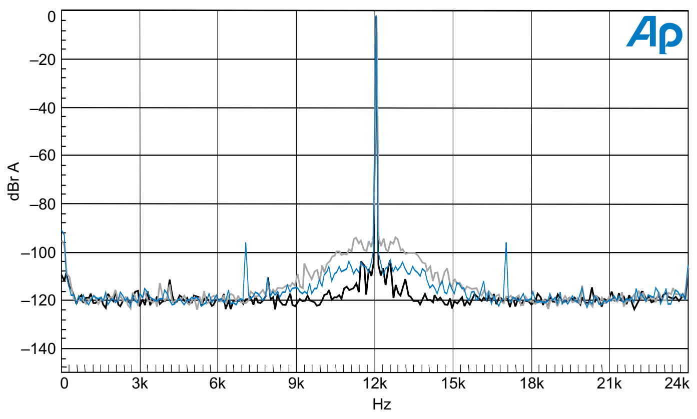


Figure 21. GREY: FFT of 12 kHz tone with 9.8 ns peak-to-peak wideband jitter; BLUE: with 3.5 ns rms sine wave jitter; BLACK: no jitter.

As an example of how the results could vary, the same tests were repeated with a different device. Figure 21 shows the FFT traces. Notice that the 5 kHz sidebands are attenuated compared with Figure 19, and the higher-frequency components resulting from the wide-band jitter are attenuated relative to Figure 20.

Other Points to Note

The wide-band jitter components shown in Figure 21 are actually at a higher level where the jitter is not attenuated at low frequencies (close to the 12 kHz tone). This apparent increase in sampling jitter compared with the earlier result is not due to jitter gain. Instead, the wide-band jitter is being aliased by the interface receiver and the clock recovery system. If, for example, the sampling clock recovery system is using a 48 kHz clock from the receiver and the jitter has a bandwidth of 200 kHz, then the jitter in the region from 24 kHz to 200 kHz is aliased into the region from 0 Hz to 24 kHz. This increases the jitter noise density within the 24 kHz region by $10\log(200/24) = 9.2$ dB.

Another feature to note is that there is significant low-frequency jitter even without the jitter stimulus, and with the sine stimulus the low-frequency jitter increases. These effects are not representative of the linear jitter transfer function but indicate low-frequency intrinsic jitter. The increase when the 5 kHz tone jitter is applied is possibly due to low-frequency noise in the jitter generation mechanism.

Sampling Jitter / Data Jitter Susceptibility.

J-test is an AES3 test signal that was developed to maximize the coherence of data patterns while at the same time providing a basic high-level stimulus tone. This test stimulates worst-case levels of data-jitter. The signal has two components, the first being an un-dithered square wave with a period of 4 samples. A cycle of this is shown here in hexadecimal notation:

```
C00000       (−0.5)
C00000       (−0.5)
400000       (+0.5)
400000       (+0.5)
```

On conversion to analog at a sample rate of 48 kHz this signal would produce a sine wave with an amplitude of –3.01 dB FS at 12 kHz. (It looks like a square wave with a peak amplitude of –6.01 dB FS but in a properly band-limited system this sequence of values represents a sine wave of amplitude –3.01 dB FS.)

This is added to the second component, an undithered 24-bit square wave of amplitude 1 LSB, made by switching between the following:

```
000000       (0)
FFFFFF       (−1 LSB)
```

This square wave is repeated at a low frequency. The frequency is not critical but, for a sample frequency of 48 kHz, a rate of 250 Hz is normally used, as that makes the signal synchronous with the AES3 channel status block of 192 samples.

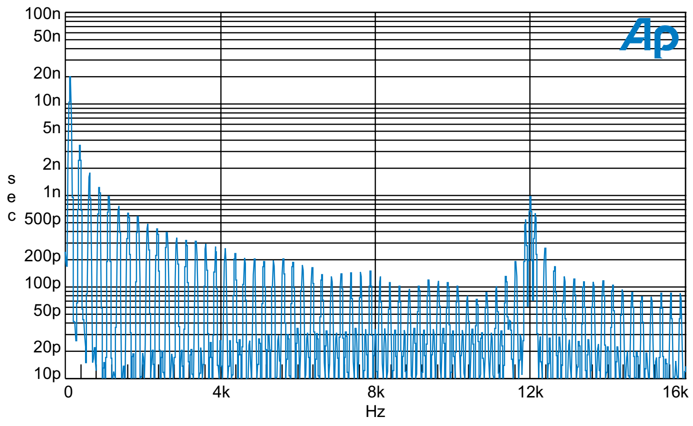


Figure 22. J-test jitter spectrum after cable simulation.

The combination of these signals results in the following 192-sample-long cycle of 24-bit data values:

```
C00000 C00000 400000 400000 (x 24)
BFFFFF BFFFFF 3FFFFF 3FFFFF (x 24)
```

The low-frequency coherent alternation in the values of the 22 LSBs produces strong jitter spectral components at 250 Hz and at odd multiples of that frequency. Figure 22, an FFT of the detected jitter signal in the Audio Precision System Two, illustrates this (but using a 384-cycle version of J-test producing a lower rate of 125 Hz). The intersymbol interference has been induced by the cable simulation. Notice that the amplitude axis is calibrated in seconds rms. This test was performed at 48 kHz. The component at 125 Hz has an amplitude of 19.91 ns. The jitter observed on the interface signal was about 35 ns peak-to-peak: this plot is of jitter at a part of the waveform where the amplitude is somewhat reduced from this.

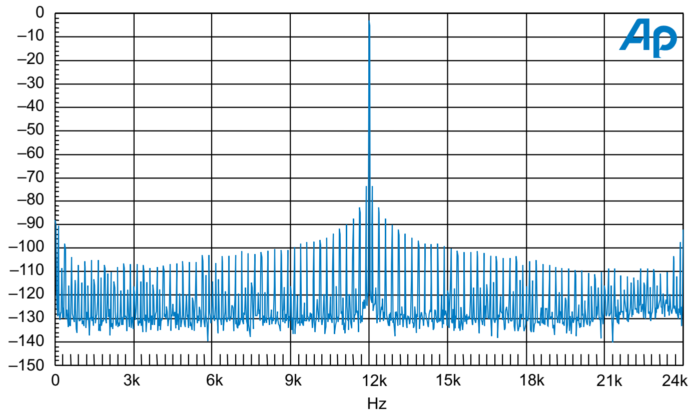


Figure 23. FFT spectrum showing jitter modulation products from J-test after a cable simulation.

Figure 23 shows an FFT of the analog output of the first test device with J-test applied. Notice that the jitter sidebands follow the interface jitter spectrum reliably, which means that the test device is susceptible to data jitter on the interface.

The shape of each sideband matches the interface jitter spectrum of the previous figure, so we can also conclude that it does not have jitter filtering within the band. The 125 Hz sidebands are each about 70 dB below the stimulus tone (67 dB for both sidebands together). This corresponds with sampling jitter at that frequency of amplitude

$$\text{antilog}\left((104 - 67)/20\right)/12 = 6 \text{ ns rms.}$$

Audibility considerations

It is one thing to be able to identify and measure sampling jitter. But how can we tell if there is too much?

A recent paper by Eric Benjamin and Benjamin Gannon describes practical research that found the lowest jitter level at which the jitter made a noticeable difference was about 10 ns rms. This was with a high level test sine tone at 17 kHz. With music, none of the subjects found jitter below 20 ns rms to be audible.[7]

This author has developed a model for jitter audibility based on worst case audio single tone signals including the effects of masking.[8] *This concluded:*

"Masking theory suggests that the maximum amount of jitter that will not produce an audible effect is dependent on the jitter spectrum. At low frequencies this level is greater than 100 ns, with a sharp cut-off above 100 Hz to a lower limit of approximately 1 ns (peak) at 500 Hz, falling above this frequency at 6 dB per octave to approximately 10 ps (peak) at 24 kHz, for systems where the audio signal is 120 dB above the threshold of hearing."

In the view of the more recent research, this may be considered to be overcautious. However, the consideration that sampling jitter below 100 Hz will probably be less audible by a factor of more than 40 dB when compared with jitter above 500 Hz is useful when determining the likely relative significance of low- and high-frequency sampling jitter.

References

1. AES3-1992—'Recommended Practice for Digital Audio Engineering—Serial Transmission Format for Two-Channel Linearly Represented Digital Audio Data' J. Audio Eng. Soc., vol. 40 No. 3, pp 147-165, June 1992. (The latest version including amendments is available from www.aes.org).

2. IEC60958-3:2000—'Digital audio interface— Part 3 Consumer applications' International Electrotechnical Commission, Geneva. (www.iec.ch).

3. Julian Dunn, Barry McKibben, Roger Taylor and Chris Travis—'Towards Common Specifications for Digital Audio Interface Jitter' Preprint 3705, presented at the 95th AES Convention, New York, October 1993.

4. Nav Sooch, Jeffrey Scott, T. Tanaka, T. Sugimoto, and C. Kubomura—'18 bit Stereo D/A Convertor with Integrated Digital and Analog Filters' Preprint 3113, presented at the 91st AES Convention, October 1991.

5. Robert Adams, Khiem Nguyen and Karl Sweetland, 'A 112 dB SNR Oversampling DAC with Segmented Noise-shaped Scrambling', AES Preprint 4774 presented at 106th AES Convention, San Francisco, September 1998.

6. Julian Dunn—'Jitter and Digital Audio Performance Measurements', Published in 'Managing the Bit Budget', the Proceedings of the AES UK Conference, London, 16-17 May 1994.

7. Eric Benjamin and Benjamin Gannon, 'Theoretical and Audible Effects of Jitter on Digital Audio Quality', Pre-print 4826 of the 105th AES Convention, San Francisco, September 1998.

8. Julian Dunn—'Considerations for Interfacing Digital Audio Equipment to the Standards AES3, AES5, AES11' Published in 'Images of Audio', the Proceedings of the 10th International AES Conference, London, September 1991. pp 115-126.

About This Series

Audio Precision has commissioned noted consultant and designer Julian Dunn to write four TECHNOTES, covering both theoretical aspects and practical techniques important to understanding and measuring digital audio signals.

With a System Two Cascade at hand, Mr. Dunn looks closely at the conversion and transmission of digitally-encoded audio and carefully examines the various links in the digital chain.

- TECHNOTE 23, "Jitter Theory," studies the causes and effects of the interface timing variations called *jitter* with a number of tests designed to characterize this pervasive malady.
- TECHNOTE 24, "Analog-to-Digital Converter Measurements," looks at key ADC parameters and behavior and includes 15 APWIN Basic procedures to run the necessary tests.
- TECHNOTE 25, "Digital-to-Analog Converter Measurements," does the same for DACs. A sidebar looks at dither. Twenty-five procedures are included.
- TECHNOTE 26, "The Digital Interface," discusses the AES3/IES60958 digital interface, examining the basic format and the means of characterizing the signal. Sidebars focus on the international standards and on synchronization considerations.

Julian Dunn took degrees in Astronomy and then Medical Electronics at London University, where he first became interested in signal processing. After graduating in 1984 he joined the BBC Designs Department, also in London. There he started to design digital audio equipment, as part of work for BBC Radio in prototyping equipment for use with the new digital audio recorders, mixing consoles and transmission systems.

After a year working at the Mullard Radio Astronomy Observatory in Cambridge, England, Julian joined Prism Sound as a consultant, where he returned to designing digital audio equipment. In 1998 he formed his own digital audio design company, Nanophon. Nanophon provides specialist consultancy in digital audio conversion, DSP software and hardware, digital audio interfacing and clock recovery systems. The Nanophon Web site is at www.nanophon.com.

Julian has presented technical papers to AES conferences and conventions on various topics in digital audio, and has been contributing to the work of the AES standards digital audio subcommittee since 1991. Since last year he has been the convener of the IEC standards working group responsible for digital audio and video interfacing standards.

Among his leisure interests Julian enjoys watching cricket, traveling the Caribbean and the maintenance and repair of old equipment of various sorts.

Julian Dunn can be contacted at tn23@nanophon.com.

NOTES

NOTES

TECHNOTES are hints and information to assist APWIN users to perform specialized tasks or unusual tests.

Audio Precision
PO Box 2209
Beaverton, Oregon 97075-2209
Tel: 503-627-0832
Fax: 503-641-8906
US Toll Free: 1-800-231-7350
email: info@audioprecision.com
Web: audioprecision.com