APPNOTE 001

MEASURING JITTER SPECTRUM WITH A DSO

A basic scope, Python, and some spreadsheets…can $50K be saved?

Rev 1.0b

6-July-20

Clockworks Signal Processing

http://clk.works/

1. TABLE OF CONTENTS

1 INTRODUCTION

Jitter has been an issue with audio since the first recordings were made on wax cylinders and wire tape recorders. 100 years ago it was called wow and flutter. Whether a continuous (analog) or discrete (digital) system, the effect is the same as modulating the waveform, which produces enharmonic content (sidebands) that generally detracts from the listening experience. Analog recording systems with their high noise and distortion levels (compared to digital) could perhaps get away with more bad things as the noise can mask low level distortion products.[1] Digital systems basically have "straight wire with gain" properties (when properly designed) can more easily expose the problems created by jitter.

For digital systems clock jitter is usually what is investigated as that can be related to how well an ADC or DAC samples; if you take the sample at the wrong instant in time you will read the wrong level, and that's distortion.

In real life it is not as simple as the above explanation; we can summarize as *all other things being equal, jitter needs to be minimized in ADC and DAC clocking.*

To minimize it, we need to measure it in a way to understand if it's a problem and if so, what the causes are.

If you've got $50,000+ to spend, any of the big name test equipment companies will happily sell you instruments that can measure sub pico-second jitter.

A garden variety DSO[2] and a little bit of software can produce results that are good enough for eliminating gross jitter problems, but for critical audio work a higher performance measurement system is needed. The techniques here would apply to using that better DSO, as well as the timing data produced by the scope's internal software would need to include a jitter spectrum and that's not a common capability, or a capability only provided by expensive add-on software.

[1] These days it's not that difficult to create low distortion audio circuits that perform a magnitude or more better than 20 years ago and cost less. The parts are simply way better, as well as circuit topologies have evolved.

[2] Nothing with turnips growing out of it. More on the scope later.

This AppNote[3] will explain the procedure to use a DSO and measure some actual systems to provide better context.

1.1 READERS DIGEST SUMMARY

TL;DR: it's good enough. Around 200 psec of instrument jitter below 1 kHz jitter frequencies and 15 psec above a few kHz. Want to know more? Read on…

1.2 BACKGROUND

There's a ton of material out there on digital jitter. This app note assumes familiarity with period jitter, cycle-cyle jitter, and TIE (Time Interval Error). For more on this you could start with:

- https://www.nxp.com/docs/en/application-note/AN4056.pdf (written for a specific NXP part but is a quick read)
- The timing 101 series, https://www.silabs.com/community/blog.entry.html/2017/08/08/timing_101_the_case-7QT5
- https://www.silabs.com/documents/public/white-papers/timing-jitter-tutorial-and-measurement-guide-ebook.pdf (another general intro)
- A dozen app-notes from Tektronix

Those articles are mostly focused on digital serial communications and worried about calculating BER (Bit Error Rate). That concern has zilch to do with the audio quality, though it is of concern when designing high speed digital serial audio interface to preserve data integrity. The standardized methods of measuring jitter do not include a spectrum definition, and the standardized measurements are allowed to be made at random intervals, which does not preserve the time domain aspects of the jitter signal.

This 3 part TI series http://www.ti.com/lit/an/slyt379/slyt379.pdf is more useful as it has high performance ADCs in mind, though in this series case it's looking at high speed "RF" converters and not audio.

[3] A bit of semantics – TechNotes are about development and use of Clockworks products. This is about a general topic, so should be called something else. AppNote sounds techie, so there you have it.

What is of interest for audio is the jitter spectrum as a singular number for period or cycle to cycle RMS jitter or Peak to Peak jitter doesn't tell the whole story. TIE is potentially more useful, though the most useful measure is the spectrum.

If all of the RMS and peak to peak[4] numbers are all low – a few tens and hundred psec at most, respectively - then you probably don't need to worry except for the most exacting applications.

The spectrum is important as human psychoacoustics comes in to play – we're much less sensitive to low frequency jitter than high frequency jitter. Jitter can act in the same way as wow and flutter and produce FM modulation as well as AM modulation, the difference being that most wow and flutter only looked up to 200 Hz[5] and digital jitter has theoretical bandwidth considerations into the MHz range due to the way delta-sigma converters work.

Another complication is that the delta-sigma architecture can reduce the impact of jitter in terms of distortion, but at the same time increase the noise floor of the ADC or DAC beyond the theoretical increase in noise floor from random jitter. Without detailed knowledge of the inner workings of those parts it's very difficult to evaluate the impact of clock jitter.

Determining jitter impact on ADC and DAC parts that have built in PLLs and reduce input clock jitter also make a simple theoretical answer elusive.

Real systems may have multiple PLLs, one at the sending end of the digital link, one at the receiving end (even if clock is carried on the link vs. recovered from the link itself), one in some intermediate processor, and another in the DAC. The interaction of these can produce jitter gain at some frequencies and attenuation at others. Systems with digital PLLs or multiple bandwidth loops act non-linearly and the analysis becomes quite complex.

Going the other way, a properly clocked ASRC right before the DAC makes all of the other jitter sources go away and the system performance is then only defined by the DAC.

In summary, considering these factors:

- Jitter spectrum for parts is never published
- The relationship between clock jitter and ADC/DAC performance for sigma-delta type converters is complex and not documented on the datasheet

[4] Peak to peak TIE can be unbounded, so it may not be helpful.
[5] Analog tape scrape flutter happens in the few kHz range but the IEC standards do not include that. Another complication is that analog tape wow and flutter could itself vary over both long and short time periods, where as digital jitter will most likely be consistent over time as the circuity's operation will not have a lot of dependencies on external factors.

- ADCs and DACs with internal PLLs can modify the jitter seen by the conversion hardware
- Multiple PLLs at the system level make analysis difficult

it becomes somewhere between impossible and freaking impossible for the system designer to know what the system performance (THD+N, DR, SNR) might actually be without building and testing.

1.3 MORE READING

If you have not already, require reading include Julian Dunn's TN-23 Jitter Theory technote published by Audio Precision (seems to have disappeared from AP's website but you can find it).[6] Though focused on AES/EBU (SPDIF) digital interfaces the basics of that paper apply to all of digital audio.

In the case of clocks there is no pattern based jitter so that aspect of TN-23 is not of direct concern unless we're looking at digital data coming from some external source and the clock is being recovered locally from an embedded (data dependent) clock signal.

Dunn's AES 1611 paper (1994)[7] presents an audibility threshold criteria, reproduced in the next section. That paper also covers the FM and AM modulation characteristics of jitter. That figure is from an earlier (1991) paper by Dunn.[8]

The Travis[9] paper provides further understanding of the jitter spectrum as well as introducing the jitter signature, which takes in to account the window of time used for measuring jitter is a consideration.

[6] Someone is keeping the website going (Julian Dunn passed away several years ago) https://www.nanophon.com/ which has several other papers available.

[7] Julian Dunn, *Jitter and Digital Audio Performance Measurements*, http://www.aes.org/e-lib/browse.cfm?elib=6111

[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. http://www.aes.org/e-lib/browse.cfm?elib=5392

[9]Chris Travis and Paul Lesso, *Specifying the Jitter Performance of Audio Components.* 117th AES convention, 2004, http://www.aes.org/tmpFiles/elib/20200506/12950.pdf

1.4 AUDIBILITY

Despite the long history of jitter the impact on audibility is not a totally closed investigation as there are many nuances to consider, starting with deciding if listening material should be digitally jittered and played back on a perfect reproduction system versus creating jitter in hardware and then use the output of various hardware systems so that the unexpected real-world interactions can come in to play.

To be clear we're only concerned with properly executed double blind listening tests and not anecdotal descriptions from snake oil pedaling $3000 USB cable makers, which have universally failed every and all actual testing as having an audible impact.[10]

An earlier attempt is Dunn's figure 1 from the 1994 paper (the only source available is a low quality scan):

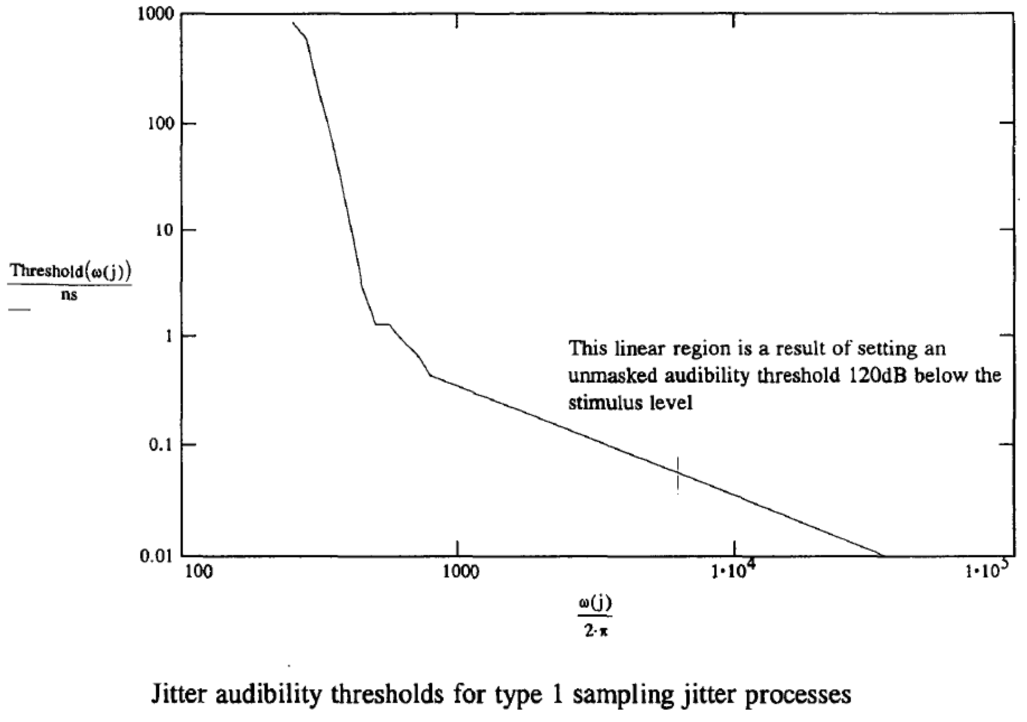


Figure 1 From Dunn's 1994 AES paper

[10] Of course absence of a positive correlation does not prove non-existence. If a test did show correlation they would be $1M richer thanks to the JREF offer, and nobody has walked away with that.

Figure 1 is a summary of a theoretically derived set of criteria in Dunn's 1991 paper, shown in Figure 2.

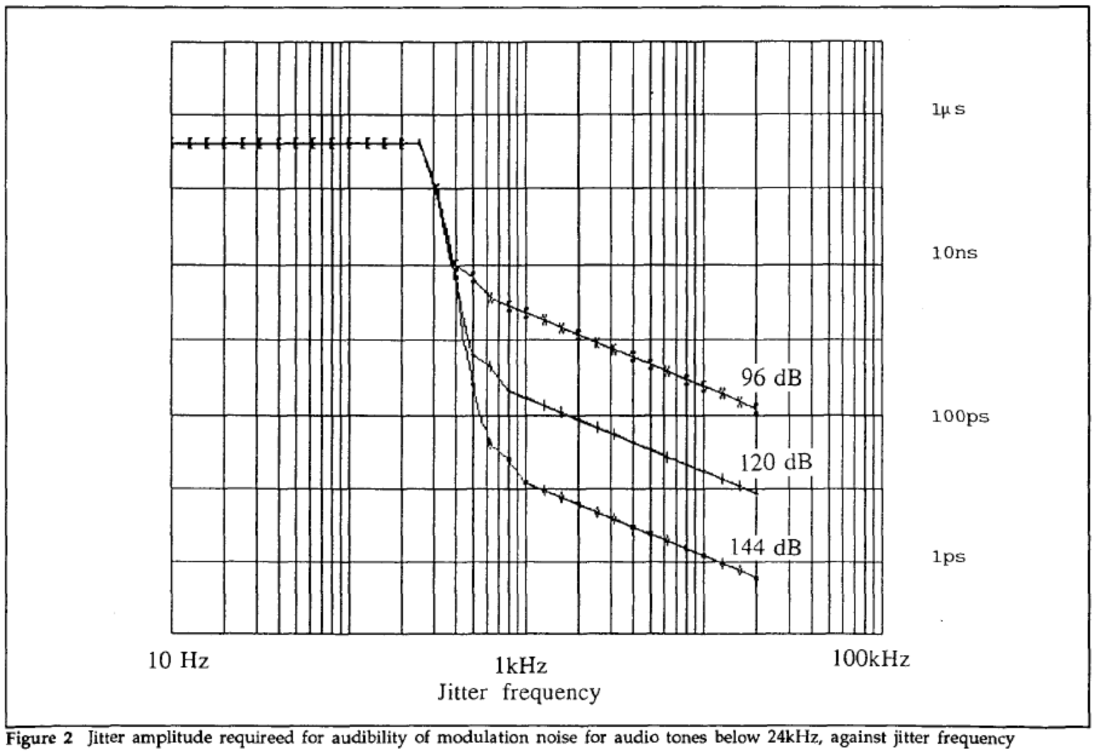


Figure 2 Jitter amplitude requireed for audibility of modulation noise for audio tones below 24kHz, against jitter frequency

Figure 2 From Dunn's 1991 paper, a theoretical set of audibility curves.

Follow-on work (1998) by Benjamin and Gannon[11] suggests a possible more relaxed set of thresholds. As with Dunn they start with a theoretical analysis as shown in Figure 3, though don't turn that in to an audibility criteria. The second part of their paper does include controlled listening experiments. Their conclusion was depending on program content jitter in the range of 30 nsec to 300 nsec RMS would not be audible; they also note that this value is higher than some of the other references they cite.

[11] Lesso, Paul; Travis, Chris, *Specifying the Jitter Performance of Audio Components*, 117th AES convention, 1998, http://www.aes.org/e-lib/browse.cfm?elib=12950

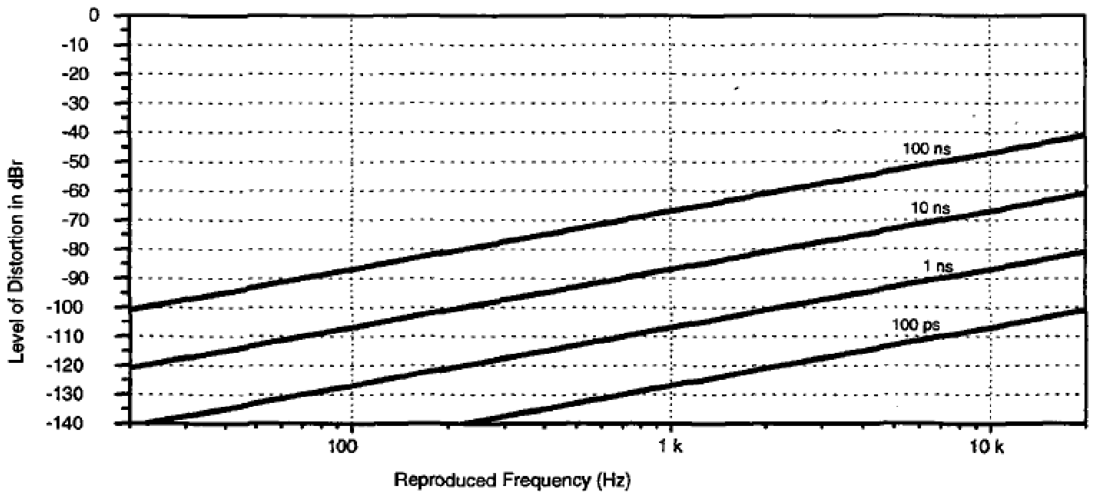


Figure 2: *Simulation of Effect of Jitter on DAC*

Figure 3 Theoretical jitter level and distortion from Benjamin and Gannon (1998).

More recently we have Kaoru Ashihara's 2005 study.[12]

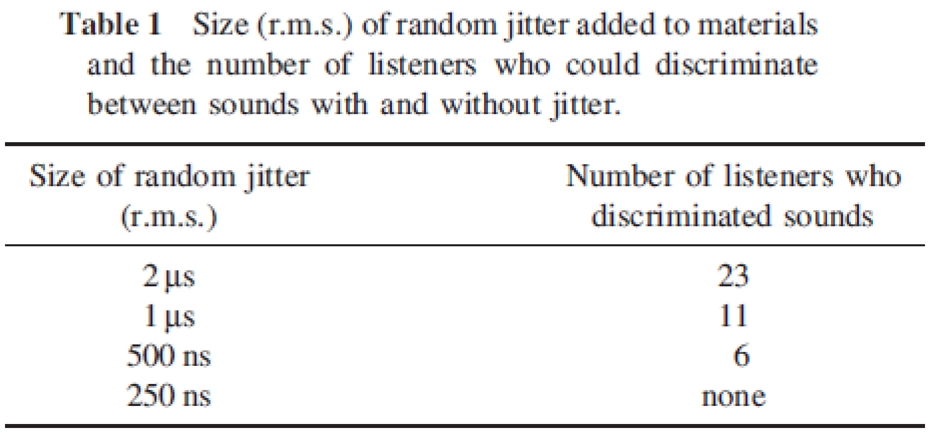


Table 1 Size (r.m.s.) of random jitter added to materials and the number of listeners who could discriminate between sounds with and without jitter.

Size of random jitter (r.m.s.)	Number of listeners who discriminated sounds
2 µs	23
1 µs	11
500 ns	6
250 ns	none

Figure 4 Summary table from Ashihra's 2005 study on jitter audibility.

A possible issue with the Ashihra's study is the jitter source is random, the question of non-spectrally flat jitter's audibility threshold seems to still be an open question.

[12] Kaoru Ashihara, *Detection threshold for distortions due to jitter on digital audio*, Acoustic Science & Technology 26, 2005.

There has been research in how to simulate jitter (to remove the unknowns of actual hardware); start with Hawksford's 2006 paper.[13]

There's a more in-depth literature survey on the Well-tempered Computer[14] website (this is not an endorsement of that site, it's called out here as a convenient list) which includes the papers cited here as well as some non-published materials.

The literature search did not turn up any recent double blind studies, this author's suspicion is that with jitter in all categories reduced to the tens of picosecond range and measurement equipment that can detect jitter induced artifacts that nobody views it as a topic that needs further work because jitter doesn't happen in well designed systems.

1.5 IT'S NOT JUST ABOUT AUDIBILITY

Unlike the early days of high performance audio, it's relatively easy to build products with inaudible levels of noise and distortion. Equipment manufacturers, particularly in the professional audio arena, turned to publishing standardized technical measurements – which in turn create a need for measuring equipment from companies like Audio Precision that could stay a generation ahead of hardware it was testing.

Some of the audio hardware developments were no doubt aided by the work in scientific and industrial applications, which have many areas where more useable resolution or higher sampling rates were needed.

Levels of jitter that we can't hear produce easily measurable distortion products (-60 dBr), and many equipment manufacturers would no doubt be horrified to ship a product with that much known distortion, regardless of audibility.

Jitter's effect of raising the noise floor and therefore reducing headroom, another concern of pro-audio applications, is not something that can be easily evaluated. It's not unreasonable to use other studies on the audibility of wideband spectrally flat (or reasonable flat) noise as proxies for what might be allowed. The degradation of measured performance will be the item of concern for equipment manufacturers.

[13] Hawksford, Malcolm J. *Jitter Simulation in High Resolution Digital Audio*, http://www.aes.org/e-lib/browse.cfm?elib=13698
[14] https://thewelltemperedcomputer.com/KB/BitPerfectJitter.htm (accessed May 2020)

1.6 OUR SUMMARY

For random jitter the case would seem to be made that relatively high levels (compared to what even low end hardware can do) of 30 nsec RMS would not be audible under any sort of imaginable or contrived test scenario. Criteria for jitter at specific frequencies is a tougher as there's no definitive study found; it's a judgement call but the theoretical arguments certainly would appear overly conservative in light of the 30 nsec RMS number. OTOH Dunn's curve from Figure 2 (1991) would be too permissive on the low end.

With high frequencies (say over 10 kHz) the distortion products will generally cluster around the tone, and human hearing is not sensitive to small amounts of high frequency content even before masking effects are considered.[15]

A lot of the early materials were focused on jitter over AES/EBU interfaces and really say more about the techniques for clock recovery ca. 2000 than what is audible when an analog voltage is produced by a DAC (or inverse for ADC).

The fallback criteria then becomes one of measurement: look for tonal components that don't belong and modification of the noise floor, and use the currently established criteria for audibility of those distortion and noise changes.[16] It is also true that aiming for very high performance levels (call that > 120 dB DR and THD+N < -100 dB) does cost more than more middle of the road performance (110 dB DR, 95 dB THD+N range). THD+N as a single number isn't always a good measure; a look at it vs. level and frequency, as well as the THD only view, can provide additional insight for the occasional system that has good numbers but has bad performance aspects.

Combing the results of Benjamin & Gannon with a limit of THD of -110 dBr creates Clockworks' proposed curve of Figure 5. If new references are discovered or mistakes uncovered this curve may change.

[15] It's way more interesting than that, we also can not distinguish tones in that frequency range. Try playing Mary Had a Little Lamb at C9.
[16] Not to imply that debate is settled on those two values, as well as the application area must be considered; studio use has different considerations than home audio playback, etc.

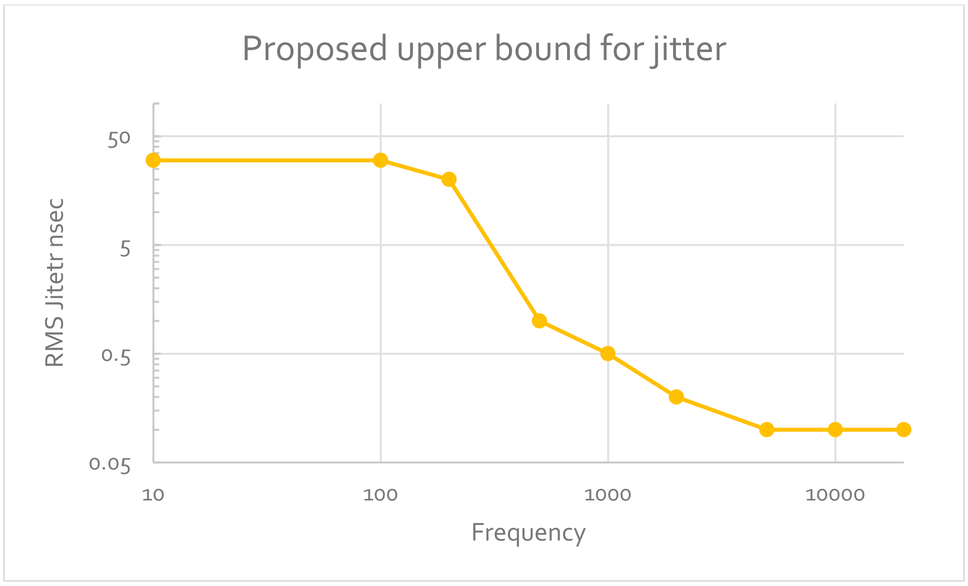


Figure 5 Clockworks jitter limit *du jour* for a performance goal of THD+N < 110 dBr below 10 kHz.

There is of course already the problem of using RMS jitter versus peak to peak, but for the jitter spectrum with sinusoidal components this should not be a problem for evaluation.

1.7 NO PROMISES…

There's a second part of this saga where the techniques described here are used to measure the jitter on the audio clocks created by ADI's AD2428 A^2B transceiver. That's just a data point of one, not enough to say the method here will work in a broader context.

1.8 FILES

Data files with simulated data that were used to validate the results are posted along with this article. The Python program is also provided. These are all provided as-is and you should validate your own setup and processing even if you think this might be right.

One word of caution if you use .wav files. The code expects .csv files, and the most obvious way to convert them via Audacity[17] will not work well because Audacity only exports 6 digits of precision.

A better method is to use Sox (command line audio tool). The command is:

```
sox input.wav output.dat
```

the .dat file has constant width columns and can be imported in to Excel to be saved as a .csv file.

2 CAPTURING THE DATA

This is specific to the scope used to write this app note, which is a Siglent SDS1204X-E. It can record 14M points at 1 GHz, which provides 14 msec of data, which gives some visibility of jitter down to around 250 Hz. It has a specified bandwidth of 200 MHz, though some reviewers report the -3 dB point is closer to 300 MHz. The lower slew rate allows for an increased impact on noise.

Another factor in accuracy is the ENOB, which even with the gain increased to ensure the scope's ADC is run across its full range is probably 6 bits.

The last uncertainty is the scope's timebase stability, which is not specified. Measurement of a crystal oscillator showed some jitter, but low enough that it was not a problem in measuring the A^2B DUT that is the focus of TechNote008 (available on the Clockworks website).

In the case of the Siglent scope the data is saved in a binary file format. Siglent provides a utility to convert that to a .csv file. It's noted if the output .csv already exists the conversion program will give odd results. Excel can not fully open the .csv file as it has too many datapoints.

The format of the .csv file can have header rows (which will be ignored) followed by a column with the sample time (in seconds) and then a column of values (in volts, though the units don't make a difference). A second value column (for the trigger channel) is ignored.

[17] Audacity was also used to create the test files.

2.1.1 REDUCING THE SAMPLE RATE

Longer record lengths would be nice to see detail in the lower frequency jitter components. Obviously a more feature rich (expensive) scope would be the best way since you could also go for wider bandwidth (probably 1 GHz would be a sane starting point) and higher sample rate (thinking > 4 GHz). Reducing the sample rate is a possibility but only if the jitter values were big enough to offset the decreased resolution.

In different times access to the better scope would have been arranged so there doesn't seem to be a point to work in the opposing direction to find the cheapest scope.

3 PROCESSING THE DATA

The program `JitterAnalysis.py` is designed to be run inside of an IDE as what would normally be command line options are just variables in the code.

The general purpose of the program is to find the time at which the threshold is crossed by interpolating between the available data points on either side of the threshold. Since this should be the mid point simple linear interpolation provides a good curve fit.

The processing looks for events (i.e. clock edge) and calculates the timing.

3.1 PROCESSED OUTPUT

The output is a csv file with 6 columns, as shown in the example.

	Time	Value	TimeAtThreshold	DeltaT	SubExpT
92	0.001	1.87E-05	0.000999994		
188	0.002	3.68E-05	0.001999988	0.001	-5.76568E-09
284	0.003	5.36E-05	0.002999983	0.001	-5.35156E-09
380	0.004	6.85E-05	0.003999978	0.001	-4.74633E-09
476	0.005	8.09E-05	0.004999974	0.001	-3.95003E-09
572	0.006	9.05E-05	0.005999971	0.001	-3.05805E-09

Figure 6 csv example output for a 1 kHz sine wave sampled at 96 kHz

The row represents the sample immediately after the trigger condition that defines a clock edge was met. The trigger condition is rising or falling edge, and threshold. The columns are:

- Col1 – sample number
- Time – the sample time as copied from the input data
- Value – the signal value as copied from the input data for the time in column 2
- TimeAtThreshold – the linearly interpolated time when the signal crossed the threshold.
- DeltaT – Current crossing time minus the last crossing time. This can be used to calculate cycle to cycle jitter.
- SubExpT – The delta T minus the expected time based on the expected frequency of the clock. This is the period jitter.

4 ANALYSIS OF THE DATA

The analysis of the data can be performed in a number of different ways. Excel is being used as it provides enough capability and is familiar to many people, though it is a bit limited as its FFT and Histogram capability are awkward to use.

4.1 CREATE A .XLSX FILE TO USE AS A TEMPLATE

While an actual template file and a bunch of scripts could be created to automate this, it's easy enough to do a SaveAs… and then copy and paste data in.

Once you have a spreadsheet with the results displayed the way you like, just make copies of it. See the example file `30hz.0001_out.xlsx`.

4.2 COPY THE DATA IN TO YOUR DUPLICATE(D) FILE

Open the .csv file produced by `JitterAnalysis.py` and copy the first 6 columns. Select your copied master file and select cell `A1` and then `Paste->values`.

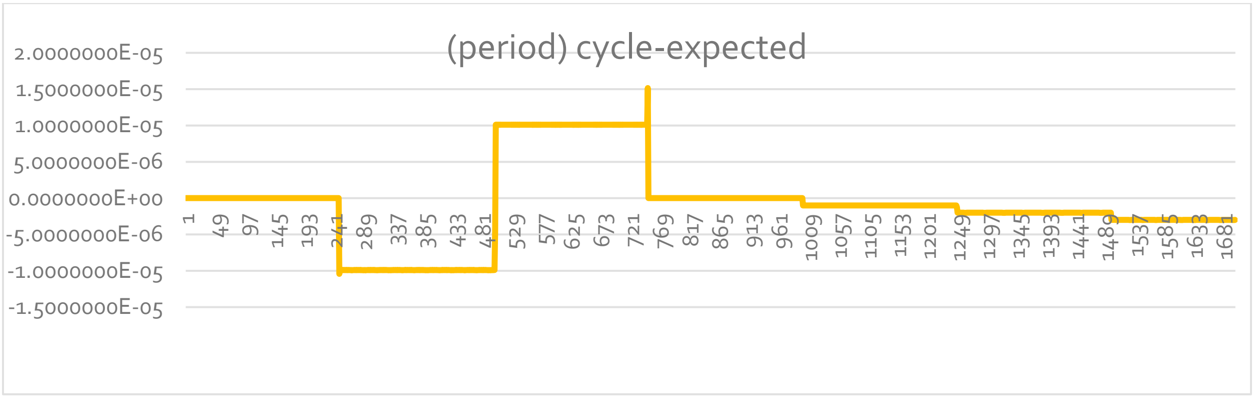
This will leave the titles and formatting changes alone.

If the file already has all of the equations you'll see the results immediately in any plots, with the exception of the FFT and Histogram, which must be re-run each time.

4.3 PLOTTING BASIC TIMING INFORMATION

The cycle to cycle period change is probably of more interest than the period time that appears in the `JitterAnalysis.py` output (column 4, Delta T). In the example this is column `I`, and is calculated by subtracting the current period length from the prior period length.

It can be confusing when looking at the plots of SubExpT (period jitter) and the Cycle to cycle jitter as they seem like they should be the same. Cycle to cycle only shows the change in period, it does not show the error. For example if the jitter was a square wave where the clock alternated between 990 Hz and 1010 Hz the cycle to cycle plot would only show a value when the clock frequency changed and would be zero everywhere else. The period jitter would clearly show the square wave shape to the changing frequency and be easier to interpret than a plot of cycle to cycle jitter.



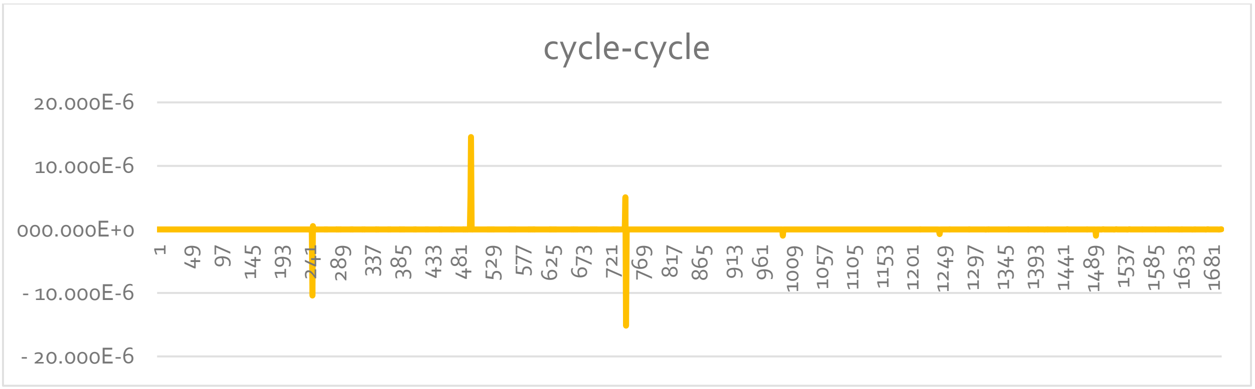


Figure 7 Period and cycle jitter versus time from simulated data of a clock that changes frequencies periodically. The small spike after 721 is a mistake in the simulated data. This illustrates how the period jitter can be more useful than just the cycle to cycle change.

The synthesized data for this test case consisted of 500msec each of the following frequencies: 1000Hz, 1010Hz, 990Hz, 1000Hz, 1001Hz, 1002Hz, and 1003Hz.

Unlike the JEDEC standard for jitter calculations, the methodology here requires consecutive values and we do not take the absolute value of the differences.

The period jitter plot from the example in Figure 7 is helpful to understand what the clock is doing, but isn't so clear on the impact of using that clock on a sampled data system. The Time Interval Error, Figure 8, which uses the computed edge time of an ideal clock compared to the measured clock, gives that insight. From the TIE plot one could derive the other plot's shapes by taking the derivative(s).

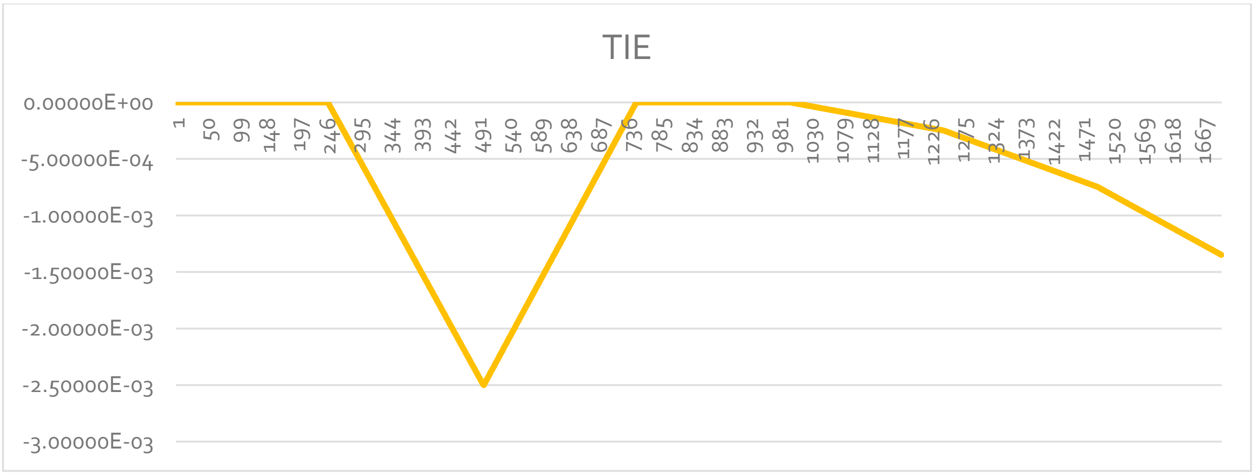


Figure 8 TIE plot for the example data from Figure 7

Mathematically, each datapoint is calculated as:

$$TIE_n = TimeOfCurrentEdge - EdgeNumber * ReferencePeriod + TimeOfFirstEdge$$

Reference period should be calculated based on the total number of edges and the measured time for them, as actual clocks will never be quite the theoretical value and even a small error in each period value can produce a large enough timing error to mask the actual jitter pattern.

If the clock frequency is running faster than expected (i.e. phase is advancing) then the edge occurs earlier in time than expected, and the TIE value is negative. If the clock is lower in frequency then the opposite occurs. The TimeOfFirstEdge accounts for the fact that the scope may center trigger so the first edge will happen at a non-zero time.

5 ADVANCED ANALYSIS

With the three types of timing information (period, cycle-cycle, TIE) the statistics of the jitter can be calculated. Jitter is theoretically unbounded but across the sample sizes used here meaningful RMS and peak to peak values can be calculated, along with a histogram to look for non-gaussian distributions, which would serve to indicate the jitter is not random (for more on that topic see the suggested readings).

A histogram does not help in determining what the underlying jitter cause may be. For that the TIE (jitter) spectrum is needed. It's obtained by taking the DFT of the TIE data. As with any other use of the DFT in analysis, care must be taken to properly interpret the results. There are two coordinates; frequency and magnitude. The frequency (DFT bin size) is determined like with any other DFT, the frequency of the clock (not the scope's sample rate!) defines the time between samples[18]

If using Excel it can only support use of the FFT and therefor the datasets must be a power of 2 in size. Zero padding them if not a full power of 2 in length will reduce the total signal energy and the results will need be scaled.

Excel does not support windowing.[19] A default rectangular window has high side lobes and that may mask lower amplitude spectral components. If you do window the data then the magnitude must be scaled to correct for that if you don't include it in the coefficient calculation.[20] Likewise unless you use a flat-top window you need to curve fit to find the magnitude of the actual peak. To find the actual exact frequency would also need a curve fit, but eyeballing it is probably close enough for general use.

5.1 A SECOND EXAMPLE

In this example a 1 kHz sine wave (representing the clock) has been jittered by a 30 Hz sinewave (the jitterer) by adding the two waveforms together. This does not change the period the way clock jitter would, but we can treat it in a similar fashion to validate the calculations.

[18] The actual edges are not evenly spaced in time – that's what jitter is. However TIE uses an ideal clock as the reference so for the purposes of analysis we can say the samples are all evenly spaced.
[19] This NI app note is a good review and includes the correction factors https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf
[20] Most places that provide window function examples don't include the correction, but read the fine print.

For a sinewave with the threshold set to zero, the small angle theorem applies and $x \approx \sin(x)$. Shifting the signal level a small amount is equivalent to shifting it in time at the threshold point. This trick wouldn't work if we used a small square wave as the jitterer as we're not changing the frequency of the 1 kHz clock, just the phase by a small and *different* amount at each edge.

The 30 Hz jitter value in this example has an amplitude of .0001. At its greatest rate of change (i.e. when the added 30 Hz sine is going through zero) this changes the period of the 1 kHz clock by 5.97 nsec. It's worth calling attention to this:

> *It is illustrative that the interferer here is 74 dB in level from the clock signal yet creates jitter that would be embarrassing to most designers. Designing systems like this with a* "it's just 1's and 0's" *outlook will lead to trouble. It's not difficult to avoid these problems, but it's also easy to unintentionally create them and testing is paramount at all stages of development.*

Here's the period jitter that is calculated from that waveform. The 30 Hz is added starting at t=0, so the greatest rate of change of the period occurs at t=0. The positive going 30 Hz value acts to shorten the period of the 1 kHz simulated clock. We see in Figure 9 is the period jitter value starts at the negative peak value, matching our expectations.

The magnitude of the period jitter is a function of both the frequency and magnitude of the clock in this simulated data.

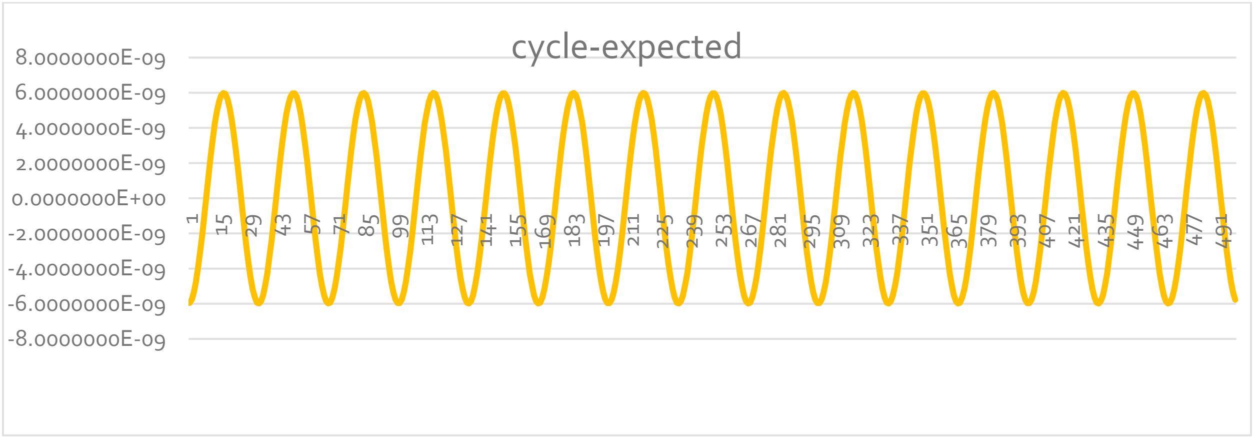


Figure 9 period jitter from adding 30 Hz sine to simulated clock to create a changing phase shift at the threshold (which is zero)

Figure 10 shows the cycle to cycle jitter, i.e. the current period minus the previous period. We can see that each period is getting longer, which makes sense since Figure 9 shows us that the relative to the expected period, each period is shorter by a smaller amount relative to the expected value.

It’s helpful to contrast these two plots with their equivalents for the example illustrated by Figure 7

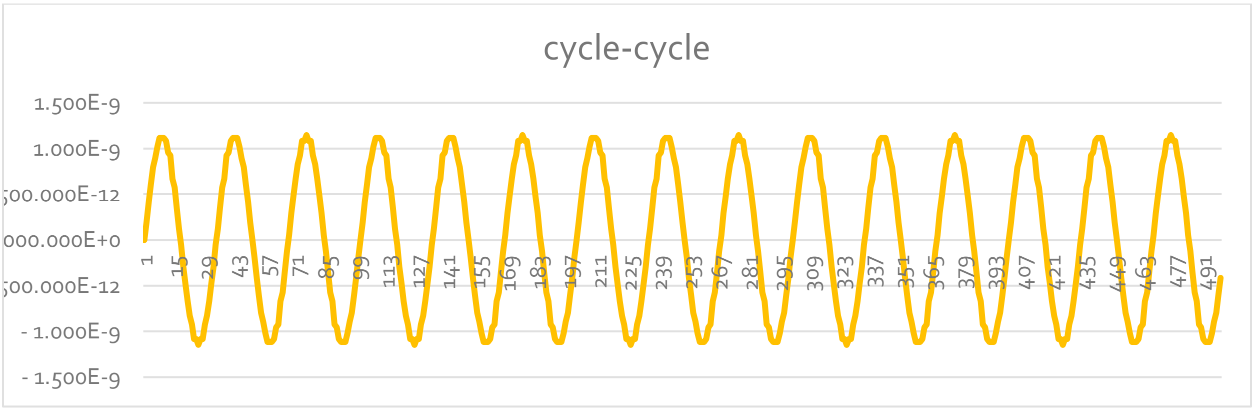


Figure 10 cycle to cycle jitter for this example.

The TIE plot of Figure 11 is perhaps the most difficult to make sense of. It represents the cumulative effect of all of the timing errors that came before it. Looking at the period plot of Figure 9 we can see that it started with a large negative time shift (because the shift is biggest at the zero crossing of the 30 Hz we used to simulate jitter). 1/2 a cycle later the total negative plus positive shift has balanced out so the TIE is now zero (near edge 15 in the plot). Now to return to the same value in Figure 9 it will take a full cycle, leading to the larger positive value than negative value.

If the added jitter started at a different phase of the 30 Hz then the shift would be different. The key point is the shape and peak to peak values of the TIE are what’s important, not where the zero value happens to be.

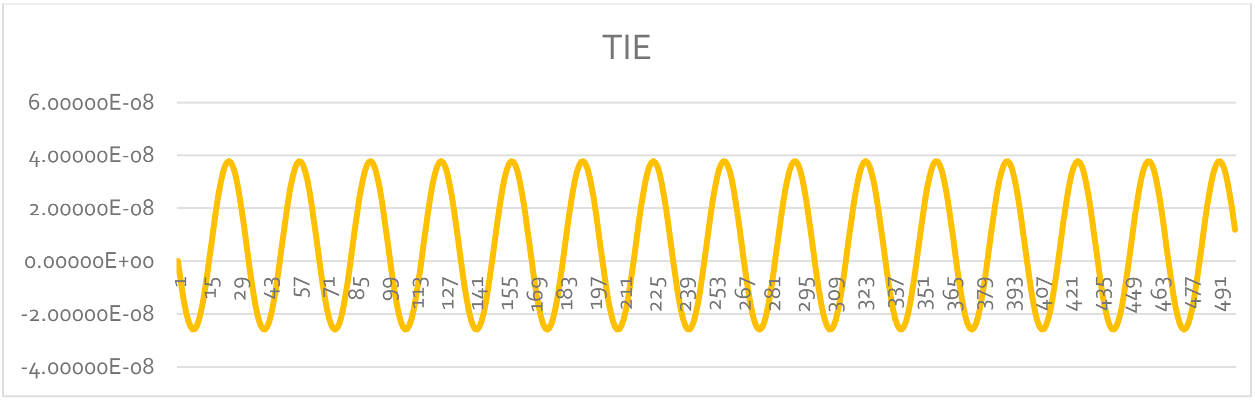


Figure 11 The TIE plot, which is the difference between the actual time and the expected time, is offset in this example.

The value of the TIE plot is it shows the cumulative affect of the jitter; in any system with only random jitter it has to stay bounded. In a real system the clock can drift and that could lead to unbounded values for TIE. Across the measurement windows used here that should not be a problem provided the average frequency is accurately calculated.

The peak to peak value for the TIE is 62.8 nsec or an amplitude of 31.4 nsec. This value may seem counter-intuitive to the approximate 1 nsec maximum of the cycle-cyle jitter of Figure 10, but is a great example of why the cycle to cycle jitter and even the period jitter (Figure 9) are of limited value in audio applications. In this example the sample clock is 96 kHz, and most people would probably say that a sample clock that gets off by 30 nanoseconds isn't a very good one, even though at 1 nsec the sample to sample error is small.

This example used sample rates and simulated clock frequencies that are low to make the example a little more intuitive. Real audio master clocks are usually 12.228 or 24.576 MHz, but we can scale the example numbers easily enough.

The TIE data is what enables calculation of the jitter spectrum. Using EXCELs FFT and plotting we get Figure 12.

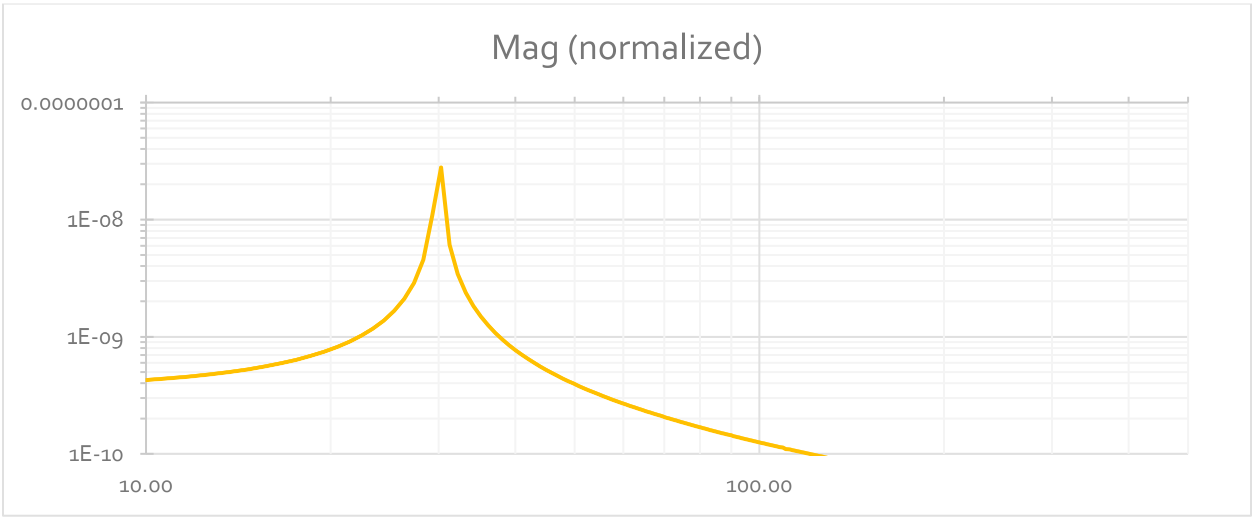


Figure 12 Plot of magnitude of the FFT of TIE (1024 points) showing the 30 Hz jitter component.

The 30Hz jitter is not an even multiple of the 1024 edge periods, and EXCEL lacks built in window functions, so the rectangular window results in the high level of skirts seen. It also means the energy is spread across a few bins, as shown in the table.[21]

FFT Frequency bin	Value
28.32	4.55E-09
29.30	10.86E-09
30.27	27.91E-09
31.25	6.11E-09
32.23	3.43E-09

Doing a crude estimate (since we know there's no noise) the peak's amplitude can be estimated as 31.5 nsec, and from the TIE plot (Figure 11) the amplitude of the timing error from jitter is 31.4 nsec, so we can consider this close enough and dispense with a more complex and exacting analysis.

6 REAL WORLD TESTS

Next we see if the scope and software can actually capture data in a way that provides useful insight. Three tests will be performed, the first of an Avermetrix Averlab Audio Analyzer's word clock output, which has RMS period jitter specified (100Hz – 40 kHz) of < 20 psec. The next test

[21] See for example https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/4293/file/exact_fft_measurements.pdf and https://dspguru.com/dsp/howtos/how-to-interpolate-fft-peak/

is of a well documented ovenized crystal oscillator, and the last for a SPDIF receiver for a knock-off[22] made in China decoder.

The DSO being used[23] is a Siglent 1204X-E with 1 GSample/sec and 200 MHz bandwidth. The lower sample rate limits the time resolution, and the bandwidth limits risetime which increases susceptibility to noise. This model is also not specified for timebase stability; the general idea is to capture long runs of data to create the TIE data that can then be spectrally analyzed and that requires a stable timebase.

6.1 USING EXCEL FOR THE ANALYSIS

The prior section just looked at the period, TIE, and cycle to cycle jitter. In looking at real devices it will be useful to look at the jitter statistics (histogram, RMS values, and max/PP) and the spectrum derived from the TIE data. The three spreadsheets created for the analysis in this section are included in the files as they provide an example that could be followed in your own analysis.

For the spectrum, a Hann window was added to remove the high sidelobes of a rectangular window, this also necessitates scaling the magnitude to accommodate the scalloping loss. However this is still not exact as frequency components centered in a bin will have higher amplitude than a component between two bins. For the purposes of the analysis here this potential factor of 2 error is ignored.

Remember that the Histogram and FFT features of Excel do not auto-update, they must be rerun each time the input data is changed.

6.2 MEASURING A KNOWN REFERENCE – WORD CLOCK

The Averlab is designed to make precision analog measurements and therefor needs an accurate time base to avoid all of the same problems that were discussed in the first part of this app note. It offers a word clock output and 96 kHz was selected to get around 1000 edges captured.

[22] The chip has a package that matches a Cirrus part typical of these type of boxes, but the chip has no identification on it. The design is 15+ years old and easy to copy compared to today's parts.
[23] Post COVID-19 the plan is to do this again on a mid range scope – 1 GHz bandwidth and 5GSample/sec type capability.

The first thing examined is the edge 5 msec after the trigger with persistence turned on. Changes in the edge position could indicate the source is jittery, the scope's timebase is jittery, or the scope's trigger isn't exact due to noise or other factors. Figure 13 shows about 3 nsec of variation, which is bigger than we might expect the peak to peak jitter for 20 psec RMS. Depending on your assumptions, peak to peak jitter would by 6x to 14x the RMS, or 120 psec to 280 psec.

Based on this we have to assume that either the scope's timebase isn't stable enough or there is trigger noise affecting the measurement.

Since the subsequent measurements are all done from the same trigger we don't need to be concerned with trigger noise.[24]

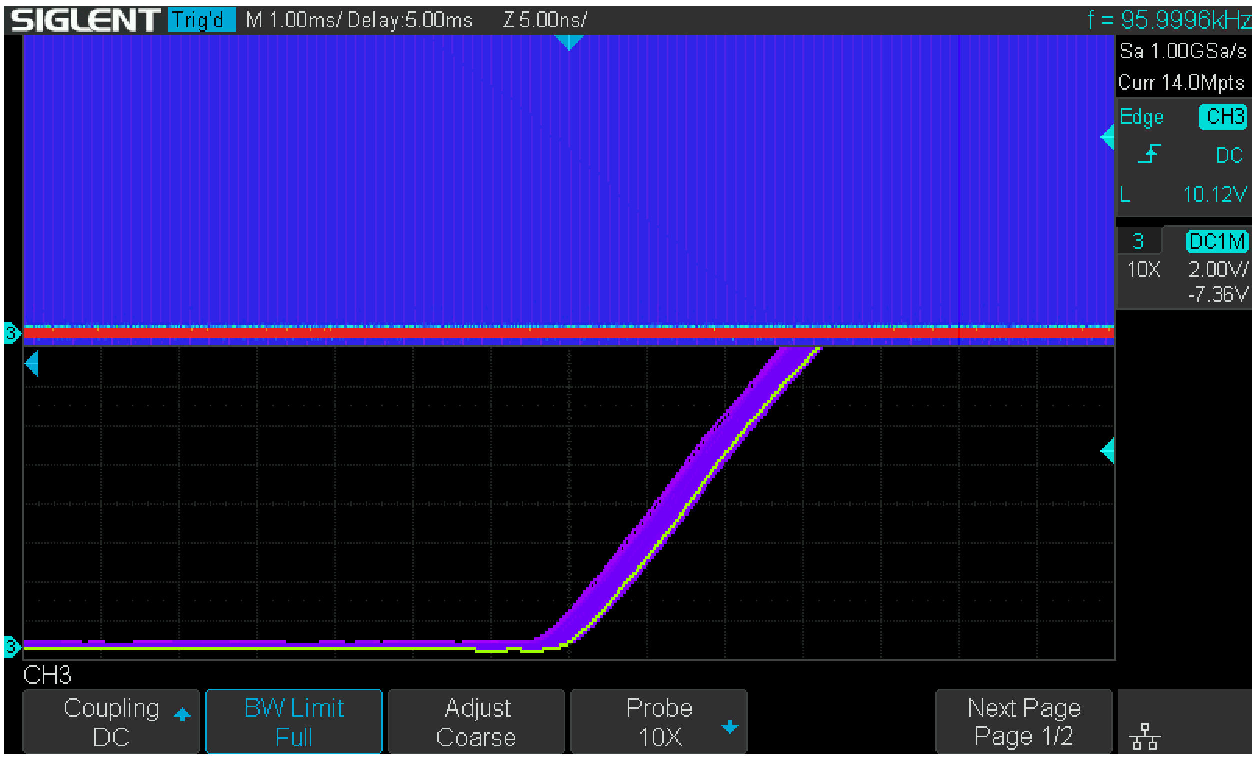


Figure 13 Averlab wordsync measured 5msec after trigger. Edge to edge jitter is about 4nsec.

There is one other analysis that can be done. The scope can perform a 1M point FFT; if there was periodic jitter then this acts like modulation and the FFT will show sidebands. Figure 14 shows that there's no obvious modulation components, so we'll expect to see that the jitter spectrum won't show strong spectral components either.

[24] Of course it would be nice to just look at an edge like in Figure 13 and eyeball the peak-peak and use that to get a rough RMS estimate

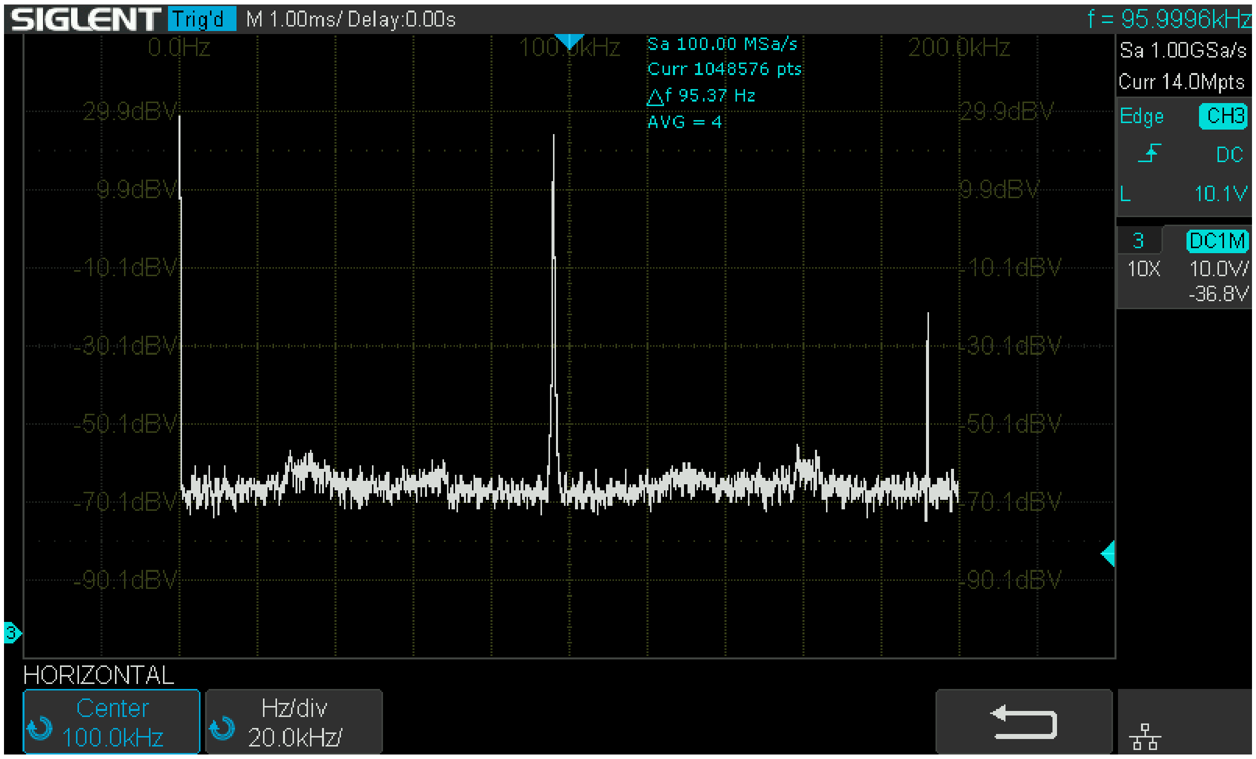


Figure 14 Spectrum of Averlab word clock does not show any tones beyond the 96 kHz and its harmonics (input is DC coupled)

Using the previously described procedure 14M points are captured, converted to a CSV, and then edge detected with the Python program, and then copied in to an Excel file.[25] Figure 15 shows the results of the three jitter period measurements plotted over time.

RMS values are computed as:

- Period 177 psec
- TIE 120 psec
- Cycle-cycle 200 psec

All three are much bigger than the specified number for the Averlab (we've assumed its published < 20 psec RMS period jitter is correct) so that would imply that the scopes timebase, resolution, and noise floor are what we're seeing.

[25] See `aver_FS_analysis.xlsx`

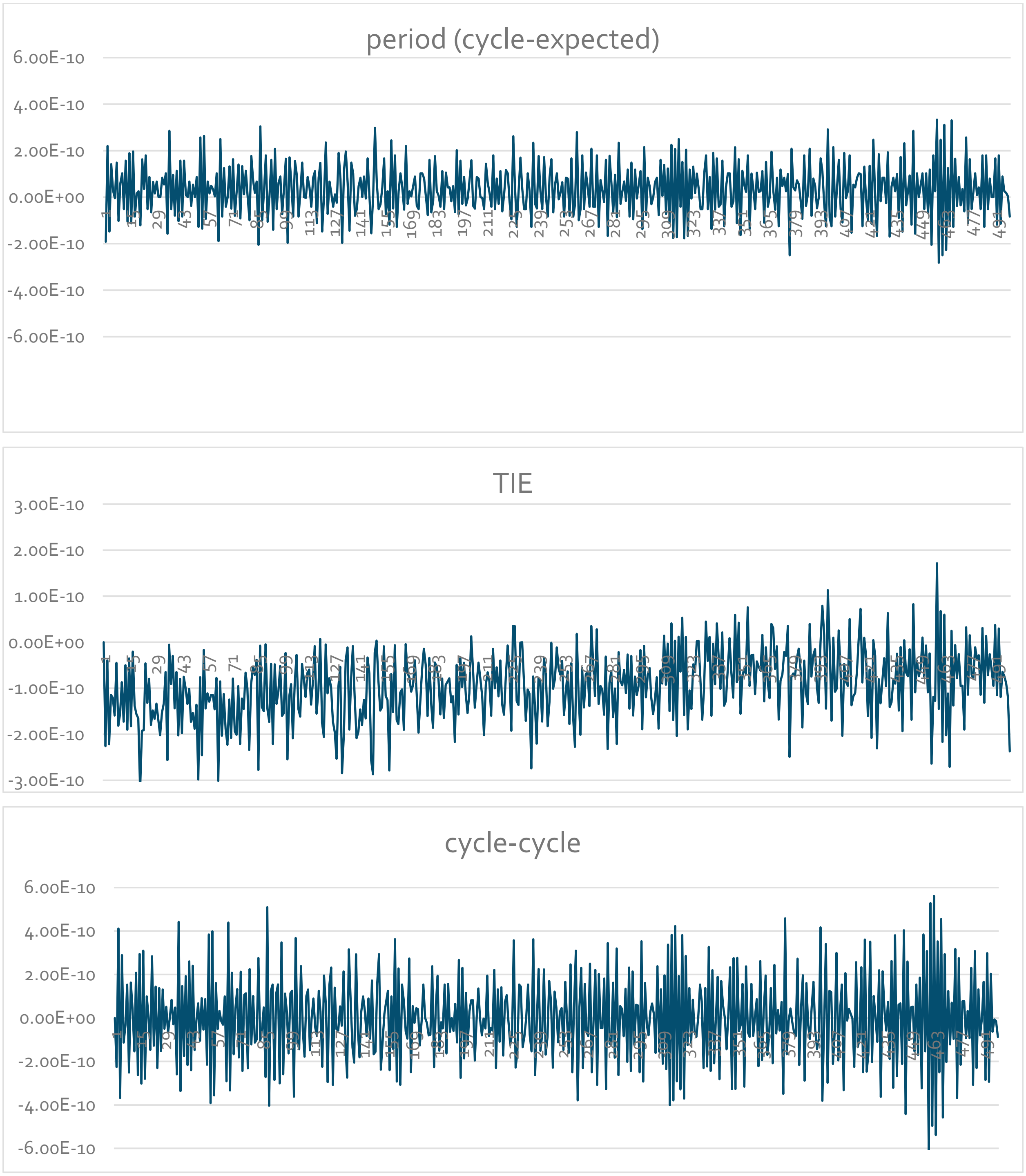


Figure 15 Period, TIE, and cycle-cycle jitter values for the Averlab word clock for the first 500 edges. Vertical axis is in seconds. (TIE plot is on a different vertical scale and does show part of a longer term trend)

These RMS numbers are within the range of normal jitter in most audio systems. If values similar to the above were measured and the jitter was spectrally flat there would be reasonable confidence that clock jitter is not a problem in the system.

To make an actual statement about the jitter would want performance about 100 times better, i.e. a system residual of a few psec so that a system with 10 or 20 psec of jitter could be measured.

Of interest is the peak to peak range of the jitter, which was determined here by looking across the data for the min and max. With only about 1200 points in the dataset this is not a totally representative determination:

- Period 615 psec
- TIE 492 psec
- Cycle-cycle 1200 psec

Again this would be the measurement floor of the DSO as the expected peak to peak is below these values by at least a factor of 2. The TIE value is not necessarily helpful as long term drift or a slight error in computed period can create a small bias that grows over the time window.

We can also look at the histogram of the cycle to cycle jitter, though again we do not have a large number of data points.

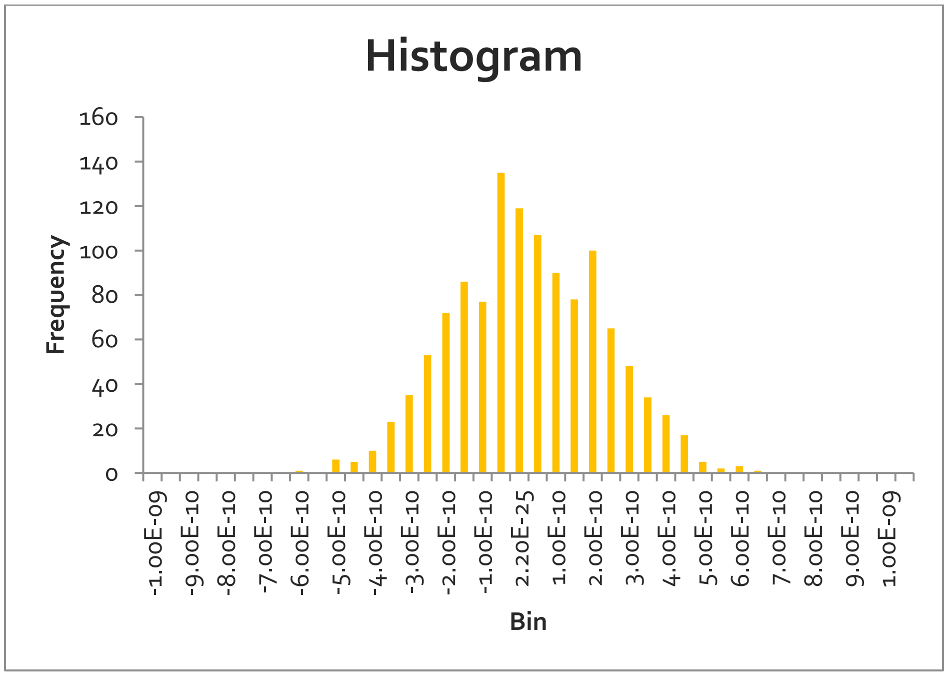


Figure 16 Histogram of the cycle to cycle jitter is limited by the 1200 data points, but doesn't appear to show an artifacts that suggest a non-gaussian distribution

The last item to examine is the spectrum (Figure 17). No obvious spectral components are present; a better analysis would capture 16 or more of the spectra and average them together to provide a better view. Given we're up against the DSOs limits that would not be expected to be particularly helpful here. The time needed to run through the steps to get to the spectrum is not conducive to gathering multiple runs. A more capable instrument would be the correct approach.

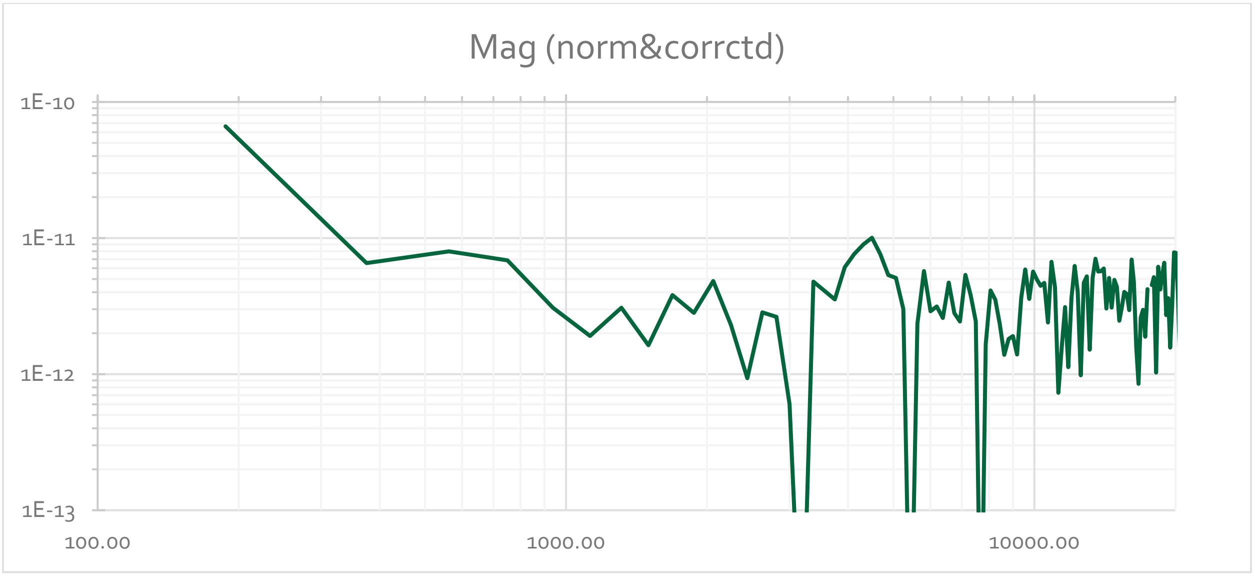


Figure 17 Spectrum of jitter (from TIE) for Averlab's word clock output. 512 point FFT, Hann window.

In summary, measurement of the Averlab did not firmly establish the test setup as we had to make an assumption about the unit's performance. The next test case will show the values come from the scope's timebase and smaller measurement windows improve the accuracy.

6.3 MEASURING A KNOWN REFERENCE – 10 MHZ

For this test a Raltron OCXO OX4120A-D3-5 10.0000 MHz oscillator was used. From its phase noise data for a bandwidth of 1 Hz to 100 kHz RMS jitter is calculated as 1.26 psec for period and 2.2 psec for cycle to cycle. Looking 5 msec after the trigger (Figure 19) we see about the same 3 nsec edge width as with the prior measurement of the AverLab unit. This would seem to confirm our assumptions about the inherent capability of the Siglent SDS-1204X-E scope.

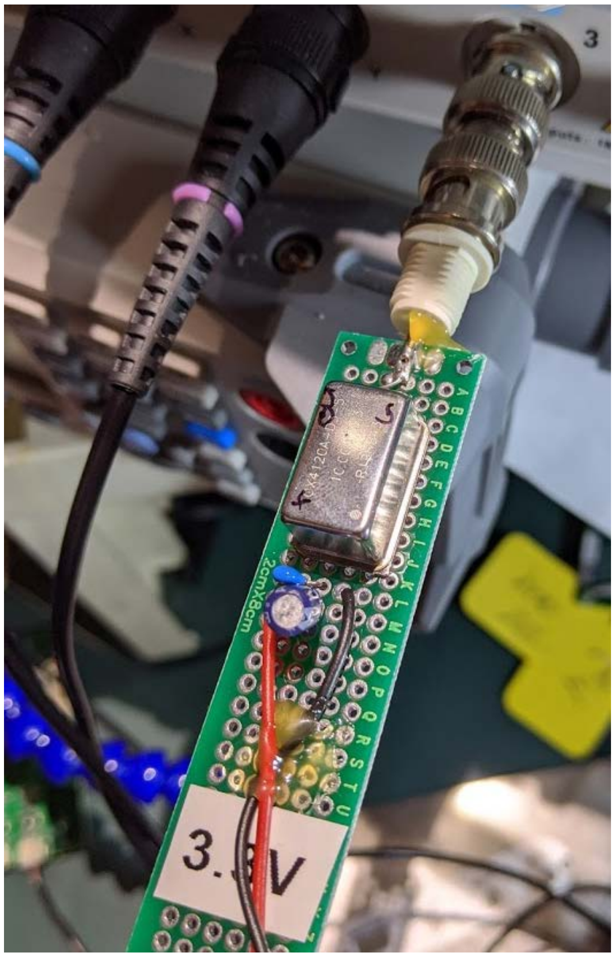


Figure 18 OCXO setup for measurement.

Different trigger edge, thresholds, and gains were used, all produced about the same results. An indication the scope's timebase may be the cause can be seen when looking at a delay of only 1 msec; if the jitter was caused by trigger noise we would expect the same 3 nsec width, but instead we see only about 300 psec of width displayed.

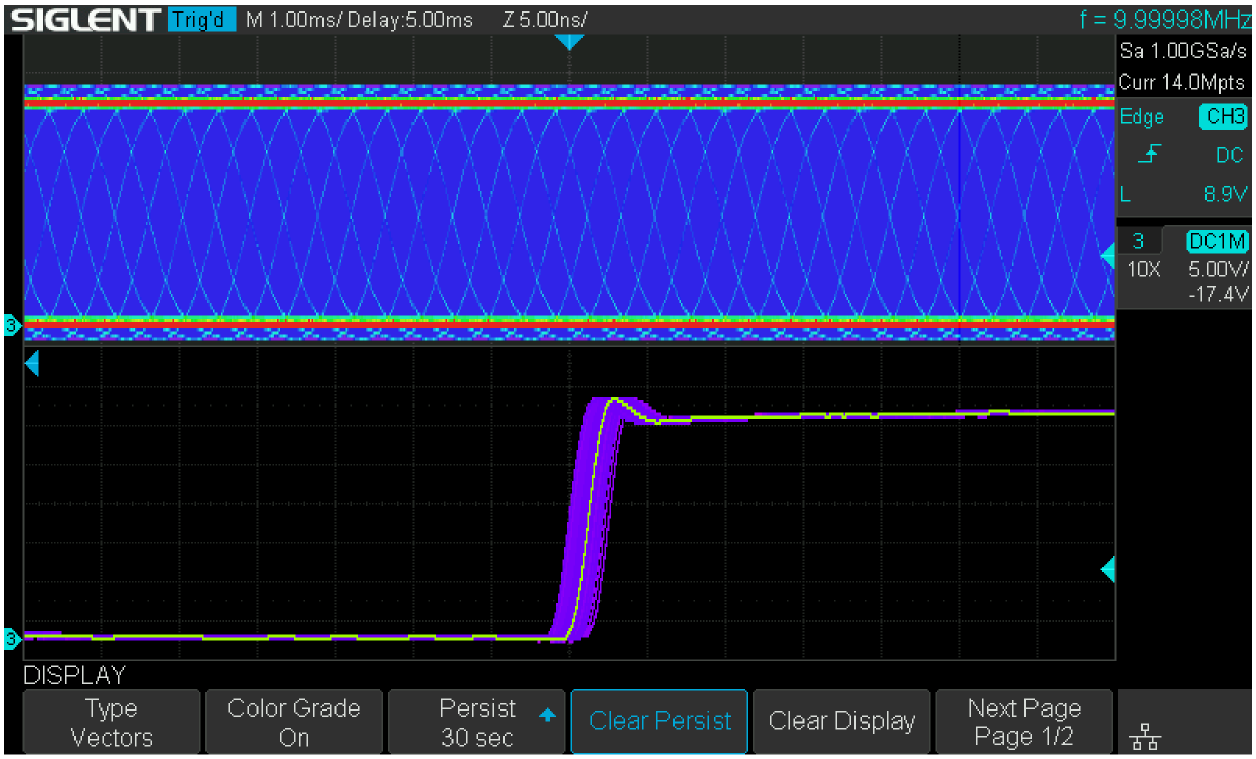


Figure 19 10 MHz OCXO edge 5 msec after trigger

For higher frequency signals we could measure over a small period as enough edges would be captured. We would lose the lower frequency resolution from the jitter spectrum, but that is less of a concern. After some experimentation the best results were obtained using the falling edge, enabling the scope's 20 MHz bandwidth, and setting the input gain 2x higher than normal.

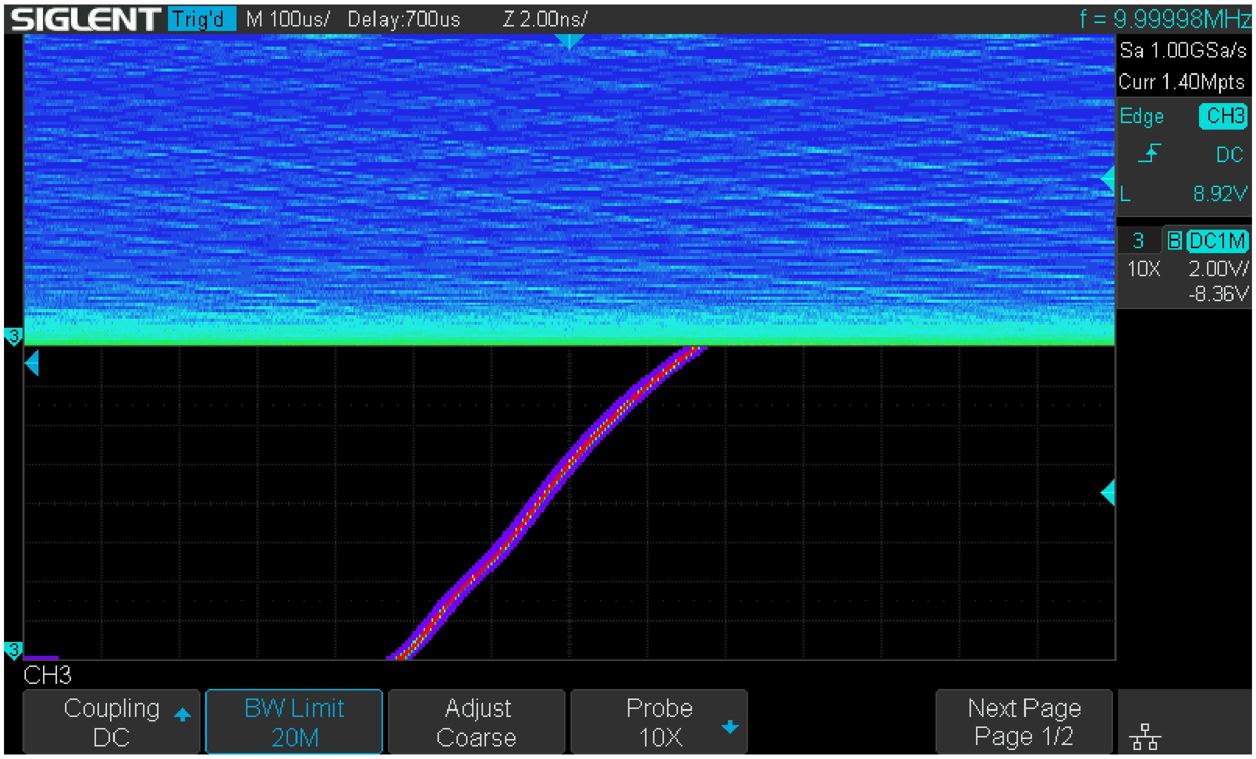


Figure 20 10 MHz OCXO edge 1 msec after trigger, the edge spread is about 1/10th the value at 5 msec (note scale change)

By restricting the range of data processed to +/- 500 usec of the trigger it would appear we can get better results, and this is backed up by the statistics:

- Period 12 psec
- TIE 200 psec
- Cycle-cycle 7 psec

With peak to peak across that window (10,000 edges) of

- Period 62 psec
- TIE 321 psec
- Cycle-cycle 111 psec

The larger TIE value seems to be explained in part by a slow frequency variation, see Figure 21. Again we can not be sure if this is the scope's timebase or the OXCO causing this. It does illustrate why a peak to peak TIE value over a long period may not be helpful in understanding jitter. OTOH the plot shows data that the cycle to cycle and period plots don't reveal.

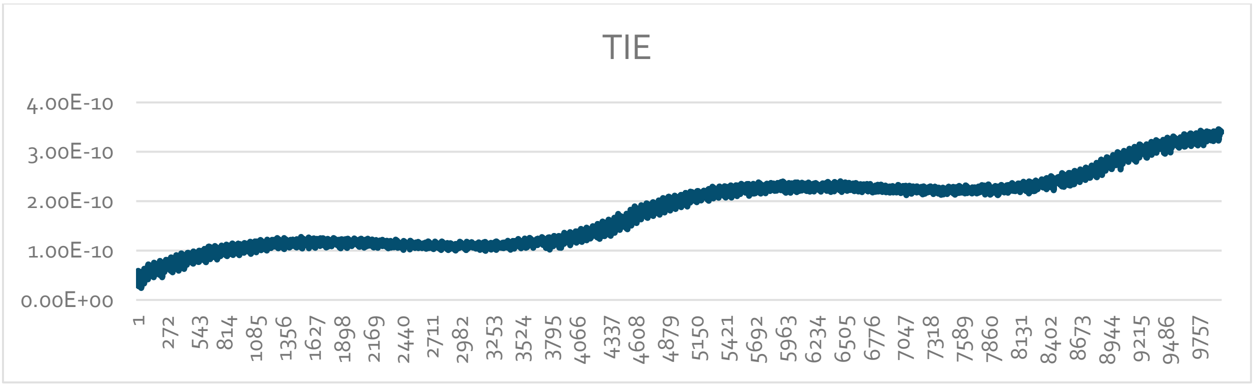


Figure 21 10 MHz OCXO over 1 msec output TIE.

The jitter plots do reveal a new issue with the current setup. Looking at Figure 22 we notice the discrete steps in the data, about 10 or 15 psec in size, but not quite constant. This is happening because the oscillator frequency and scope sample period are related by a factor of 100, along with the limited vertical step size and the way the interpolation is performed to estimate when the signal crossed the threshold. You could think of the combined affect like a quantization error, but as things drift slightly it changes the measurement.

It's a sign that we're really scraping up against the limits of what we can do with the bandwidth and scope sample rate limits.

For the TIE data it means the spectrum is dominated by this quantization noise and we can't learn anything about the spectrum of the jitter. The spectrum of the actual clock signal is clean, so we wouldn't have expected to see spikes in the jitter spectrum.

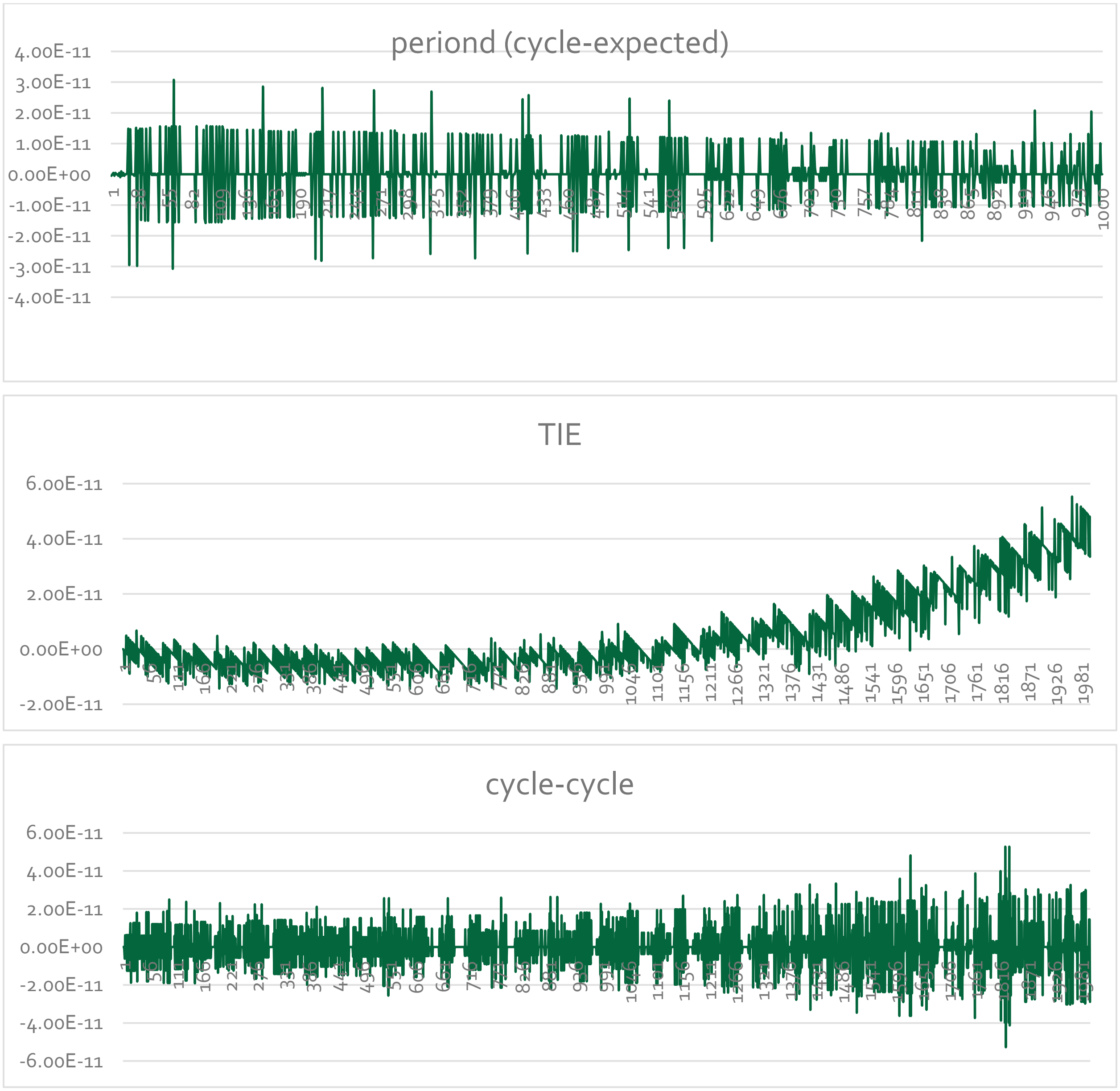


Figure 22 10 MHz OCXO jitter plots reveal the limits of the measurement setup

6.3.1 SUMMARY OF ANALYSIS OF THE REFERENCE SOURCES

To look at jitter spectral components down to around 500 Hz the full 14 M sample data set of the scope is needed, which implies the longer time window and means the system noise floor will be about 200 psec. We can detect a device with RMS TIE at this level as the two jitters will add.

For higher jitter frequencies (a few kHz) we can use only 1M samples and reach jitter down to around 15 psec.

While it would be great to have sensitivity across the whole audio bandwidth, these limits do happen to correspond with the audibility criteria (Figure 5) that allows for higher jitter at lower jitter frequencies.

6.4 SPDIF RECEIVER WORD CLOCK TEST

One of these:

HD AUDIO RUSH
DIGITAL SURROUND SOUND DECODER
DOLBY DIGITAL PRO LOGIC II
dts Digital Entertainment
RAYMAX
DC/5V
Input
2.1/5.1
1 — Spdif — 2
Power
D1
D2
D3
Aux

Figure 23 Zepthus 5.1 decoder

was pulled apart so that the recovered word clock and bit clock (next section) could be probed. SPDIF was used as it's more likely to create jitter than ToS. A CD (you know, the shiny disc things?) was played as the source for the SPDIF.

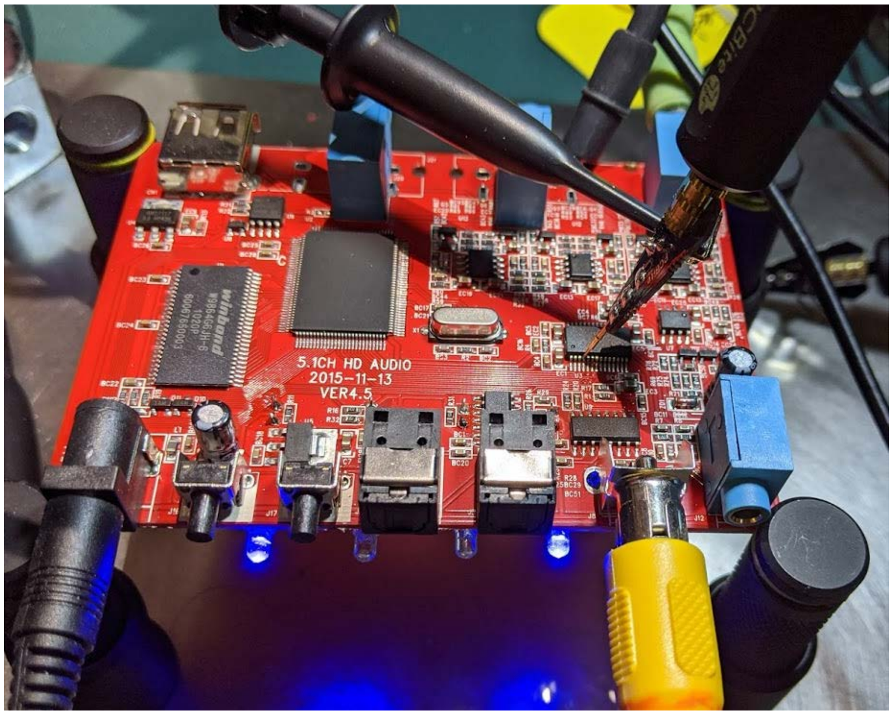
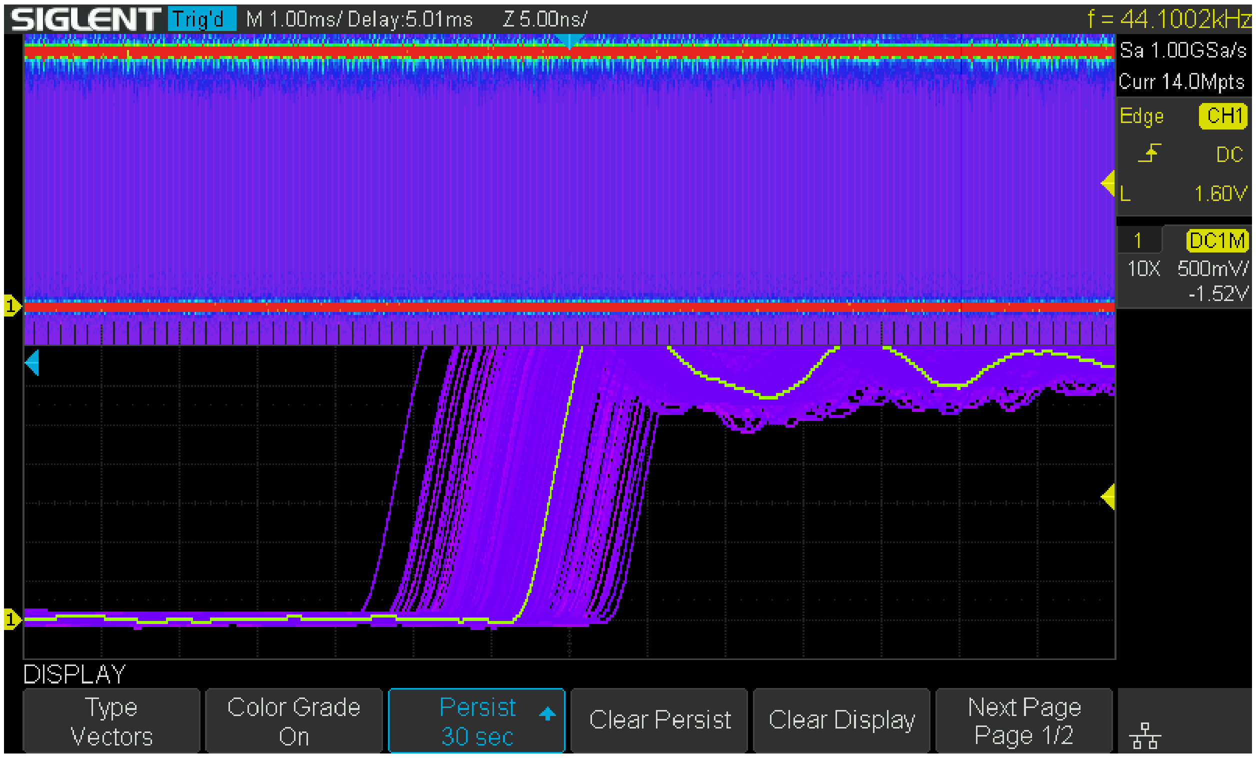


Figure 24 Mounted up for probing. The flying ground lead was not desirable but there weren't a lot of options for a quick experiment


SIGLENT Trig'd M 1.00ms/ Delay:5.01ms Z 5.00ns/ f = 44.1002kHz

Sa 1.00GSa/s
Curr 14.0Mpts
Edge CH1
DC
L 1.60V

1 DC1M
10X 500mV/
-1.52V

DISPLAY

Figure 25 Word clock 5 msec after trigger shows considerable jitter.

The first measurement was to look at the edge 5 msec after the trigger. From the reference measurement (Figure 13) we know a few nsec would be normal with this scope. Here we see around 15 nsec of variation, an indication that something isn't as good here. A look at the spectrum (Figure 26) confirms that there is some sort of clock modulation as we would not expect noise to have a symmetrical pattern under the fundamental (compare with Figure 17).

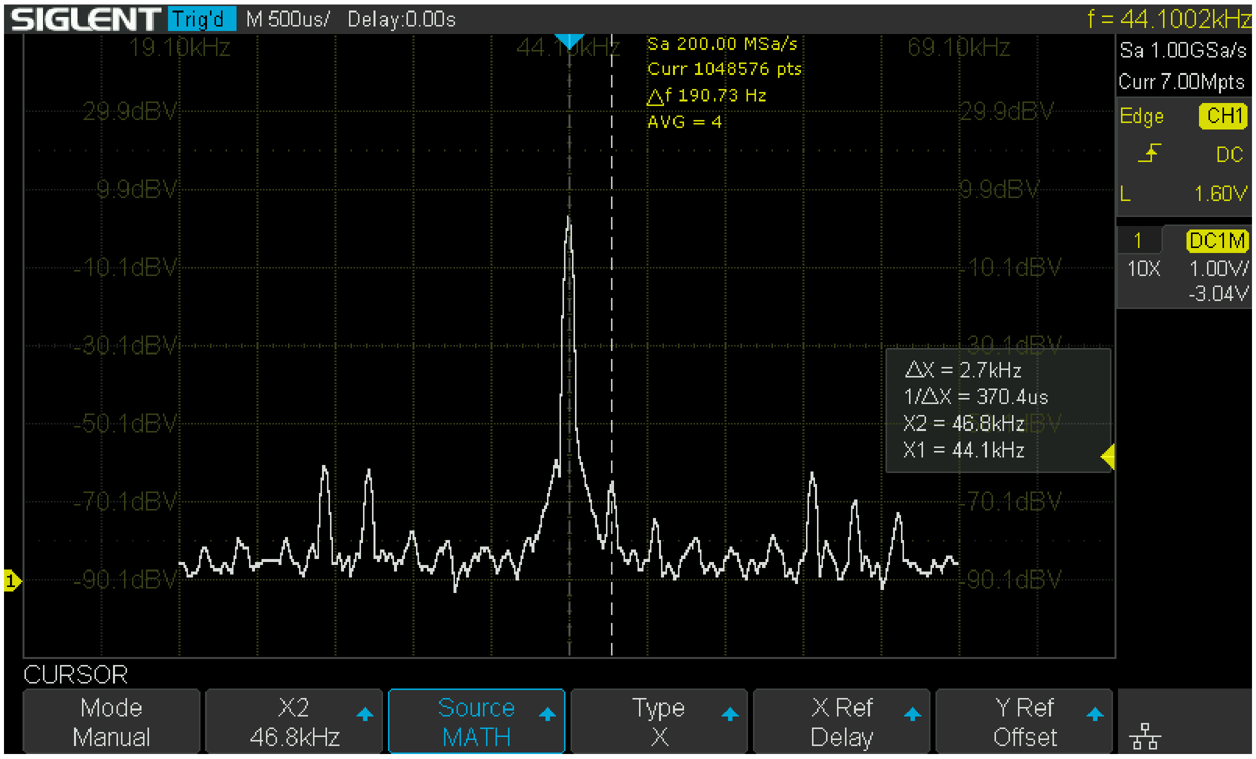


Figure 26 Spectrum of word clock shows effects of clock modulation

From this spectrum we would expect to see jitter spectral components, but first lets look at the basic measurements.

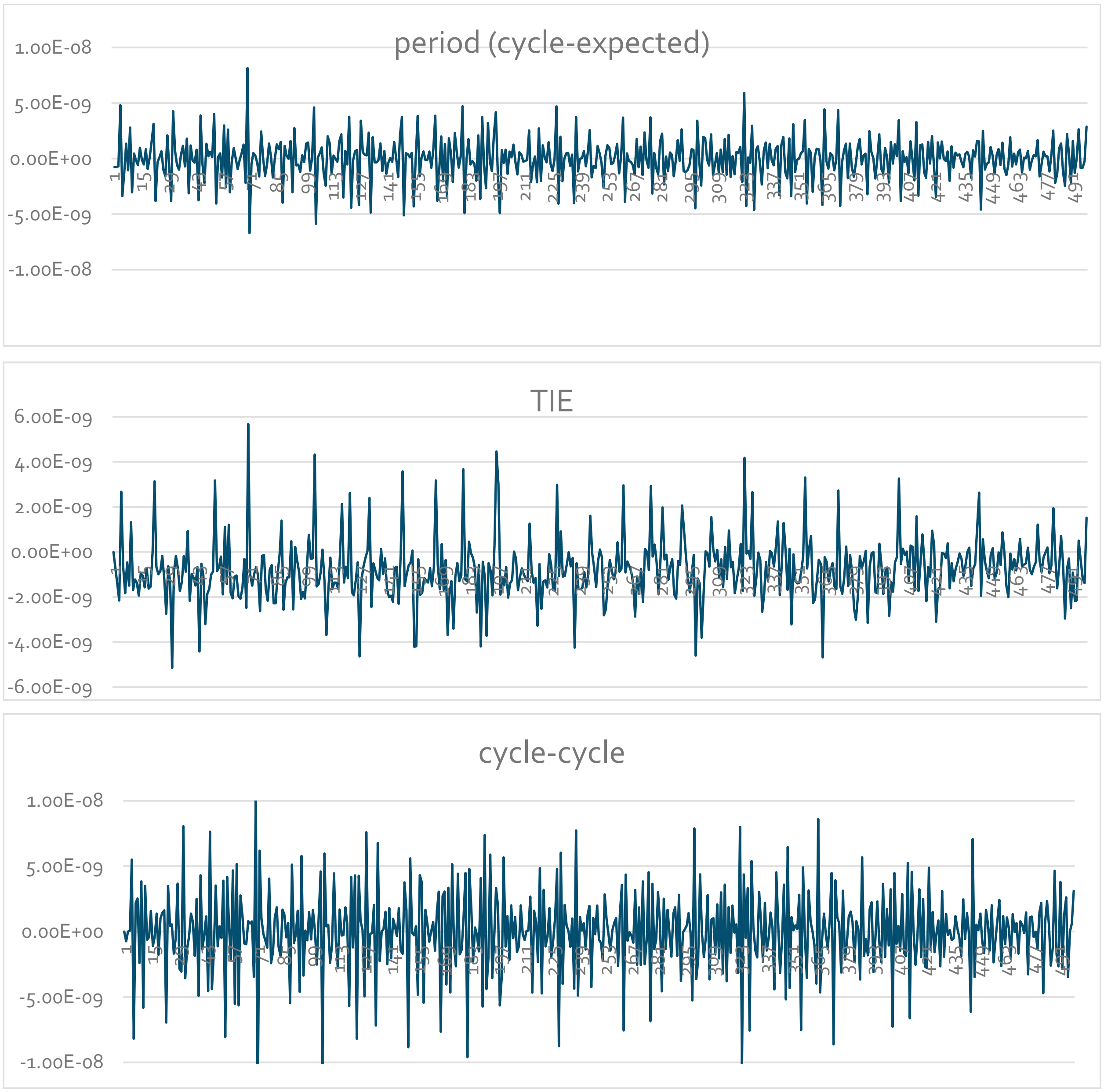


Figure 27 Period, TIE, and cycle-cycle jitter values (vs. edge number) for the SPDIF decoder box word clock for the first 500 edges. Vertical axis is in seconds. Compare with Figure 15, where period and cycle vertical axis limits are +/-300 psec and here the plot limits are +/-10 nsec.

From the plots this device has jitter a factor of 10x over our reference, so the jitter noise floor of our measurements system will only have a small impact on measurements. The histogram plot (Figure 28) also shows the degraded performance.

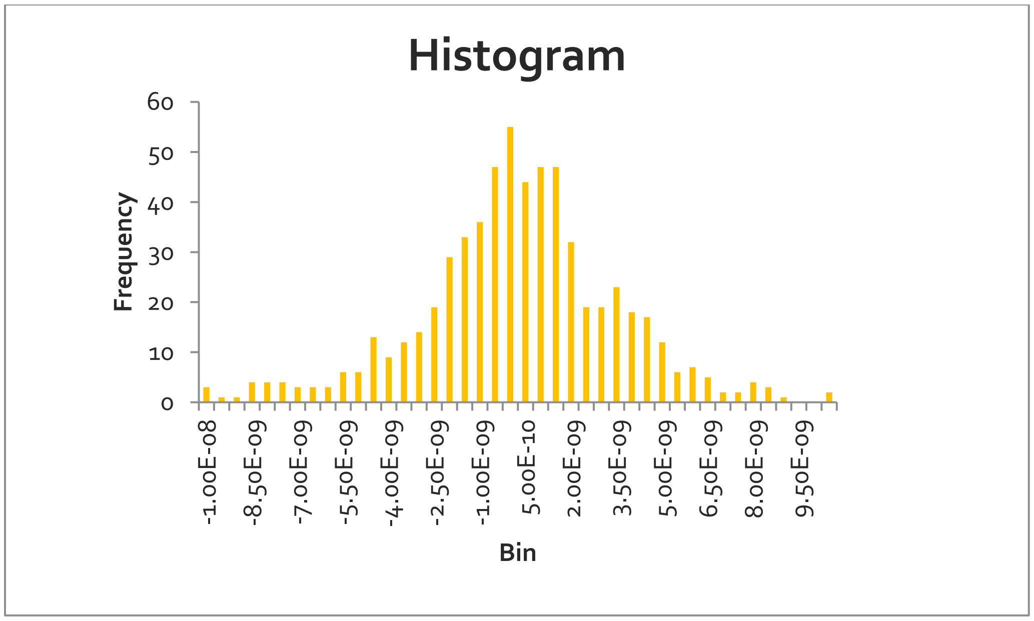


Figure 28 Histogram of cycle to cycle jitter. The scale is 10x of the one shown for the reference system in Figure 16. Only 600 samples in this data set leads to variation in appearance from an idealized curve/

The calculated RMS values are:

- Period 1.9 nsec
- TIE 1.5 nsec
- Cycle-cycle 3.3 nsec

And corresponding peak to peak (from max and min across 600 samples)

- Period 15 nsec
- TIE 14 nsec
- Cycle-cycle 29 nsec

Looking at the jitter spectrum in Figure 29 we see a peak at 2756 Hz (bin size is 86 Hz) which matches will with the measurement from the clock spectrum in Figure 26. Though the cause is unknown, it is 1/16th of the 44.1 kHz sample rate so it's plausible that it's related to hardware or data dependency.

The second largest peak at 12.8 kHz doesn't have an obvious numerical relationship with the clock period or the underlying 2.822 MHz bit rate, however the peaks on either side of it seem to share the same 2756 Hz spacing implied by the largest peak.

As with the first example, capturing multiple datasets and averaging could provide a better picture of the peaks versus noise components of the spectrum.

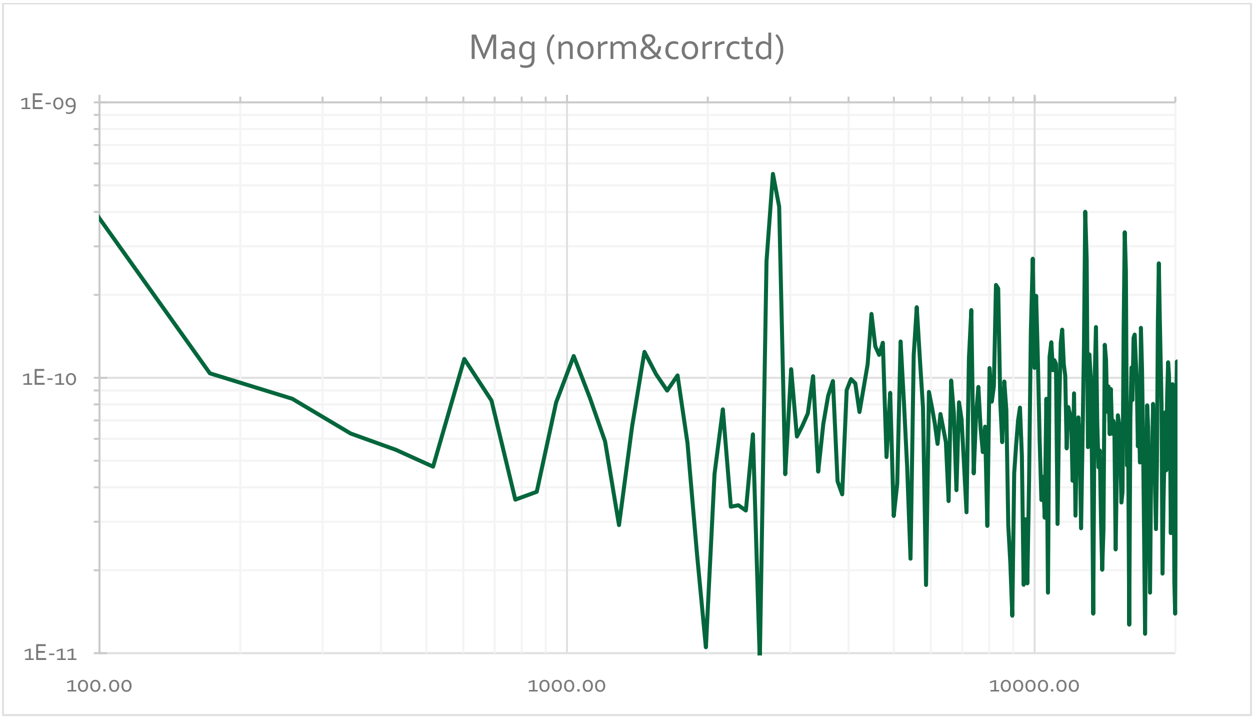


Figure 29 Spectrum of jitter (from TIE) for the SPDIF converter word clock. 512 point FFT, Hann window. The large peak is at 2756 Hz.

6.5 SPDIF RECEIVER BIT CLOCK

This example uses the same hardware as the prior example, except the signal analyzed is the 2.822 MHz (44.1 kHz * 2 * 32) bit clock that would be divided down to produce the word clock analyzed above. We would expect that the jitter would be similar as dividing down does not remove random jitter components, and it's possible for the jitter in a divided down clock to have higher jitter. Figure 30, the observed edge jitter, from an eyeball comparison with Figure 25 would appear to not reject that hypothesis.

Figure 31 likewise suggests a closer look at the spectrum to see if there may be clues to the jitter there. The apparent 480 kHz modulation does not fit any obvious multiple for a 44.1 kHz frame rate.

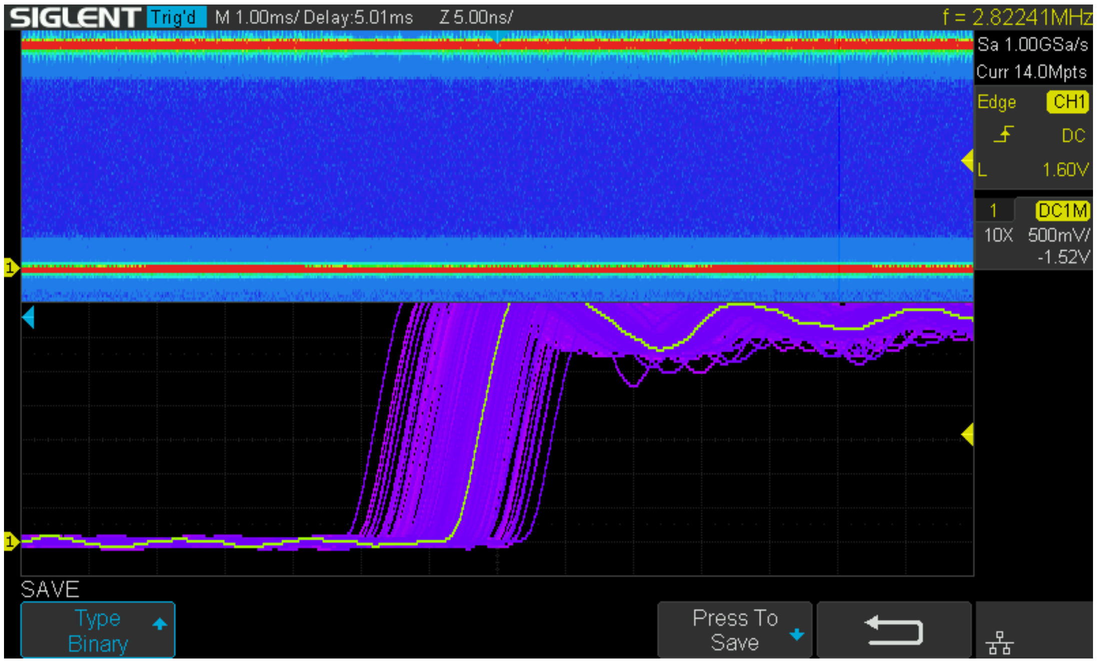


Figure 30 SPDIF receiver bit clock edge 5 msec out.

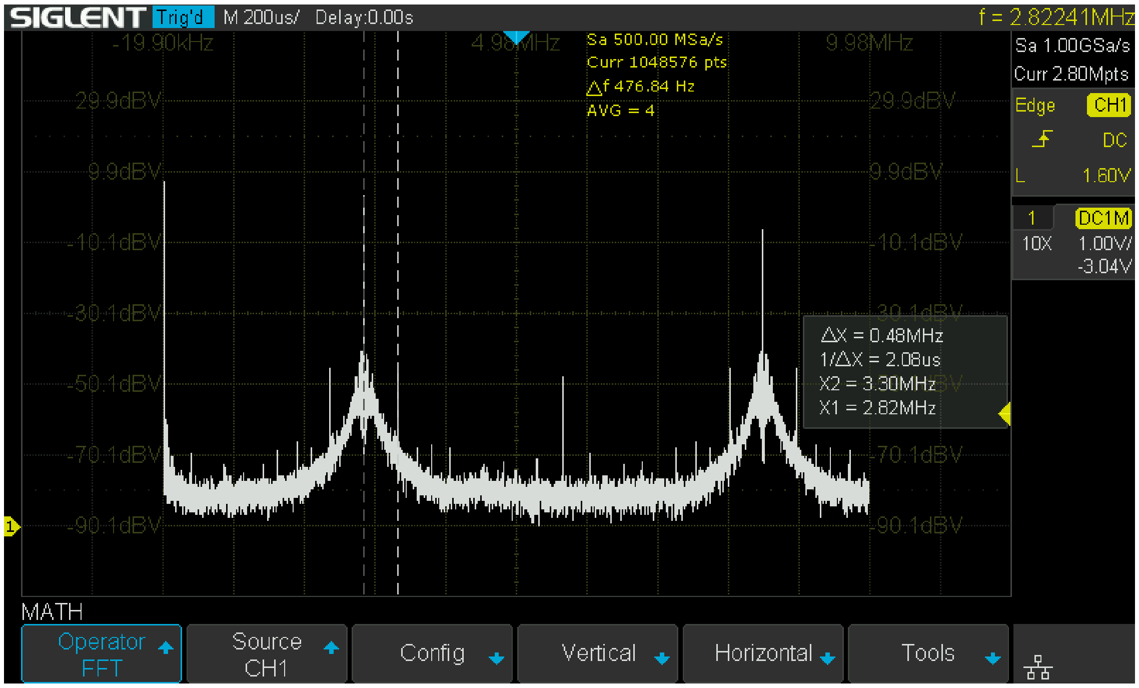


Figure 31 SPDIF receiver bit clock spectrum showing 0.48 MHz tones and high skirts on the 2.822 MHz fundamental

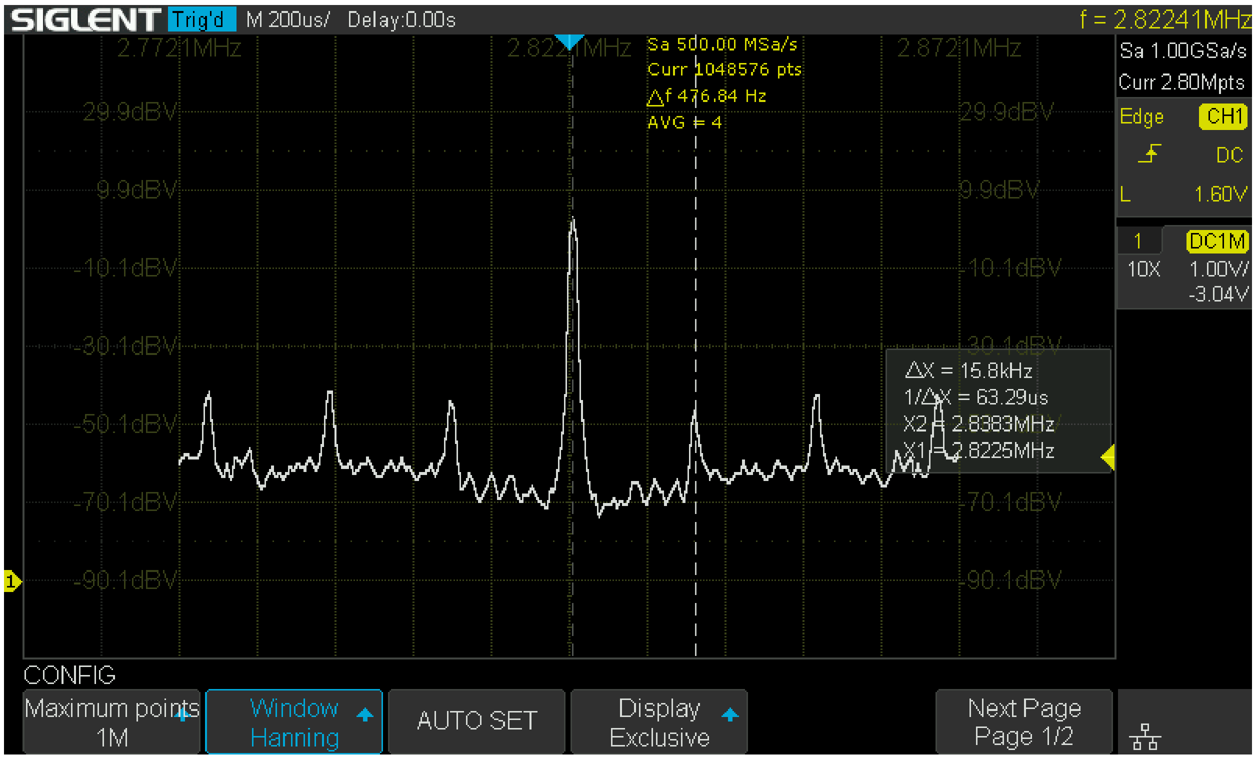


Figure 32 Bit clock spectrum showing modulation components.

The zoomed in FFT clearly shows clock modulation, while the measurement shows as 15.8 kHz the FFT bin size pf 476 Hz is large enough that this component could be related to the 3rd spectral spike seem in the frame sync jitter spectrum. Again there's no obvious integer relationship with the 2.822 MHz bit rate, suggesting the cause may be from some unrelated hardware creating jitter through power supply or EMI coupling.

Figure 33 shows the jitter plots.

The calculated RMS values are:

- Period 570 psec
- TIE 1.59 nsec
- Cycle-cycle 650 psec

And corresponding peak to peak (from max and min across 20000 samples)

- Period 5.6 nsec
- TIE 16 nsec
- Cycle-cycle 7.9 nsec

The larger sample size does affect the expected peak to peak value we will measure for a signal with Gaussian statistics (Figure 34).

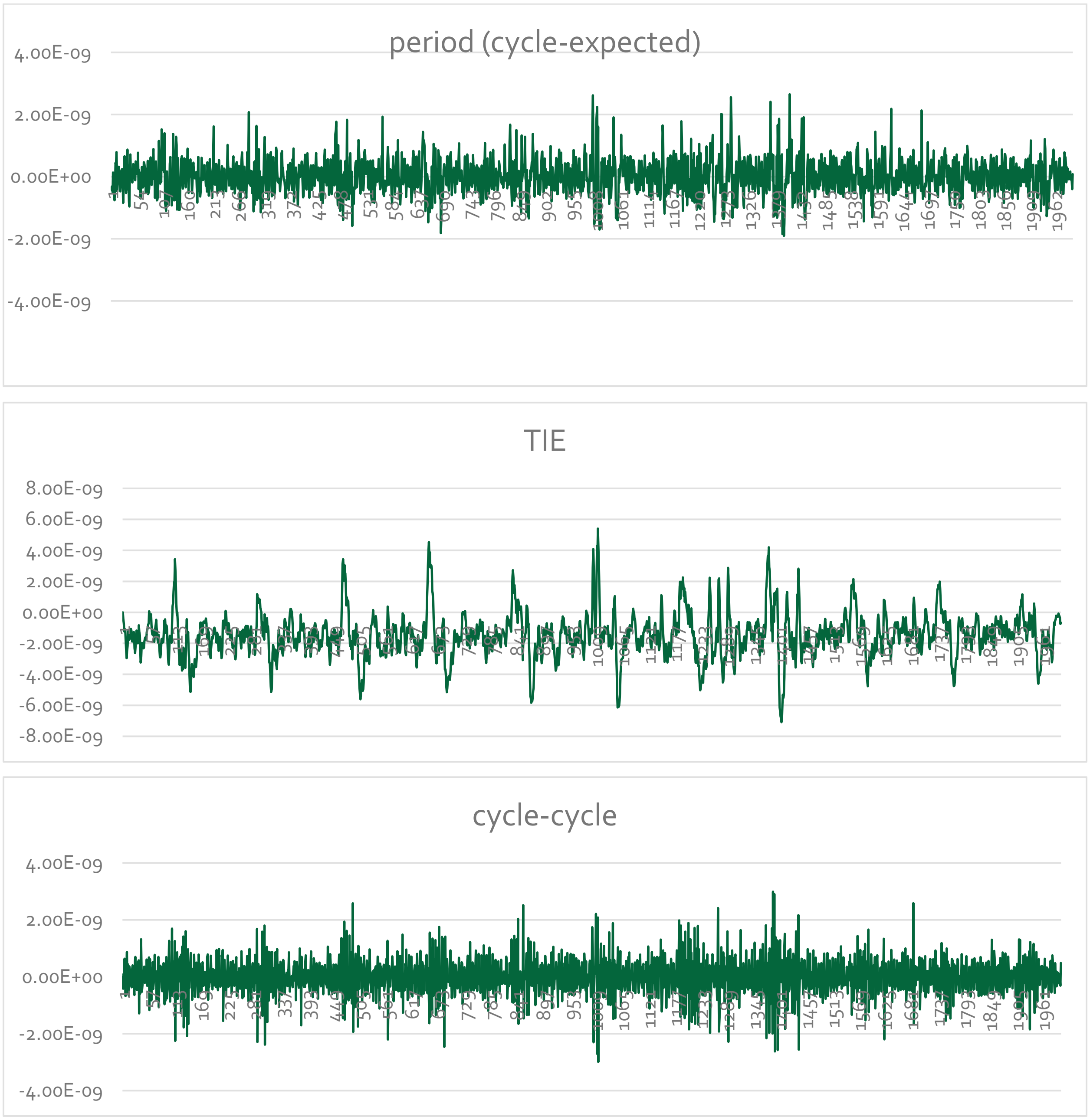


Figure 33 These plots of the bit clock have a longer record length than the plots for word clock in Figure 27. TIE is at a 1/2 scale to the other two plots.

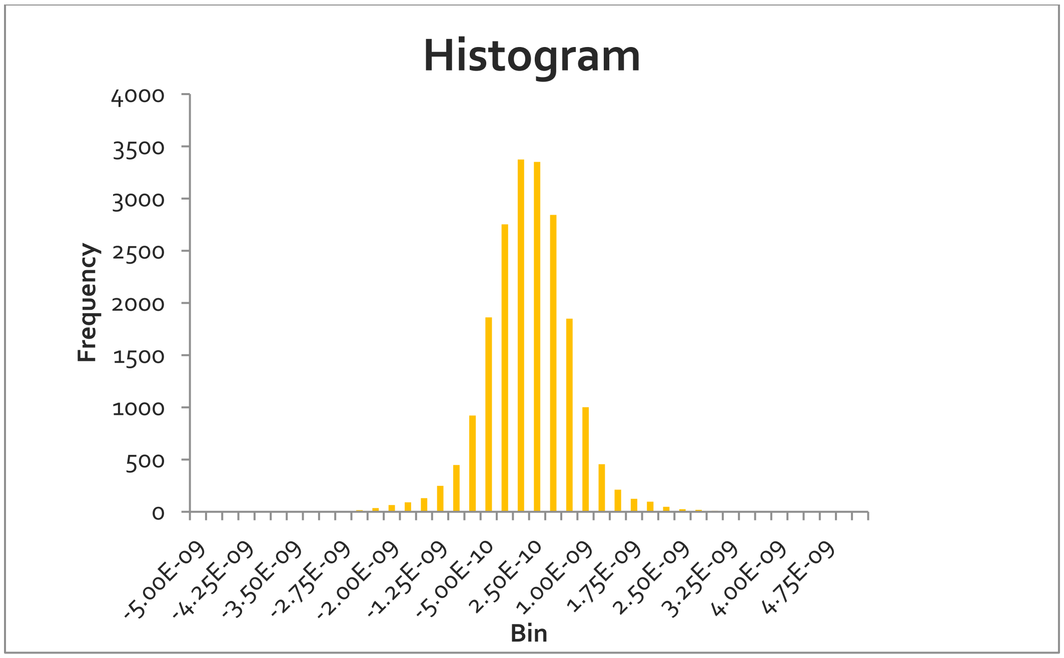


Figure 34 cycle-cycle jitter for 20000 samples

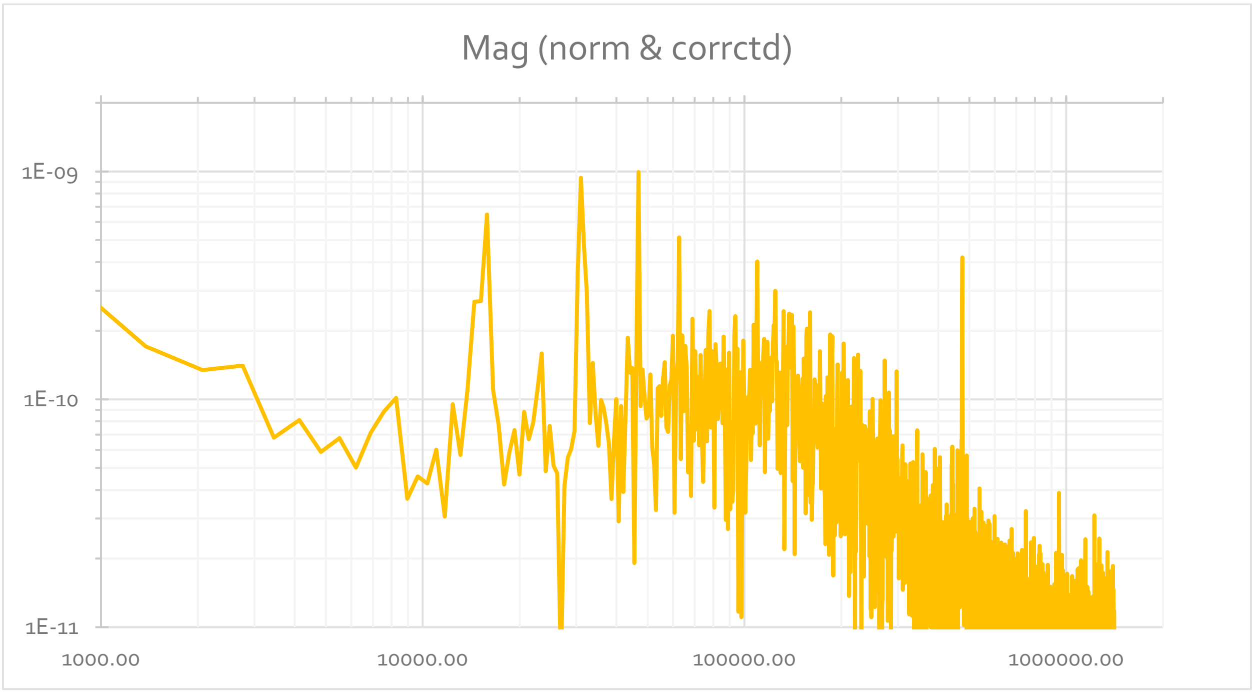


Figure 35 TIE spectrum for BLCK of decoder box

6.5.1 SUMMARY OF THE SPDIF CONVERTER

As we might have expected from a cheap knock-off based on older devices, the jitter performance is substandard compared to what we would find in current audio systems. The performance numbers are 5x or more above the measurement limits. The 2.756 kHz component in the frame sync at 590 psec RMS would be greater than our proposed threshold requirement (Figure 5) of around 200 psec, as would the cluster of tones in the 12 – 20 kHz portion of the jitter spectrum. We see these components in the bit clock spectrum, along with some higher frequencies.

Which of these jitter numbers matter would require knowledge of the DAC. The frame sync should not affect the DAC clocking; since it's divided down from the bit clock we're just using it to better see the low frequency jitter components of the bit clock, which probably forms the basis for the output clock, though most likely via a PLL to create a 256 Fs clock.

7 NEXT

The techniques developed here will be used to diagnose jitter from an AD2428 A^{2}B transceiver being used in an audio application. The results are published on the Clockwork's website TechNotes page: https://clk.works/information/technotes/

8 PYTHON PROGRAM LISTING

For convenience we've included it here.

```
import csv
import pandas as pd
# Xms and Xmx may need to be increased to run this, and 64 bit python as well

#############################
# Copyright (C) 2020 by Clockworks Signal Processing LLC http://clk.works
# Permission to use, copy, modify, and/or distribute this software for any purpose
# with or without fee is hereby granted.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
# TO THIS SOFTWARE INCLUDING ALL IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE
# FOR ANY SPECIAL, DIRECT, INDIRECT,
# OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA
# OR PROFITS, WHETHER IN AN ACTION OF
# CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH
# THE USE OR PERFORMANCE
# OF THIS SOFTWARE.
#
# Original Author: Elizabeth LaMacchia
#
# This version release 7-may-2020
# See Clockworks appnote0001 for a descritpion and use of this program.
# settings are all made via variables.
#
# Threshold should be the midpoint of the signal.
# expected frequency should mathc the exact measured frequency or a small
# bias will creep in
# To get TIE from the output of this program please see the spreadsheet examples.

threshold = .85  # for a scope it may be the scope's raw value, so actual would be 10x this
(with 10x probe)
freqexp = 10000000 # the expected frequency of the edges. Suggest running a 2nd time with the
actual from measurements

# no .ext on filename, we'll add suffixes later

filename= r'20MHZBW_1msec_OCXO'  # you can put a full path here with \ for windows
input_file = filename  + '.csv'

# rise | fall
edge = 'fall'

# don't process header
nNumRowToSkip = 6  # match to your source data

# have to deal with variable number of columns.  Col 1 = time, col 2 = val.
# Some rows could have 3 or 4 cols.
df  =  pd.read_csv(input_file,  skiprows=[i  for  i  in  range(0,nNumRowToSkip)],  header=None,
names=['Time', 'Value', 'ch2'] )
print("csv read complete")
#df.columns = ['Time', 'Value', 'ch2']
df = df.drop(['ch2'], axis = 1)   # would be better to not waste time reading it
                                     # on the 1st place but so far can't make that work
```

```
df['Time'] = df['Time'].astype('float64')
df['Value'] = df['Value'].astype('float64')

ExpectedTime = 1/freqexp
print("Expected period (usec): " + str(ExpectedTime*1000000) )

df.loc[df['Value'] > threshold, 'CurrentState'] = 'high'
df.loc[df['Value'] <= threshold, 'CurrentState'] = 'low'

df['PrevState'] = df.CurrentState.shift(1)
df['PrevValue'] = df.Value.shift(1)
df['PrevTime'] = df.Time.shift(1)

if edge == 'rise':
    df.loc[(df['CurrentState'] == 'high') & (df['PrevState'] == 'low'), 'CrossPt'] = 'Y'

if edge == 'fall':
    df.loc[(df['CurrentState'] == 'low') & (df['PrevState'] == 'high'), 'CrossPt'] = 'Y'

df = df[df.CrossPt == 'Y']

#linear interpolation to find x: exact time crossing threshold
slope = (df['Value']-df['PrevValue'])/(df['Time']-df['PrevTime'])
exacttime = (threshold - df['PrevValue'])/slope + df['PrevTime']

df.insert(loc=len(df.columns), column='TimeAtThreshold', value=exacttime)

df['DeltaT'] = df.TimeAtThreshold - df.TimeAtThreshold.shift(1)

df = df.drop(['CurrentState', 'PrevState', 'PrevValue', 'PrevTime', 'CrossPt'], axis = 1)

df['SubExpT'] = df.DeltaT - ExpectedTime

NumberOfEdges = str(len(df.index))

print(df)

print("Number of edges is: " + NumberOfEdges)

df.to_csv(filename +'_out.csv')

# that's all
```