Audio precision®
Beaverton, Oregon

AUDIO.TST®

SYSTEM ONE AND TWO NEWSLETTER

Volume 14 Number 1
December, 1999

In This Issue...

Tech Note Series by Julian Dunn

Audio Precision is pleased to be publishing a series of articles over the next year written by one of the world's premier designers and consultants in digital audio, Julian Dunn.

First up will be "Jitter Theory," available in the spring of 2000, followed soon by "Analog to Digital Conversion." Later in the year we will distribute "Digital to Analog Conversion" and "Digital Interface."

Like all Audio Precision technical material, you will be able find Julian Dunn's articles on the Audio Precision Web page at www.audioprecision.com as PDF documents. If you would prefer hard copy, please contact us at sales@audioprecision.com or 1-800-231-7350.

The articles in the series will also be condensed and printed as installments in *AUDIO.TST*, beginning with "Jitter Theory—Part 1" in this issue.

Jitter Theory—Part One

Interface Jitter

By Julian Dunn

Introduction

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*.

Jitter can affect a digital audio signal in two broad realms: in the *interface* and in *sampling*. In this issue of *AUDIO.TST* we will look closely at *interface jitter*, saving *sampling jitter* for the March 2000 issue.

What is jitter?

Jitter is the variation in the time of an event—such as a regular clock signal—*from the ideal.* The zero-crossing transitions of many of the pulses in the data stream are seen to vary

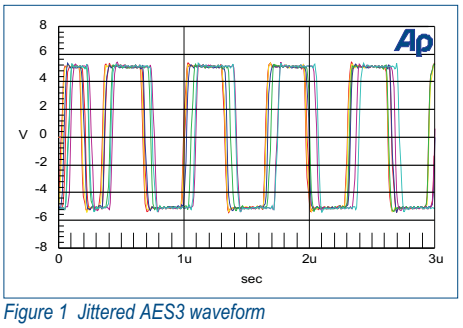


Figure 1 Jittered AES3 waveform

dynamically from an ideal clock timing; essentially, jitter is phase modulation of the transitions of the data stream.

When very little jitter is present, the pulse transitions are moved back or forth by only a small measure of time. When the jitter is increased the transitions have a larger range of time displacement. In discussing jitter, then, the *jitter amplitude* is expressed in units of *time*, not *voltage*, which can lead to some surprising graph coordinates (time vs. time, for example) when viewing jitter waveforms.

Jitter frequency is the rate at which this phase-shifting is happening. Like other noise or interference signals, the jitter modulation can be a pure and regular sine wave, a complex waveform or have a completely random character. The jitter signal can be extracted from the pulse stream and studied for clues as to its source using familiar analysis tools.

The Unit Interval

The *unit interval* (UI) is a measure of time that scales with the interface data rate, and is a convenient term for 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 UI = 1 / (128 x 48000) = 163 ns

The UI is used for several of the jitter specifications in AES3 (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.

How can you see jitter?

If you are fortunate enough to have your perfect reference clock available on a nearby spigot, you can compare the clock with the jittered signal in the time domain, perhaps using dual oscilloscope traces, for example. However, this can be difficult to see.

If the scope is triggered from the reference clock and the 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*. The opening in an eye pattern is narrowed by the time spread of the pulse transitions. A narrow eye, then, indicates jitter.

In practice, there are often no ideal clocks to compare with, and real jitter measurements must be

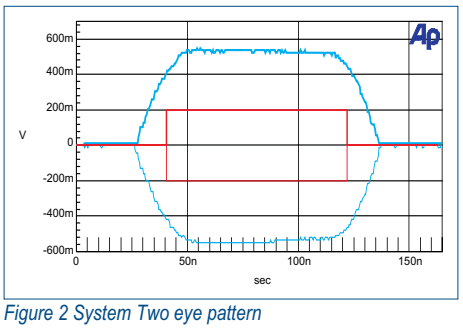


Figure 2 System Two eye pattern

self-referenced—made relative to the signal itself. This is done by approximating the ideal clock by phase-locking a relatively low-jitter oscillator to the jittered signal or real clock, using a phase-locked loop (PLL).

An oscilloscope without an external trigger will render a different form of self-referenced view of the digital signal, indicating *interval variations* rather than jitter. There is a relationship between the two, but at some frequencies jitter will not be shown and at others the jitter amplitude will appear doubled. This technique can be very misleading, being particularly insensitive to low frequency jitter.

Using DSP techniques, the DSP analyzer's internal clock can be locked to the incoming signal to approximate the ideal clock reference. The DSP analyzer can then capture the signal (and its jitter) very accurately and can be set to display the pulse stream in the time domain or as an eye pattern, or the analyzer can extract the jitter

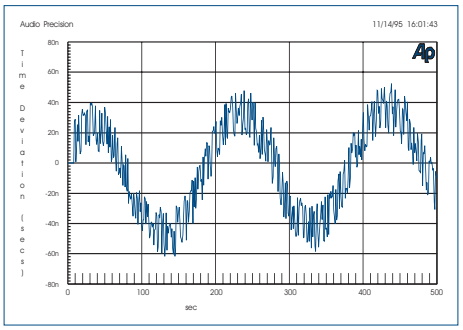


Figure 3 5 kHz jitter vs. time

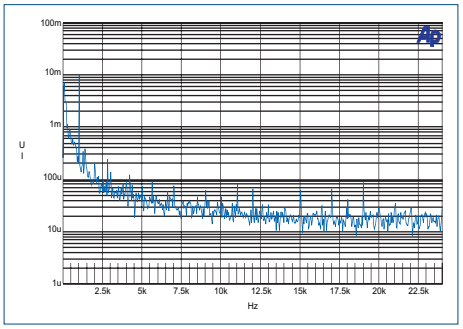


Figure 4 FFT spectrum analysis of jitter signal

signal for simple display or subject it to more powerful techniques such as FFT spectrum analysis.

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.

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.

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 alone. Quartz oscillators have low phase noise, 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 in order to synchronize with the external reference. The intrinsic jitter of the oscillator is 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.

Cable-induced jitter

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.

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

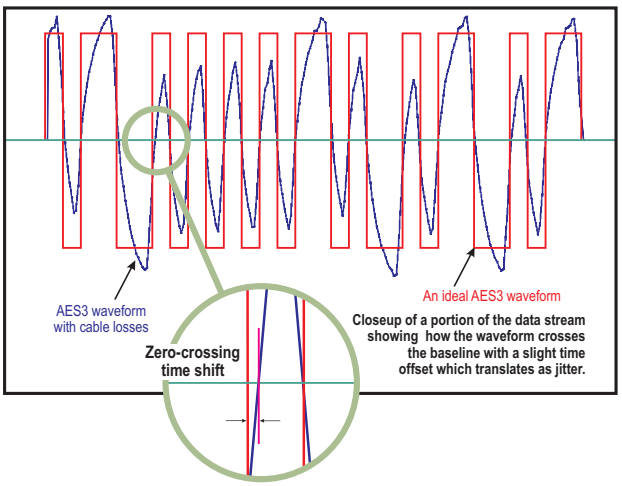


Figure 5 AES3 ideal waveform with cable-affected waveform overlaid

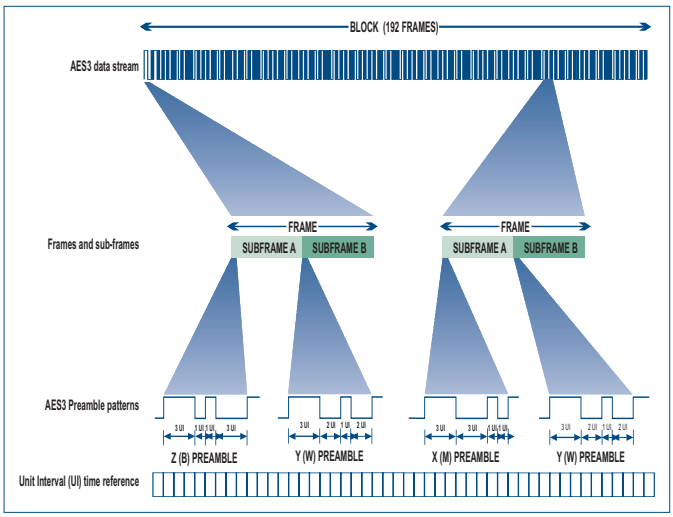
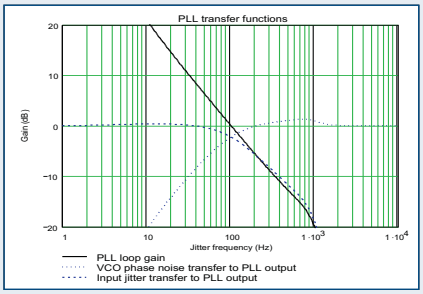


Figure 6 AES3 data pattern. Note Y preambles are identical in every frame

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 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.



in a small static delay to the signal which could be ignored. However, it would only be so if the pulse stream were perfectly regular; a string of embedded ones or zeros, for example. But this is not the case: real pulse streams consist of bit patterns which are changing from moment to moment, and in the presence of cable losses this gives 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 the *Intersymbol Interference* sidebar on the next page.)

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 8 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. 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 7.

The five traces on Figure 7 are all of the same part of the B subframe Y

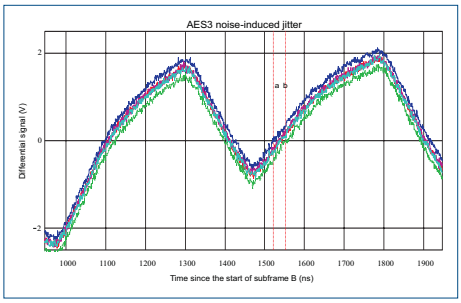


Figure 7 AES3 Noise-induced jitter

Intersymbol Interference

Figure 8 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 blue signal represents "1-1-1," the magenta is "1-1-0," the cyan "1-0-0," the red "0-1-0" and the green "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. 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 blue, magenta and cyan traces have a transition starting at 1465 ns (9 UI) from the subframe start because they have an initial "1" in their data. The red and green 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 markers "a" and "b" indicate that the times of the zero-crossings from those transitions are 1705 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."

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.

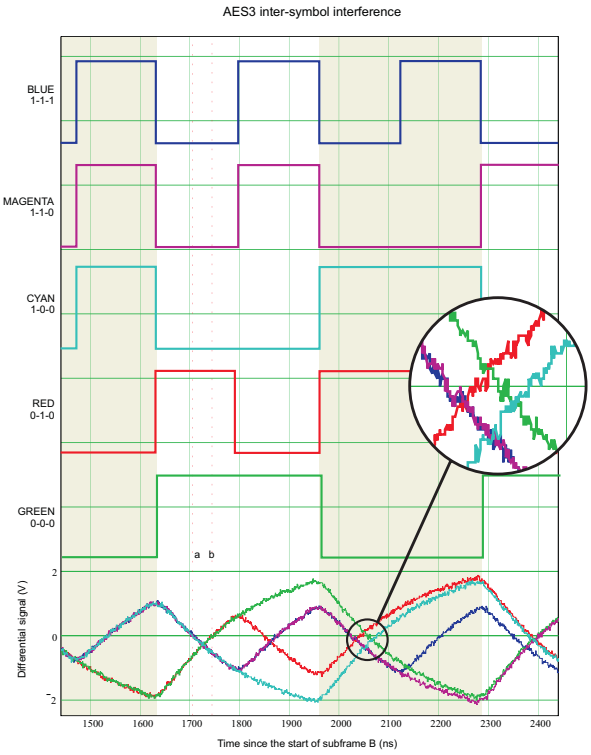


Figure 8 AES3 Intersymbol Interference

This interference is more complex after the second bit symbol (about 2050 ns from the start of the subframe, also in expanded 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.

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 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.

A clock-recovery PLL has a low-pass characteristic analogous to a mechanical flywheel: it responds or "tracks" well to slower changes, and filters out faster changes.

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.

The AES3 interface specification defines a *jitter tolerance template*, shown in Figure 9. The line on the graph represents a lower limit for receiver jitter tolerance to sinusoidal jitter of the frequency 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.

Jitter gain and the jitter transfer function

For a device that is synchronized to

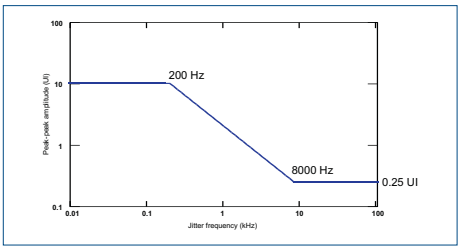


Figure 9 AES3 jitter tolerance template

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

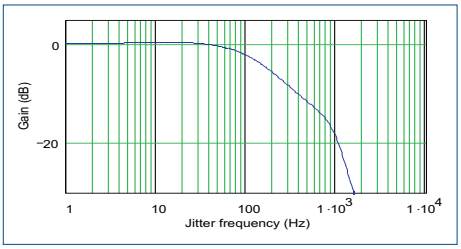


Figure 10 Jitter transfer function

frequency.

Figure 10 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.

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

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.

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

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

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.

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 convenor 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, which old equipment includes his own 28-year-old Morris Marina, which jitters a bit.