Siglent SDS1104X-E Review

Source: https://www.eevblog.com/forum/testgear/siglent-sds1104x-e-in-depth-review/

Introduction

The new Siglent SDS1x04X-E series SPO have entered the world market and appear to have the potential of setting a new price/performance standard in the entry level class.

The most striking features are:

- 4 channels
- Sample rate **2 x 1GSa/s** or 4 x 500MSa/s
- Bandwidth 100MHz or **200MHz**
- Memory depth **2 x 14Mpts** or 4 x 7Mpts
- Up to >100k Wfm/s (up to **>400k** in sequence mode)
- Vertical true full resolution sensitivity down to **500µV/div**
- Segmented memory up to **80000 segments** for **history** and **sequence** mode
- **Full speed mask test**
- **1Mpts FFT** with SA-oriented user interface
- Automatic **measurements** use full record length with **no decimation**
- **Bode plotter** for network analysis
- Full connectivity over USB and **LAN**, optional WiFi
- Built-in web server
- Optional external AWG SAG1021 up to 25MHz
- Optional **16 channel MSO** with 1GSa/s up to 14Mpts

Siglent were kind enough to offer me an early SDS1104X-E for evaluation and I couldn't resist finding out whether this scope has the potential to meet the high expectations. The SDS1x04X-E series scopes aren't expensive, yet they are certainly not the cheapest DSOs around. But then again, I figure many of us would be looking for the best value for money rather than just the cheapest price. As can be seen from the listing above, these models have a whole lot to offer – certainly more than the average DSO in this class.

In any case this appears to be a very interesting product and I felt it deserves an in-depth review. Because of my very limited spare time this has actually started long before the scope was finally released. Consequently, most of the review is based on a pre-production hardware and early firmware, which raises the question how to deal with issues encountered during evaluation. Of course it would not be fair to point out bugs of an unfinished firmware that was never meant to be released to the public.

Thankfully it turned out to be not a big problem, as most of the functionality was of surprising high quality already, with only minor issues that did not prevent me from doing my tests the way I had them planned and getting the results I'm going to present now. Of course there were (and sure as hell still are) a number of flaws, particularly in the areas that are either completely new or have undergone a major re-design:

- Serial decoding had been redesigned and looks promising, but needs further refinement.
- MSO Option was not yet available.
- Bode Plotter first shot, good concept, but certainly far from finished yet.
- XY-mode currently some emergency solution that will see some major re-design.
- Sequence mode is currently broken, but Siglent promised a fix within a short time.

A full and final review of these functions listed above will be published later, after a firmware with the necessary fixes has been released.

Differences to SDS1202X-E

Even though the Siglent SDS 1x04X-E has a lot in common with the SDS1202X-E, it is still a newer design and one step further in some areas. It particularly has got some exclusive features and this is also reflected by the new front panel layout. The obvious differences are listed below:

Connectivity. The SDS1104X-E now offers a 2nd USB host port at the rear, which is handy for connecting one of the new options: the WiFi dongle or the SAG1021 AWG (or any other Siglent SDG device). This way, the front panel USB connector remains free for an USB memory stick.

Sbus Connector. This is where the new SLA1016 logic probes are connected. This will add 16 digital channels with a max. sample speed of 1GSa/s and up to 14Mpts memory.

Shared vertical gain and position controls. Some people might not like this, but there is no alternative in a compact 4-channel DSO like this. I don't see a big problem either; for real work, separate controls don't offer a big advantage in my experience. In R&D I set up the channels within a minute or so in order to monitor the critical signals. Then I tweak the hardware and/or firmware until I get the desired result, which might take hours or even days without touching the vertical settings. If anything, I'm more likely to alter timebase, trigger, math or measurements rather than channel gain and position. Other input channel related settings like input coupling and bandwidth limit are not directly accessible anyway. A service technician on the other hand might often change gain and/or position when probing many different test points in sequence, but this is usually done with just one single channel where separate controls would not make a difference.

Search. Now this scope finally got the first implementation of a search function, which allows finding certain trigger conditions within a long record. It does not search across the history though (where we would need it the most) and I really hope we'll get that eventually with a later firmware.

Navigate button group. This is finally the solution for the old problem of dialing in a huge amount of delay with the horizontal position control. The new navigate function is essentially an automatic knob twiddler that works in three speeds and it can also browse through search events and history frames. I've learned to appreciate this feature pretty quickly.

No Ext. Trigger input. Other than the SDS2x04X, which still offers an external trigger input even for the four channel models (albeit on the rear side of the instrument), the SDS1x04X-E doesn't have that anymore. But then again, with up to 20 possible channels (4 analog + 16 digital), this is only a very minor loss.

To sum it all up, the obvious differences between the SDS1x04X-E and the older SDS1202X-E are:

- 4 channels
- No ext. Trigger input
- Sample rate **2 x 1GSa/s**
- Memory depth **2 x 14Mpts**
- **Search** function
- Dedicated **Navigate** buttons
- **Bode plotter** for network analysis
- Built-in web server
- Optional WiFi
- Optional external AWG SAG1021 up to 25MHz
- Optional **16 channel MSO** with 1GSa/s up to 14Mpts

Base Functionality

Before looking at all the bells and whistles, I always want to make sure that the base functionality is sound and the scope will do a proper job, also as a substitute for analog scopes. After all we want a reliable tool.

Display

SPO (Super Phosphor Oscilloscope)

Obviously a major task of any oscilloscope is to display waveforms. Early DSOs did just that in a very restricted way – this was one of the many reasons why particularly analog engineers disliked them and rather stuck with their trusted analog scopes, because these were not only free of limitations and artifacts, but also provided a lot more information than just timing and waveform: it was the trace intensity, which depends on the trace speed and frequency and simply comes down to the fact that a certain spot on a CRT screen will be brighter when it is hit by the electron beam longer and/or more often than others.

DSOs eventually got more memory and became fast enough (i.e. could capture many thousand waveforms per second) so that a similar behavior as the analog scope could be emulated. This is universally known as intensity grading and for digital scopes yet another variant could be added, known as color grading. The quality of the implementations differs a lot between various DSO models and every vendor has its own name for this technology. Siglent calls it SPO (Super Phosphor Oscilloscope).

The Siglent SPO technology is nothing new and has proved to give excellent results. We already know it from the existing Siglent SDS 1kX(-E) and 2k(X) scopes. Yet I shall review it once again just for completeness and because it's the basis for all the future demonstrations in this article.

Most modern DSOs provide some sort of intensity grading (though some implementations barely deserve that name), even in the entry level class. Not all do color grading though, which can be very beneficial especially for displaying narrow pulses or steep edges in general. This Siglent SPO offers both. It also offers the choice of dots and vectors display as well as persistence.

Dots mode just shows the individual sample points as they have been acquired by the ADC. Vectors mode interconnects these points in order to create a contiguous trace without gaps, which in turn slows acquisition down a bit. On the other hand, this scope is fast enough, so that we can make do without vector mode most of the time, because the sheer amount of dots from many acquisitions mapped to the screen at the same time will give the impression of a contiguous trace as well. Furthermore, we can always do an acquisition in dots mode and then turn on vectors in stop mode later.

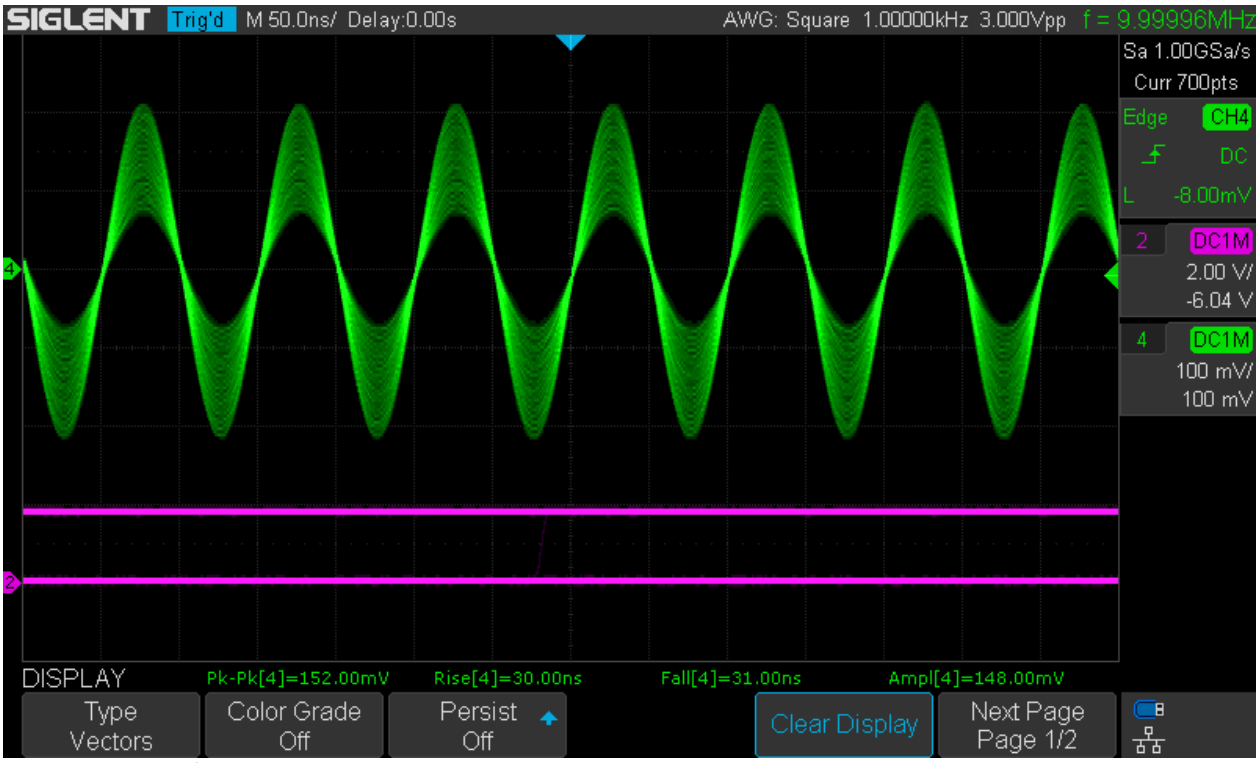
Now let's examine a popular example with an amplitude modulated signal.
A 10MHz sine carrier is 50% amplitude modulated with a 1kHz sine and we look at the carrier first.

Channel 4 shows 7 periods of the modulated carrier and because of the fast waveform update rate, we see all possible states of the carrier at once. The intensity of the trace is higher at the regions that are more frequently hit, which is of course at the edges, but also at the extremes of the envelope because of the sine as a modulation waveform.

Cannel 2 displays a sync signal for the modulation waveform, but this is only 1kHz and of course not synchronous when triggering on the 10MHz carrier.

MOD_10MHz_1kHz_50%_50ns_Vect_IG

Of course it looks different with other modulation waveforms, like a triangle (ramp with 50% symmetry) for example.



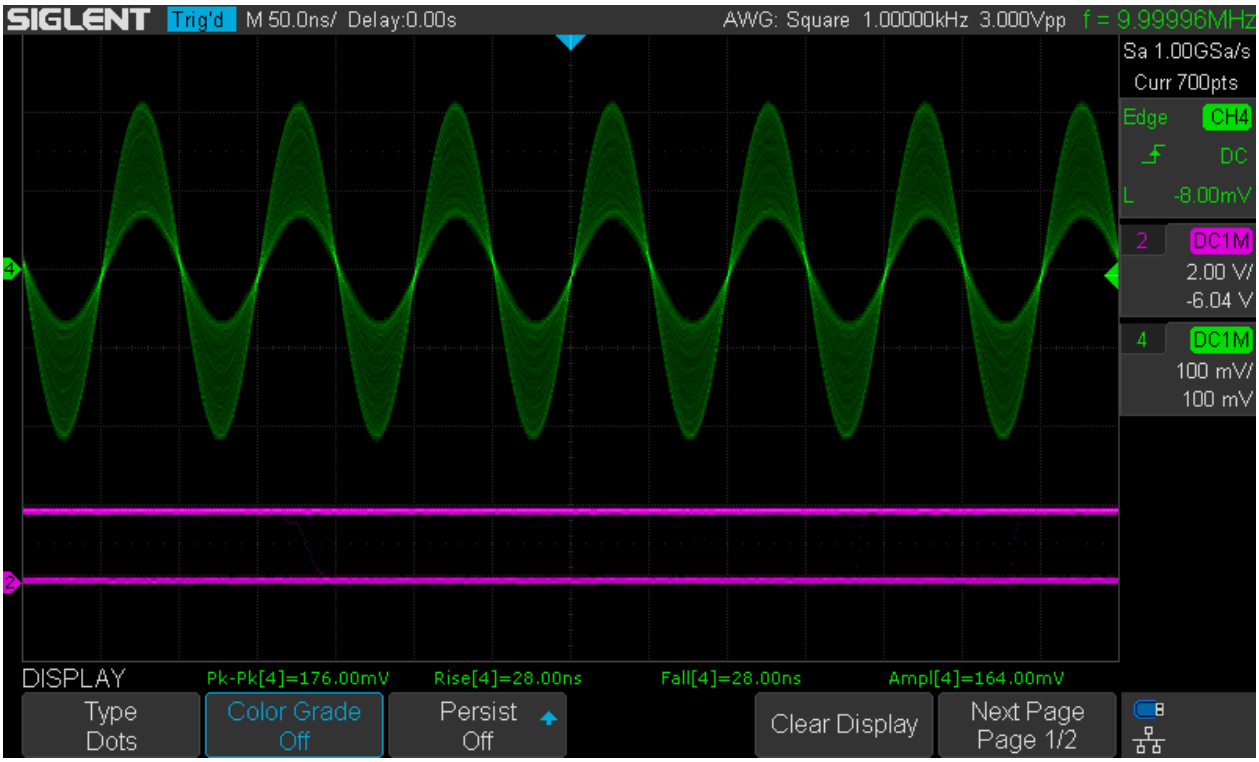
MOD_10MHz_1kHz_50%_50ns_Vect_IG_ramp

As expected, the trace intensity is now pretty much uniform in the vertical direction.

Another extreme can be seen by selecting a squarewave as the modulation source.

MOD_10MHz_1kHz_50%_50ns_Vect_IG_square

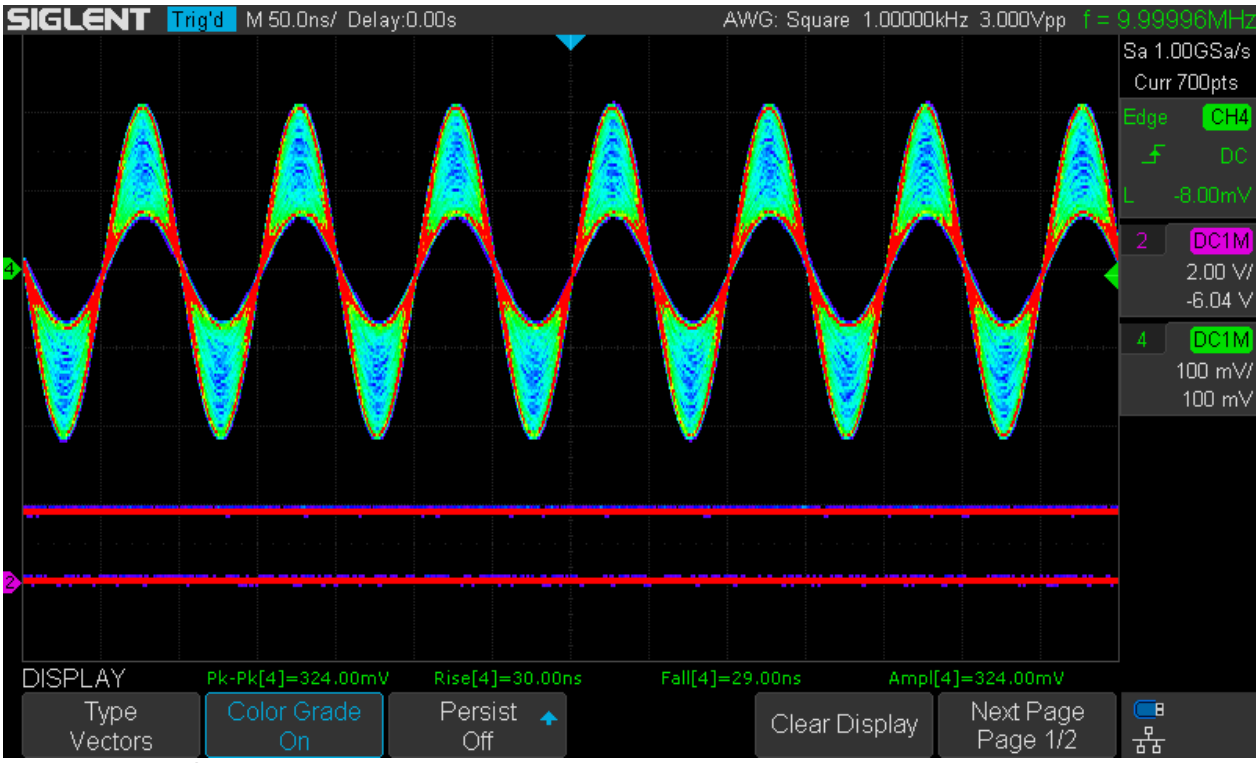
Now only the extremes are displayed, because a squarewave modulated carrier just switches between two discrete amplitude levels. Let's go back to the sine modulation with dots instead of vectors:



MOD_10MHz_1kHz_50%_50ns_Dots_IG

Without any interpolation, the trace image gets a bit dimmer, but it still works, hinting on the high speed acquisition in this DSO.

Now let's see what it looks like with color grading:



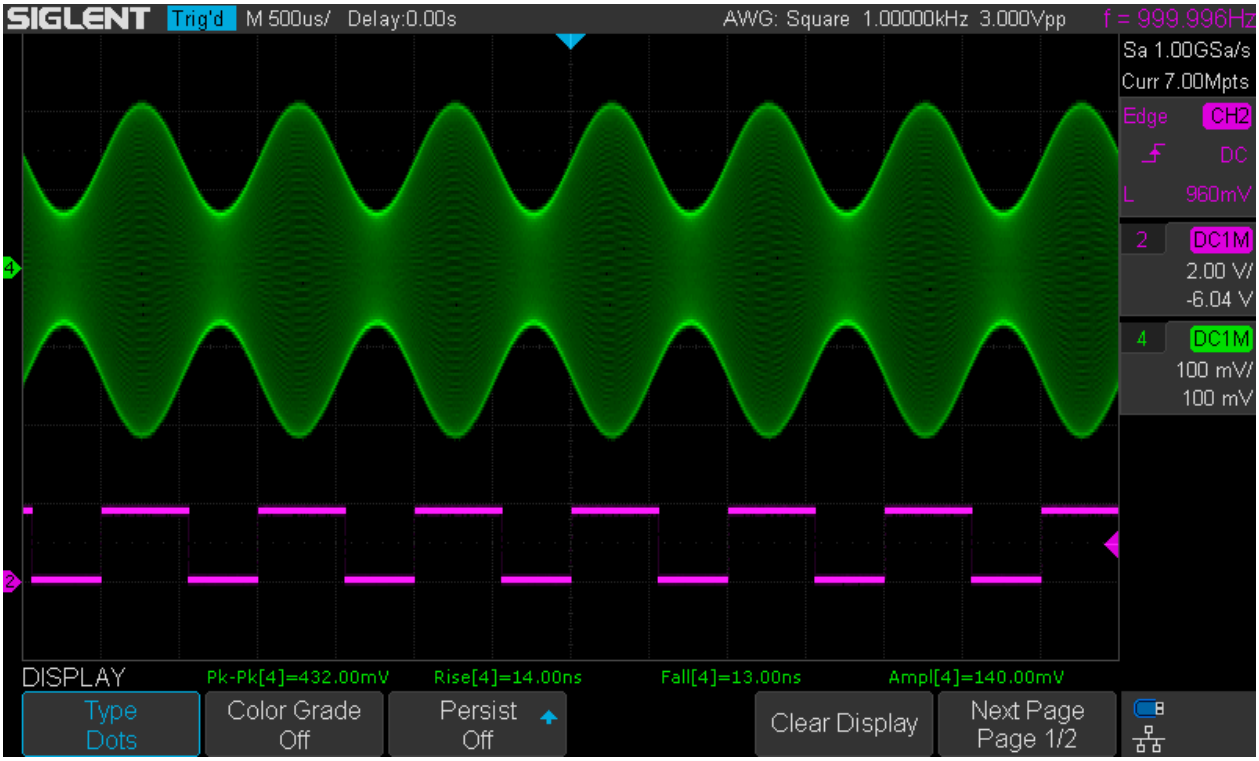
MOD_10MHz_1kHz_50%_50ns_Vect_CG

For unknown signals, this might give an even better insight what's really going on.

Up to now, we have used a 50ns/div timebase and triggered on the 10MHz carrier. We can trigger on the modulation signal and use a much slower timebase accordingly.

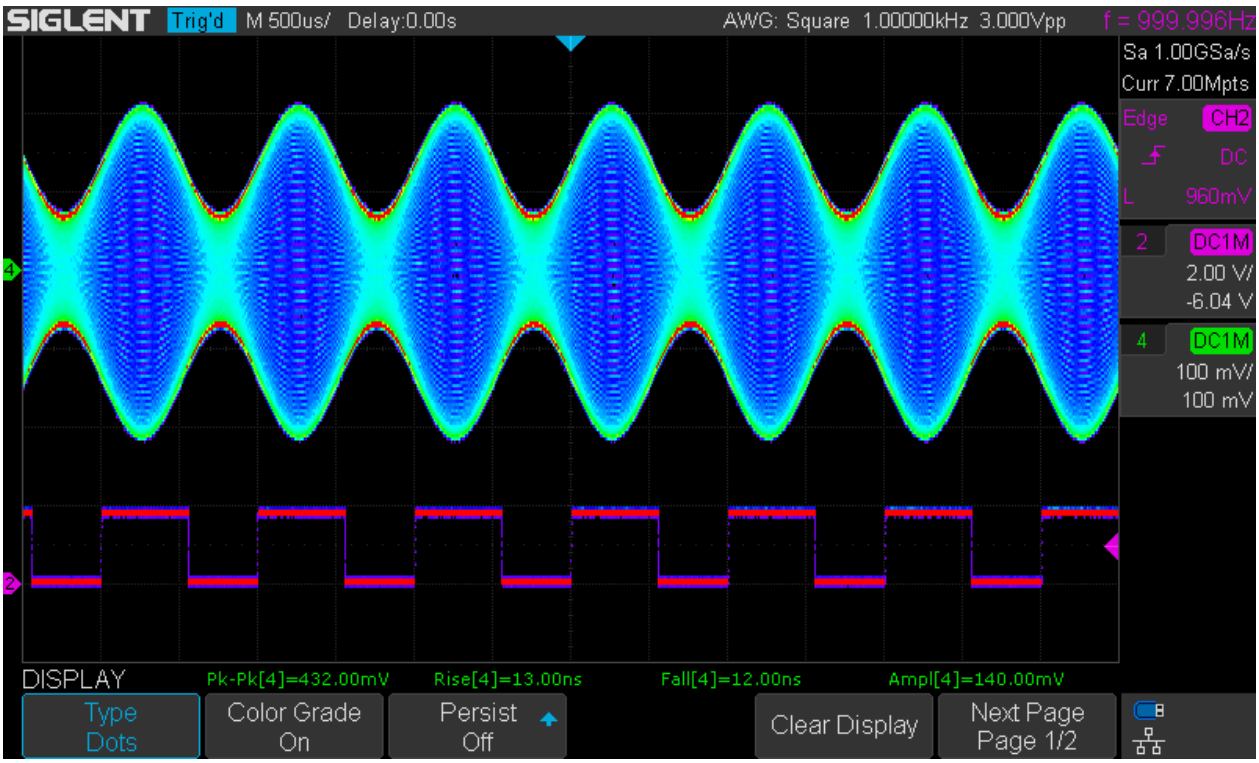
In this case we are triggering on the sync signal on channel 2 and the timebase is set to 500µs/div. At this relatively slow speed, the room inside the envelope is naturally always filled by the carrier trace, but we still need a high acquisition rate in order to accomplish the intensity grading.

Only Dots mode is shown here, because at slow timebase settings like this, it absolutely makes no difference – vectors mode looks exactly the same.



MOD_10MHz_1kHz_50%_500us_Dots_IG

The same situation as above, but this time with color grading:



MOD_10MHz_1kHz_50%_500us_Dots_CG

Display Modes

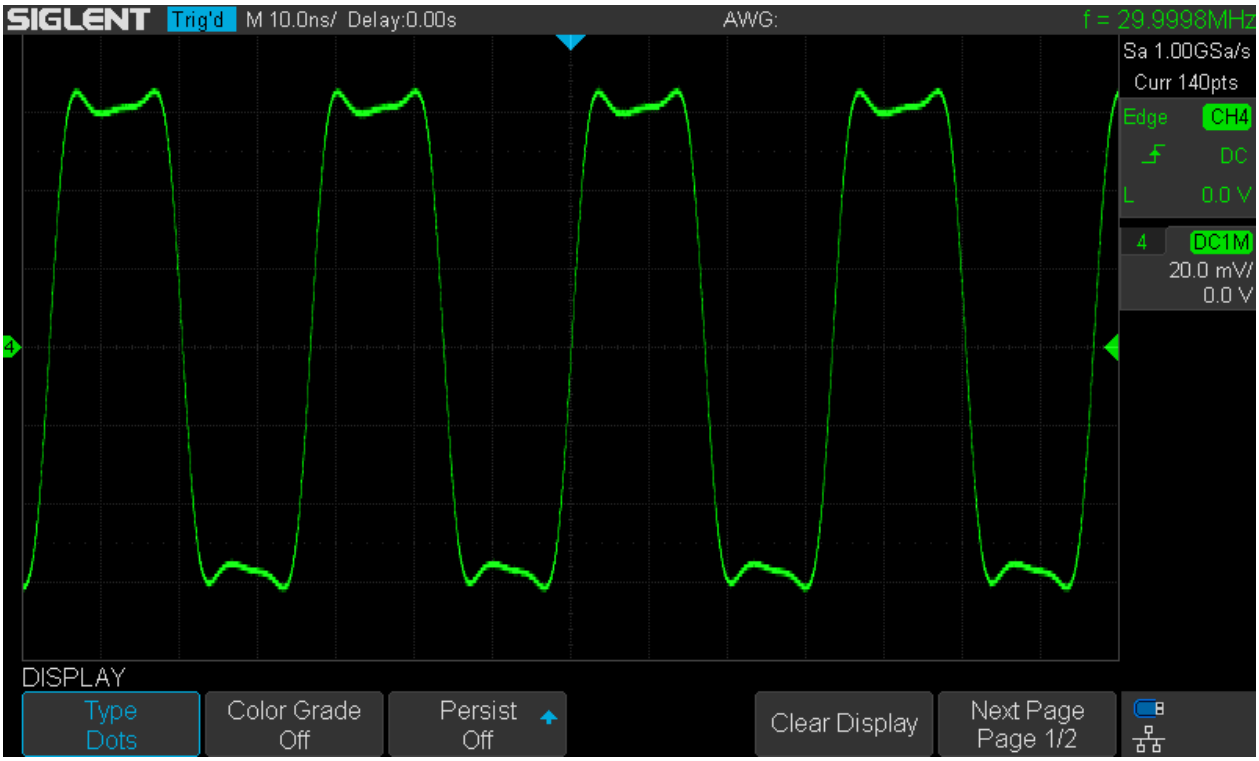
The Siglent X-series scopes have a number of settings that affect signal representation on the screen that might be a bit confusing at first. So right at the start, we shall have a closer look at them in order to learn how to setup the scope for optimum results.

Some of these settings will be discussed again in the *Acquisition* chapter, but a brief overview will be given here together with some basic rules that should be easy to remember. It is highly recommended to stick with these rules and only go a different route for very good reasons.

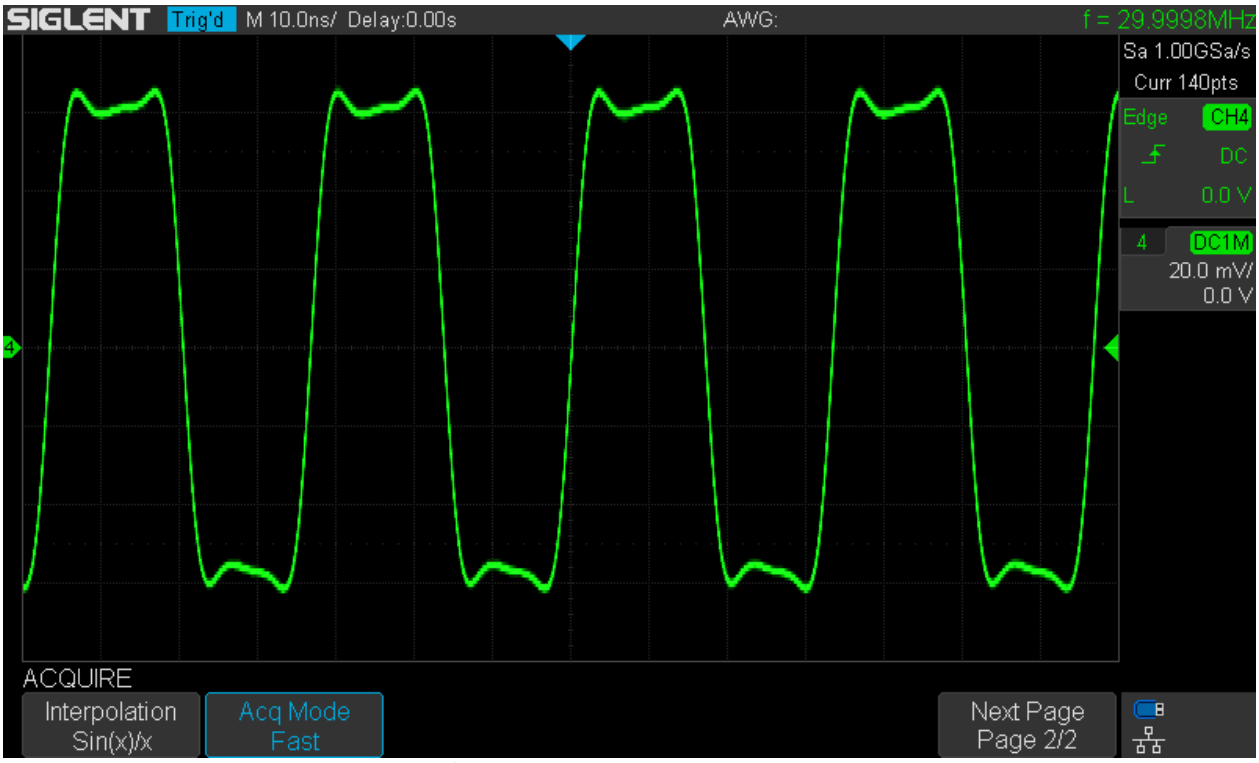
- **Acquire – Interpolation**: always use Sin(x)/x for analog waveforms, x for (digital) pulses.
- **Acquire – Acq. Mode**: Fast is standard, use Slow only if for some reason you do not want to see multiple traces after a signal drop-out in normal trigger mode. Be aware that this will significantly decrease waveform update rate.
- **Display – Type**: For normal Run mode, use Dots whenever feasible to get highest fidelity for signal reproduction as well as highest speed. Use Vectors when you need to aid visibility for fast edges, narrow pulses, in stop mode and some special functions like Acquisition Mode Slow, in History and for FFT.
- **Display – Color Grade**: Use this whenever you like, especially for aiding visibility on fast edges and narrow pulses – particularly useful for screenshots. Be aware that is not the best setting for multi-channel acquisition because channels look indistinguishable on the screen.

Some examples shall demonstrate the effect of various combinations of the settings mentioned above. A 30MHz square wave with extremely fast edges (<100ps) and Sin(x)/x reconstruction is used for all the following tests.

Let's start with a pair of screenshots for Acquisition Mode Fast, Display Type Dots versus Vectors and Color Grade Off.



SDS1104X-E_Square_30MHz_10ns_fast_dots



SDS1104X-E_Square_30MHz_10ns_fast_vectors

The signal representation is very similar, just the fast edges are a little brighter. Generally, intensity grading works better with dots in many cases.

Here is a similar pair of screenshots for Acquisition Mode Fast, Display Type Dots versus Vectors and Color Grade On.

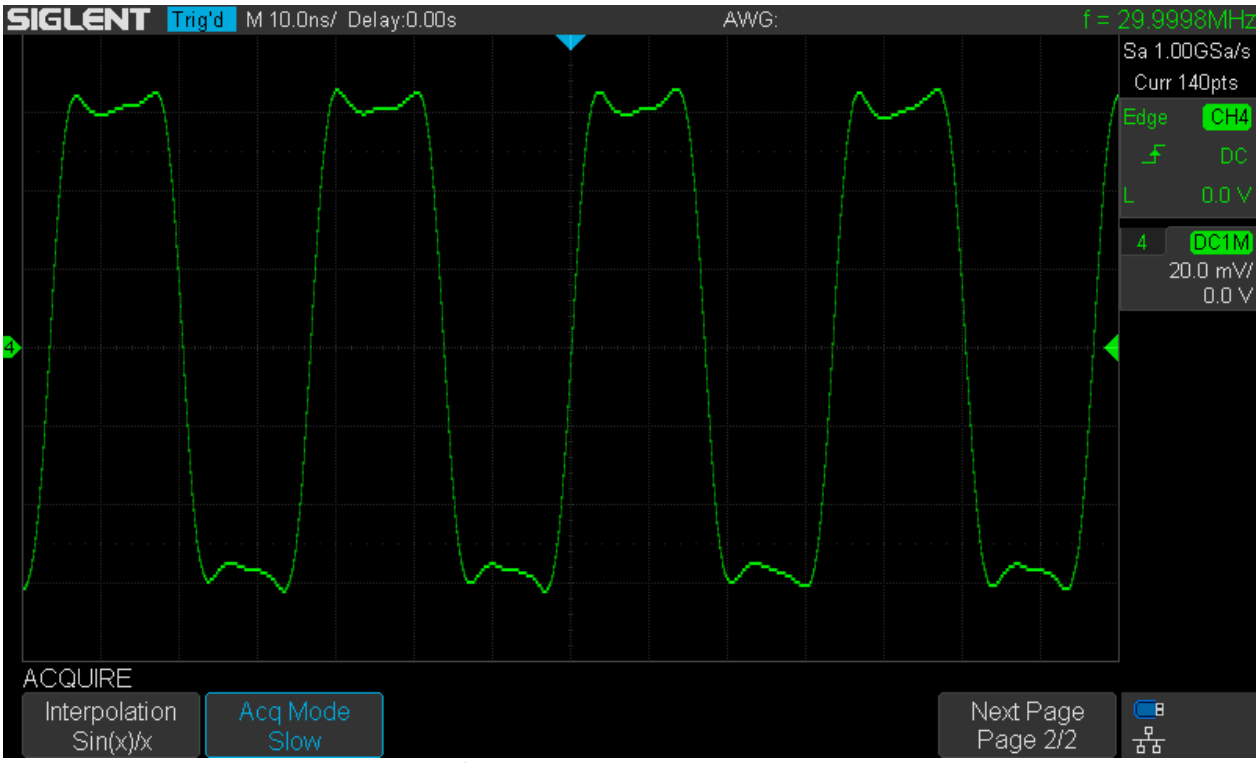
SDS1104X-E_Square_30MHz_10ns_fast_dots_color

SDS1104X-E_Square_30MHz_10ns_fast_vectors_color

Again the signal representation is similar, just with vectors the fast edges appear in the same color as the rest of the signal. In general, color grading works better with dots in many cases.

Here comes a pair of screenshots for Acquisition Mode Slow, Display Type Dots versus Vectors and Color Grade Off.

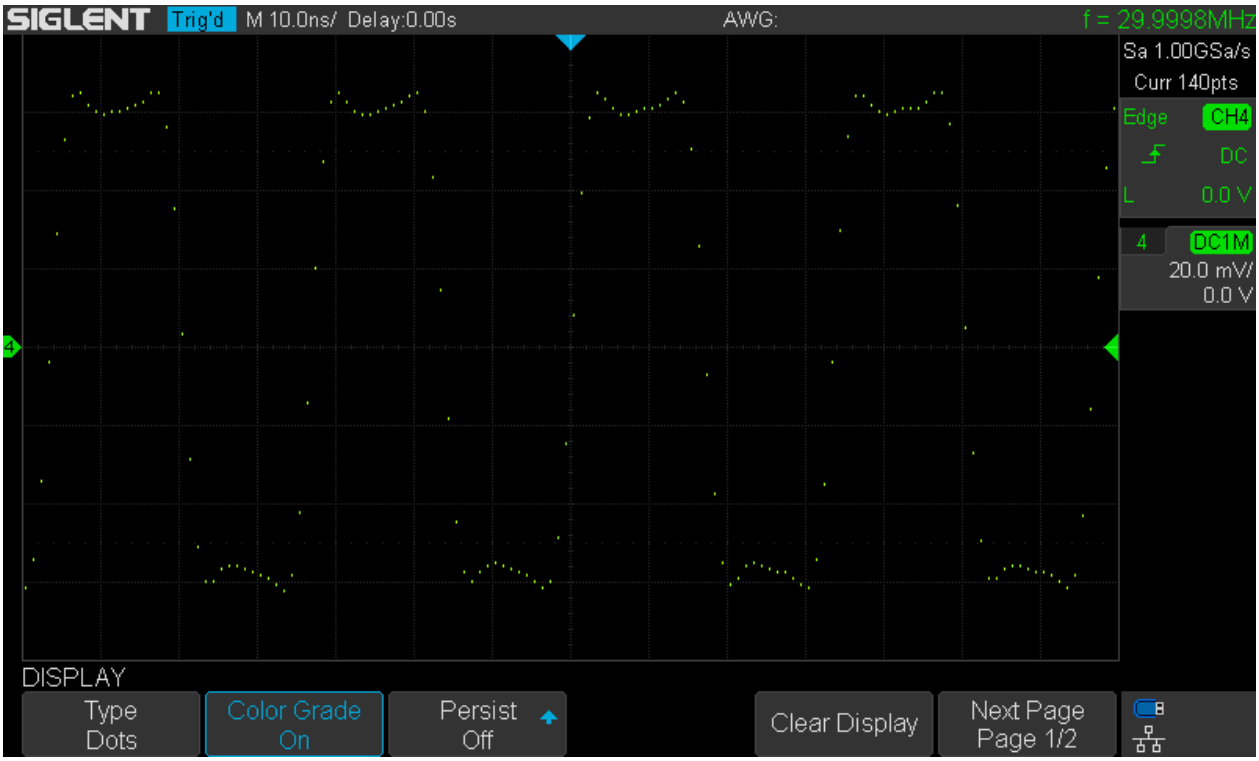
SDS1104X-E_Square_30MHz_10ns_slow_dots



SDS1104X-E_Square_30MHz_10ns_slow_vectors

In Slow Mode, the difference is striking and we would use Dots only in special situations when we need to see the true samples of the input signal. We still get some intensity grading though.

Here is a similar pair of screenshots for Acquisition Mode Slow, Display Type Dots versus Vectors and Color Grade On.



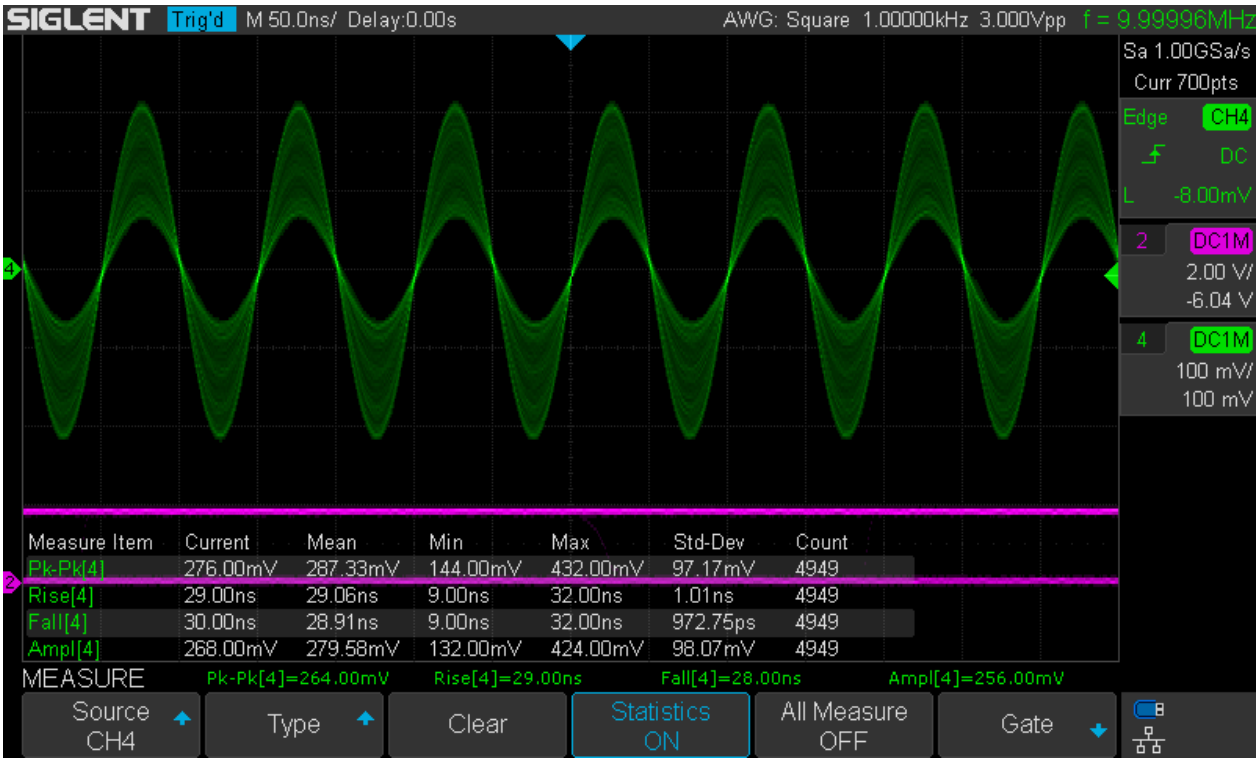
SDS1104X-E_Square_30MHz_10ns_slow_dots_color

SDS1104X-E_Square_30MHz_10ns_slow_vectors_color

Once again, the difference is striking and we would use Dots only in special situations.

A first glance at measurements

Have you noticed the measurements for peak-peak and amplitude for the examples in the SPO section? They are different in every screenshot. That's only natural, as the measurements can only be valid for one single acquisition, whereas the amplitude of the carrier changes all the time because of the modulation. To get meaningful measurements in this situation, we need to enable statistics.



MOD_10MHz_1kHz_50%_500us_Dots_IG_stat

Now we can see the minimum, maximum and mean value of the amplitude measurements. From these figures, it becomes obvious that it's indeed a 50% modulation, as min is roughly 50% of mean and max is 150% respectively.

Persistence Mode

Persistence is another feature for emulating analog scopes, because CRT screens always had some persistence by nature and then there have been special analog storage scopes with very long persistence times and the ability to explicitly clear the screen. This is just a means to make rare events better visible. Instead of being displayed on the screen for just the fraction of a second, we can keep them displayed for a certain time, 1s, 5s, 10s, 30s and infinite in case of the SDS1104X-E. Of course there is also a menu button to clear the display, which is useful whenever a persistent image is no longer needed.

Here's an example how persistence can be used to detect rare events.

Channel 4 still displays the already familiar amplitude modulated waveform, and it is worth noticing it is not affected by the persistence mode. This is an important feature, as persistence doesn't kill intensity or color grading while still displaying unusual events.

An unsynchronized pulse, 10ns wide, is fed into channel 2 at a rate of 10Hz and it rarely can be seen for the blink of an eye at the given timebase of 50ns/div. We see just a flat line almost all the time, just as in the screenshot below.

Pulse_10Hz_10ns_P0

Now let's turn on 30s persistence and wait a minute, just to see what happens.

Pulse_10Hz_10ns_P30

Now at a certain point we can even see 3 pulses at the same time, If channel 2 was expected to actually be just DC, then the persistence mode gives us all the information to trigger on the undesired signal and

subsequently look for correlated actions that might cause this. So it's a nice tool for troubleshooting, but certainly not the only one in this DSO.

Horizontal Zoom

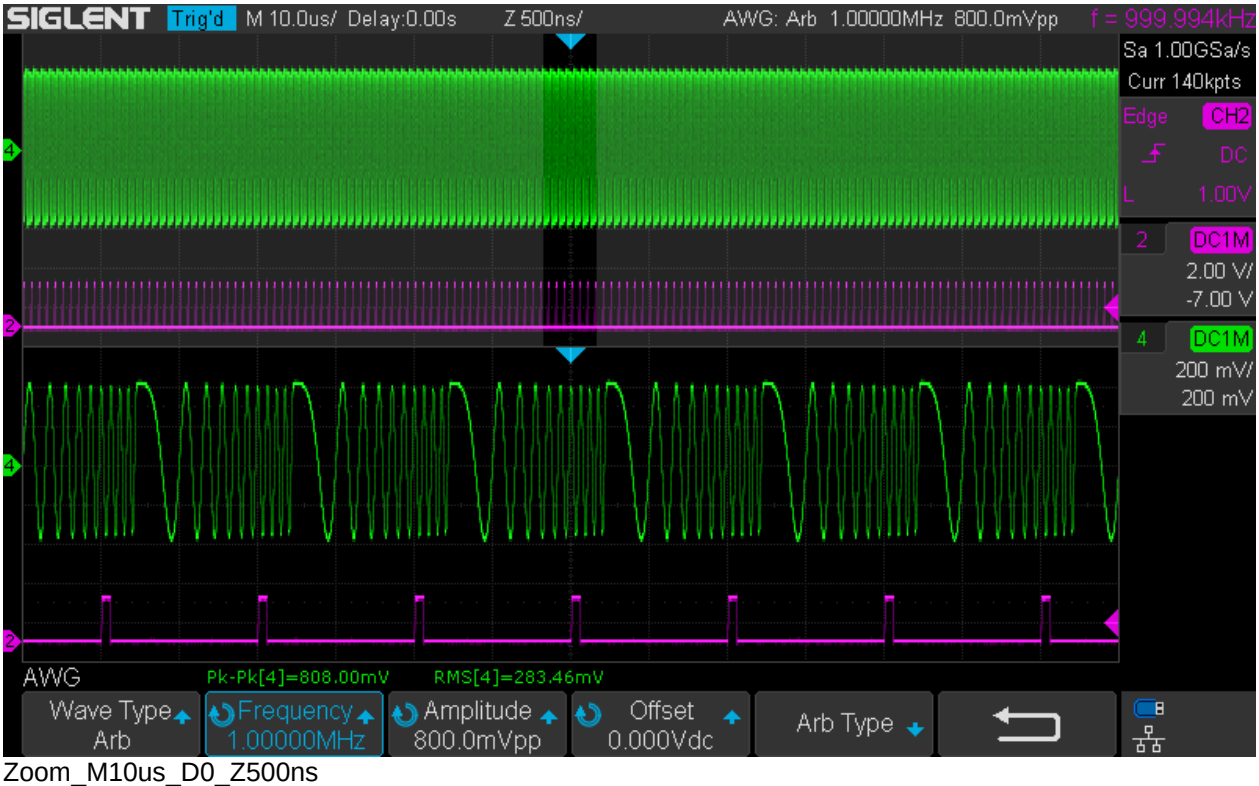
Horizontal zoom is an important feature, particularly for deep memory scopes. There are two different methods to accomplish this: Zoom Mode as well as changing timebase in stop mode after acquisition has been completed.

Zoom Mode

Zoom Mode allows the observation of signal details at high time resolution while still keeping an eye on the entire acquisition period at the same time. This works real time in run mode.

As an example, we can take a look at the Chirp signal at 1MHz from the optional SAG1021 AWG fed into channel 4. This is a complex signal that cannot be properly displayed using a simple edge trigger, so for this demonstration the sync signal from the SAG1021 is fed into channel 2 and the scope triggers on that. Of course there are certain means as well as advanced triggers that would allow us to trigger from the main signal, but that will be dealt with later in a separate chapter.

The following screenshot shows the full 140kpts record at 10µs/div in the (upper) main window and the 500ns/div zoom window below. Now we can see the entire record, spotting any interesting events or anomalies, while inspecting signal details at the same time. Of course we can move the zoom window to any position we like within the record.

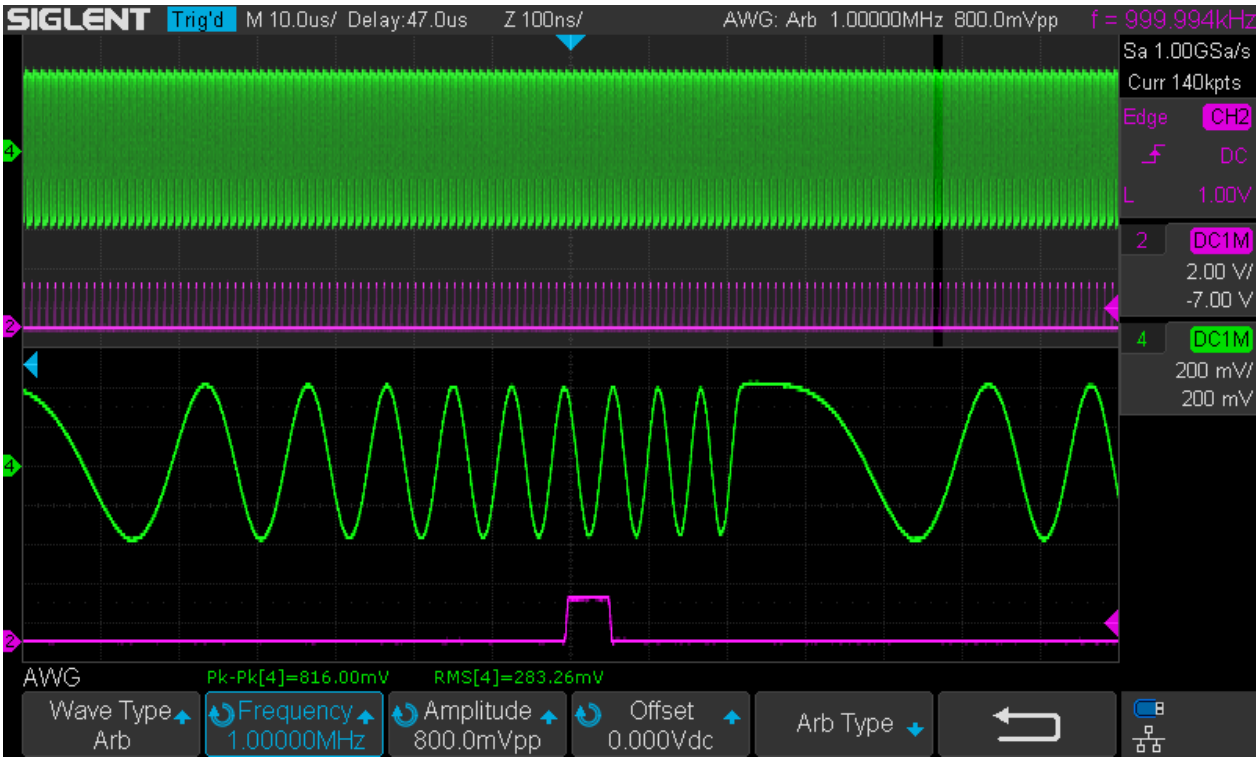


Zoom_M10us_D0_Z500ns

Positioning the zoom window is normally done by adjusting the horizontal position control, which can become tedious very quickly, especially with high zoom factors. The solution is to start with a lower zoom factor, center the zoom window upon the event of interest and then zoom in further as needed.

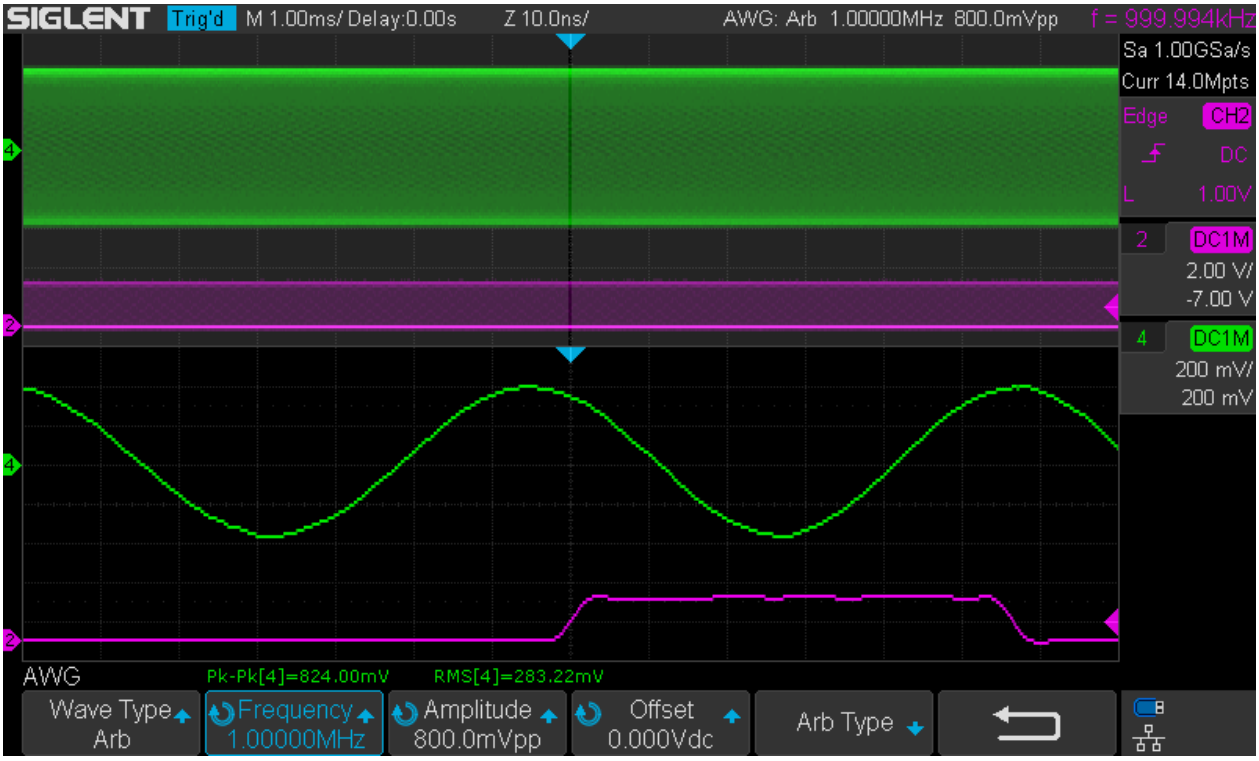
This scope also provides a group of dedicated buttons for navigation, which make life a lot easier whenever a certain portion of a long list or long record is to be inspected. This is particularly the case with

the time axis, where we can zoom in up to 1:1000000, i.e. would have to scroll through 1 million screen widths to get from one end of the record to the other.



Zoom_M10us_D0_Z100ns_D47us

Finally there is a screenshot showing a 1:100000 zoom ratio, i.e. a 10ns/div zoom window in a 1ms/div record. This is rather extreme already.



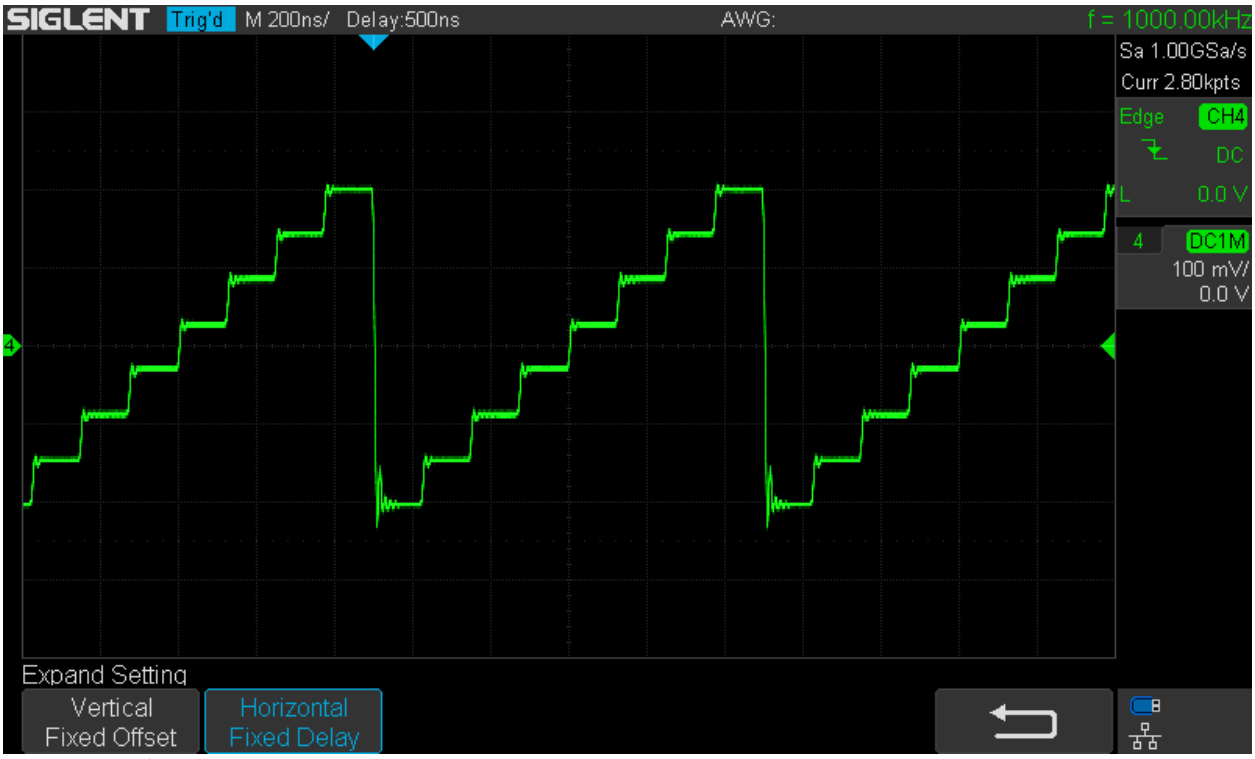
Zoom_M1ms_D0_Z10ns

Changing Timebase in Stop Mode

We can stop acquisition anytime by putting the scope into stop mode and then inspect signal details at higher time resolution by simply selecting a faster timebase. This is easy as long as we are interested in the signal details at the trigger point and the trigger position is at the center of the screen. In that case, we can zoom in and out by just changing the timebase.

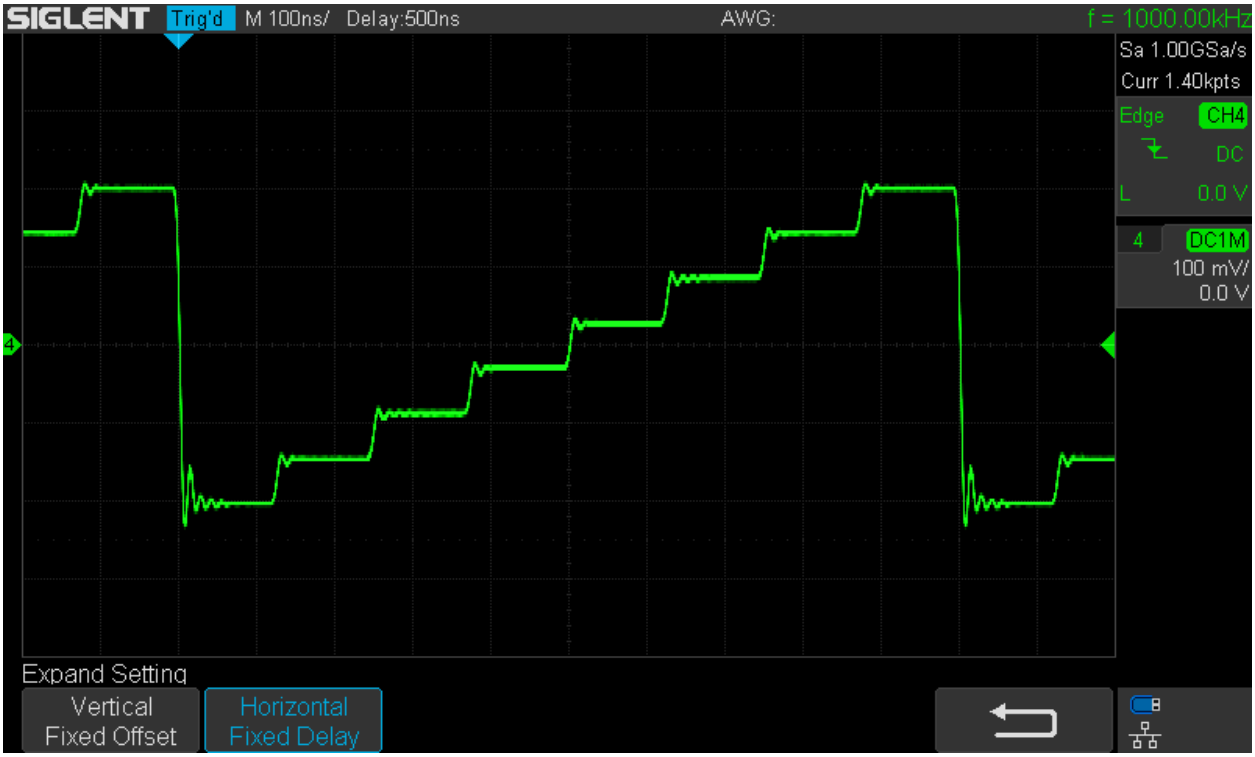
Sometimes we need to look at a signal detail at a point of the trace that has a certain delay with regard to the trigger point. By zooming in (lowering the timebase) the point of interest will eventually move out of the visible screen area. This is easy to handle though, as we just need to adjust the horizontal position control in order to bring the signal detail to the center of the screen.

In the following example we are interested in the rising edge of the stair step in the middle of the waveform, but are triggering on the falling edge. Of course we could easily trigger on the desired rising edge directly, so this is just to demonstrate a situation as described above, where we set a certain delay (500ns in this example) to bring the interesting part to the center of the screen width.



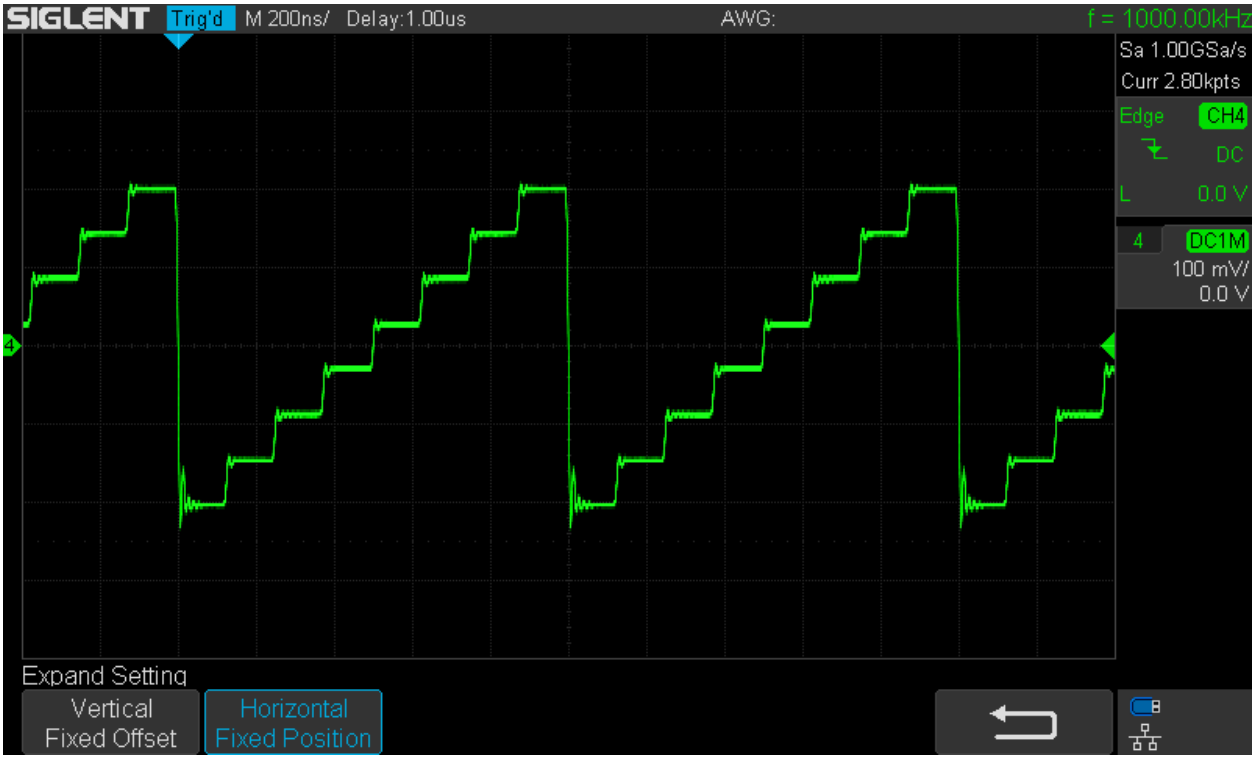
HPos_0_Delay+500ns_M200ns_FD

Now we can once again zoom in and out by just altering the timebase and the signal detail of interest will stay at the center of the screen. This is called Constant Delay mode and is the default behavior.



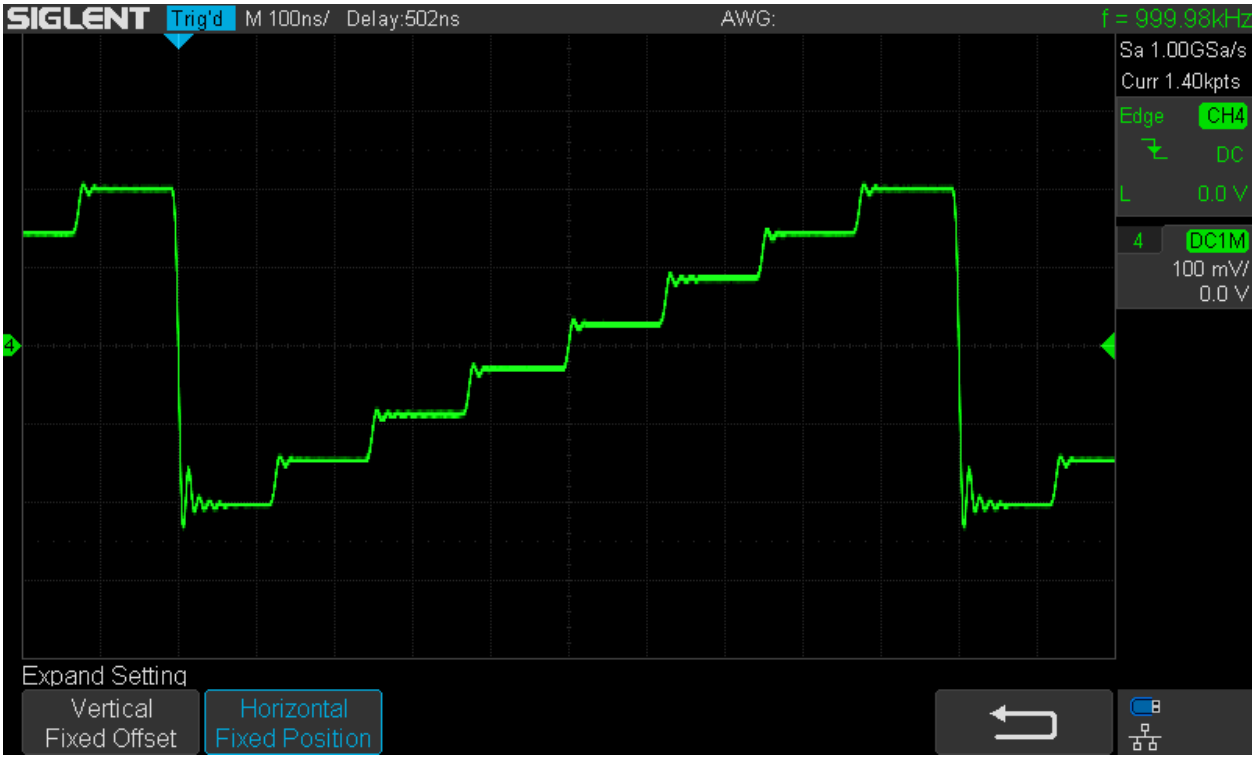
HPos_0_Delay+500ns_M100ns_FD

What if for some reason we want the trigger reference position not to be at the center, but at e.g. 10% of the screen width, because it's mainly the portion after the trigger that we are interested in? In default mode, the trigger position changes together with the timebase to maintain a constant delay. To deal with that scenario, we have to define a fixed reference position instead.



HPos_-5div_Delay+5div_M200ns_FP

Again we can zoom in/out by altering the timebase and the trigger position will stay put.



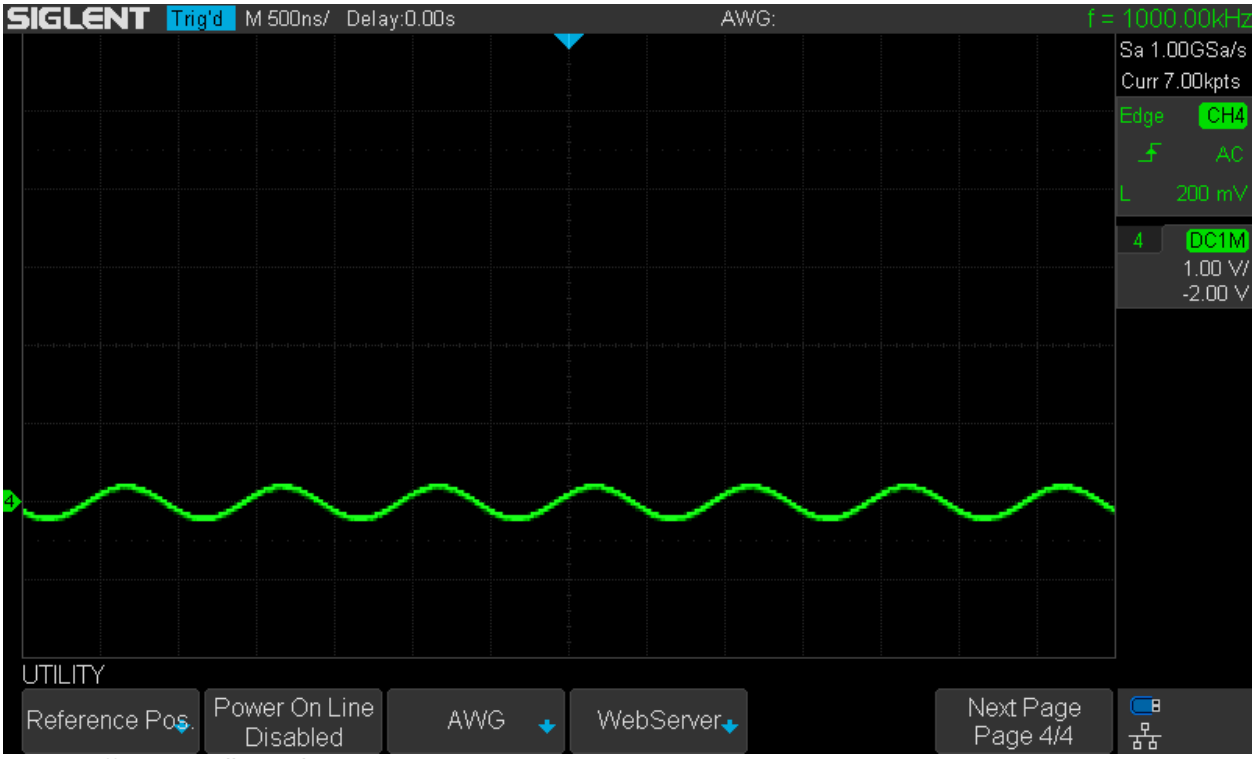
HPos_-5div_Delay+5div_M100ns_FP

We still have no means to deal with scenarios where we want to look at a signal detail at a certain delay with regard to the trigger position *and* the trigger position not being at the center of the screen. I've already submitted a change request and got the confirmation that this will be implemented in a future update.

Vertical Zoom

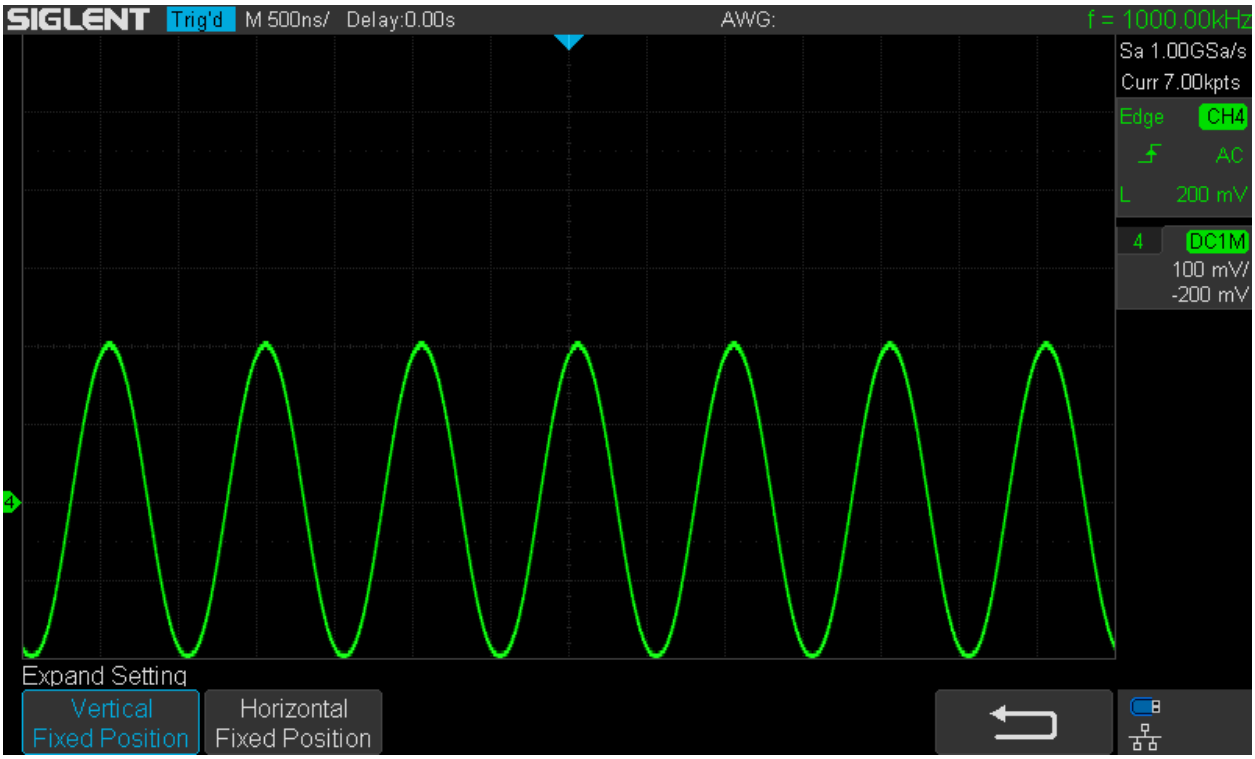
Vertical zoom may be required for closer inspection of signal amplitude details. It can be done by altering vertical gain/sensitivity in run as well as stop mode. This is easy as long as we are interested in the signal details at the vertical trace position, which can be set at any height within the screen. In that case, we can zoom in and out by just changing the vertical sensitivity. This is called Constant Position mode and is the default behavior.

The example below shows a 400mVpp sine at the gain of 1V/div and at a position of -2 divisions, set by the position control in constant position mode.



VPos_Off0_Pos-2div_Gain1V_FP

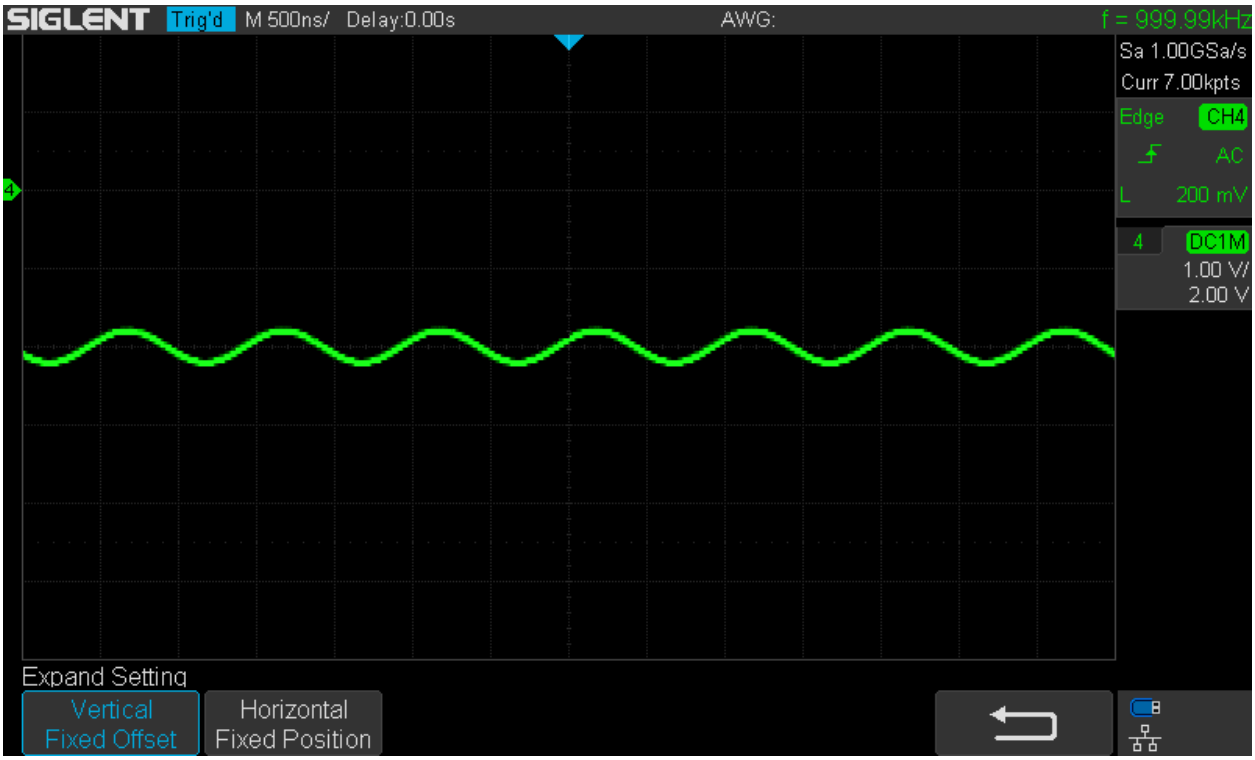
If we want to magnify the waveform, we can increase the vertical sensitivity to 100mV/div and everything works as expected: the trace position remains constant.



VPos_Off0_Pos-2div_Gain100mV_FP

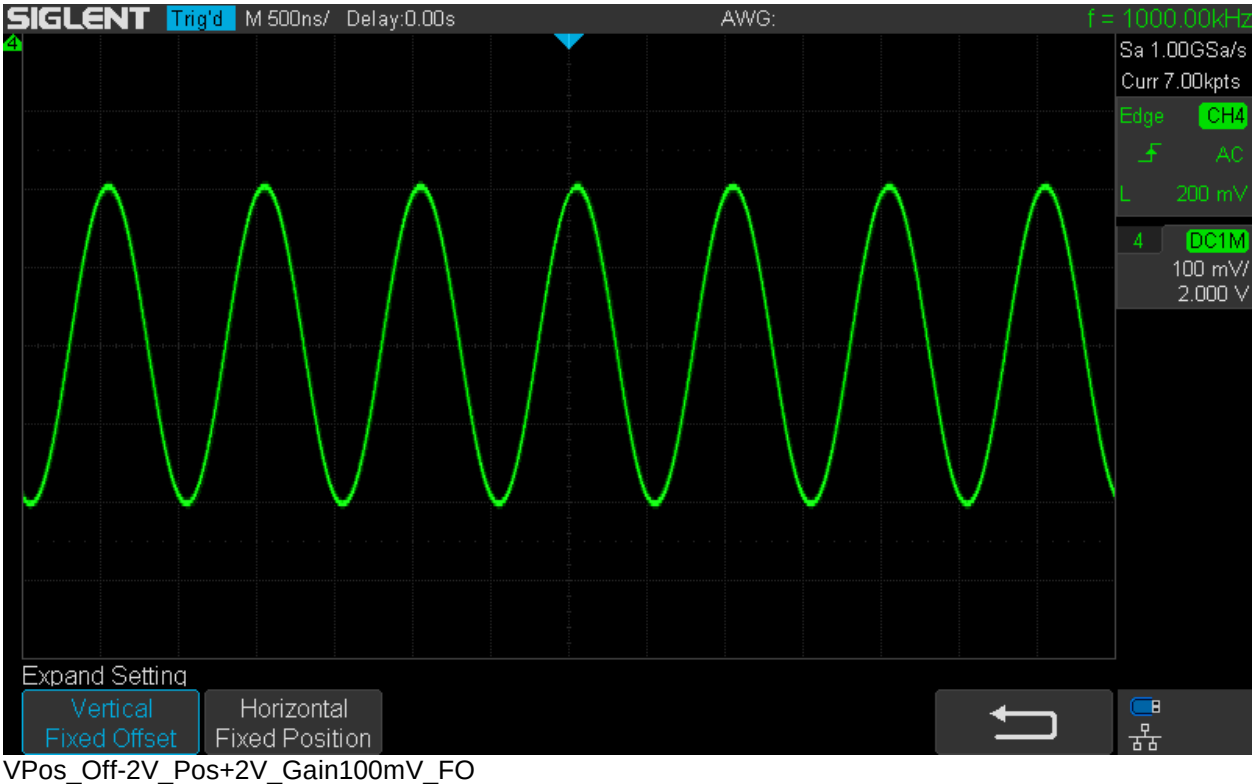
Sometimes we need to look at a signal detail at a point of the trace that has a certain offset with regard to the trace position. By zooming in (increasing the gain) the point of interest will eventually move out of the visible screen area. To avoid this, we need to change the “Expand Setting” strategy to Constant Offset

and then use the vertical position control to bring the signal detail to the center of the screen. In the example below, the signal has a -2V DC offset that is compensated with the channel position control by setting a +2V offset at 1V/div.



VPos_Off-2V_Pos+2V_Gain1V_FO

Once again we can zoom by changing the vertical gain and the trace position remains centered.



VPos_Off-2V_Pos+2V_Gain100mV_FO

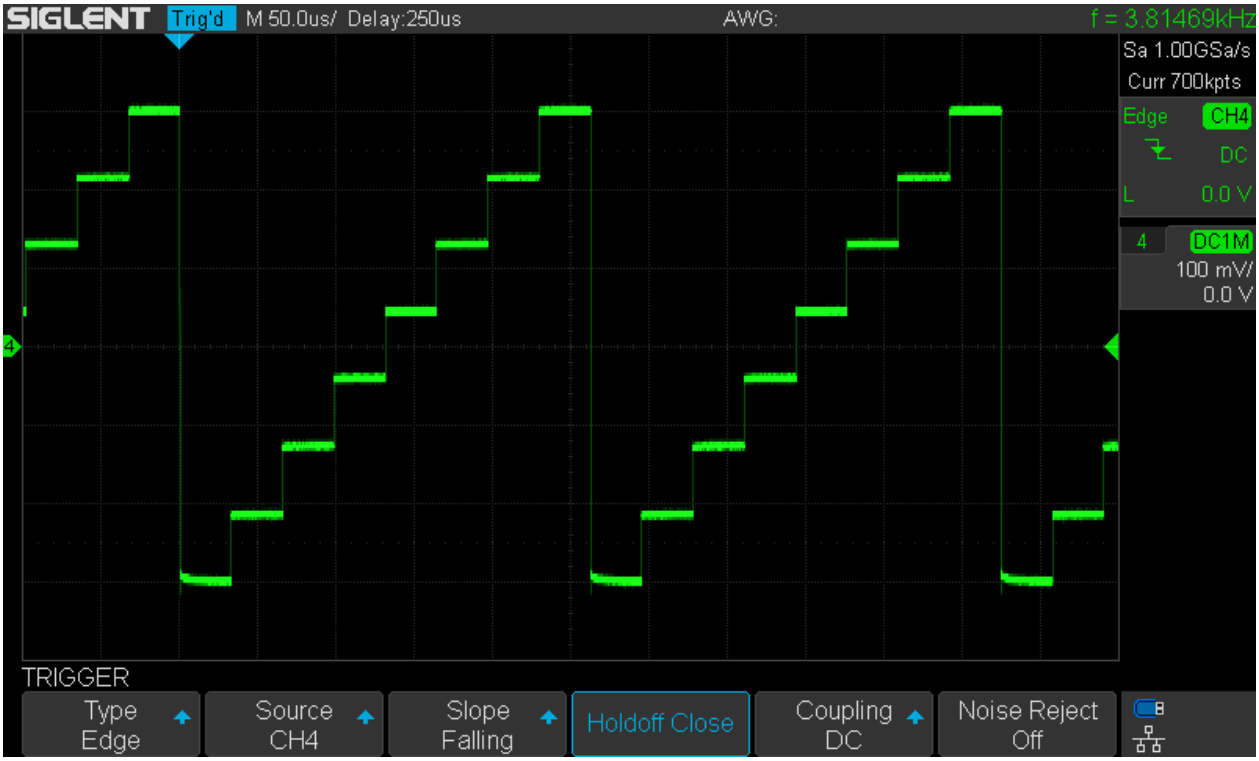
Of course this is again just a demonstration, as we could easily get rid of the DC offset by just changing the input coupling to AC, but it would be different with a unipolar pulse train for example.

We still have no means to deal with scenarios where we want to look at a signal detail at a certain offset with regard to the trace position *and* the trace position not being at the center of the screen. I've already submitted a change request and got the confirmation that this will be implemented in a future update.

Reference Waveforms

Sometimes we want to compare a certain waveform in a circuit before and after a tweak or under various conditions in general. This would be the main application for reference waveforms.

Here is a staircase waveform generated by an AWG in "Play Mode", which means that all sample points are output at max. sample speed. This causes a rather odd fixed output frequency, but at the same time it avoids the errors and artifacts introduced by the normal DDS process.

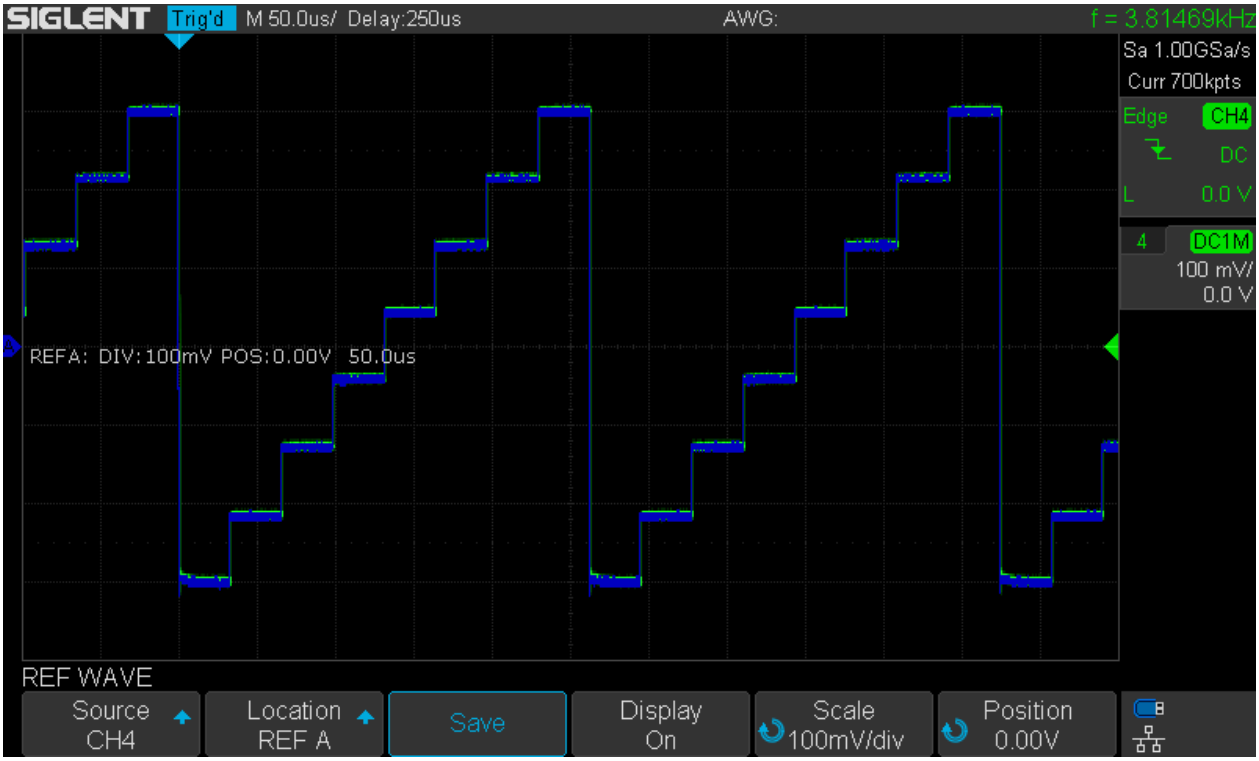


SDS1104X-E_REF_Stair_Orig

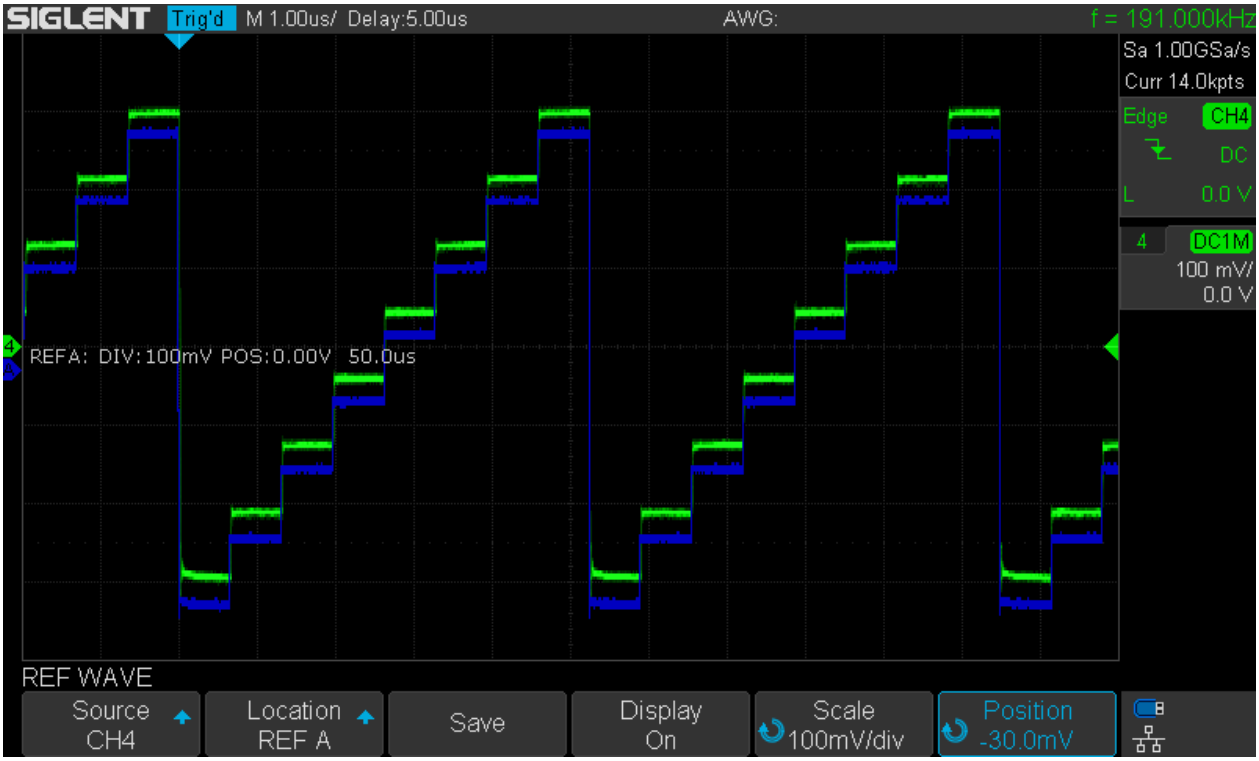
So this is the clean waveform that is ideally suited to act as a reference that could be compared to the same waveform output in DDS mode at a vastly different frequency.

Reference mode is enabled by pressing the [**Ref**] button on the front panel and then after selecting the appropriate channel and the reference location, the *Save* soft menu button can be pressed. This saves the current waveform on the selected channel to the selected internal location (REF A in this example). This internal reference memory is permanent, so the stored reference waveforms will be retained over a power cycle of the instrument.

The screenshot below shows the original waveform together with the reference on top.



SDS1104X-E_REF_Stair_Save



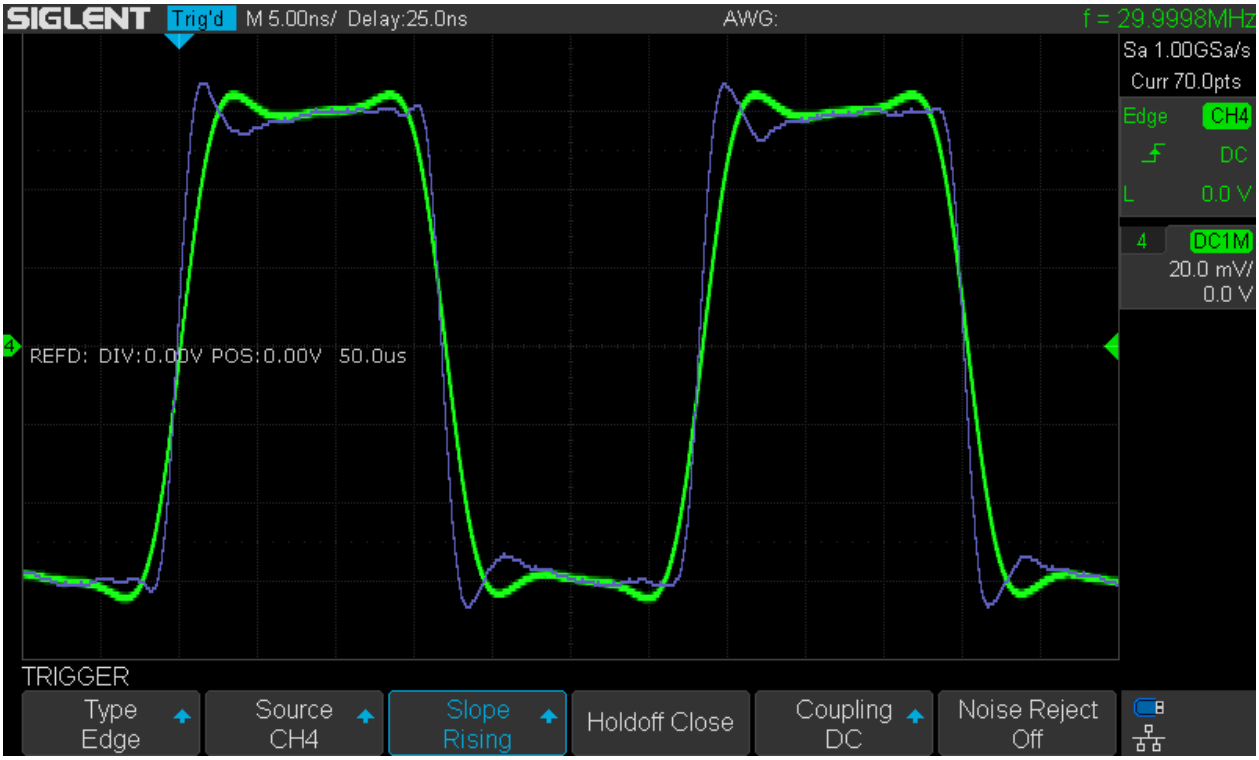
SDS1104X-E_REF_Stair_Compare

The screenshot above shows the reference waveform recalled from REF A together with the same waveform created in DDS mode at a much higher frequency than before. As expected, the differences are pretty much invisible.

The reference waveform has been shifted down a bit in order to make both waveforms clearly distinguishable. This also demonstrates the possibilities to adapt the position and scale of the reference waveform, so that offset and gain of a circuit can be compensated.

We can change the timebase as has been done in this example, but the reference waveform will always remain the same. In some rare cases it might be useful to change the timebase for the reference waveform, but this would require a re-sampling of the waveform, which is not going to give good results for a small dataset like this. The reference files are just a few kilobytes in size and high quality re-sampling would require to save the entire acquisition data, which might be up to 14Mpts (up to 140Mpts on a SDS2kX), which is too high a prize for satisfying a rare use case.

Finally, a practical example shows the difference between a 200MHz SDS1202X-E and the 100MHz 1104X-E when rendering a 30MHz squarewave with ultra fast edges (<100ps). The reference waveform has been saved to an USB stick on the SDS1202X-E and imported into REF D on the SDS1104X-E. There is a minor bug as the imported reference waveform shows an original timebase of 50µs, whereas 5ns would be correct of course. Other than that, the difference in bandwidth clearly shows; the reference waveform contains much more high frequency details. But then, for a fast signal like that, even 237MHz (the actual -3dB bandwidth of the SDS1202X-E) is not nearly enough to show the true waveform – but at least it's a lot closer to the truth. For most applications the exact waveform does not matter as long as the zero crossings and the amplitude are reasonably correct.



SDS1104X-E_REF_Square_30MHz_SDS1202X-E

Acquisition

Memory Depth

The SDS1104X-E offers 4 different memory depths.

14k, 140k, 1.4M and 14M points for interleaved mode, i.e. only one channel per channel group (channel groups are 1+2 and 3+4 respectively) is in use. In other words, this is any combination of (Ch.1 or Ch.2) and/or (Ch.3 or Ch.4).

7k, 70k, 700k and 7M points for individual mode, i.e. all channels can be used in parallel.

In order to capture fast signal details even at slow time bases, we normally want to use the maximum memory available and never touch that setting again. Yet there are some more advanced functions and special use cases, where we might indeed want to limit the memory available for a single acquisition, e.g. in Sequence mode and for segmented recording in general, as well as limiting the number of sample points for FFT or speeding up acquisition at slow timebase settings.

Acquisition Modes

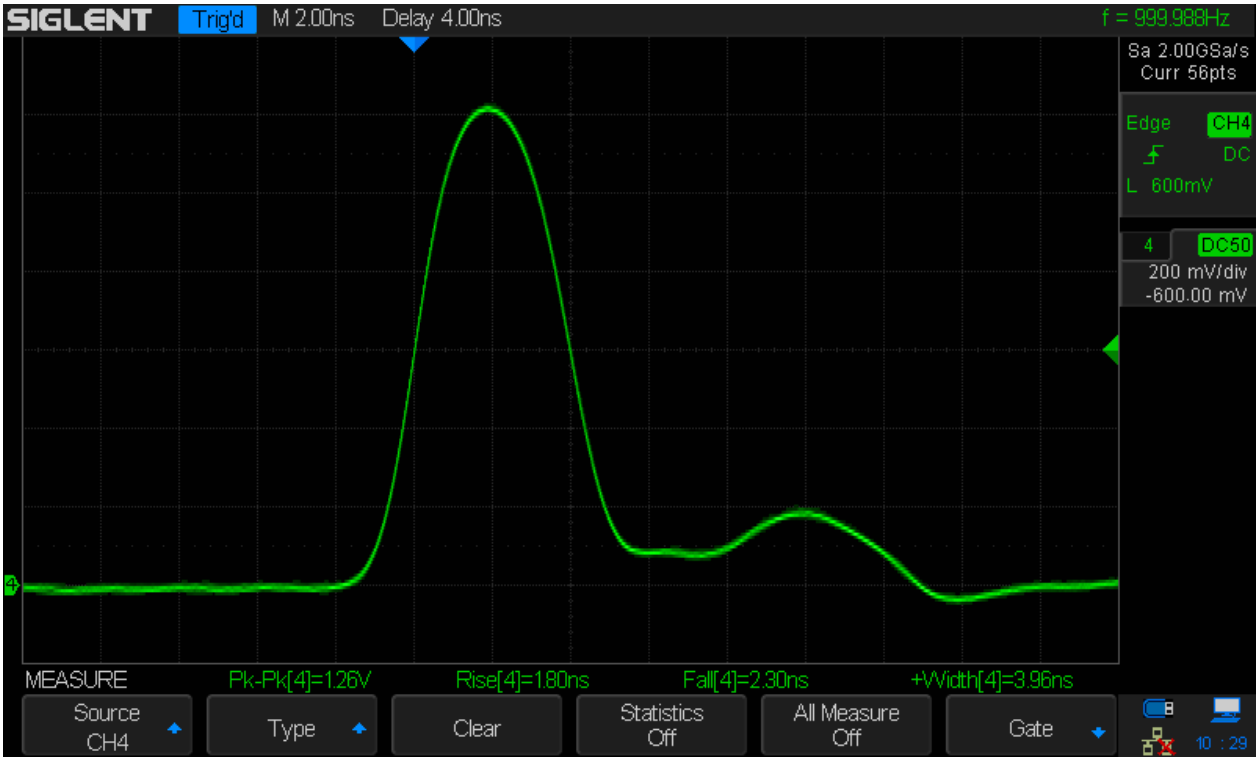
There are 4 dedicated basic acquisition modes, which determine the way captured data from the ADC is preprocessed (and decimated, if necessary). On top of that, roll mode also qualifies as acquisition mode, but could be looked at as an extension to normal and peak detect modes. It also has a dedicated physical button on the front panel.

Normal

As the name suggests, this is the most commonly used acquisition mode. ADC data is processed at full sample rate and stored in acquisition memory as long as there is enough space available. For 7/14Mpts memory depth, this is the case up to a 1ms/div. At slower timebase settings, sample data have to be decimated, that means just using only every n^{th} sample in order to virtually reduce the sample rate so the data can fit the available memory.

Normal mode provides a faithful reproduction of the analog input signal within the constraints of the effective sample rate, which is always displayed in the top right corner of the screen together with the currently used record length.

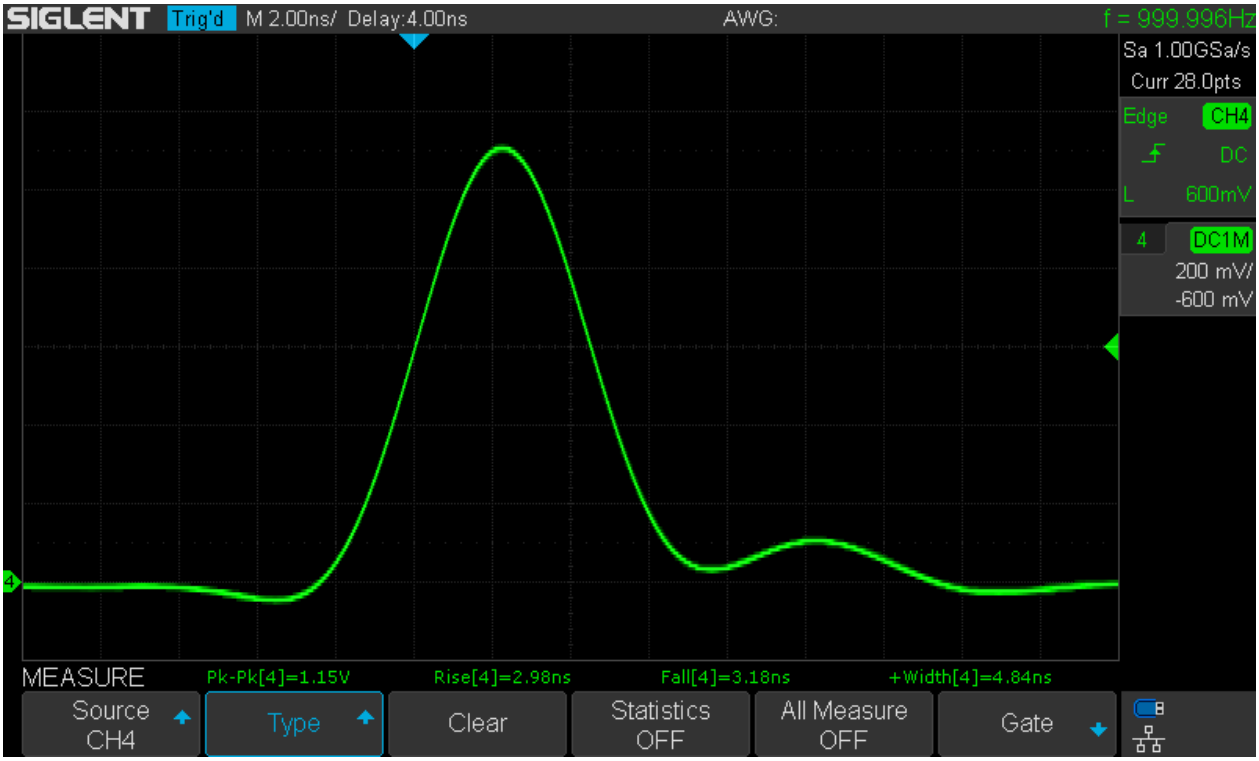
This is a good place to show the test pulse used for some of the subsequent tests – and at the same time, demonstrating the constraints of a 100MHz bandwidth. The test pulse is about 4ns wide and 1.2V in amplitude, and it is generated at a rate of 1kHz. This is how it looks on a Siglent SDS2304X, which happens to have a 3dB bandwidth of 375MHz at 200mV/div vertical sensitivity.



Pulse_1kHz_4ns_BW375MHz

The automatic measurement shows the amplitude a little too high because of the undershoot caused by a suppressed 2nd pulse – this is one of the many flaws of the pulse generator in use and should just be ignored here. Pulse width measurement should be fairly accurate and transition times are pretty fast – even though fall time also suffers from the before mentioned flaw.

Anyway, let's just keep these numbers in mind when we compare the results with the SDS1104X-E, which provides a 3dB bandwidth of 109MHz at 200mv/div:



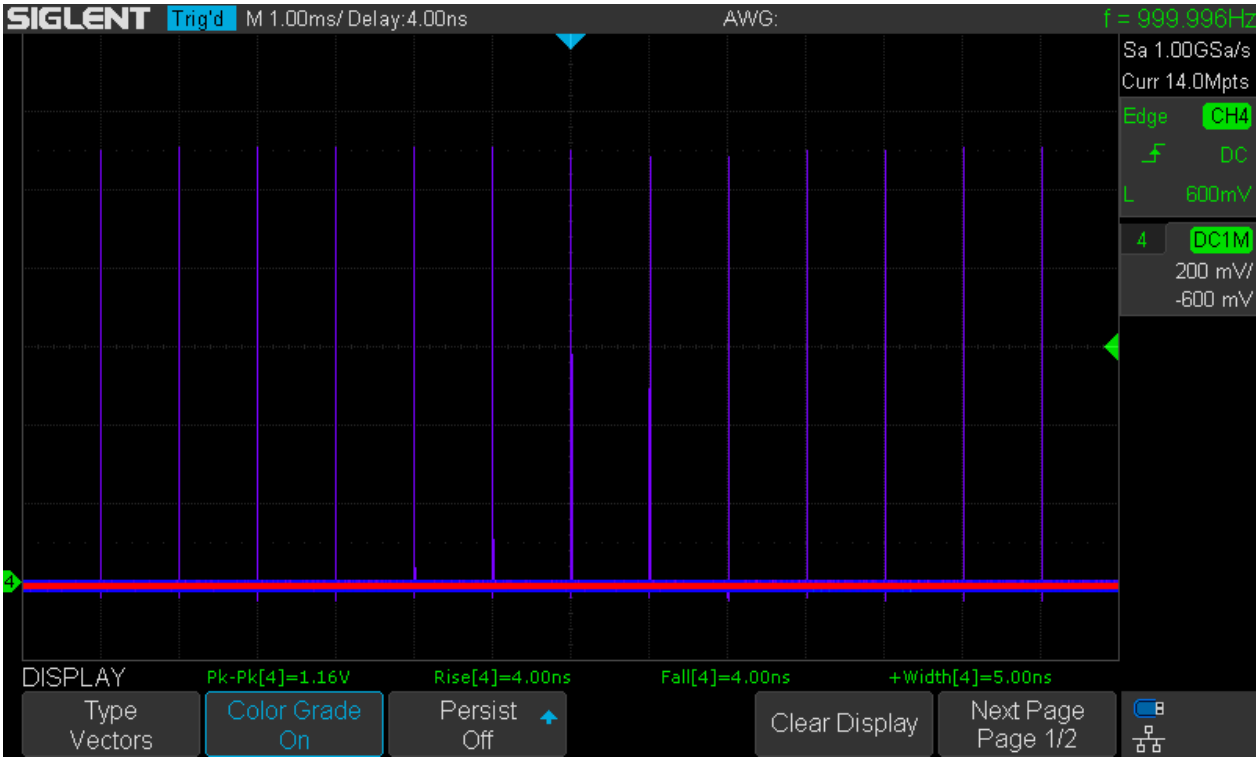
Pulse_1kHz_4ns_BW109MHz

As can be seen, the trace looks much smoother and the full amplitude is not reached – automatic measurements only sees 1.15Vpp. Unsurprisingly, transition times are about 0.9ns higher and also the pulse width is stretched by the same amount. Since the SDS1104X-E does not provide a 50Ω input impedance, an external pass-through termination is used for this and all following tests, where a direct connection from a signal source to the scope is made.

In any case, this is a fairly steep and narrow pulse and even though it exceeds the capability of a 100MHz scope to accurately characterize it, we're still able to use it for the following tests without problems.

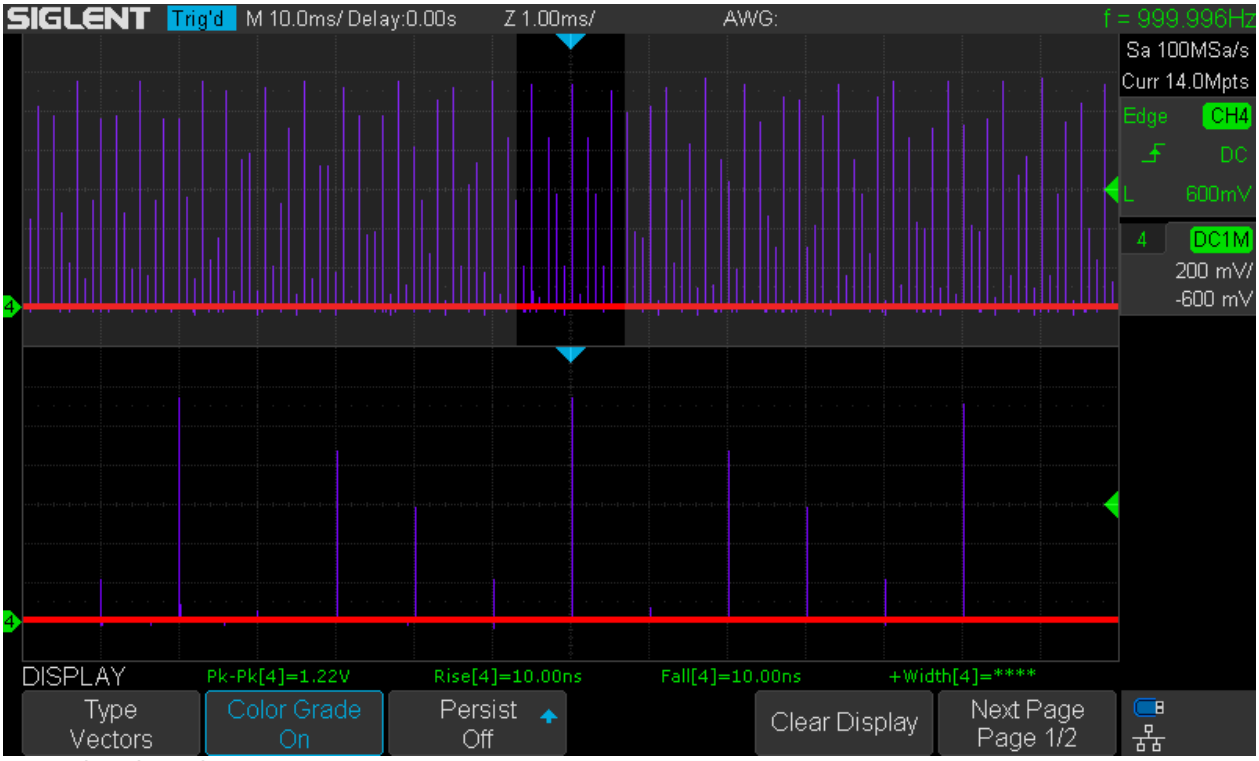
First we look at the pulse train at a timebase of 1ms/div. In this scenario, the full 14Mpts of acquisition memory are utilized and the full sample rate of 1GSa/s is still available. As expected, the pulses are captured without problems. Color graded display is used for all the pulse tests, as this provides a much better visibility in the screenshots.

Please note that the automatic measurements are fairly accurate (within the constraints of the limited bandwidth) and we get proper results in the realm of single digit nanoseconds even though the timebase is 1ms/div and the total record length covers a 14ms time interval.



Normal_Pulse_1kHz_4ns_1ms

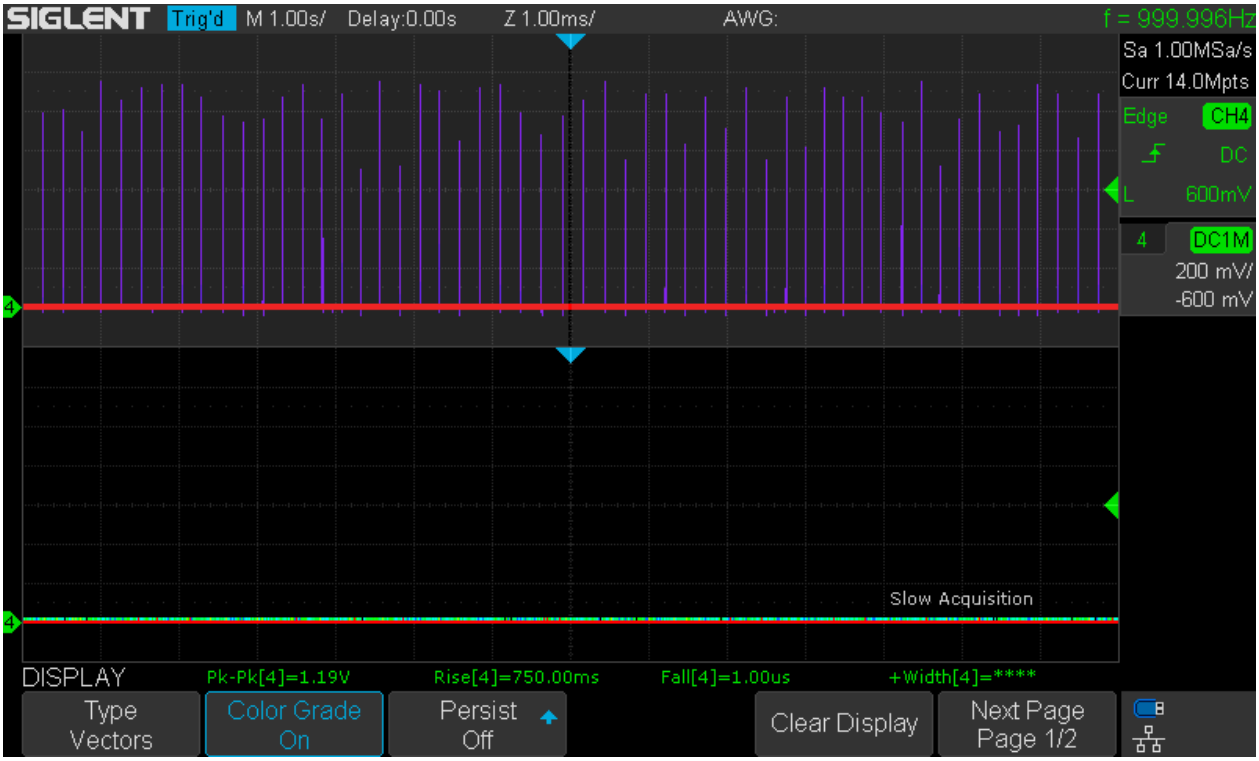
Lowering the timebase to 10ms/div reduces the sample rate to 100MSa/s. A zoom window of 1ms/div is used to provide a direct comparison with the first screenshot.



Normal_Pulse_1kHz_4ns_10ms

As a result, we get massive amplitude variations and it should be clear that sampling every 10ns will miss some of the 4ns wide pulses completely – and it actually shows in the zoom window.

Lowering the timebase even further to 1s/div reduces the sample rate to only 1MSa/s. Notice that the scope would automatically enter roll mode for timebase settings slower than 20ms/div, but we can easily bring it back to normal mode by pressing the [**Roll**] button so that it is not illuminated anymore, thus exiting roll mode.



Normal_Pulse_1kHz_4ns_1s

The main window consistently shows 4 pulses per second (where it should actually be 1000) and the amplitudes vary quite a bit. The zoom window doesn't show a single pulse – we don't even see the trigger event. How could that be? Well, the ADC and the digital trigger system always work at full speed, so trigger performance doesn't change. But the subsequent data decimation just throws away 99.9% of the samples (1MSa/s vs. 1GSA/s), hence the probability to catch the trigger event is only 0.1%.

Peak Detect

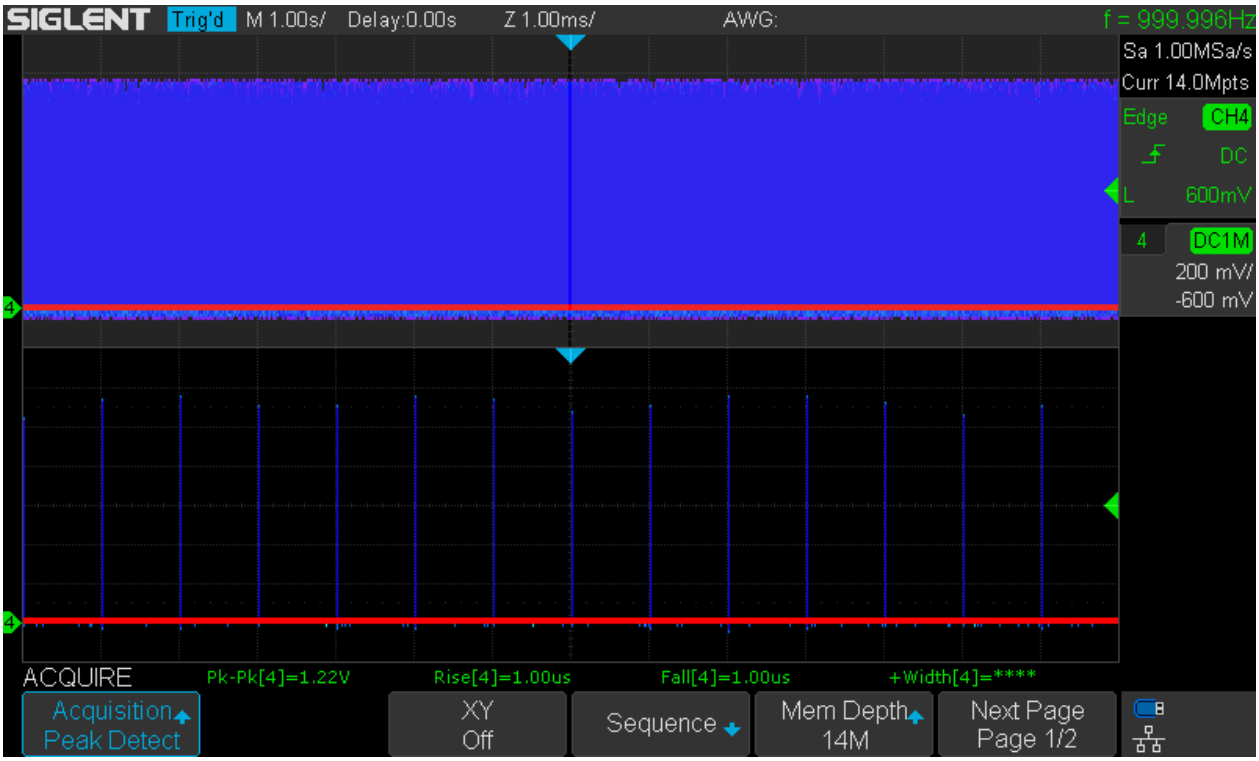
Sample rate drops at slow time bases, which raises the risk of losing important signal details, like spikes and glitches. Peak detect mode uses a different strategy for data decimation: instead of using every n^{th} sample from the ADC stream, one min/max pair out of *2n* samples is stored. This way we cannot miss a transition; on the other hand we don't get a faithful reproduction of the input signal anymore.

In actual fact, Peak Detect mode is nothing more than a crouch that no one would miss in a DSO with sufficient acquisition memory. Unfortunately, the memory size requirements go through the roof pretty quickly, so this isn't a realistic option for the time being:

The SDS1x04X-E provides 14Mpts for 1GSa/s, thus full sample rate up to 1ms/div
The SDS2000X provides 140Mpts for 2GSa/s, thus full sample rate up to 5ms/div

The SDS1x04X-E has 100s/div as the slowest timebase setting – this would require 1.4Tpts of acquisition memory to retain the full sample rate and make peak detect superfluous.

Now let's repeat the 1s/div test, where normal mode has failed completely. Once again, horizontal timebase is 1s/div and swept mode is used instead of roll mode.



Peak_Pulse_1kHz_4ns_1s

Now this is something completely different. Even at a sample speed of only 1MSa/s, the main window is completely filled with pulses and the zoom window shows that all the pulses are indeed captured. The amplitude measurement is fairly accurate, but transition times are far off and the system cannot determine a pulse width at all. Why is it so?

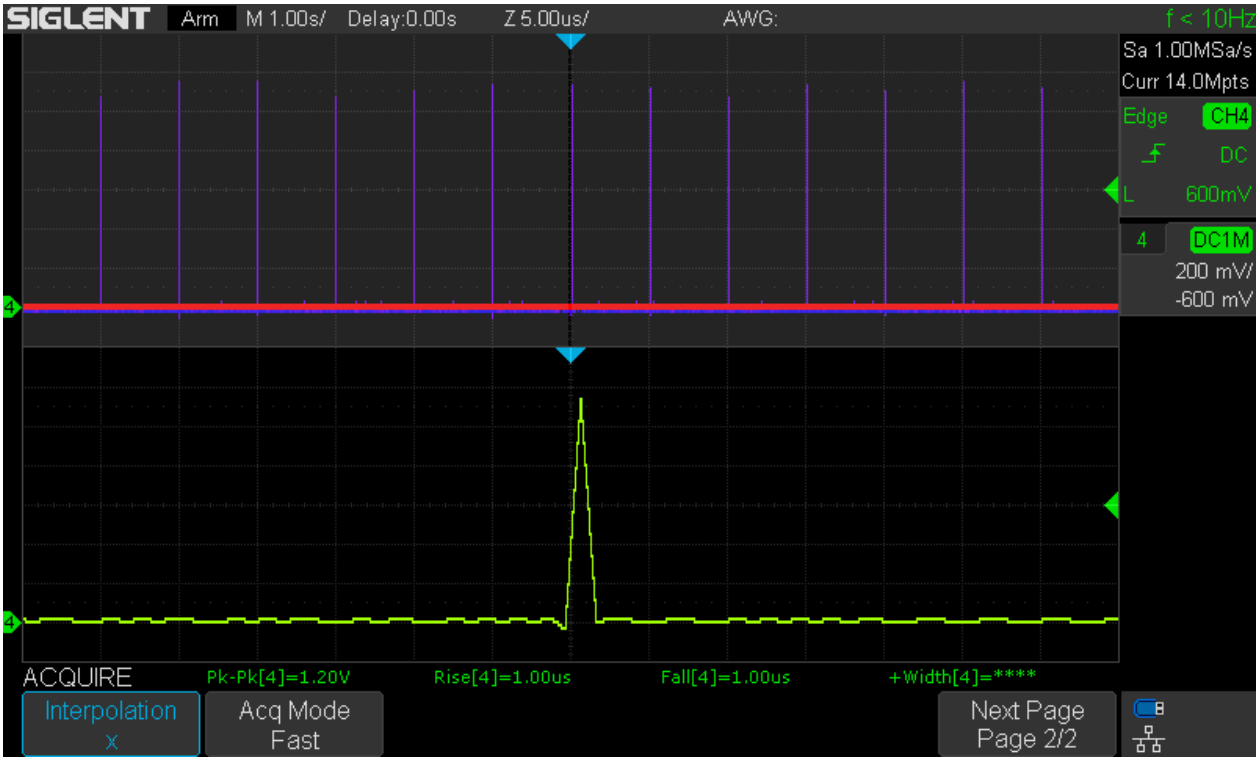
As already stated earlier, peak detect mode doesn't provide a true representation of the input signal. Apart from the measurements, this wasn't obvious yet in the screenshot above, but it gets immediately revealed if we zoom in a little further.

At 5µs/div, we can clearly see the Sinc pulse resulting from the sin(x)/x reconstruction of the severely undersampled and misaligned data produced by the peak detect decimation scheme. According to what we can see in the zoom window, the pulse would indeed have 1µs transition times, and the pulse width would be somewhere around 1µs as well. Even when it's totally wrong, why can't the scope measure and display the pulse width?

The answer gets immediately clear when looking at the next screenshot, where sin(x)/x reconstruction has been turned off and simple x interpolation is used. We can see that the peak detect data doesn't actually define a pulse, but just a triangular spike, which cannot have a pulse width by definition.

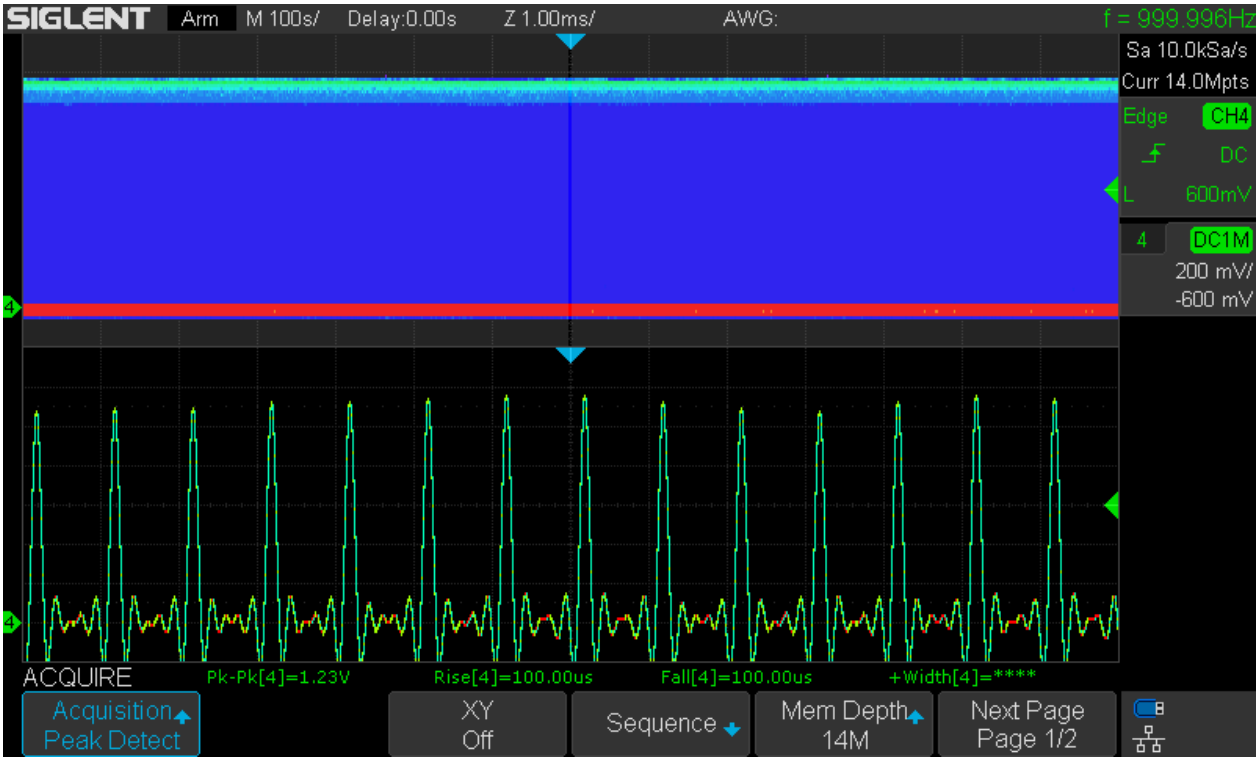
It is important to understand such pitfalls and always get suspicious when the scope seems to fail – it might just be the test scenario in itself being invalid. The moral of the story, never trust any pulse measurements when in peak detect mode – at least not for measurement results close to the actual effective sample speed. For the example here, it's not by accident that the measurements read 1µs; this simply happens because the effective sample rate is 1MSa/s, which means 1µs sample interval and no measurement can provide a better resolution than that.

Peak_Pulse_1kHz_4ns_1s_Z5us



Peak_Pulse_1kHz_4ns_1s_Z5us_x

Finally we take it to the extremes and try a timebase of 100s/div – this is the maximum setting for the SDS1104X-E. Time for an extensive coffee break…



Peak_Pulse_1kHz_4ns_100s

It works – just. We still get a pulse every millisecond, but the waveform distortion is already visible without the need to zoom in any further. At a sample speed of only 10kSa/s, the pulses cannot be narrower than 200µs at their base. If the repetition rate of the pulses were a little higher, then even the fundamental frequency of the pulse train would exceed half the sample rate, thus violating the Nyquist criterion. As a result, we would get an aliased signal. Nevertheless the main window wouldn't look any different and hint us on the heavy signal activity going on.

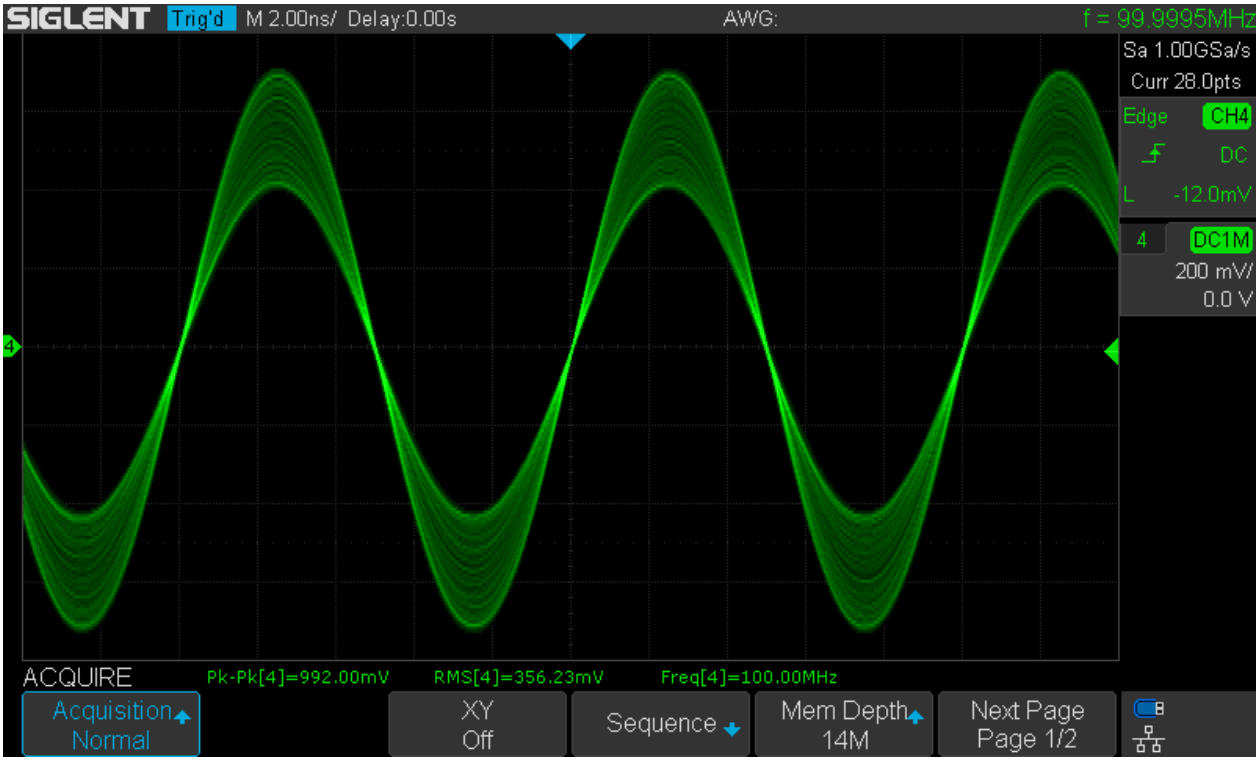
CAUTION: As it just fits the topic, a warning appears in place here. Never ever use peak detect together with FFT mode. Peak detect mode violates a fundamental requirement of digital signal acquisition, it is the evenly spaced samples with regard to the time axis. This produces some artificial signal and the FFT of this does not reflect the reality anymore.

Average

Average mode provides a gliding average over a number of subsequent acquisitions (records). The SDS1104X-E provides a choice of 4, 16, 32, 64, 128, 256, 512 and 1024x averaging. This will not limit the frequency response of the scope, but act as a high-pass filter to any signal modulation. We could also say that it irons out any signal variations from one acquisition to the next, which also includes noise.

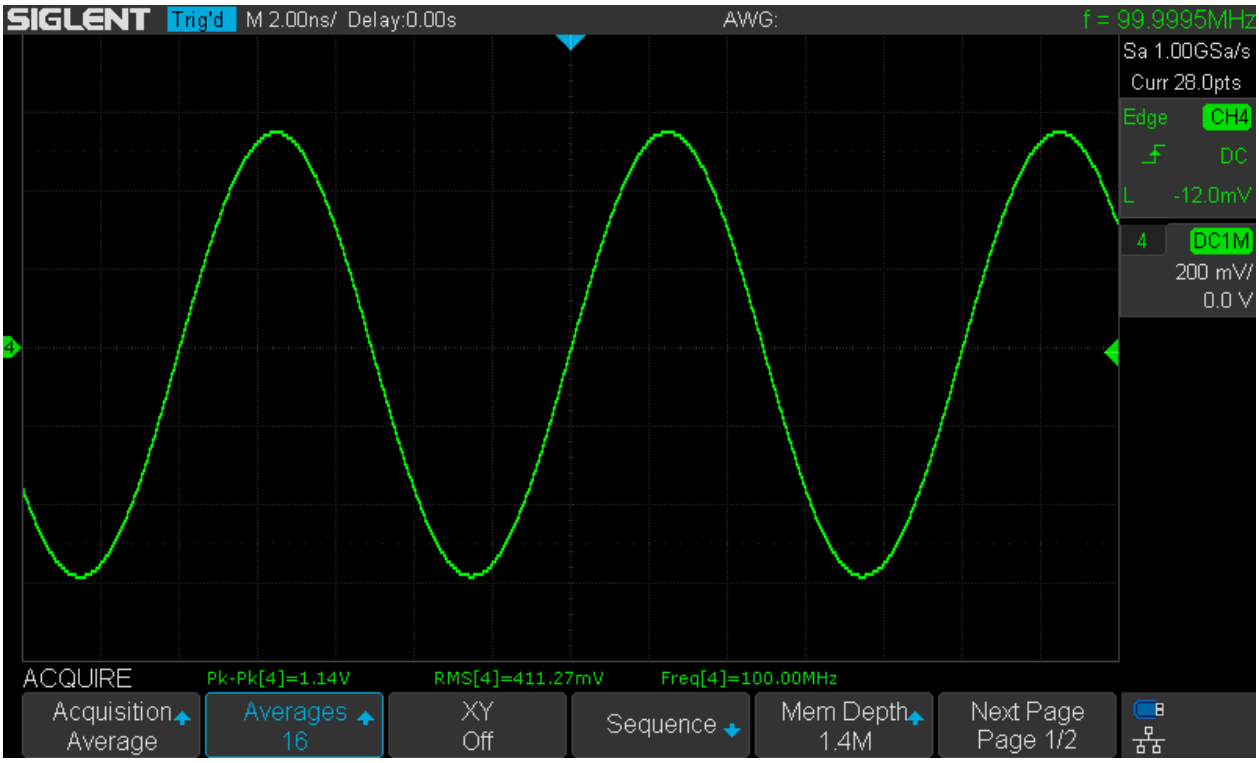
Consequently, average mode is a great means to clean up noisy static signals without limiting the acquisition bandwidth. Since it requires a lot of memory and processing power, the maximum acquisition length is limited to 1.4Mpts (interleaved mode, 700kpts with all 4 channels in use).

Let's have a look at a 100MHz carrier 25% amplitude modulated with a 400Hz sine in normal acquisition mode, shown in the screenshot below. The amplitude measurements are more or less meaningless, as they are changing all the time at high speed, according to the modulation.
The actual carrier level is 500mVrms, but the scope will always show less as we're approaching its bandwidth limit.



AVG_100MHz_AM25%_400Hz_normal

The next screenshot shows the very same signal, but this time in average mode with 16 averages selected. Amplitude variation is only a few pixels now and we also get reasonable measurements.

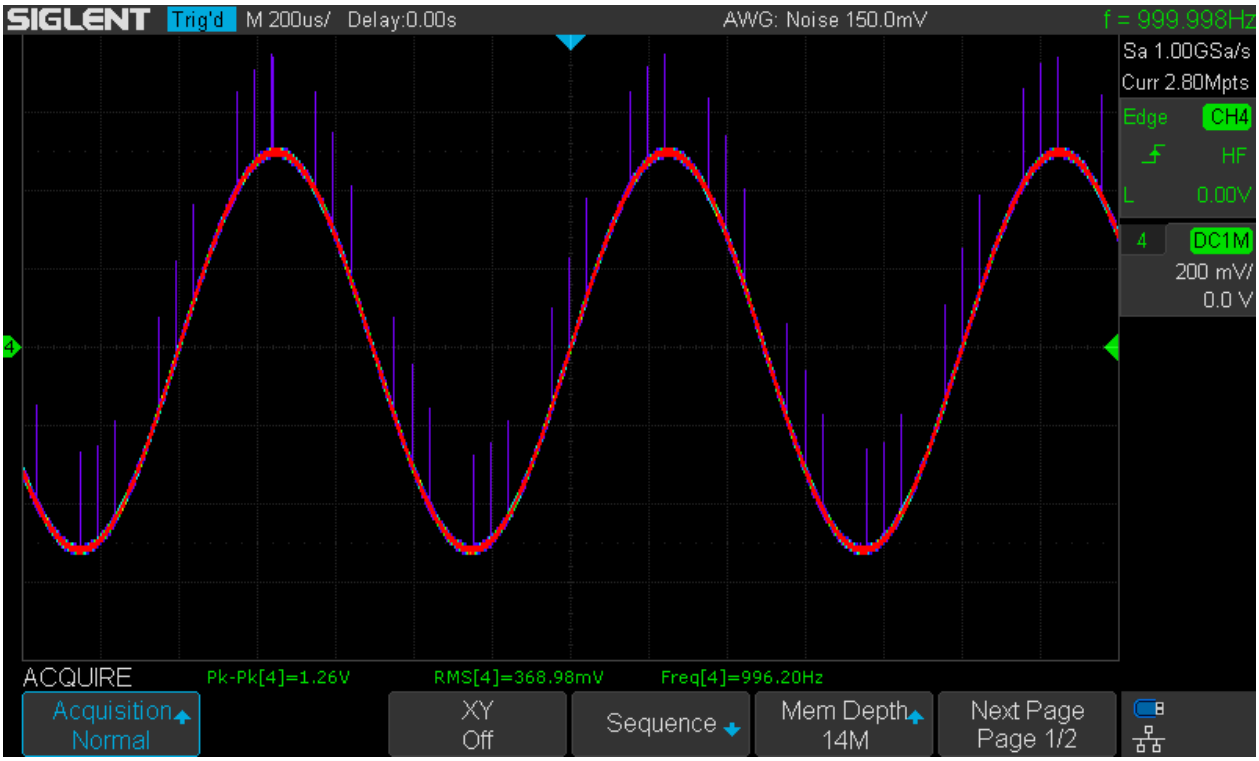


AVG_100MHz_AM25%_400Hz_avg16

With the maximum of 1024 averages, we get an absolutely static signal, but no point in showing a screenshot here, as it would look exactly the same and this is also true for the measurements.

Now let's examine the averaging effect on noise – we're using spikes here. It is important that the noise must not be related to (being in sync with) the signal, otherwise it would be treated as a part of the signal and not be removed by averaging.

In this example, a 1kHz sine is used with 20ns pulses superimposed at a repetition rate of 4981Hz, so they occur at different spots of the signal in each acquisition. Color graded display mode is used once again to aid visibility in the screenshots.



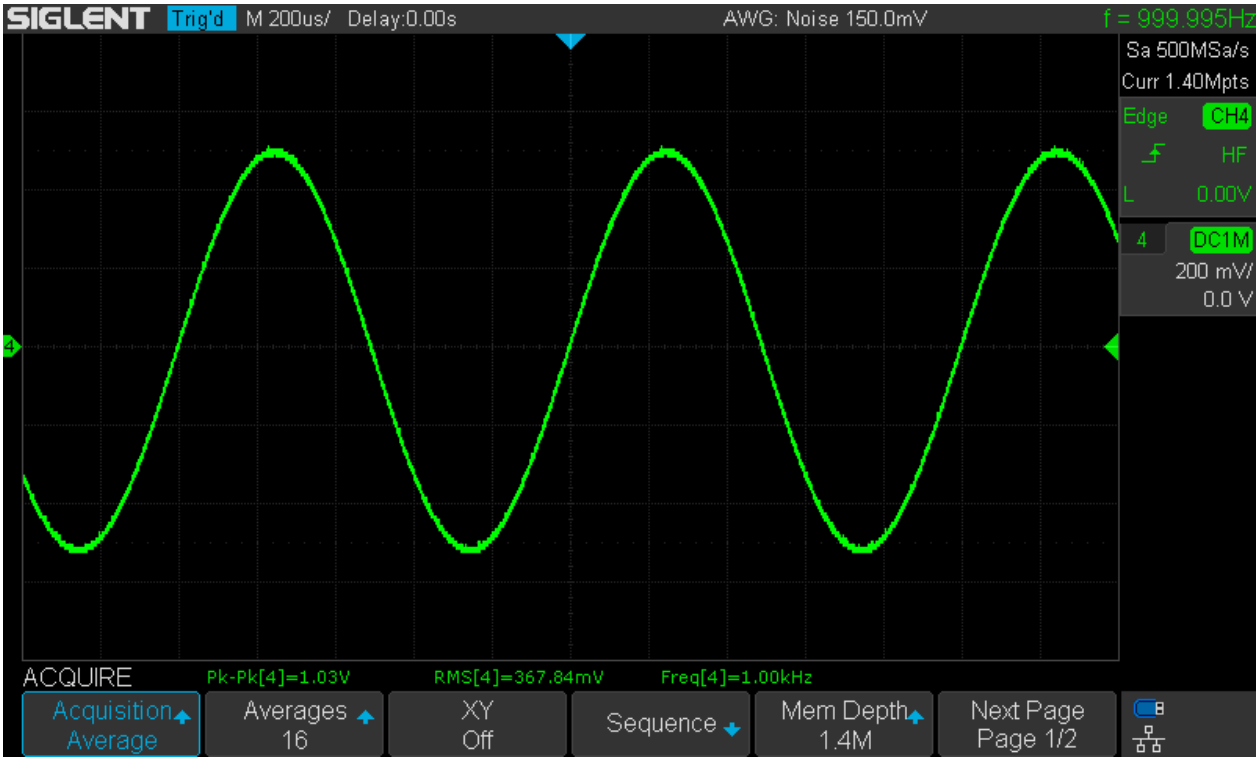
AVG_1kHz_Sp4981Hz_normal

Now turning 16x averaging on, the spikes disappear completely.

We have a standard display now, because a filter mode like average does neither support color nor intensity grading. This is not a big surprise, as there is no scan speed/frequency information available, only the result of the (mathematical) signal processing.

It is worth noticing that triggering on noisy signals isn't straight forward and we won't get stable triggering with the default settings. Fortunately, the SDS1104X-E provides plenty means to deal with situations like this, which will be discussed in more detail in a later section.

Hint: for this example, simply HF-reject edge trigger was used, as can also be seen in the screenshots.



AVG_1kHz_Sp4981Hz_avg16

Eres

Eres stands for “Extended Resolution” and is some specific kind of FIR filter applied to the sample data of each individual record. Up to 3 bits resolution enhancement in steps of 0.5 is available. This will limit the signal bandwidth, but has no direct impact on modulation other than possibly cutting the upper modulation sideband. We could also say, Eres acts as a low-pass filter with somewhat obscure parameters. Therefore, I don’t particularly like this mode, yet some investigations cannot hurt.

The table below shows the measured bandwidth of the SDS1104X-E at all available Eres bit settings for a sample rate of 1GSa/s in interleaved mode (only one channel per channel group enabled).

Siglent SDS1104X-E			
SR = 1GSa/s	1 dB	3 dB	6 dB
Eres Bits	BW [MHz]	BW [MHz]	BW [MHz]
0,5	64,00	102,00	135,00
1	49,00	83,00	111,00
1,5	29,00	51,00	70,00
2	15,00	27,00	36,00
2,5	8,00	13,60	18,70
3	4,00	6,90	9,40

Measured Bandwidth vs Eres bits interleaved

It is obvious that the input bandwidth of the SDS1104X-E is the limiting factor here, so any bandwidth results >70MHz will most likely not be valid for e.g. the SDS1204X-E.

A 2nd series of measurements shows the Eres bandwidth for 500MSa/s in individual mode (both channels in a group enabled), see the table below.

Siglent SDS1104X-E			
SR = 500MSa/s	1 dB	3 dB	6 dB
Eres Bits	BW [MHz]	BW [MHz]	BW [MHz]
0,5	34,00	87,00	137,00
1	30,00	64,00	108,00
1,5	22,60	43,10	63,30
2	13,60	24,90	35,00
2,5	7,40	13,10	18,20
3	3,80	6,70	9,20

Measured Bandwidth vs Eres bits individual

Surprisingly, the upper bandwidth limit of this Eres implementation doesn't seem to be closely related to the sample rate resulting from the channel configuration (individual/interleaved). But it is still proportional to the effective sample rate resulting from timebase and memory depth changes, as is shown in the table below for interleaved channel configuration:

Siglent SDS1104X-E -3dB BW Eres Interleaved				
SR [Sa/s]	1,00E+9	500,00E+6	250,00E+6	100,00E+6
Eres Bits	[Hz]	[Hz]	[Hz]	[Hz]
0,5	108,00E+6	98,00E+6	62,00E+6	25,40E+6
1	87,00E+6	57,00E+6	28,50E+6	11,40E+6
1,5	53,00E+6	28,00E+6	14,00E+6	5,58E+6
2	27,30E+6	13,90E+6	6,90E+6	2,77E+6
2,5	13,90E+6	6,90E+6	3,45E+6	13,90E+6
3	6,93E+6	3,45E+6	1,73E+6	680,00E+3

Measured Eres Bandwidth vs Samplerate

In LeCroy scopes, Eres bandwidth can be calculated by multiplying the Nyquist frequency (half the effective sample rate) by a factor specific for the resolution enhancement in bits:

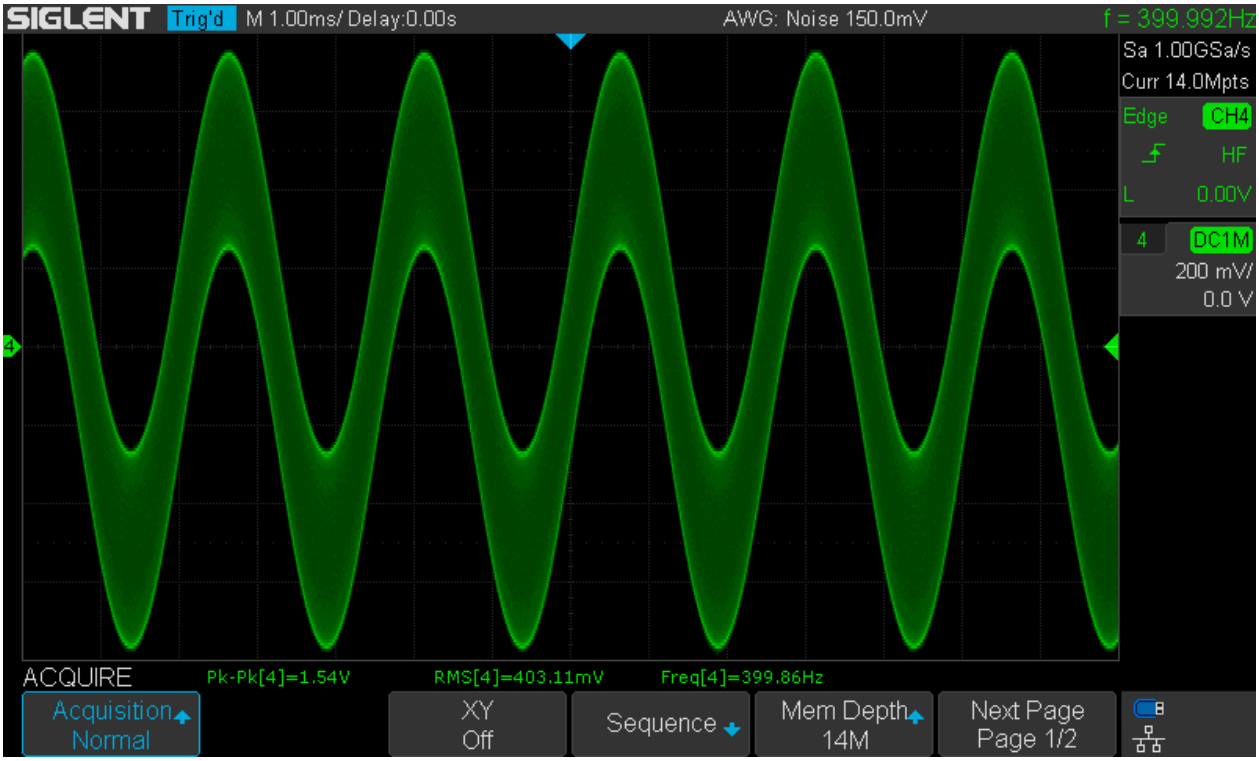
Resolution Enhancement [Bits]	-3dB Bandwidth (x Nyquist]
0.5	0.5
1	0.241
1.5	0.121
2	0.058
2.5	0.029
3	0.016

This could be used as a coarse guide, but the bandwidth calculated from these parameters is slightly (5 - 15%) higher than what can actually be achieved on the SDS1104X-E (apart from the obvious bandwidth limit of a 100MHz scope).

Eres mode may be helpful to clean up noisy dynamic signals, where averaging cannot be used. The maximum acquisition length is limited to 1.4Mpts (interleaved mode, 700kpts with all 4 channels in use).

Beware that for the time being, Eres mode provides no true resolution enhancement. This is mainly because of the display interface which is still limited to 8 bits. Siglent have promised to change this at some point in the future, but this is treated as a low priority task, so it may take a while.

To demonstrate the effect of Eres acquisition mode, a 400Hz sine with a 56MHz sine superimposed was used. The screenshot below shows what it looks like in normal acquisition mode.



Eres_normal_400Hz_56MHz

With just 0.5 bits resolution enhancement we can already see a filtering effect.

Eres_05_400Hz_56MHz

The next screenshot shows the same signal with 2 bits of resolution enhancement, which gets completely rid of the high frequency signal components and cleans up the waveform quite nicely.

Eres_20_400Hz_56MHz

Be aware that Eres mode only supports 1.4Mpts memory length and the sample rate drops down to just 100MSa/s in this example. The effective bandwidth limits in this example are therefore 25.4MHz for 0.5 bits and 2.77MHz for 2 bits.

Roll Mode

This is not in the *Acquisition* menu, but has a dedicated button on the front panel. It also is not a completely independent mode, but has to be combined with normal or peak detect.

Consequently, roll mode is a non-triggered continuous acquisition in either normal or peak detect mode, similar to a strip chart recorder. The big advantage of this mode is the total lack of blind time, but there are several restrictions to keep in mind:

- Max. acquisition memory depth is limited to 1.4Mpts
- Max. sample rate cannot be higher than 2MSa/s
- Only available for time bases 50ms/div and slower
- Cannot work with Average or Eres acquisition modes

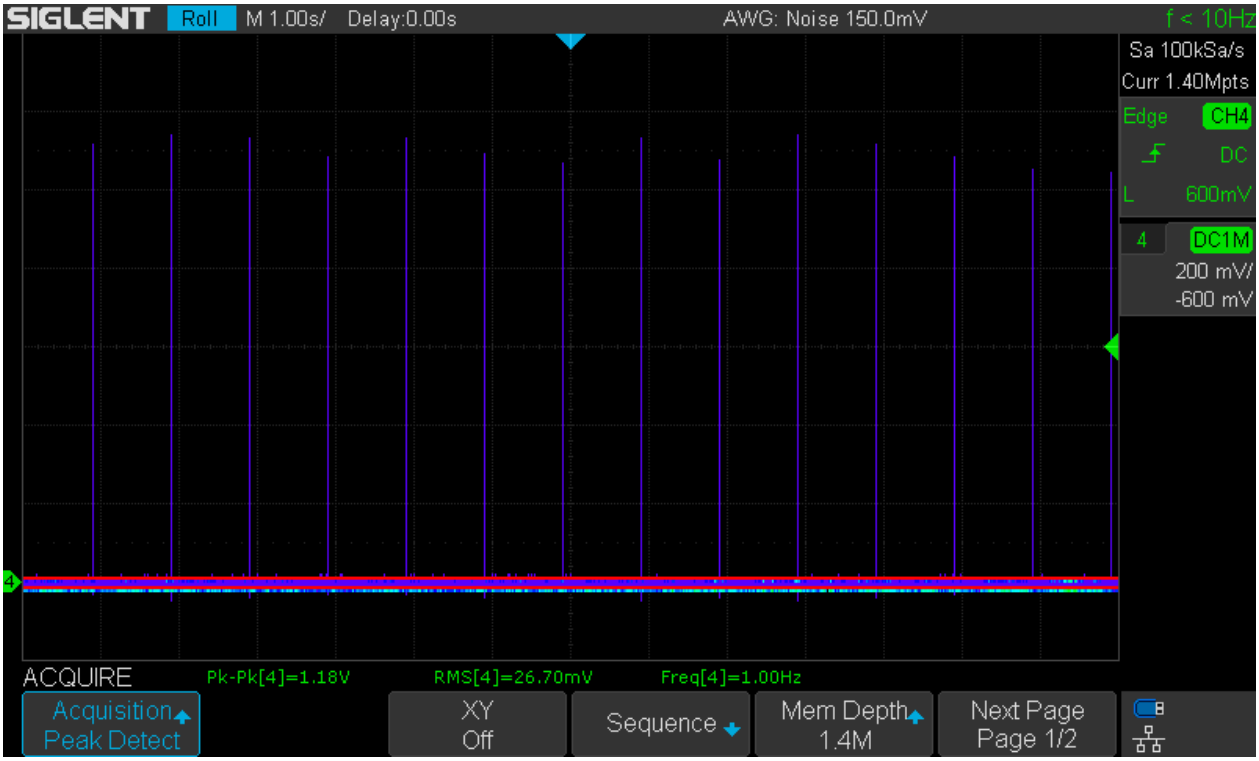
Roll mode is automatically engaged for timebase settings of 50ms/div or slower, but can be manually disabled, so that normal Y-t mode is still available. In these cases, a “Slow Acquisition” warning together with a red activity bar shows on the screen.

Roll mode works continuously and untriggered as expected only in auto trigger mode. If normal trigger is used, the acquisition is still triggered, which seems to defy the purpose, but other manufacturers seem to offer this feature as well and there might actually exist some odd application for it.

The screenshot below shows an example of normal roll mode at 1s/div.

Roll_normal

In the 2nd example, roll mode is combined with peak detect mode. Once again a 4ns wide pulse is fed into channel 4 at a repetition rate of 1Hz. At 1s/div and with only 100kSa/s, all pulses are visible with only slight variations in amplitude. Color mode has been used to aid the visibility of the narrow pulses in the screenshot.



Roll_peak

X-Y

In contrast to analog scopes, X-Y mode on a DSO is quite different and in any case not straight forward. The timebase is still active and important, as the X-Y picture is based on regular Y-t acquisitions of Ch.1 and Ch.2 (and/or Ch.3 and Ch.4, as the SDS1104X-E can show two independent X-Y traces in parallel, provided that all signals on all channels can be properly captured at the same timebase).

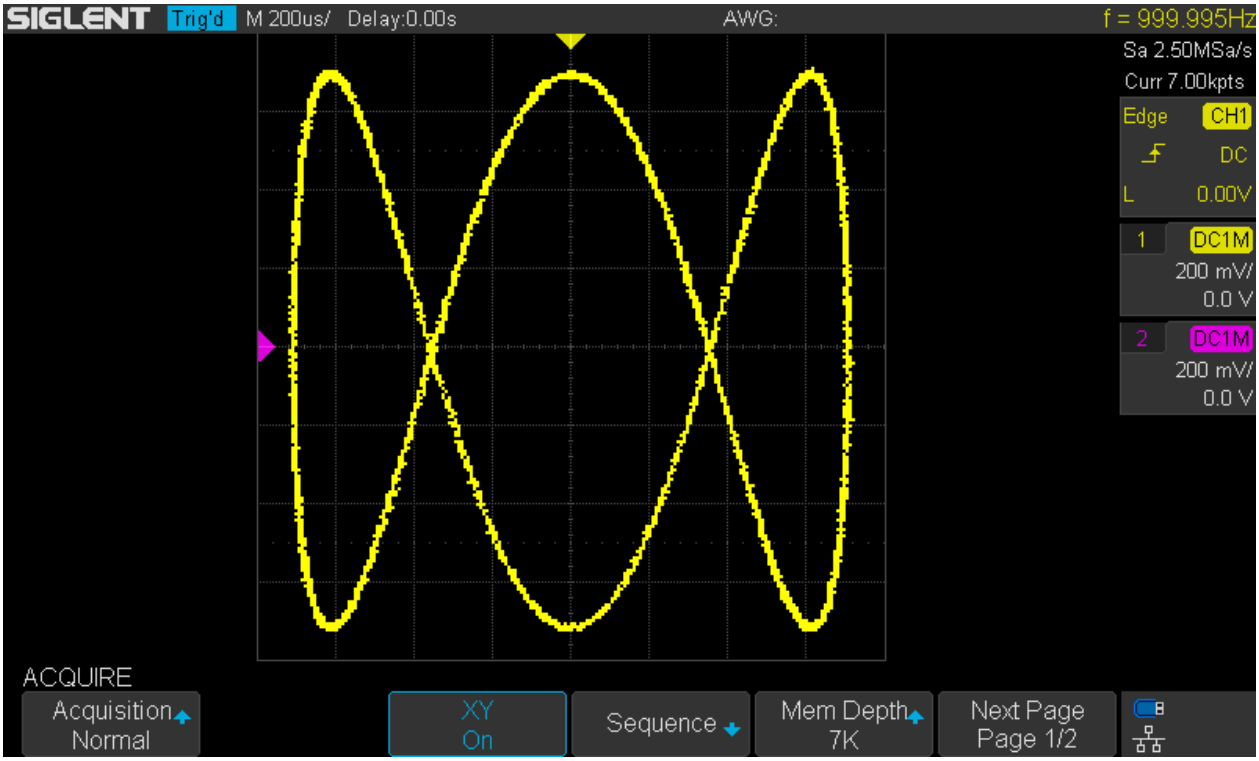
The SDS1104X-E is a bit special as it allows long memory in X-Y mode. Long memory considerably slows down operation, which is already slow anyway, at least by analog scope standards. Consequently, we want short memory (<700kpts, ideally just 7kpts) whenever possible. We just need to be careful as we might introduce aliasing with complex signals.

As a result, we have to carefully adjust our signals and memory depth so that the screen shows at least one full period of each signal involved and no aliasing occurs. At least we get a very reasonable update speed with short memory. I've yet to find a reliable way to estimate the exact update rate, but it is certainly fast enough to do the usual phase comparisons in realtime.

Siglent is going to implement the X-Y processing in hardware, thus speeding this mode up dramatically with some future firmware update.

Once the signals are properly set up in Y-t mode as shown in the first screenshot, we get the expected Lissajous figure in X-Y mode shown in the 2nd screenshot. The XY trace is fat and grainy, but what else can we expect from barely 8-bit data (it's actually just 200 visible codes) stretched on a 700x400 pixel screen area? In Y-t mode we have still full resolution on the X-axis and intensity grading helps masking the fact that every ADC code uses two horizontal pixels at once. In contrast, every X/Y data point uses occupies 4 screen pixels at once. Still I'm convinced the X/Y appearance could be improved.

XY_signals



XY_display

Interpolation

There is a choice between x-interpolation and sin(x)/x reconstruction in the *Acquisition* menu. We can also chose either dots or vectors in the *Display* menu, where the first setting could be located as well, since both choices are closely related and do not actually affect acquisition in any way.

However, it is important to understand what these menu items do and how to use them properly. Refer to the explanations in the sections below.

Dots + x

Display Type Dots and Interpolation x shows the true ADC samples and nothing else.

Dots + sin(x)/x

Display Type Dots and Interpolation sin(x)/x usually behaves exactly the same as Dots + x, i.e. only the original samples are visible. But in rare instances it also shows a bunch of additional display points that are the result of a digital sinc filter reconstruction. This is clearly a bug and I have only observed it once; either of the two behaviors would be acceptable, but it should be consistent all the time.

Vectors + x

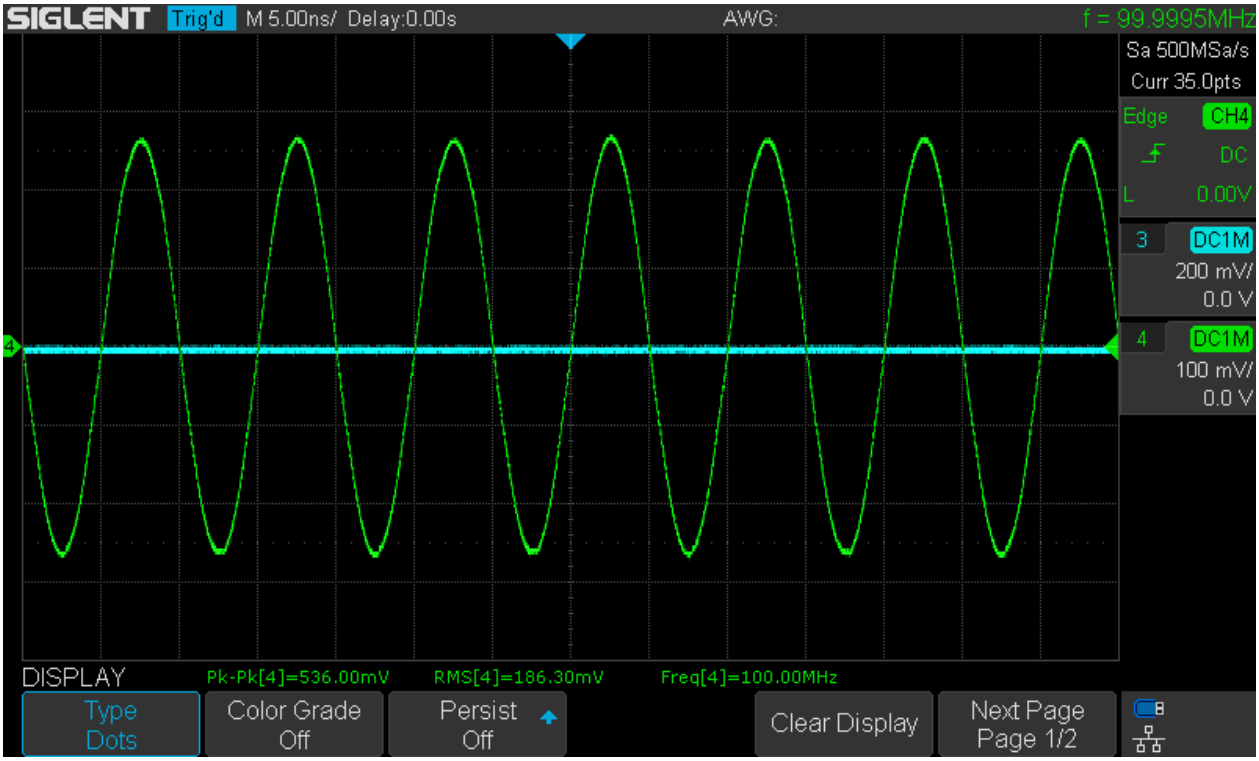
With Display Type Vectors and Interpolation x, a bunch of additional display points is generated as a linear interpolation from one sample to the next. In other words, the existing ADC samples are interconnected by straight lines. This is useful for “beautifying” signals like pulses and ramps, especially when signal components close to or above Nyquist cause ringing and show the Gibbs phenomenon.

Vectors + sin(x)/x

With Display Type Vectors and Interpolation sin(x)/x, a bunch of additional display points is generated as the result of a digital Sinc filter reconstruction. In other words, the existing ADC samples are fed into a numerical Sinc filter, which should ideally act as a brick-wall low-pass at the Nyquist frequency. This is the right mode for faithful reconstruction of sinusoidal waveforms near the Nyquist frequency.

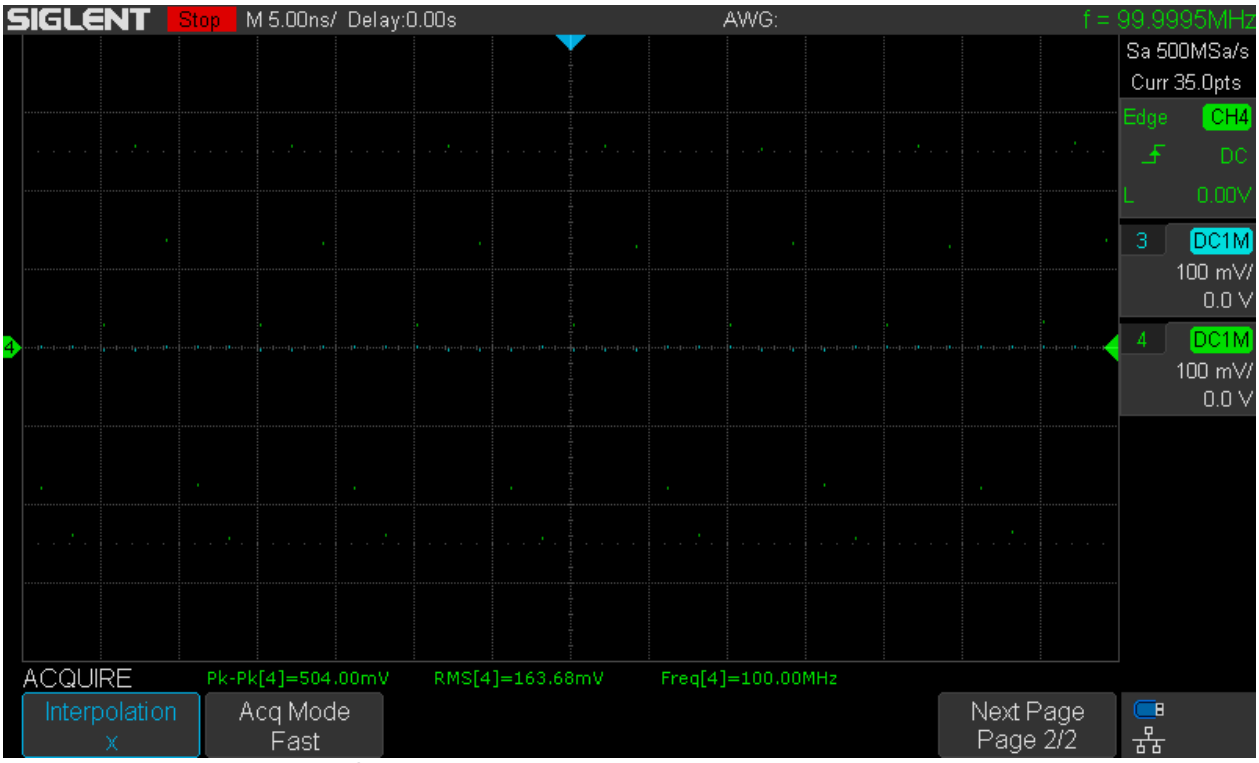
Examples

The next two screenshots show a 100MHz sine sampled at 500MSa/s (because both channels in a channel group are in use) and displayed in dots mode.



INT_100MHz_500MSa_5ns_dots

The dots display yields a perfectly fine result as long as the scope is in run mode. If stopped, we only see a single acquisition record and there are only a handful of isolated dots.



INT_100MHz_500MSa_5ns_dots_stop

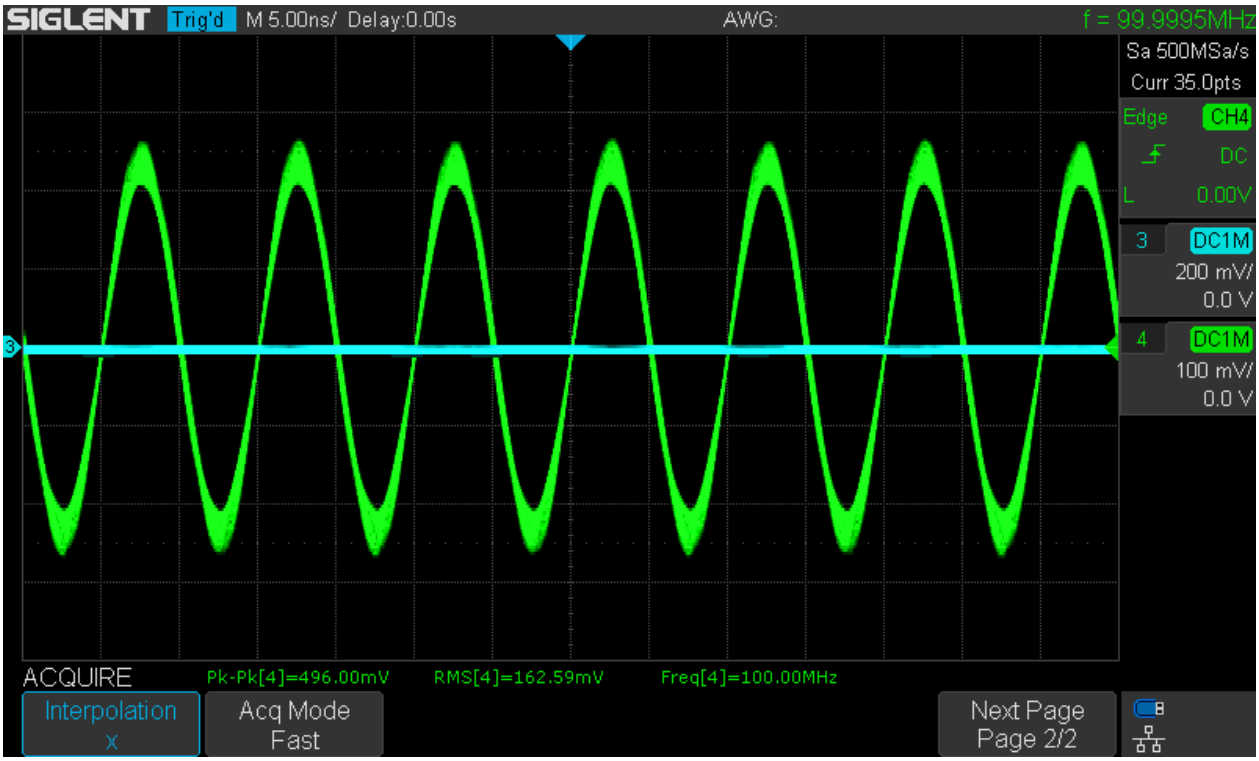
The next screenshot shows display mode vectors together with x-interpolation. This looks quite different to the dots display we've seen before. With vectors and linear x interpolation, we seem to get an amplitude-modulated waveform rather than a stationary signal.

Dots display works flawlessly just because we only see the true ADC samples. Because of the fast acquisition rate, we get a lot of acquisition records mapped into each display frame and the position of the samples continuously changes, because there is no sync between signal and sample clock. This yields plenty of samples (= displayed dots) to give the impression of a contiguous line.

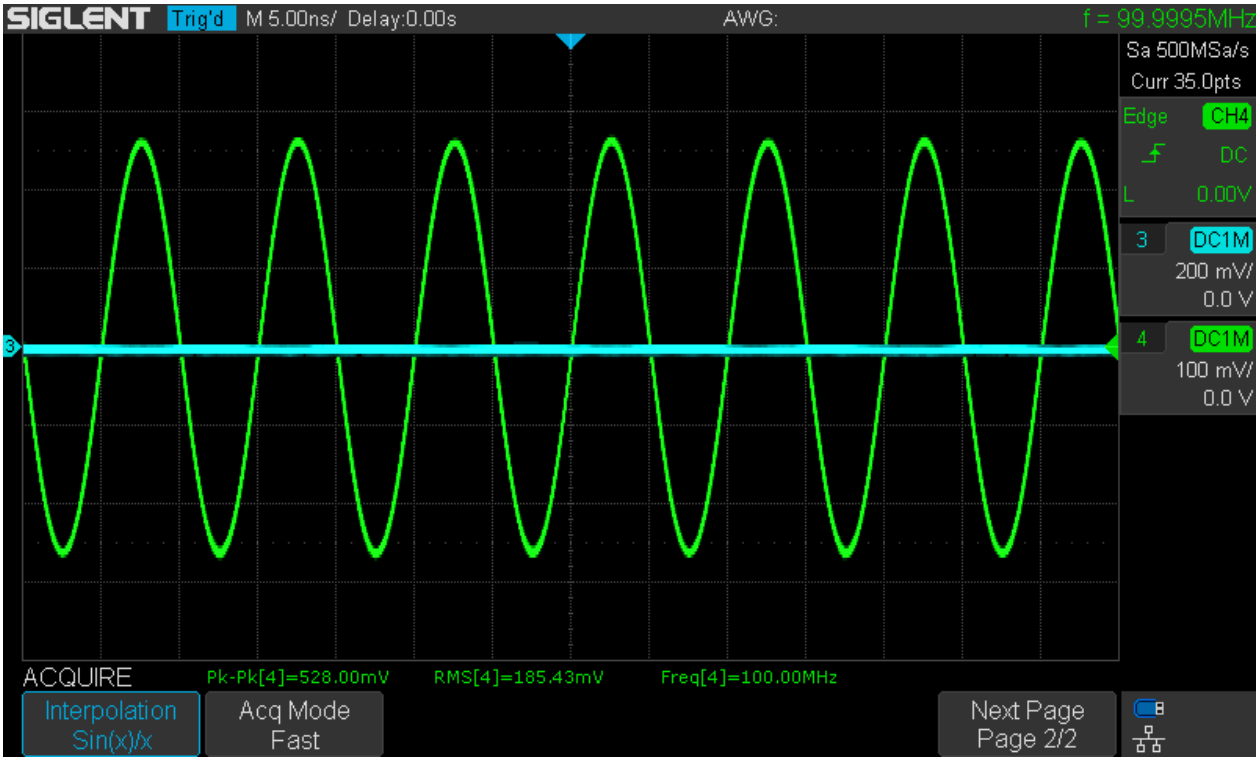
Dots display always gives a faithful representation of the input signal, as long as there are no signal components exceeding the Nyquist frequency (half the sample rate).

Vectors display together with x-interpolation does not show the truth though. This is because interpolation has added artificial data points that do not match the real signal. Consequently, the real ADC data mixed with the artificial interpolation date sum up to something completely new. So just keep in mind:

Linear (x) interpolation only works reasonably well for signals up to 1/10 of the sample rate.



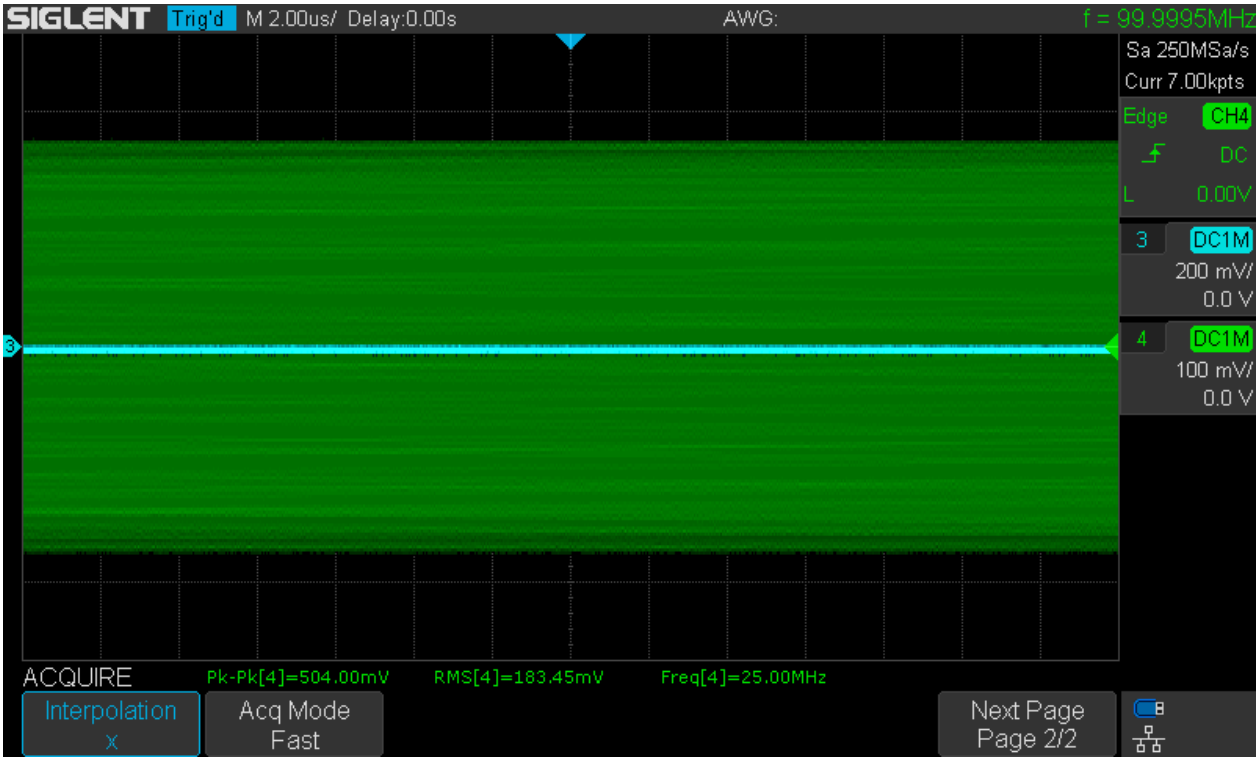
INT_100MHz_500MSa_5ns_x



INT_100MHz_500MSa_5ns_sinc

The screenshot above shows the same signal in vectors display mode, but this time with sin(x)/x reconstruction. The result is pretty much the same as with dots mode, but in contrast it will work on a single record and consequently show a contiguous line even in stop mode.

Finally, let's see how close the sinc filter implemented in the SDS1104X-E comes to an ideal brick-wall filter. Today's benchmark appears to be an acceptable reconstruction at 40% of the sample rate. With both channels in a group, e.g. all four channels in use, the max. sample rate is 500MSa/s and 40% of this would be 200MHz. Since the frontend bandwidth is limited to some 100MHz, we need to test this at an even lower sample speed of 250MSa/s. To accomplish this, I've limited acquisition memory depth to 7kpts and selected a time-base of 2µs/div.



INT_100MHz_250MSa_2us_x

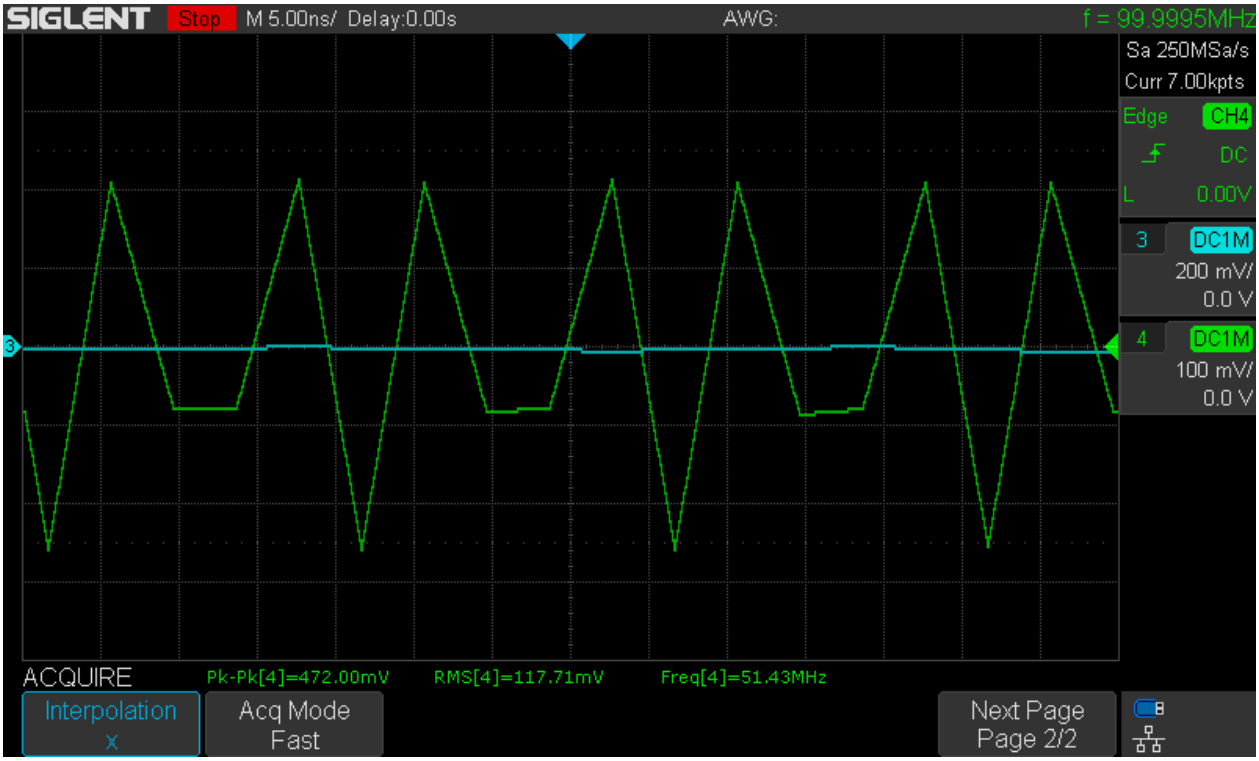
Of course, with a time-base that slow we cannot see anything meaningful. So we have to stop the acquisition and then zoom into the waveform by simply adjusting the horizontal time-base to 5ns/div.

Now we can try both display methods even after the acquisition has been stopped.

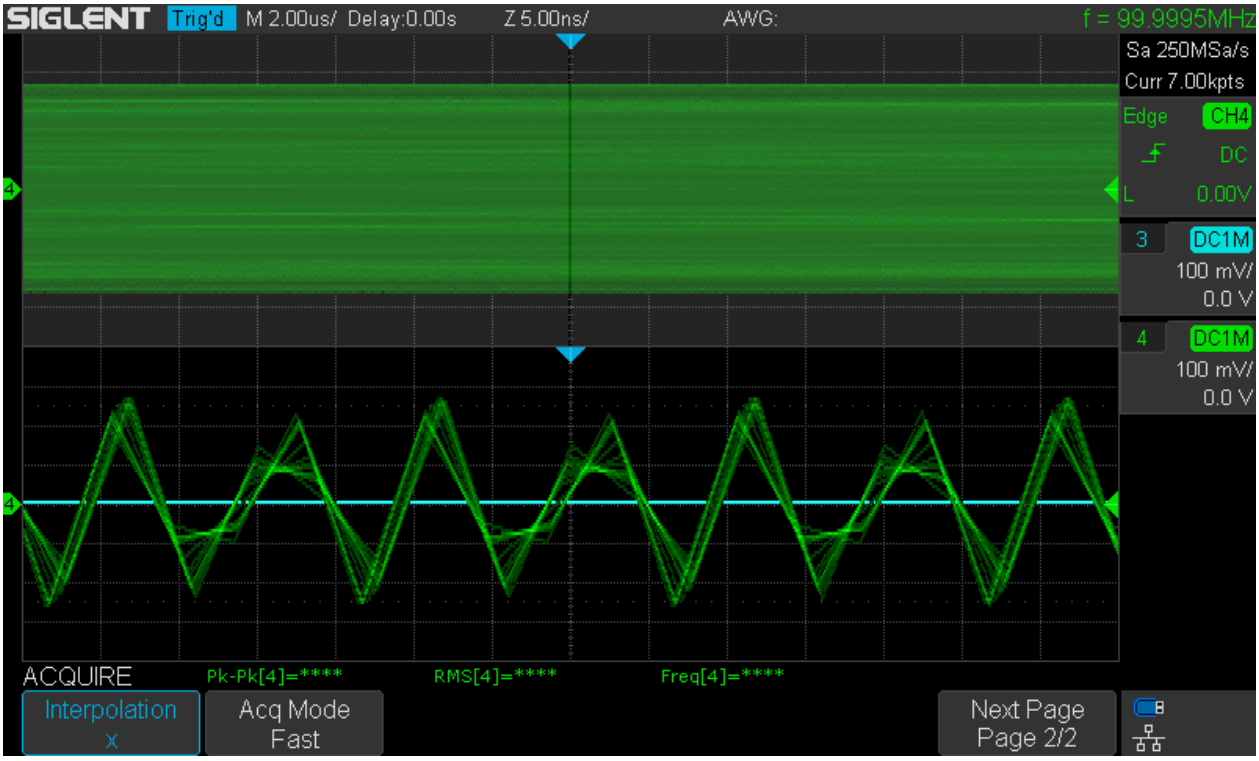
The first following screenshot shows vectors with linear x-interpolation. The similarity with the original signal is quite low – to say the least – and we already get an idea how this would look like in run mode.

The second following screenshot actually shows what it looks like in run mode. For this the zoom mode has been used with 2µs/div main time-base in order to get the required sample-rate of 250MSa/s and a zoomed time-base of 5ns/div to display an image comparable to the previous ones.

As can be seen, the waveform is now a complete mess and any similarity with the real signal would be purely by chance.

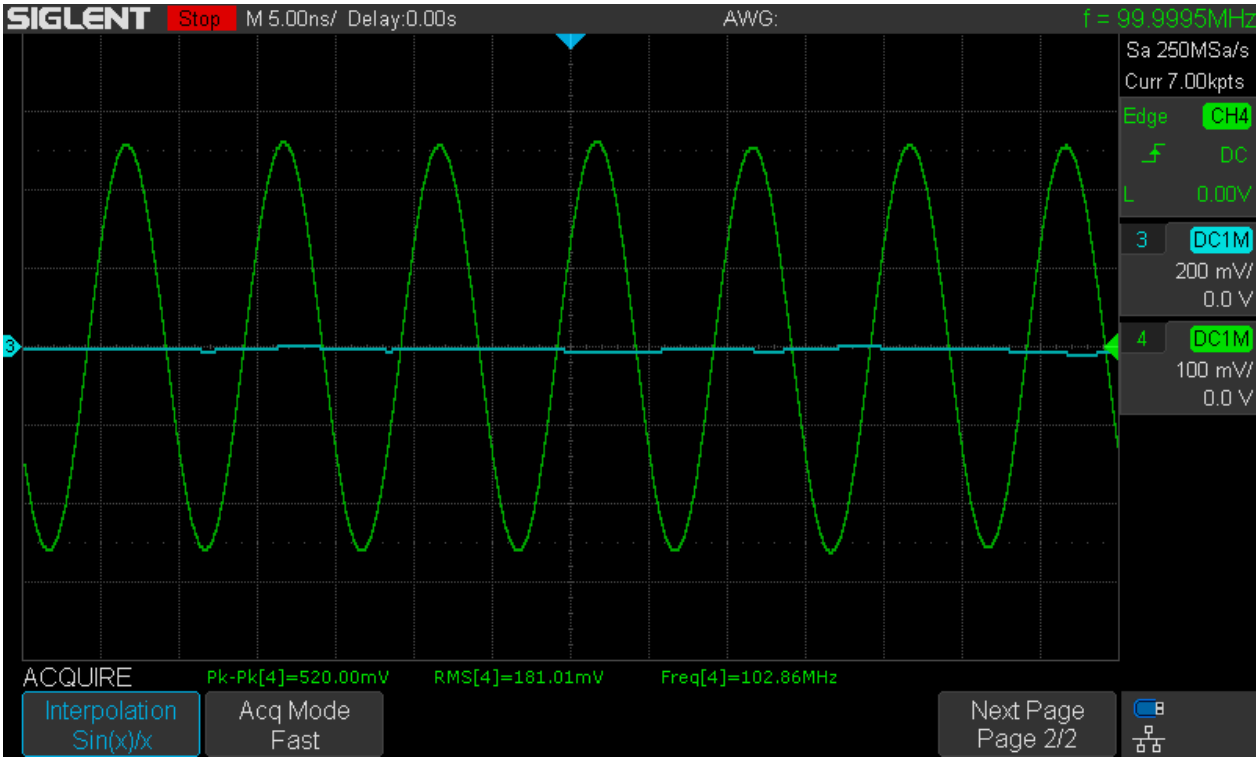


INT_100MHz_250MSa_5ns_x



INT_100MHz_250MSa_2us_x_Z5ns

Here's the same situation, i.e. a 100MHz sine-wave captured at 250MSa/s and displayed at 5ns/div with proper sin(x)/x reconstruction.



INT_100MHz_250MSa_5ns_sinc

Finally an example where simple x-interpolation gives better usable results when compared to sin(x)/x reconstruction:

INT_2MHz_stepup_100ns_x

INT_2MHz_stepup_100ns_sinc

Signal Handling

This section deals with the DSO properties related to the frontend design and calibration, hence mainly amplitude accuracy and bandwidth. Even though an oscilloscope is not a precision measurement device, its accuracy specifications are still in the same ballpark as analog meters that were universally used several decades ago – and then as now there are many tasks where we just don't need more than that. Yet we want to be sure that our instrument meets its specifications under all practical circumstances and we can rely upon the test results we're getting out of it.

DC Accuracy

The first test looks at the DC accuracy of the trace display and the automatic measurements for all available vertical gain settings. Environment temperature was 23°C and a self-calibration has been performed after several hours of warm-up one day before the test started.

The table below shows the results of all measurements. For each vertical gain setting, measurements have been performed for both polarities and three offsets (zero and ±3 divisions) with and without input signal respectively, resulting in a total of seven measurements per range. The reason why measurements with offset have been included is the probably widely unknown fact that not all DSOs will pass this test, so I wanted to be thorough.

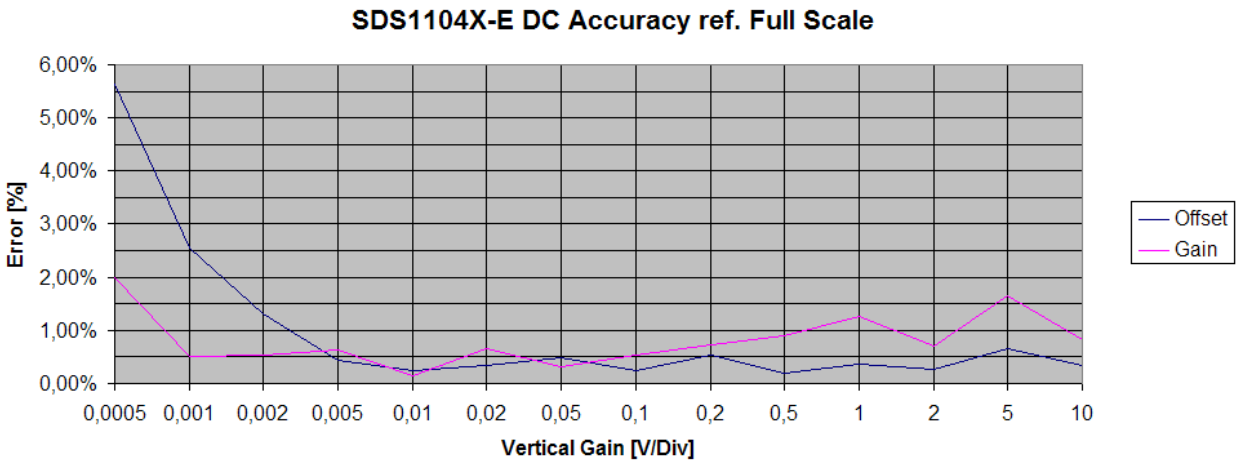
All ranges are well within spec, the green results are even within 25% of the specified error margin. If we look at the error figures, it becomes clear that what we mostly see is nothing more than the inevitable ±1LSB error of the 8-bit ADC and this DSO can indeed compete with the most expensive analog meters.

Below there is also a graph providing a quick overview of the maximum offset and gain errors with respect to the vertical gain setting. The table always shows the worst error figures out of all 7 measurements.

Setting	Signal	Error		
Gain [V/div]	Step [V]	Offset [V]	Offset [%FS]	Gain [%]
0,0005	0,0015	225,0E-6	5,63%	2,00%
0,001	0,003	205,0E-6	2,56%	0,50%
0,002	0,006	209,0E-6	1,31%	0,54%
0,005	0,015	174,0E-6	0,44%	0,64%
0,01	0,03	192,0E-6	0,24%	0,14%
0,02	0,06	540,0E-6	0,34%	0,65%
0,05	0,15	1,9E-3	0,48%	0,31%
0,1	0,3	1,9E-3	0,24%	0,54%
0,2	0,6	8,6E-3	0,54%	0,73%
0,5	1,5	7,4E-3	0,18%	0,89%
1	3	28,9E-3	0,36%	1,27%
2	6	43,7E-3	0,27%	0,71%
5	15	261,0E-3	0,65%	1,66%
10	30	276,0E-3	0,35%	0,82%

Out of spec.
Within spec.
Error less than 25% spec.

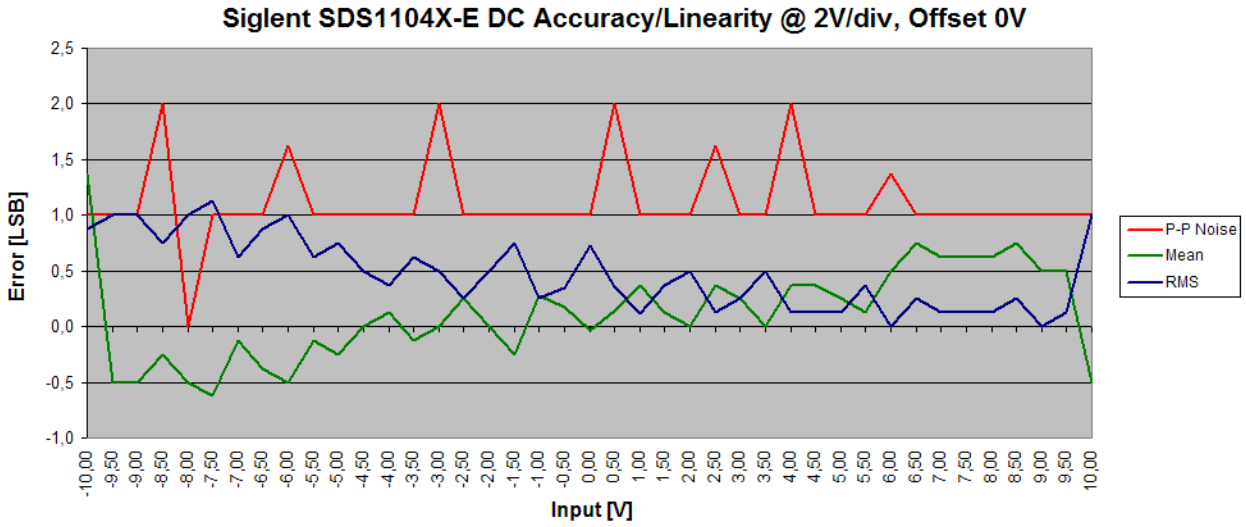
SDS1104X-E_DC_Ranges_Accuracy



SDS1104X-E_DC_Accuracy_FS

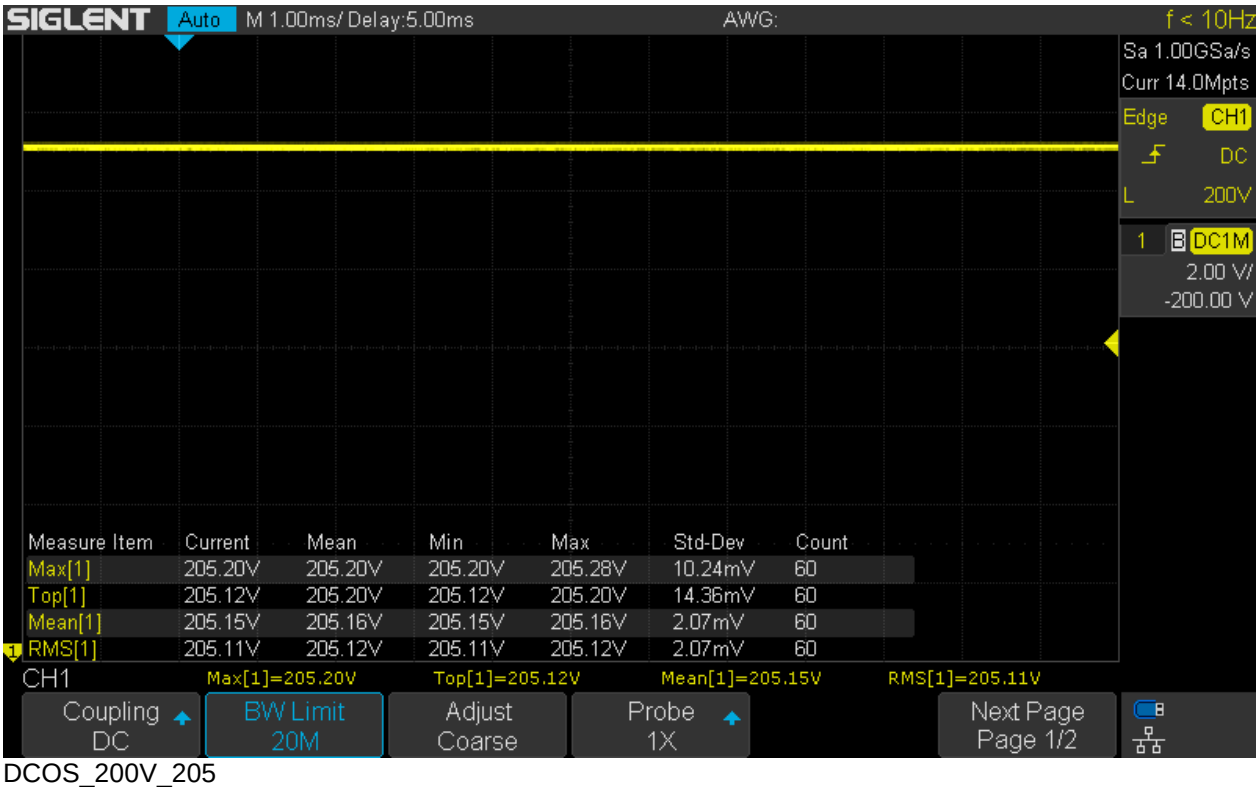
The next test looks at the linearity over the full ADC range. At a vertical gain of 2V/div and with zero offset, an accurate input voltage from -10V to +10V in steps of 0.5V has been applied and measured. Please keep in mind, that the visible display range is only ±8V, according to 200 LSB of the ADC. Consequently, this ±10V test covers 250 LSB, hence almost the entire 8-bit number range and possibly eating into the calibration headroom. Results are shown in LSB instead of %, because that makes immediately clear how close to the physical limits they actually are.

Peak-peak ADC noise is a little on the high side with up to 2 LSB in several spots, yet mean and RMS values stay within ±1 LSB pretty much throughout the range.



SDS1104X-E DC_Linearity@G2V

Finally, a practical test measuring an accurate DC voltage of +205V ±20ppm at the scope input directly is shown. To do this, the vertical gain has been selected as 2V/div, because this is the most sensitive range that allows an offset up to ±200V. And that's exactly what has been used here. With an offset of -200V, the +205V input voltage causes a deviation of +2.5 divisions from the center, and automatic measurements should indicate a voltage of +205V. As this is pure DC, Max, Top, Mean and RMS measurements should all give pretty much identical results – and they certainly do. The error of the mean measurement is <0.08%!

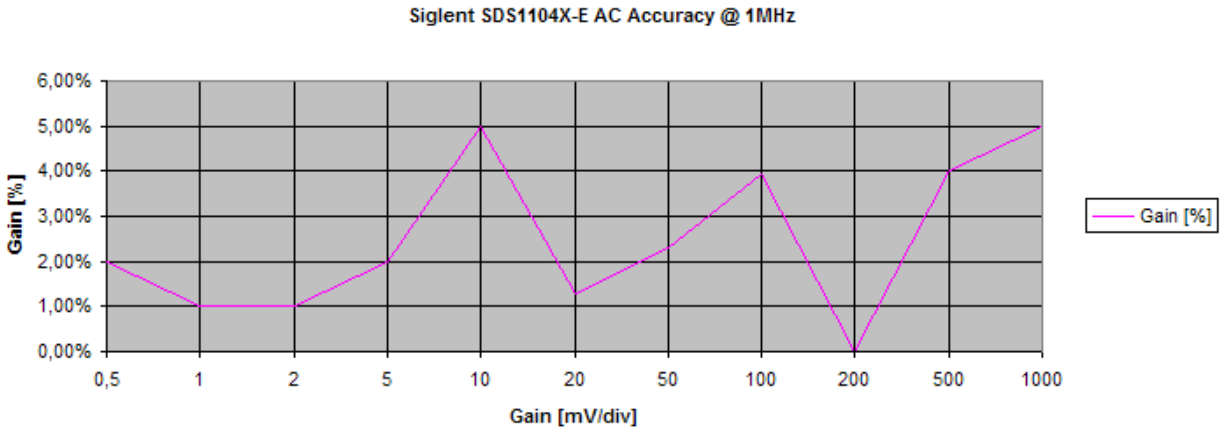


DCOS_200V_205

AC Accuracy

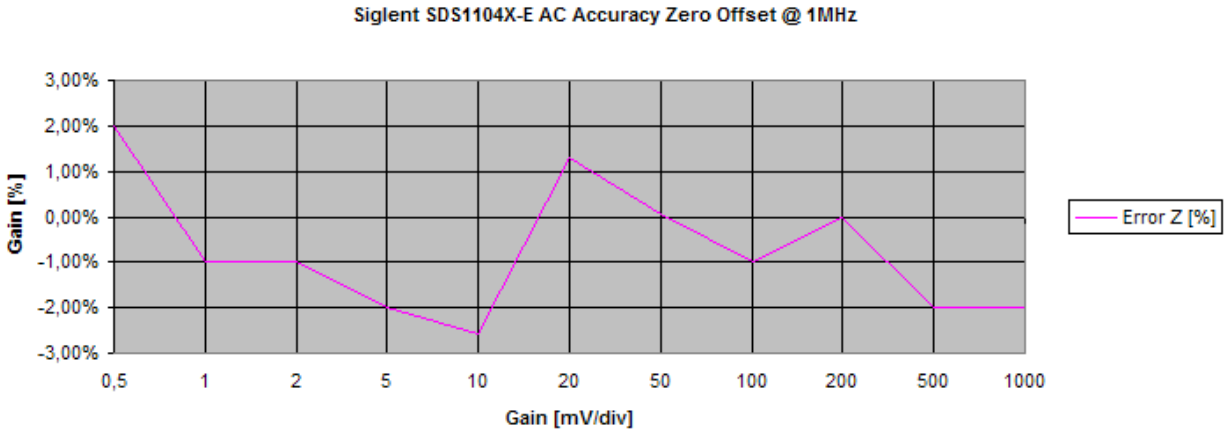
This test looks at the AC accuracy of the trace display and some automatic measurements for most of the available vertical gain settings. Environment temperature was 23°C and a self-calibration has been performed after several hours of warm-up one day before the test started.

The graph below shows the results of all measurements. For each vertical gain setting, measurements have been performed for three offsets (zero and ±2 divisions) , with and without input signal respectively, resulting in a total of 7 measurements for each gain setting. The reason why measurements with offset have been included is the probably widely unknown fact that not every DSO will pass this test.



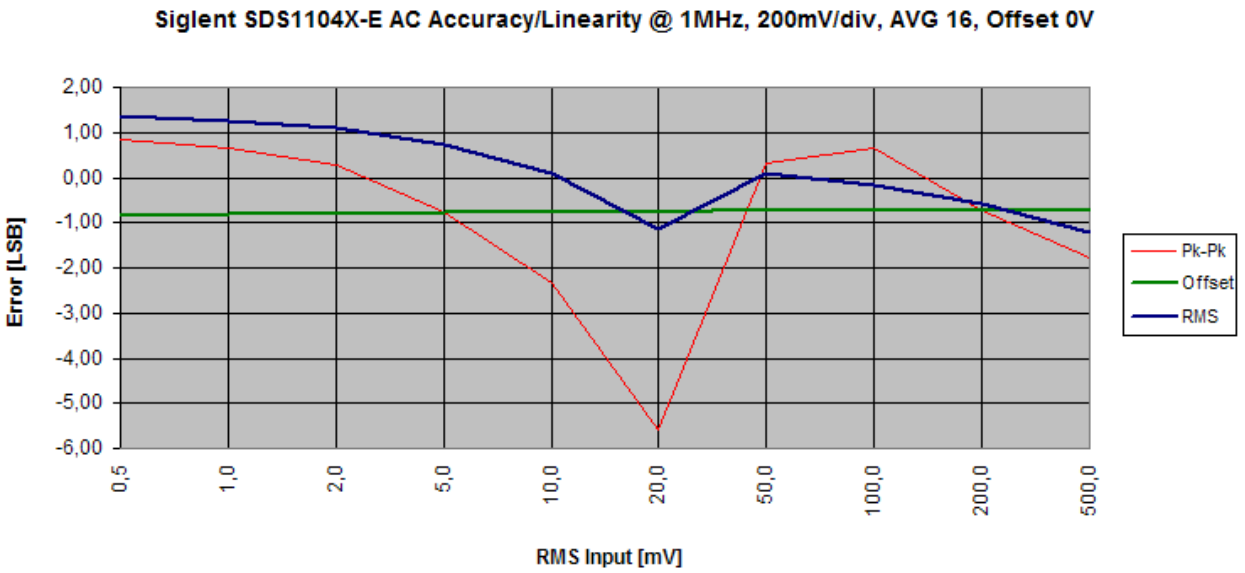
SDS1104X-E_AC_Accuracy@1MHz

With a maximum error of 5%, all tested ranges are well within spec, which would be 1dB or approximately 10.9%. If results are limited to the tests with zero offset, the error never exceeds 2.5% as can be seen in the diagram below.

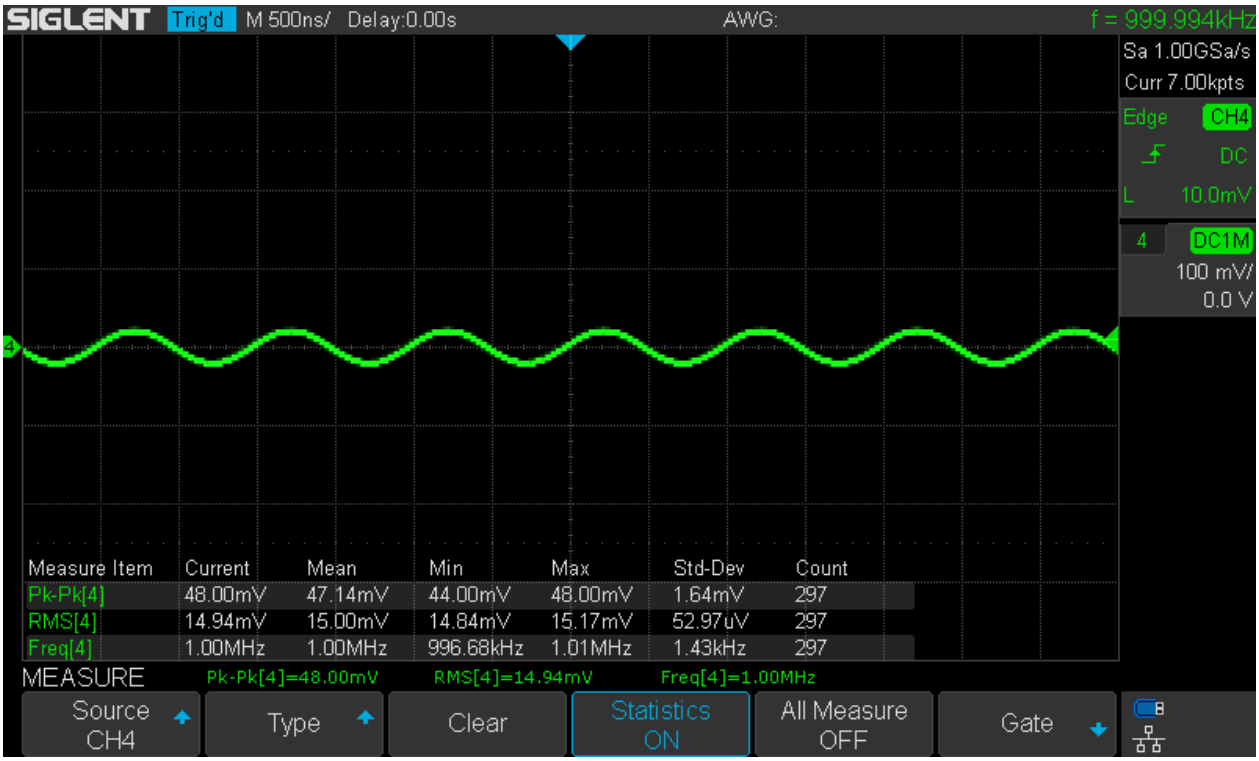


SDS1104X-E_AC_Accuracy_Zero_Offset@1MHz

Next is the AC linearity test which should answer the question how low a signal can be measured with reasonable accuracy. The graph looks odd because the error is measured in LSBs of the ADC and the test covers an absolutely unrealistic 60dB range, whereas only some 26dB are actually usable. This is still a very respectable result, as it actually means we can measure a signal with just 0.5 divisions peak to peak amplitude pretty accurately. A screenshot has been added to demonstrate just this. The automatic measurement of the 15mVrms input signal is pretty much spot-on despite the low amplitude of only half a division peak to peak.

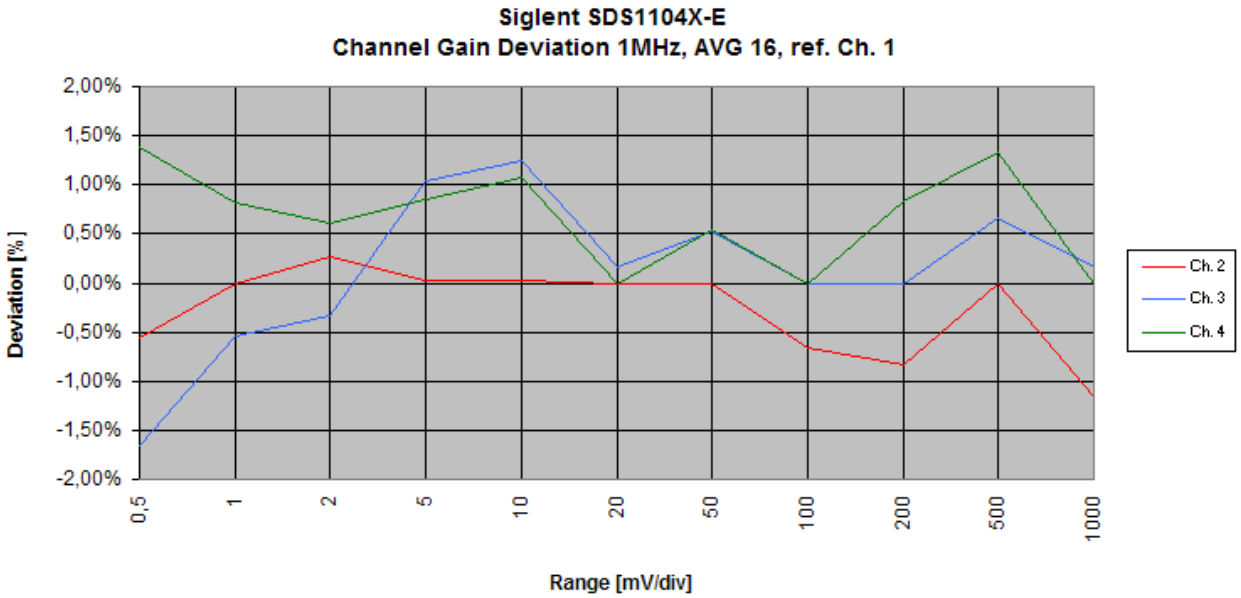


SDS1104X-E AC_Lin@1MHz_200mV_Avg16



SDS1104X-E_AC_Accuracy_1MHz_05div

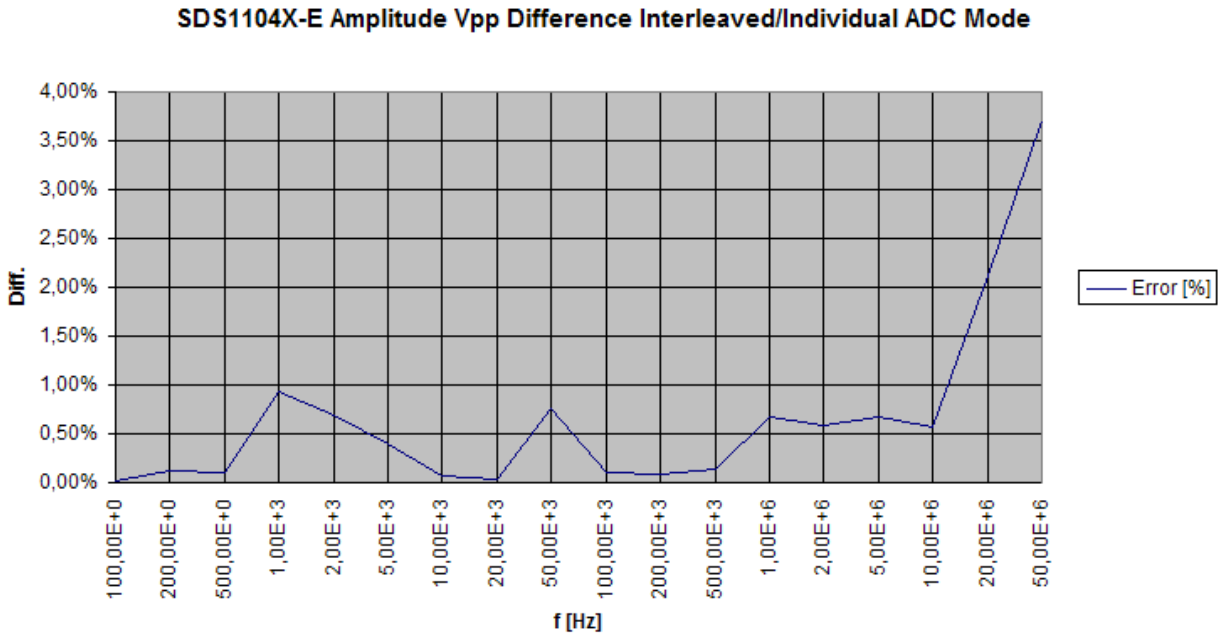
We also want to check the vertical gain deviation of channels 2 to 4 with reference to channel 1. The test is done at a frequency of 1MHz for all vertical gain settings up to 1V/div. The differences are fairly low and certainly acceptable, particularly for a cheap entry level scope like this. Except for Ch.2 at 500µV/div, the error does not exceed 1.5%.



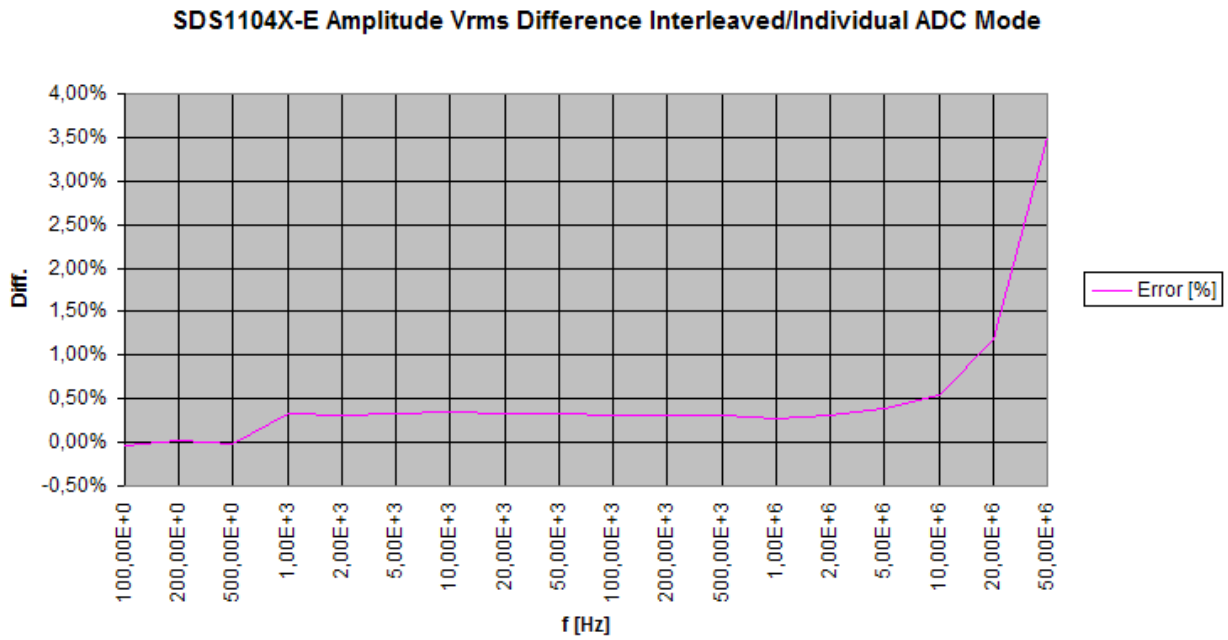
SDS1104X-E_Channel_Gain_Deviation@1MHz

Finally we want to see the impact of the ADC mode (interleaved/individual) on the signal amplitude. The key for this is the number of active channels. With only one active channel per channel group, the ADC and memory resources that would normally be used for the second channel are now associated with that single active channel (interleaved mode), thus giving us twice the sample rate and acquisition memory. Of course this affects the measured signal amplitude, particularly at high frequencies.

The following graphs show the deviation of peak to peak and rms measurements between one and two channels per group for frequencies from 100Hz up to 50MHz.



SDS1104X-E_Amplitude_Vpp_Difference



SDS1104X-E_Amplitude_Vrms_Difference

The deviation is pretty much negligible for frequencies up to some 10MHz and still well within specifications above that. It comes as little surprise that the deviation rises significantly at high frequencies, where the difference in sample rate has indeed a substantial impact on measurement accuracy. But this is also the area where a +2/-3dB error is specified, so there's still no reason to complain.

Bandwidth

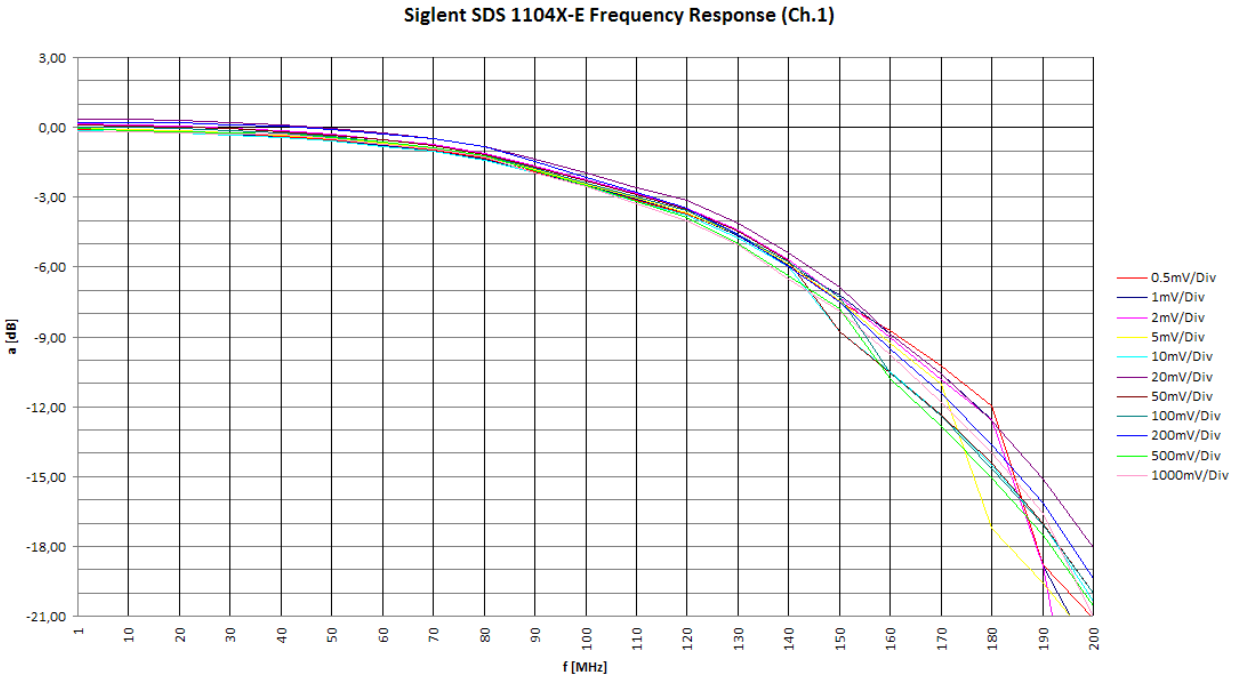
For this test, only the 100MHz SDS1104X-E was available. The table below shows the measurement results for all vertical gain settings up to 1V/div.

Vert. Gain	BW [MHz]		
	1 dB	3 dB	6 dB
0.5mV/Div	74,00	110,00	141,00
1mV/Div	74,00	110,00	141,00
2mV/Div	74,00	110,00	141,00
5mV/Div	74,00	110,00	141,00
10mV/Div	74,00	109,00	141,00
20mV/Div	74,00	111,00	141,00
50mV/Div	74,00	109,00	141,00
100mV/Div	74,00	109,00	141,00
200mV/Div	78,00	109,00	139,00
500mV/Div	76,00	109,00	138,00
1000mV/Div	78,00	109,00	138,00

SDS1104X-E_BW

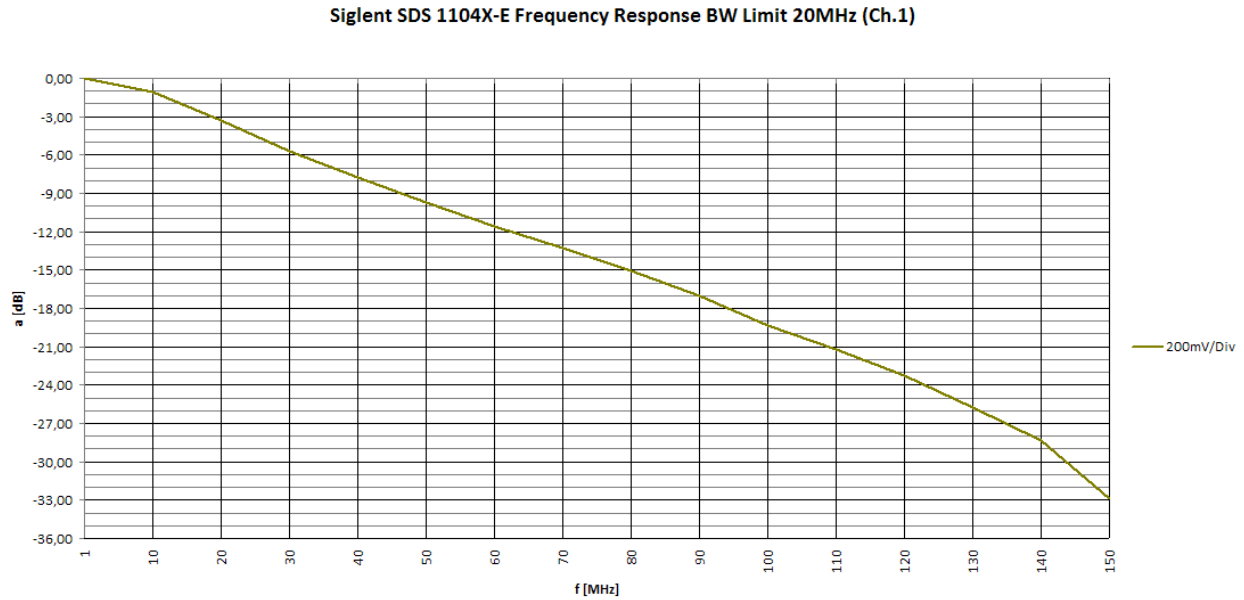
Since there has to be an explicit bandwidth limiter somewhere in this scope, it comes as no surprise that the actual bandwidth is only slightly higher than specified.

The graph below shows the individual frequency responses up to 200MHz for all tested vertical gain settings from 500µV/div to 1V/div.



SDS1104X-E_BW_Graph

The next graph shows the frequency response at 200mV/div with 20MHz bandwidth limit activated.



SDS1104X-E_Frequency_Response_BW_Limit_20MHz

Actual 3dB bandwidth limit has been measured as 18.6MHz, which is well within the specified tolerance of ±40%.

I do not have any data for the SDS1204X-E, but its frontend is most likely pretty much identical to the SDS1202X-E whose bandwidth measurement data can be found in the table below.

Vert. Gain	BW [MHz]		
	1 dB	3 dB	6 dB
0.5mV/Div	204,00	240,00	290,00
1mV/Div	204,00	240,00	290,00
2mV/Div	203,00	240,00	289,00
5mV/Div	200,00	236,00	285,00
10mV/Div	199,00	236,00	285,00
20mV/Div	201,00	237,00	285,00
50mV/Div	198,00	233,00	283,00
100mV/Div	192,00	227,00	280,00
200mV/Div	200,00	243,00	314,00
500mV/Div	196,00	238,00	310,00

SDS1202X-E_BW

Input Impedance

Bandwidth measurements are always compromised by parasitic impedances, which we'll take a closer look at in this section.

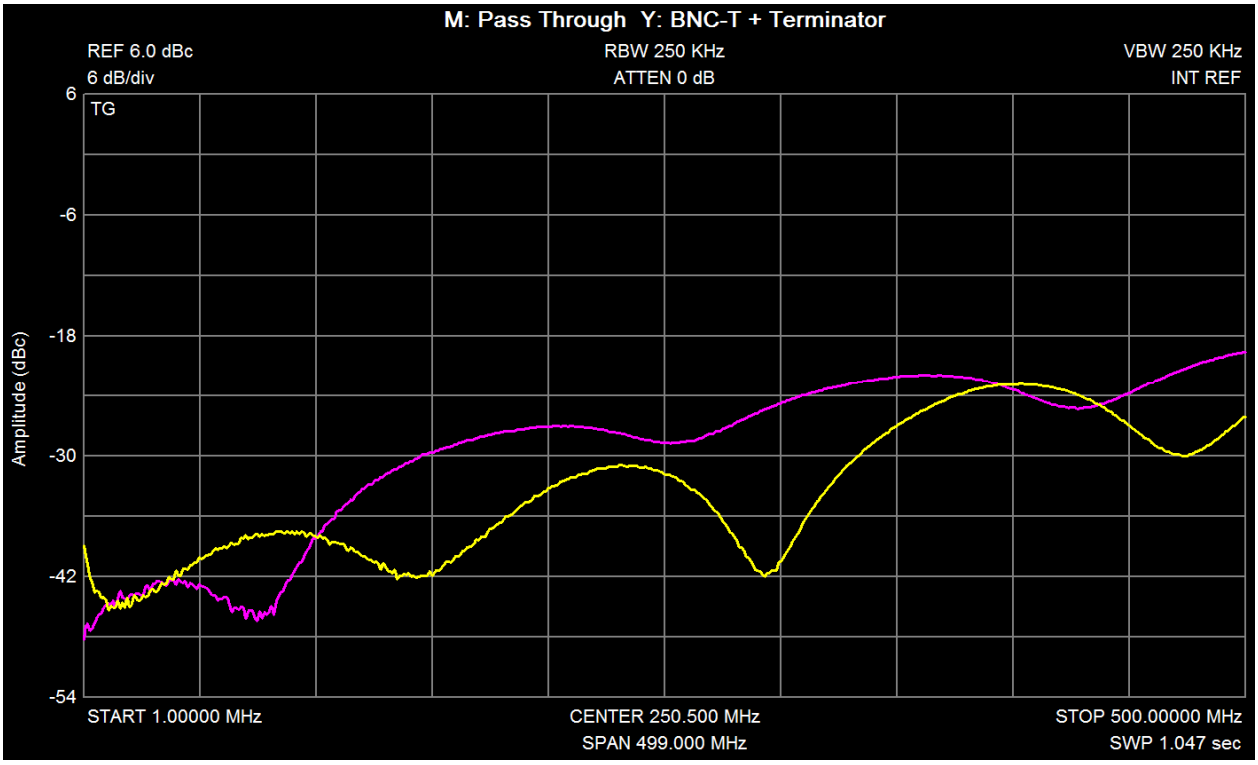
If we use the probes, then the input capacitance of the probe will present an increasing load to the signal source at higher frequencies, which lets the amplitude drop and might even cause signal distortion. In most cases, the stray inductance of the ground connection adds to that and turns the probe into a series resonance circuit. This is why it is essential to use the probe BNC adaptor whenever the probe bandwidth is to be determined.

For the majority of tests, a direct BNC connection is used and this has it's pitfalls as well. In general purpose high frequency test & measurement instruments, 50Ω impedance for the inputs is common. The same is true for coax cables e.g. RG58 used in the lab. A flat frequency response, thus high signal fidelity can only be obtained if the entire connection from the signal source to the scope input is precisely 50Ω throughout. Signal generators usually have a 50Ω output, laboratory coax cables and connectors are 50Ω, but scope inputs are often not – particularly not in low cost general purpose scopes. Better general purpose scopes provide a switchable 50Ω input impedance, but e.g. the SDS1000X-E does not.

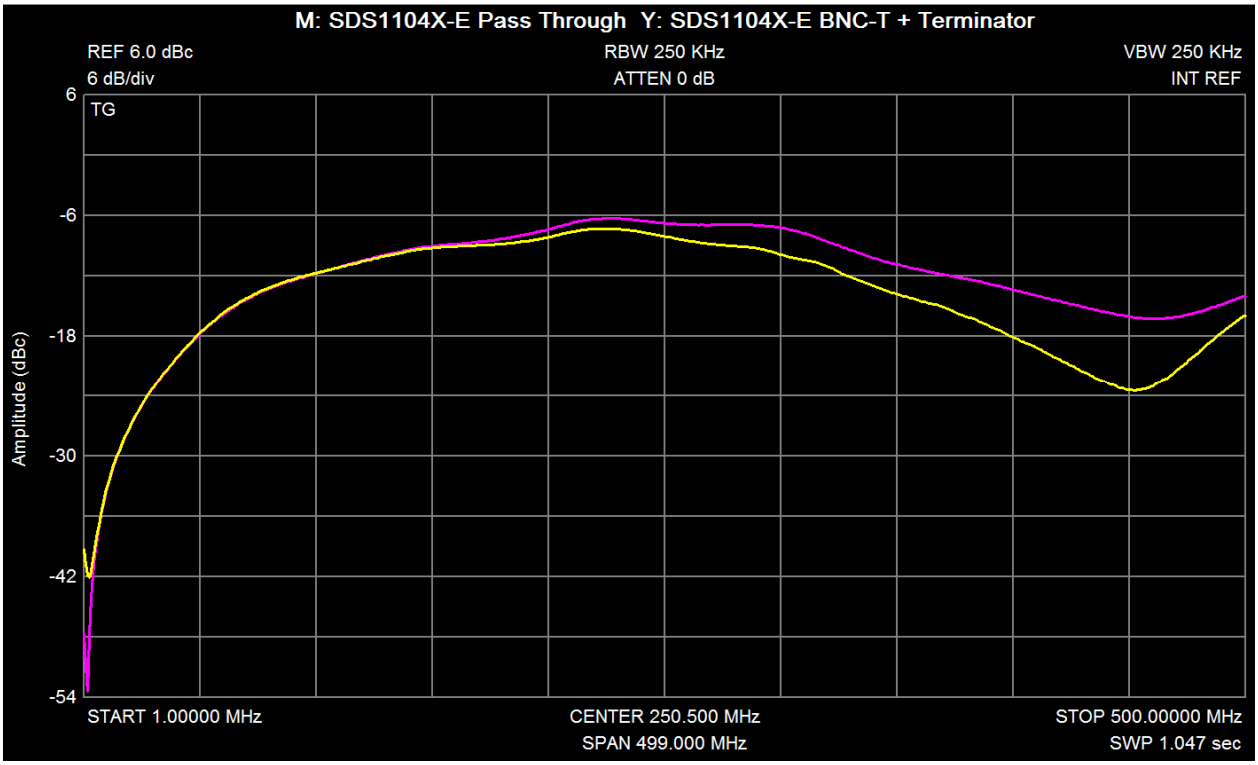
In theory, we can use an external 50Ω pass-through terminator to make any scope 50Ω compatible. The 1MΩ Input resistance would be negligible in this case, but the input capacitance (15pF for the SDS1104X-E) is not. The 3dB corner frequency of this combination is 212MHz. There are several adverse effects because of the input capacitance of the scope:

1. Additional attenuation at higher frequencies because of the capacitive loading. It would be -3dB at 212MHz for the SDS1104X-E.
2. The end of the BNC cable is not properly terminated anymore at higher frequencies. Depending on the cable length this will cause ripple in the overall frequency response.
3. The two topics mentioned above would apply for most scopes that can set the input impedance to 50Ω internally. For an external pass-through termination, there is some distance between the 50Ω resistor and the input capacitance, causing an even worse mismatch and making matters all the more obscure, but this most likely only plays a role at really high frequencies >1GHz.

Sometimes people don't have an external pass-through terminator available and use a BNC-T + BNC Termination. Let's compare the two methods. The pass-through terminator is a RS-456-150-50Ω which is the Tyco part B35 X13 • 999 X99 – unfortunately there's no specification. The equivalent end terminations are specified with a VSWR of 1.1 up to 1GHz. The terminator on the BNC-T is not specified either, but at one point I have bought a bunch of different BNC terminators, kept the best and threw away the rest. So it is a good one. Now let's see how the two solutions compare – just the through termination, not plugged into the scope. Magenta is the RS-456-150-50Ω, yellow the BNC-T with selected terminator at one end. Both solutions work fine up to 500MHz, with a return loss >20dB, equivalent to a VSWR <1.25. The 2nd screenshot shows the return loss of both solutions when plugged into the scope input.



RL_Ext_Comp



RL_SDS_Ext_Comp

As expected, this is significantly worse, but still not quite as bad as suspected. It is interesting though that the BNC-T works slightly better than the dedicated pass-through terminator. Return loss is particularly bad at 220MHz, where it is almost down to 6dB, equivalent to a VSWR of 3. Interestingly, it gets better again at higher frequencies.

Particularly for the SDS1104X-E we get a return loss of some -9dB and VSWR <2.2 at 100MHz. The situation is only really satisfactory for frequencies up to some 50MHz.

Low Frequency Response

There are at least two concerns about the low frequency response in an oscilloscope:

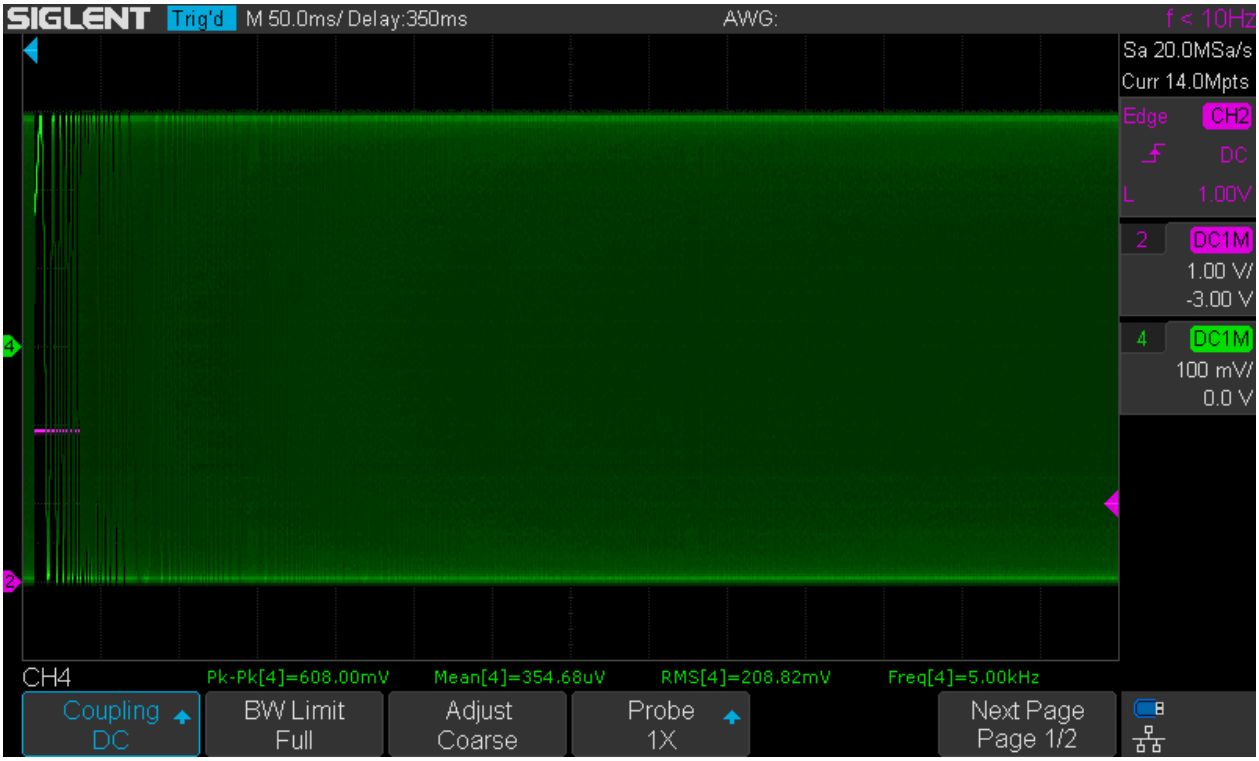
- Lower bandwidth limit with AC input coupling
- Gain flatness at the crossover region in the split path input buffer

The first topic is easily measured and the Siglent 1104X-E has its lower bandwidth limit at about 1.2Hz for -3dB and 3Hz for -1dB. This is a good choice, as with these values AC coupling covers most practical AC signal applications, while response to DC offset voltage changes is still reasonably fast.

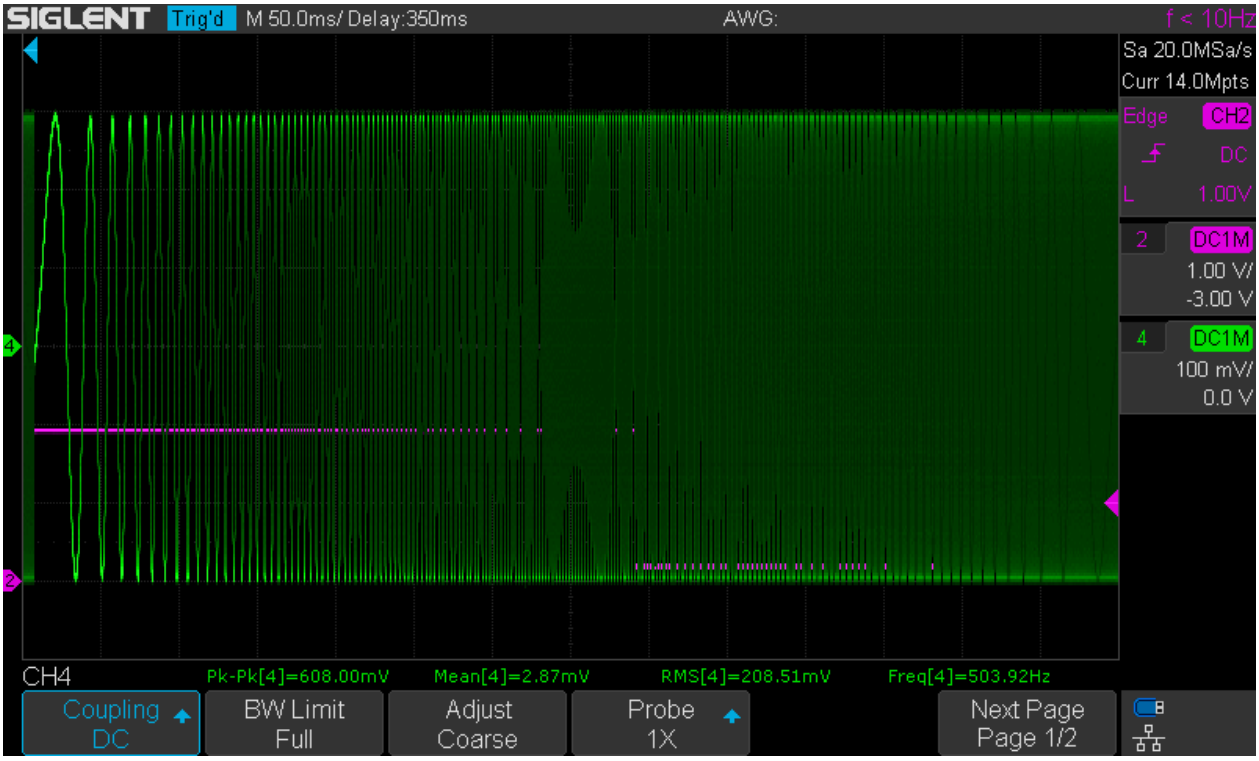
A quick check of the proper design and adjustment of the crossover networks for the split path input buffer can be done by viewing a 100Hz square wave. This is independent of input coupling, so DC coupling is used again.

SDS1104X-E_LF_Square_100Hz

This doesn't look particularly bad, though not 100% perfect, but this could also be the signal source. Another test method is just checking the frequency response in the ranges 10Hz to 10kHz and 1kHz.



SDS1104X-E_LF_SWEEP_10Hz-10kHz



SDS1104X-E_LF_SWEEP_10Hz-1kHz

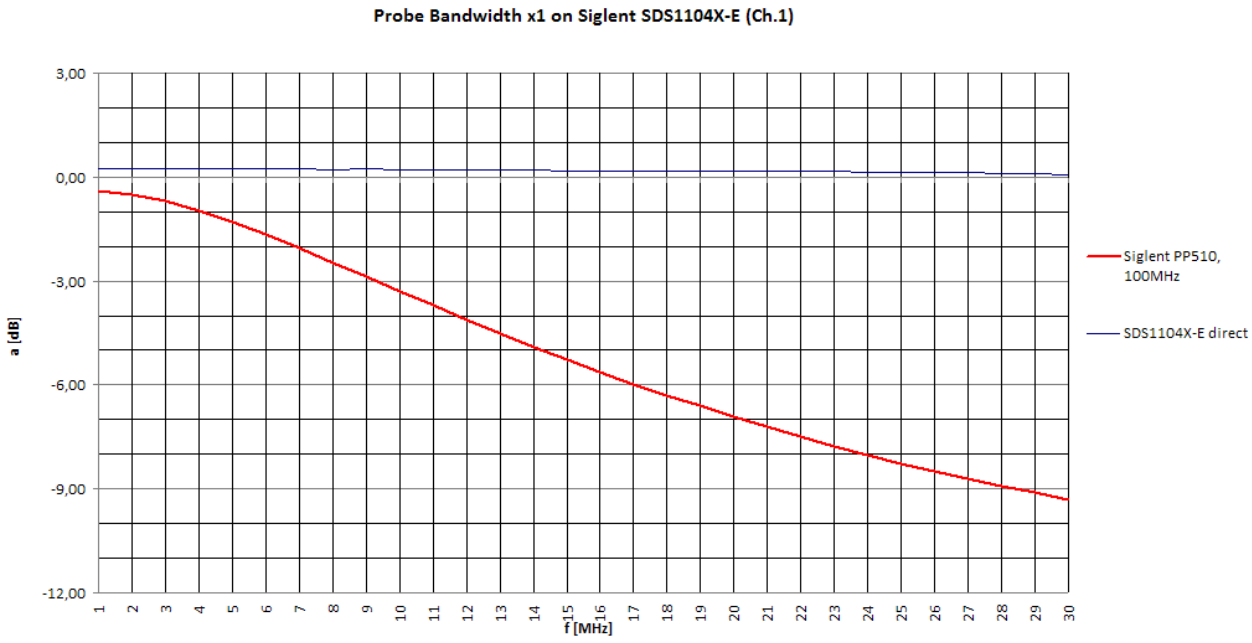
Frequency response is certainly flat. So the split path input buffer is well designed and properly adjusted.

Probe Bandwidth PP510

The Siglent SDS1104X-E came with slim 100MHz probes PP510 with switchable attenuation x1 and x10.

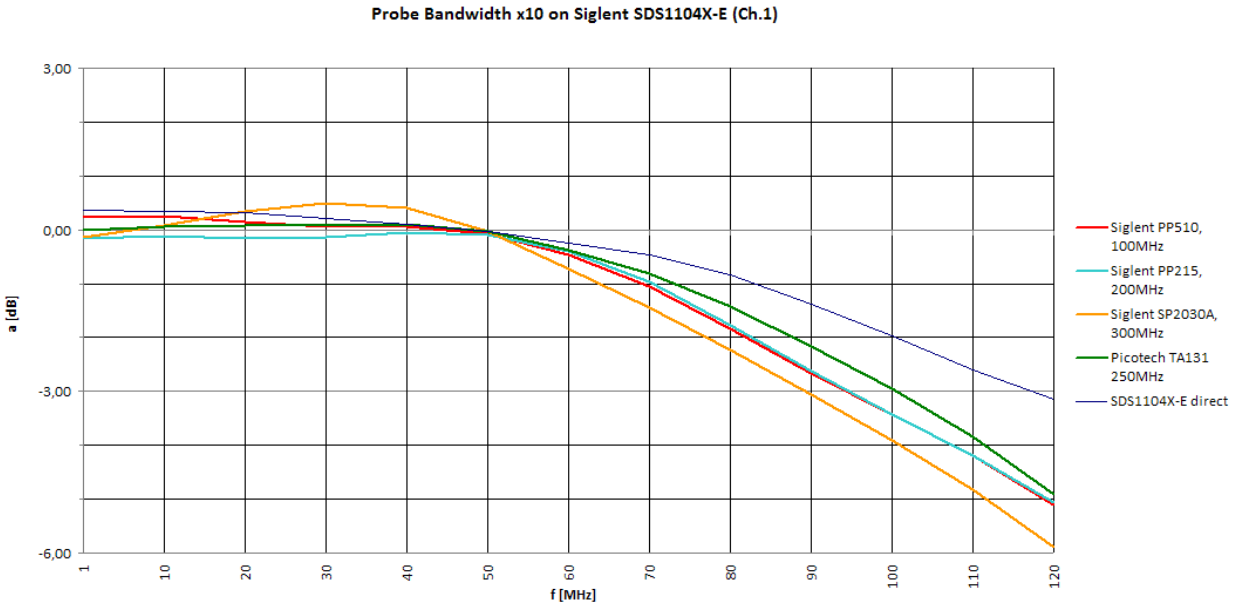
As we all know, x1 mode is of limited use because of the high capacitive loading (approx. 100pF) and limited bandwidth. The most common application would be checking power supply rails for ripple and noise.

The following graph shows the frequency response with a 3dB bandwidth slightly above 9MHz for 50 ohms source impedance.

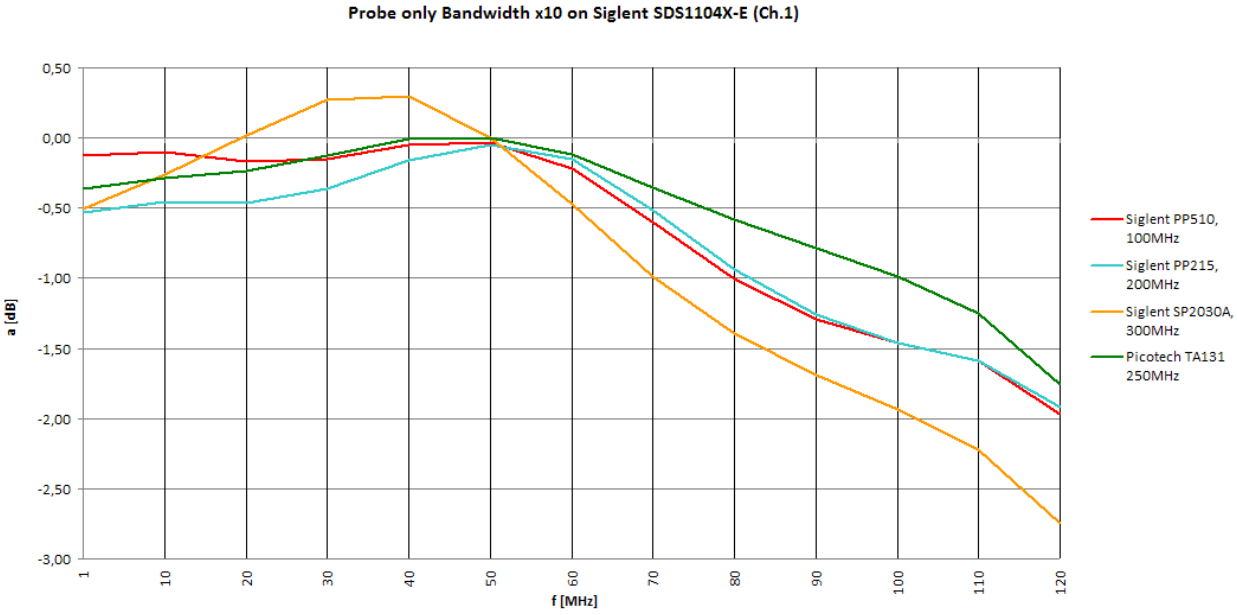


SDS1104X-E_Probe_x1_BW

The following screenshot shows the frequency response of the SDS1104X-E with 4 different x10 probes and with direct coax connection as a reference.



SDS1104X-E_Probe_x10_BW



SDS1104X-E_ProbeOnly_x10_BW

The screenshot above shows the difference between each probe and the direct connection. The following probes have been tested:

- Siglent PP510, 100MHz
- Siglent PP215, 200MHz
- Siglent SP2030A, 300MHz
- Picotech TA131, 250MHz

It can be seen that the supplied standard probe PP510 is not bad at all; up to 50MHz it is clearly the best choice. Above that frequency, the Picotech TA131 performs a little better, but the difference is still just 0.5dB @ 100MHz.

Some might think a faster probe is always better, but that's clearly not the case when looking e.g. at the SP2030A. At the same time, this very same probe works perfectly fine together with an SDS2304X scope and extends its bandwidth up to 450MHz! Probes should be matched with the particular scope input characteristics and the Siglent PP510 certainly is a pretty good match for the SDS1104X-E. Yet it should be noted that the differences are not huge and probe matching is generally a less critical issue for slower scopes.

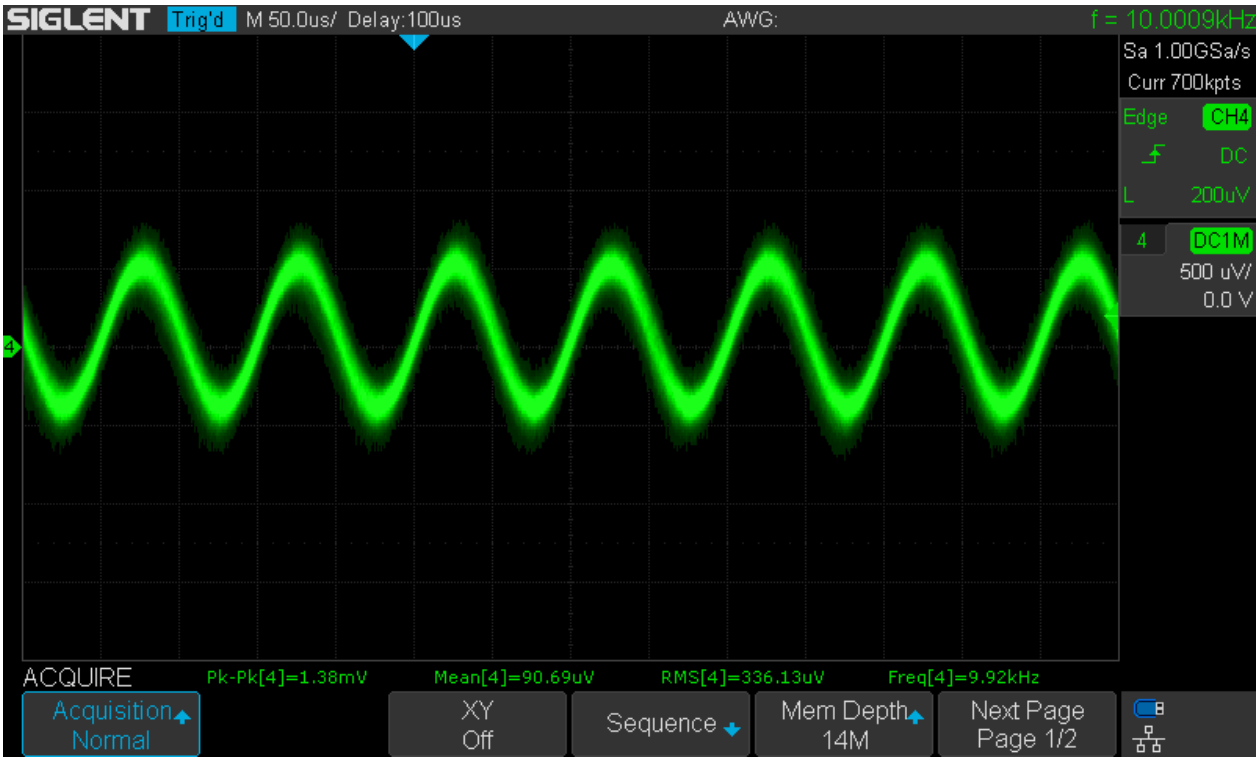
Noise

This scope provides an exceptional high input sensitivity of 500µV/div at full bandwidth. Of course the actual sensitivity is limited by the input noise of the scope – which cannot be low in an overvoltage-protected high impedance input amplifier. This also shows by the fact that the scope noise is more or less independent of the external load impedance. In other words, it remains the same, whether the input is shorted to ground or left open.

Whenever screenshots are published to demonstrate the low noise of a scope, we usually can assume that the scope settings have been carefully selected to make the results look good. Noise performance is affected by so many parameters, like bandwidth, memory depth, sample speed and screen update rate, which are all directly proportional to the exhibited noise amplitude.

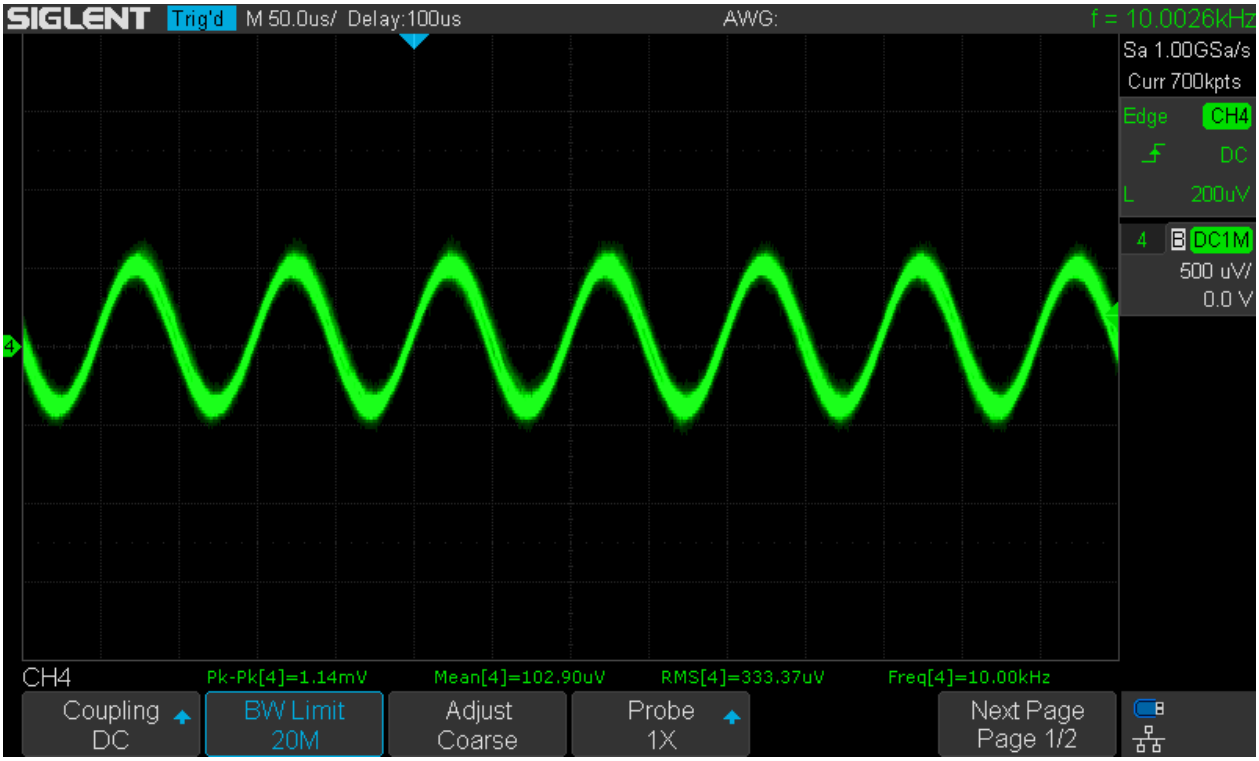
In this review, we do the opposite by using a fairly slow timebase, where the 1/f noise already takes its toll and we can see 700kpts mapped to the screen, yet some 300 waveforms per second, resulting in a total of about 7.7Mpts per screen refresh – which is quite a lot of data to look at in a single picture.

The screenshot below shows a 10kHz sine with only 1mVpp amplitude. The trace is fat and noisy, the peak to peak measurement reads much too high (+38%) because of the noise, but Vrms and frequency measurements are pretty close and overall, this is still a fairly decent result.



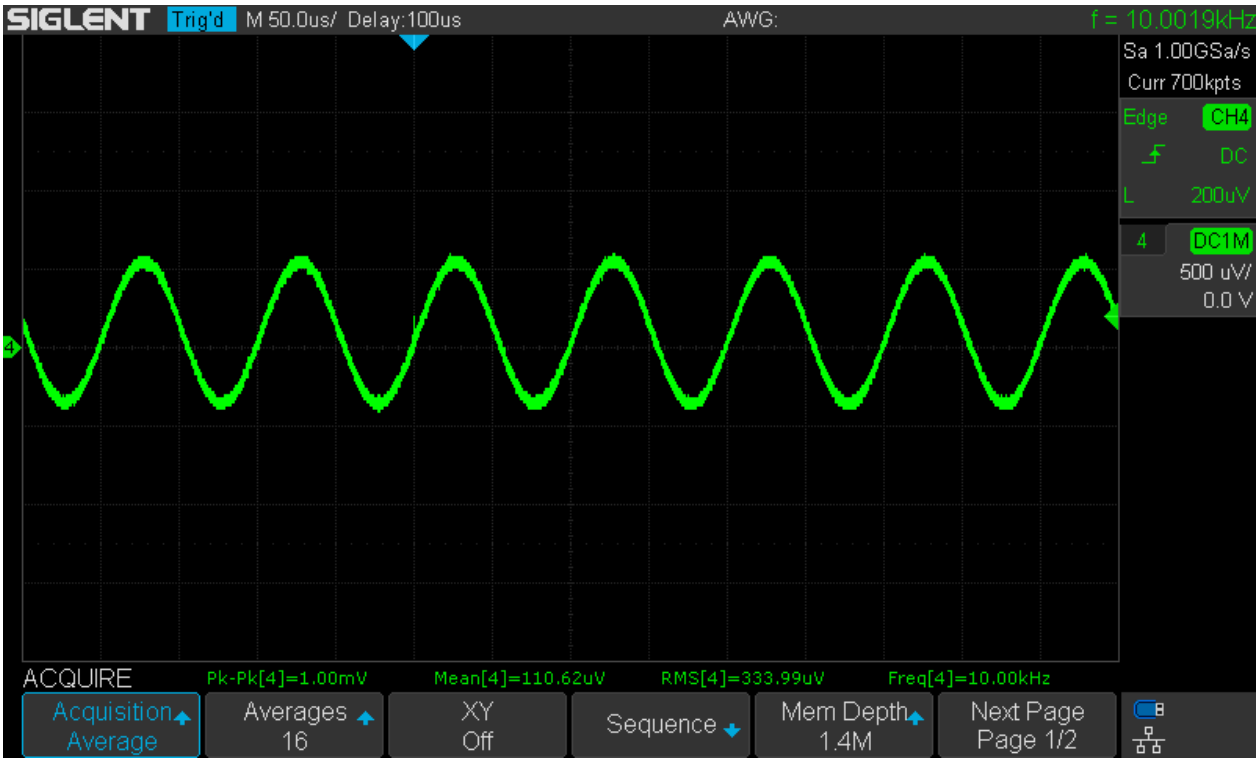
Noise_10kHz_1mVpp_normal_full

With the 20MHz bandwidth limit, noise is significantly reduced already. Even the very sensitive peak to peak voltage measurement gets much closer and exhibits only 14% error now.



Noise_10kHz_1mVpp_normal_limit

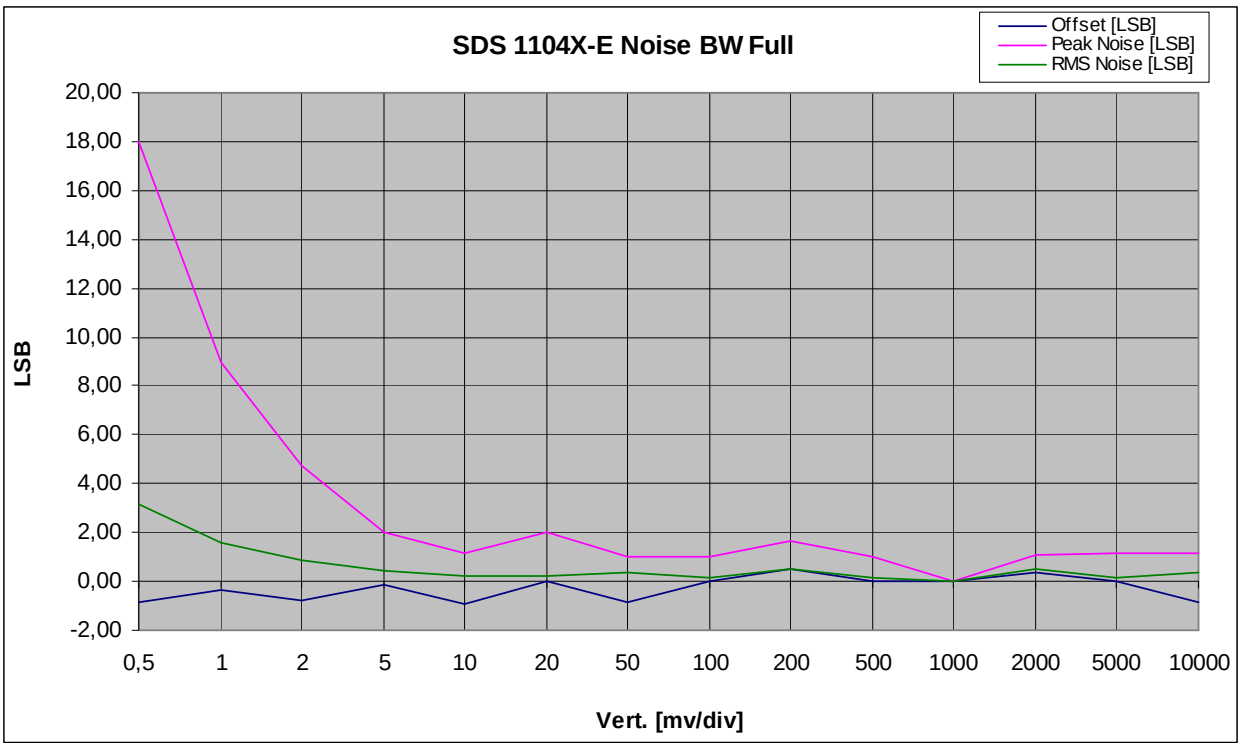
Instead of an input bandwidth limit, we can use the average acquisition mode. With just 16 averages, noise isn't an issue anymore and all automatic measurements are pretty much spot-on.



Noise_10kHz_1mVpp_avg16_full

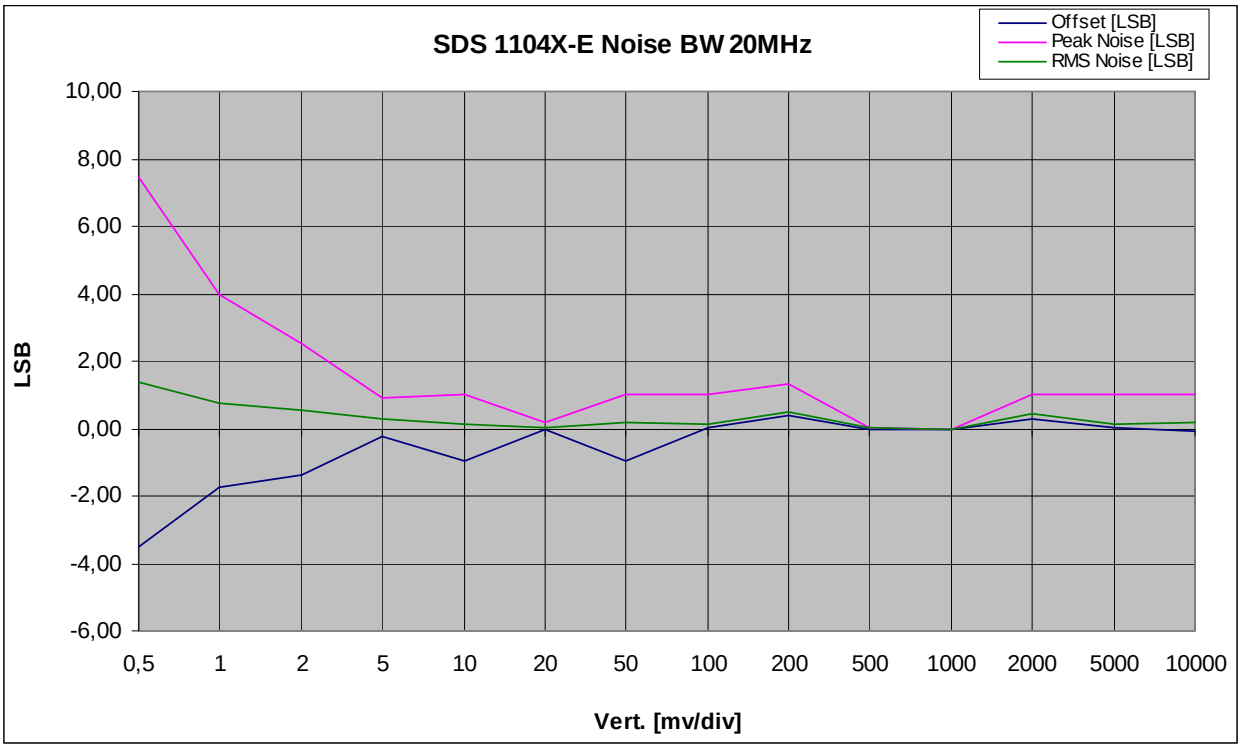
What has been demonstrated before is just one isolated use case – and with a signal frequency of just 10kHz certainly hasn't hit the sweet spot. So a more detailed examination is in order. The following diagrams show offset, peak-peak and RMS noise for all vertical gain settings at 50ns/div, for full bandwidth and also with 20MHz bandwidth limit.

SDS 1104X-E Noise BW Full								
Gain [mV/div]	**Noise [mVpp]**	**Noise [mVrms]**	**Mean [mV]**	**Offset [mV]**	**Offset [LSB]**	**Peak Noise [LSB]**	**RMS Noise [LSB]**	**LSB [mV]**
0,5	0,360	0,063	-0,018	-0,02	-0,88	18,0	3,1	0,020
1	0,360	0,064	-0,015	-0,02	-0,38	9,0	1,6	0,040
2	0,380	0,069	-0,061	-0,06	-0,76	4,8	0,9	0,080
5	0,403	0,084	-0,023	-0,02	-0,12	2,0	0,4	0,200
10	0,464	0,090	-0,377	-0,38	-0,94	1,2	0,2	0,400
20	1,590	0,195	0,022	0,02	0,03	2,0	0,2	0,800
50	2,000	0,702	-1,700	-1,70	-0,85	1,0	0,4	2,000
100	3,980	0,633	0,121	0,12	0,03	1,0	0,2	4,000
200	13,100	4,000	4,120	4,12	0,52	1,6	0,5	8,000
500	19,940	3,030	-0,527	-0,53	-0,03	1,0	0,2	20,000
1000	0,200	0,011	-0,001	0,00	0,00	0,0	0,0	40,000
2000	84,000	38,100	29,100	29,10	0,36	1,1	0,5	80,000
5000	229,000	30,500	5,120	5,12	0,03	1,1	0,2	200,000
10000	468,000	152,000	-328,000	-328,00	-0,82	1,2	0,4	400,000



SDS1104X-E Noise BW_full

SDS 1104X-E Noise BW 20MHz								
Gain [mV/div]	Noise [mVpp]	Noise [mVrms]	Mean [mV]	Offset [mV]	Offset [LSB]	Peak Noise [LSB]	RMS Noise [LSB]	LSB [mV]
0,5	0,150	0,028	-0,070	-0,07	-3,50	7,5	1,4	0,020
1	0,160	0,031	-0,070	-0,07	-1,75	4,0	0,8	0,040
2	0,200	0,044	-0,111	-0,11	-1,39	2,5	0,6	0,080
5	0,180	0,057	-0,051	-0,05	-0,26	0,9	0,3	0,200
10	0,400	0,058	-0,390	-0,39	-0,98	1,0	0,1	0,400
20	0,140	0,008	-0,001	0,00	0,00	0,2	0,0	0,800
50	2,000	0,323	-1,940	-1,94	-0,97	1,0	0,2	2,000
100	3,980	0,618	0,111	0,11	0,03	1,0	0,2	4,000
200	10,730	3,770	3,080	3,08	0,39	1,3	0,5	8,000
500	0,600	0,025	-0,002	0,00	0,00	0,0	0,0	20,000
1000	0,000	0,000	0,000	0,00	0,00	0,0	0,0	40,000
2000	80,000	34,000	22,000	22,00	0,28	1,0	0,4	80,000
5000	200,000	31,000	5,750	5,75	0,03	1,0	0,2	200,000
10000	405,000	80,500	-38,000	-38,00	-0,10	1,0	0,2	400,000



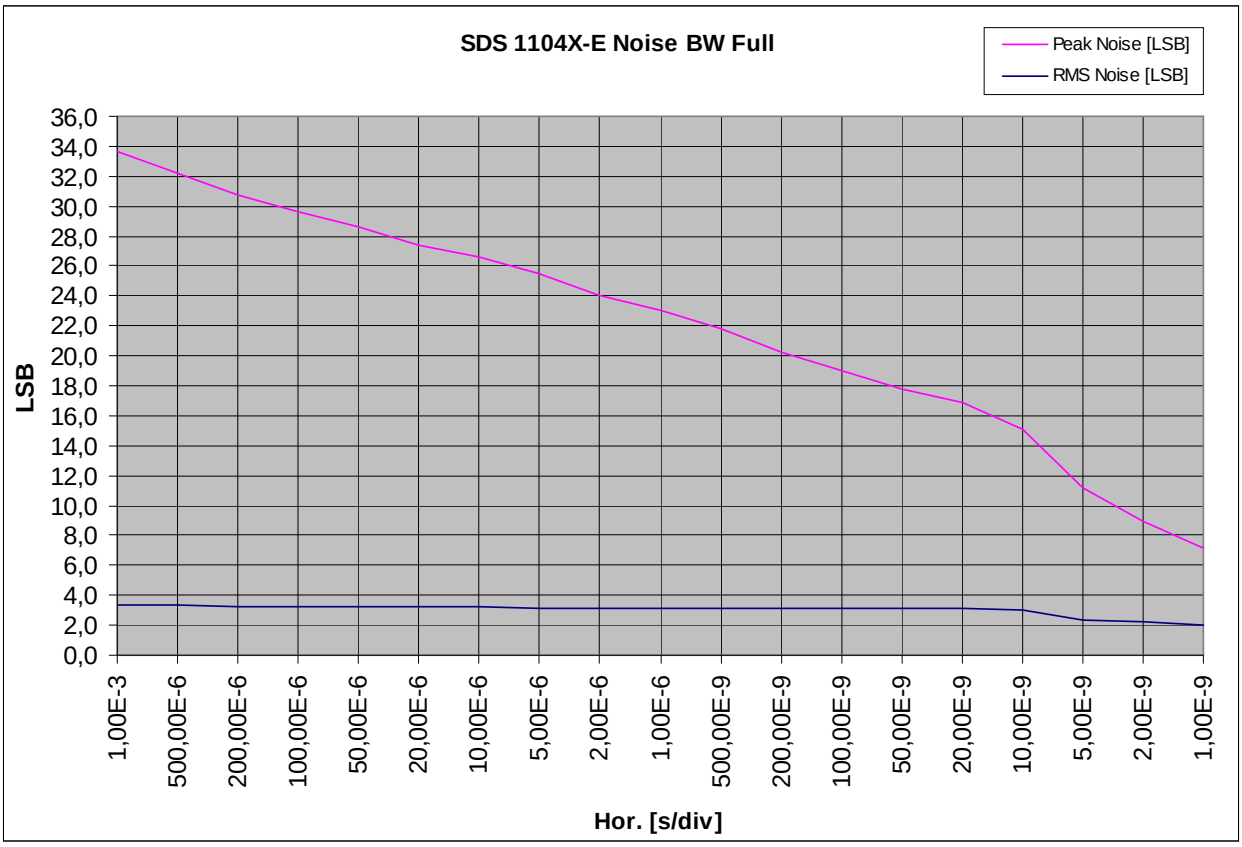
SDS1104X-E Noise BW_20M

It turns out that the ranges 10, 50, 100 and 500mV/div and 1 to 10V/div are really low nose with a peak amplitude of no more than 1ADC LSB. The 1V/div is particularly almost noise-free.

Noise not only depends on the total bandwidth, but also the lower bandwidth limit, for at least two reasons.

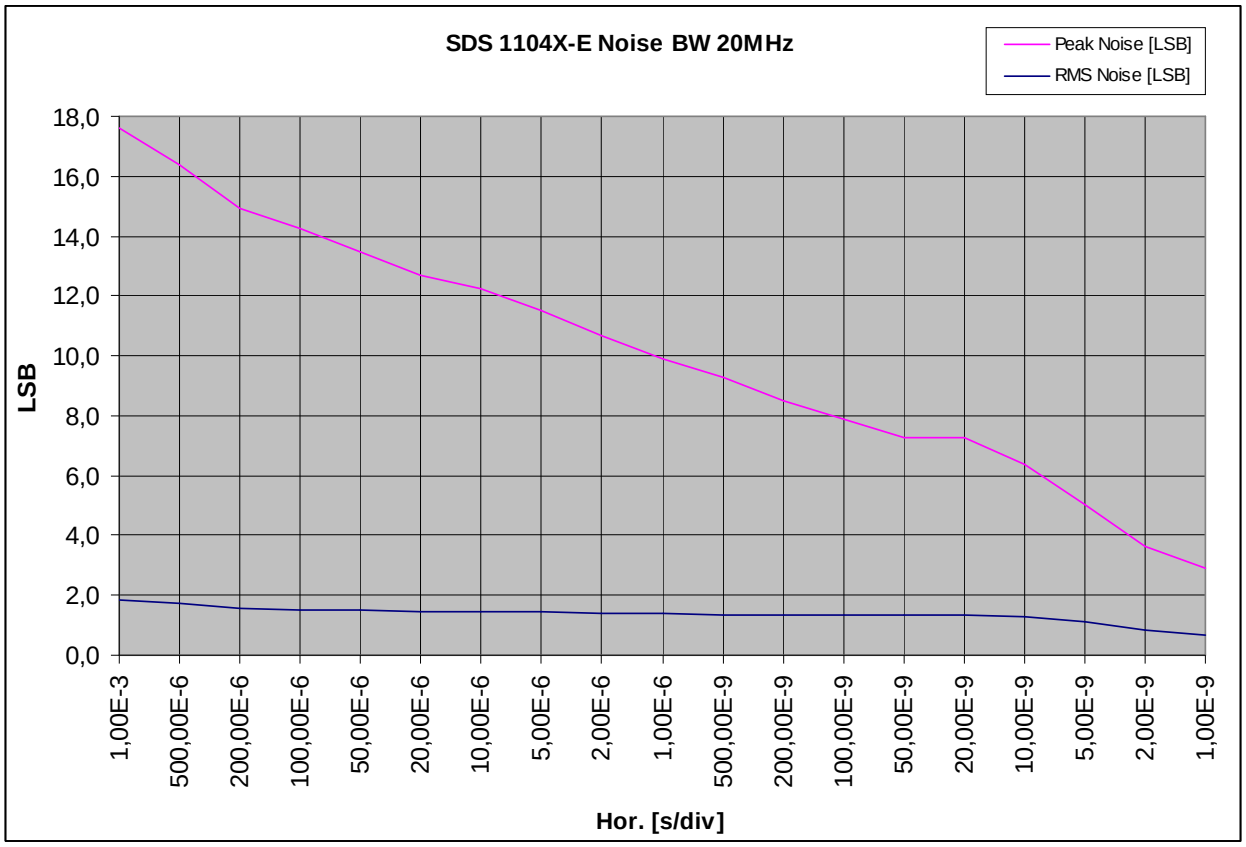
- 1/f noise of the high impedance FET input amplifier
- DC overvoltage protection for the DC path of the input amplifier results in high source impedance and attenuation on top of that, which needs to be compensated by additional amplifier gain.

SDS 1104X-E Noise, Full BW			Max.SR	1,00E+9	Max. Mem	14,00E+6	
Timebase [s/div]	Noise [mVpp]	Noise [mVrms]	Peak Noise [LSB]	RMS Noise [LSB]	LSB [mV]	Mem [Pts]	NOISE fc_low [Hz]
1,00E-3	0,673	0,068	33,7	3,4	0,020	14,00E+6	71E+0
500,00E-6	0,643	0,067	32,2	3,3	0,020	7,00E+6	143E+0
200,00E-6	0,615	0,065	30,8	3,2	0,020	2,80E+6	357E+0
100,00E-6	0,592	0,064	29,6	3,2	0,020	1,40E+6	714E+0
50,00E-6	0,573	0,064	28,7	3,2	0,020	700,00E+3	1E+3
20,00E-6	0,548	0,064	27,4	3,2	0,020	280,00E+3	4E+3
10,00E-6	0,533	0,064	26,7	3,2	0,020	140,00E+3	7E+3
5,00E-6	0,509	0,064	25,5	3,2	0,020	70,00E+3	14E+3
2,00E-6	0,480	0,063	24,0	3,2	0,020	28,00E+3	36E+3
1,00E-6	0,460	0,063	23,0	3,1	0,020	14,00E+3	71E+3
500,00E-9	0,435	0,063	21,8	3,1	0,020	7,00E+3	143E+3
200,00E-9	0,405	0,062	20,3	3,1	0,020	2,80E+3	357E+3
100,00E-9	0,381	0,062	19,1	3,1	0,020	1,40E+3	714E+3
50,00E-9	0,356	0,062	17,8	3,1	0,020	700,00E+0	1E+6
20,00E-9	0,337	0,062	16,9	3,1	0,020	280,00E+0	4E+6
10,00E-9	0,302	0,060	15,1	3,0	0,020	140,00E+0	7E+6
5,00E-9	0,223	0,047	11,2	2,4	0,020	70,00E+0	14E+6
2,00E-9	0,180	0,044	9,0	2,2	0,020	28,00E+0	36E+6
1,00E-9	0,143	0,039	7,2	2,0	0,020	14,00E+0	71E+6



SDS1104X-E Noise_vs_time BW_full

SDS 1104X-E Noise, BW 20MHz			Max.SR	1,00E+9	Max. Mem	14,00E+6	
Timebase [s/div]	Noise [mVpp]	Noise [mVrms]	Peak Noise [LSB]	RMS Noise [LSB]	LSB [mV]	Mem [Pts]	NOISE fc_low [Hz]
1,00E-3	0,352	0,037	17,6	1,9	0,020	14,00E+6	71E+0
500,00E-6	0,328	0,034	16,4	1,7	0,020	7,00E+6	143E+0
200,00E-6	0,298	0,031	14,9	1,6	0,020	2,80E+6	357E+0
100,00E-6	0,285	0,030	14,3	1,5	0,020	1,40E+6	714E+0
50,00E-6	0,270	0,030	13,5	1,5	0,020	700,00E+3	1E+3
20,00E-6	0,254	0,030	12,7	1,5	0,020	280,00E+3	4E+3
10,00E-6	0,245	0,029	12,3	1,5	0,020	140,00E+3	7E+3
5,00E-6	0,230	0,029	11,5	1,5	0,020	70,00E+3	14E+3
2,00E-6	0,214	0,028	10,7	1,4	0,020	28,00E+3	36E+3
1,00E-6	0,198	0,028	9,9	1,4	0,020	14,00E+3	71E+3
500,00E-9	0,186	0,027	9,3	1,4	0,020	7,00E+3	143E+3
200,00E-9	0,170	0,027	8,5	1,3	0,020	2,80E+3	357E+3
100,00E-9	0,158	0,027	7,9	1,3	0,020	1,40E+3	714E+3
50,00E-9	0,145	0,027	7,3	1,3	0,020	700,00E+0	1E+6
20,00E-9	0,145	0,027	7,3	1,4	0,020	280,00E+0	4E+6
10,00E-9	0,128	0,026	6,4	1,3	0,020	140,00E+0	7E+6
5,00E-9	0,101	0,023	5,1	1,1	0,020	70,00E+0	14E+6
2,00E-9	0,073	0,017	3,7	0,8	0,020	28,00E+0	36E+6
1,00E-9	0,058	0,014	2,9	0,7	0,020	14,00E+0	71E+6



SDS1104X-E Noise_vs_time BW_20M

The graphs above show input noise for all timebase settings from 1ms/div to 1ns/div at maximum vertical sensitivity of 500µV/div. We can see a continuous 1/f characteristic for the peak to peak noise amplitude. In contrast, the RMS reading is fairly constant and clearly doesn't tell the full story.

Trigger

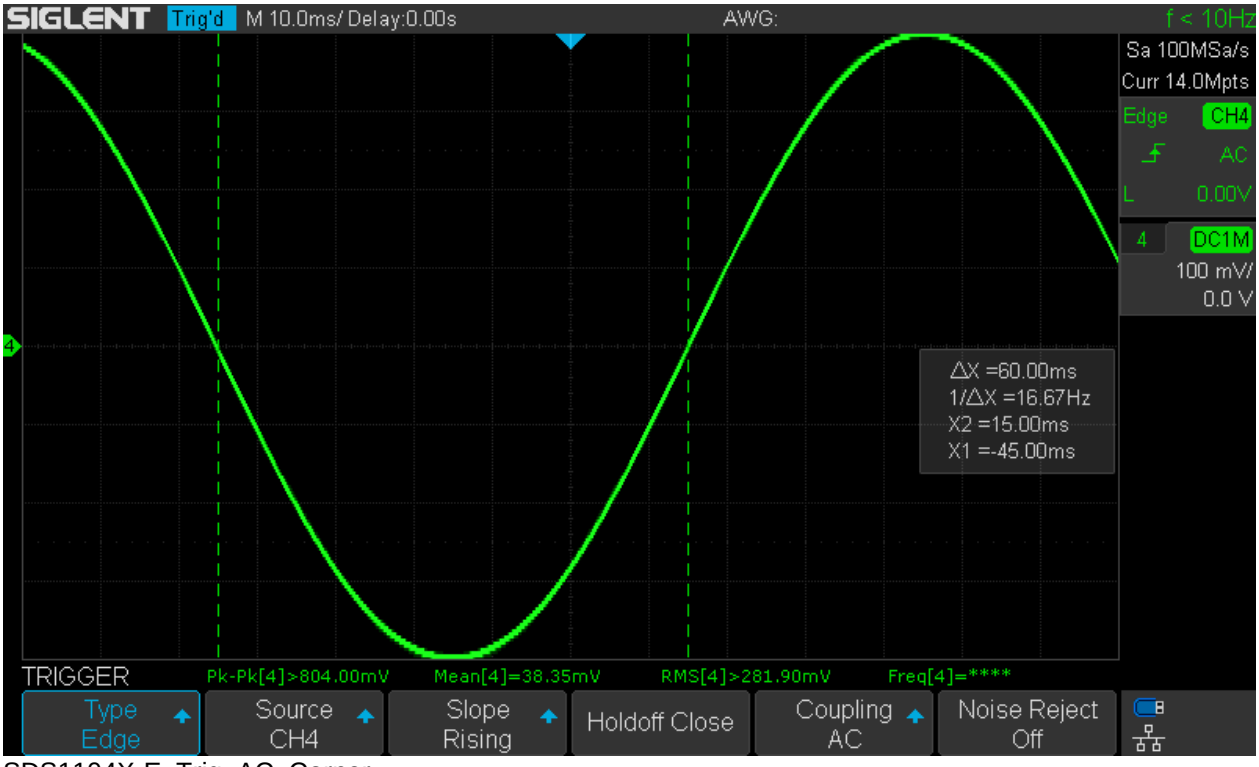
The trigger is one of the most important features of any oscilloscope, since we can only analyze a signal that we are able to trigger in a reliable and stable manner. Consequently, this section deals with the trigger system and its various options in the Siglent SDS1104X-E, which has a modern fully digital trigger system. Expectations are high, because triggering has been excellent on previous Siglent X-series DSOs.

Trigger Coupling

This gives us the opportunity of filtering the sampled input signal data (by means of digital signal processing) before the actual trigger condition is checked. If this is done in a proper way that suits the application, we can often get a stable trigger even on complex and/or noisy signals. There are four choices of trigger signal coupling and consequently filtering:

- DC: The entire input frequency range – no filtering.
- AC: High-pass filter with a lower corner frequency of about 8Hz
- LFRJ (LF-Reject): High-pass filter with a corner frequency of about 2MHz
- HFRJ (HF-Reject): Low-pass filter with a corner frequency of about 2.2MHz

The corner frequency of AC coupling can be determined by the phase shift between trigger point and signal.



SDS1104X-E_Trig_AC_Corner

Cursor measurement is used to determine the phase shift. The difference between both cursors X2-X1 is half the period of the input signal, hence its frequency is 1/(2 x 0.06) = 1/0.12 = 8.33Hz. The easier way is to just divide the frequency calculation of the cursor measurement by two: 16.67Hz/2 = 8.33Hz.

We have 45deg phase shift when the trigger reference point sits at precisely 75% of the distance between X1 and X2. Since X1 – Ref. = 45.00ms and X2 – Ref. = 15.00ms and 3 x 15.00 = 45.00, this condition is perfectly met in the screenshot above.

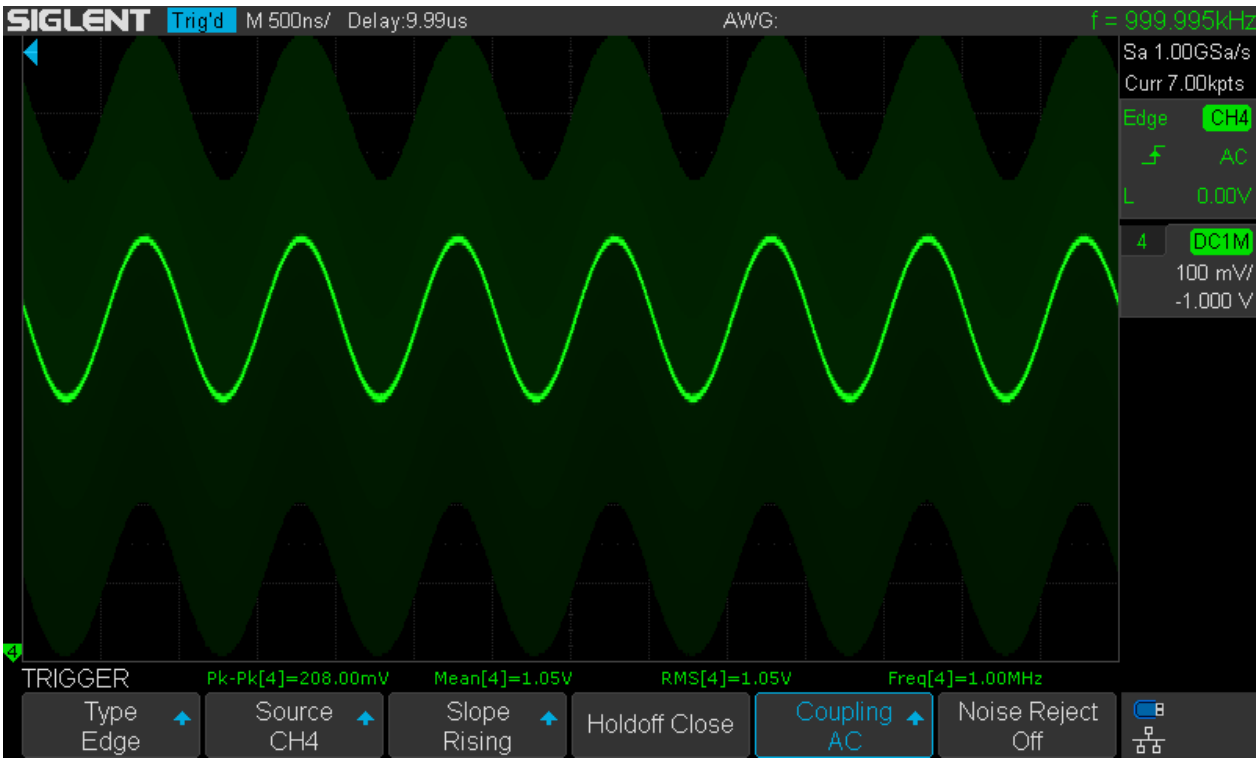
For LF/HF-Reject trigger coupling, the corner frequencies cannot be determined in the same way, as apparently a higher order filter is implemented. Instead, a reliable triggering on a 1div peak-peak signal has been used as criterion in order to get the numbers listed above.

AC Trigger Coupling

Most of the time we use DC coupling for the trigger, and there is not much to tell about it, other than it works just as it should. We rather want to examine AC trigger coupling now.

Why and when would we need AC coupling for the trigger at all? Usually, we make that choice for the channel input and if we select AC coupling there, the trigger will inevitably be AC coupled as well. So there we already have the answer – we have the opportunity to force the trigger into AC coupling, even when the corresponding input channel is DC coupled. This can be useful for AC signals that have a DC offset that we want to watch on the screen. The offset might change with time and we still don't want to lose triggering.
AC trigger coupling does not display a trigger level indicator, which I'm not happy with, but Siglent don't seem to be willing to change that.

The following test uses a 200mVpp 1MHz sine wave that is superimposed on a 600mVpp symmetrical ramp signal at 100mHz, which acts as a variable DC offset here. As if this weren't enough, this signal has a fixed offset of +1V on top of that, which needs to be removed by means of the vertical position control.



SDS1104X-E_Trig_AC

With DC trigger coupling, triggering would only occur about 1/3 of the total time in this scenario and even then the horizontal position would not be stable because of the ever changing trigger level (relative to the AC portion of the input signal).

It's totally different if we use AC trigger coupling. The trigger level is set to zero which is actually always equivalent to the mean level of a symmetrical input signal. With this, triggering occurs always at the same point on the X-axis, no matter what the DC offset or low frequency instantaneous signal level is. The waveform constantly changes its vertical position on the screen, but remains stable on the time axis.

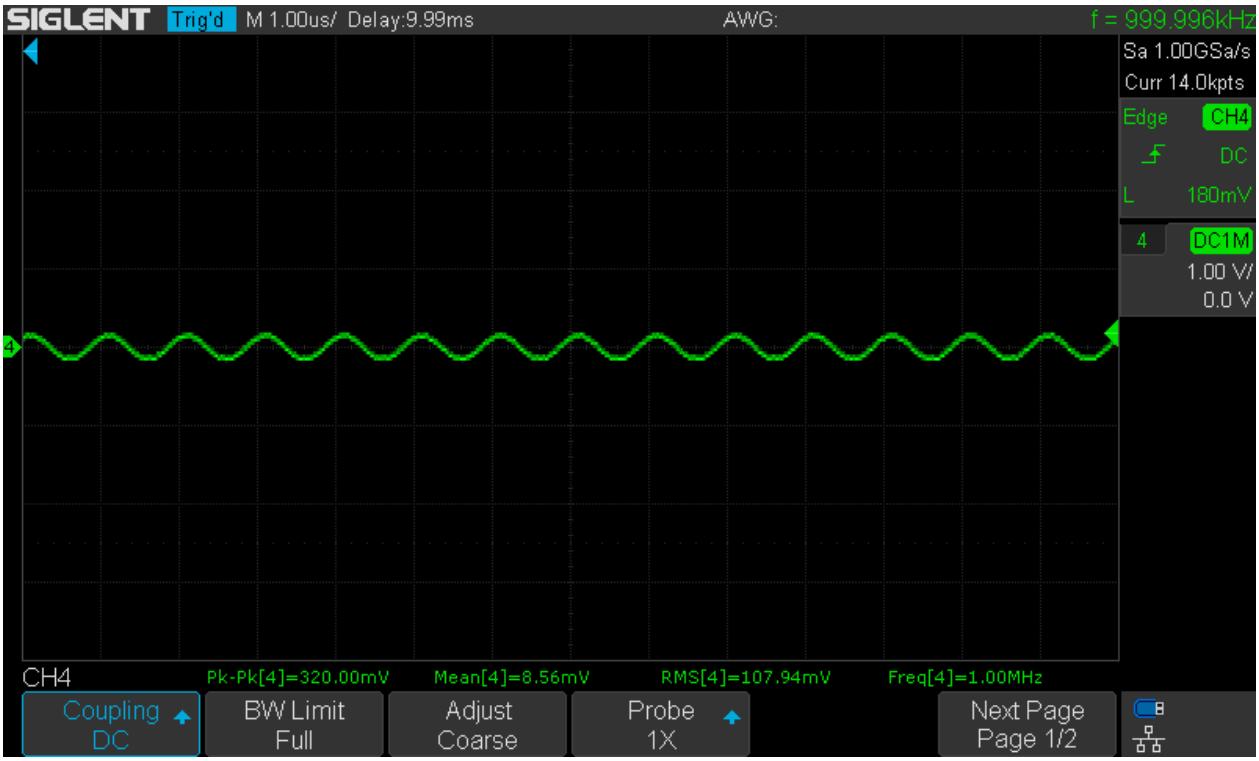
Persistence mode has been used for the screenshot above, to indicate the slow vertical movement of the signal.

Trigger Sensitivity

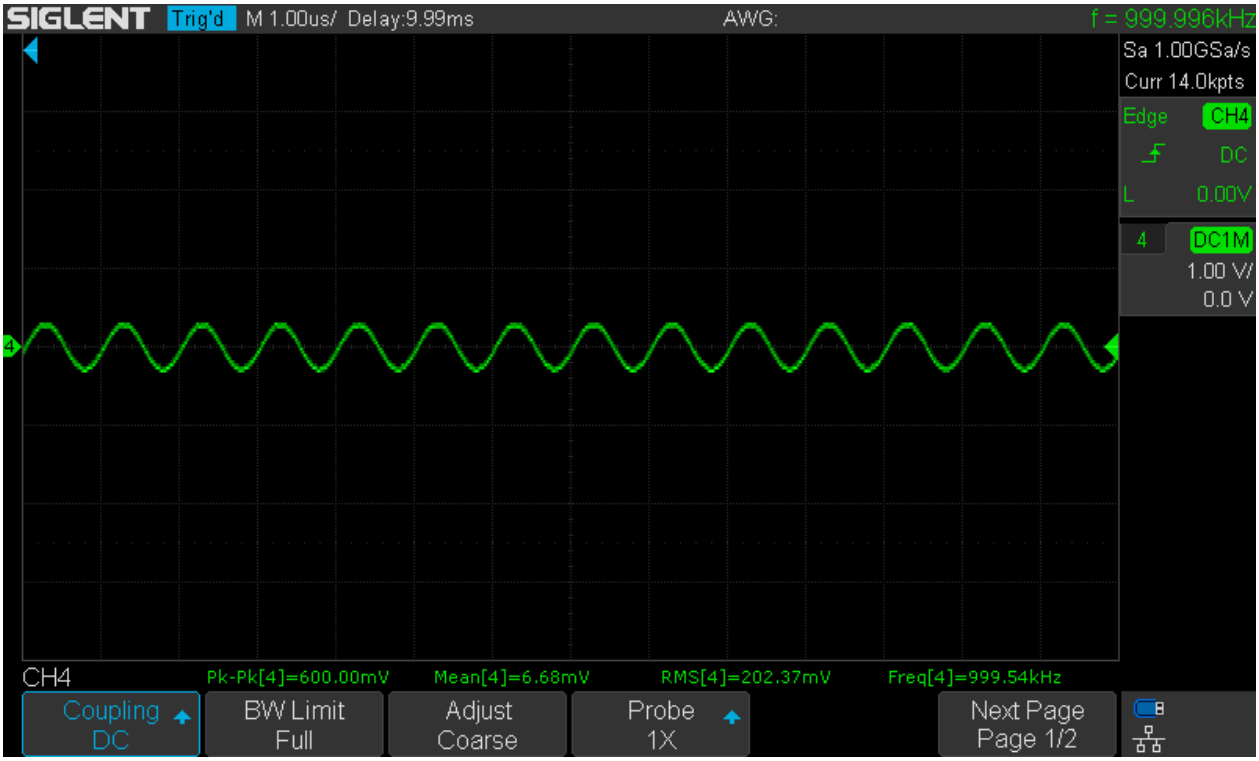
Trigger sensitivity in Edge mode depends on the Noise Reject setting (hysteresis), but also the trigger level, whether it is automatically set (by pushing the trigger level knob) or manually tweaked. Here are the sensitivities expressed in vertical divisions for all possible combinations:

- Noise Reject off, manual trigger level: **0.32** div.
- Noise Reject off, automatic trigger level: 0.60 div.
- Noise Reject on, manual trigger level: 0.88 div.
- Noise Reject on, automatic trigger level: 1.60 div.

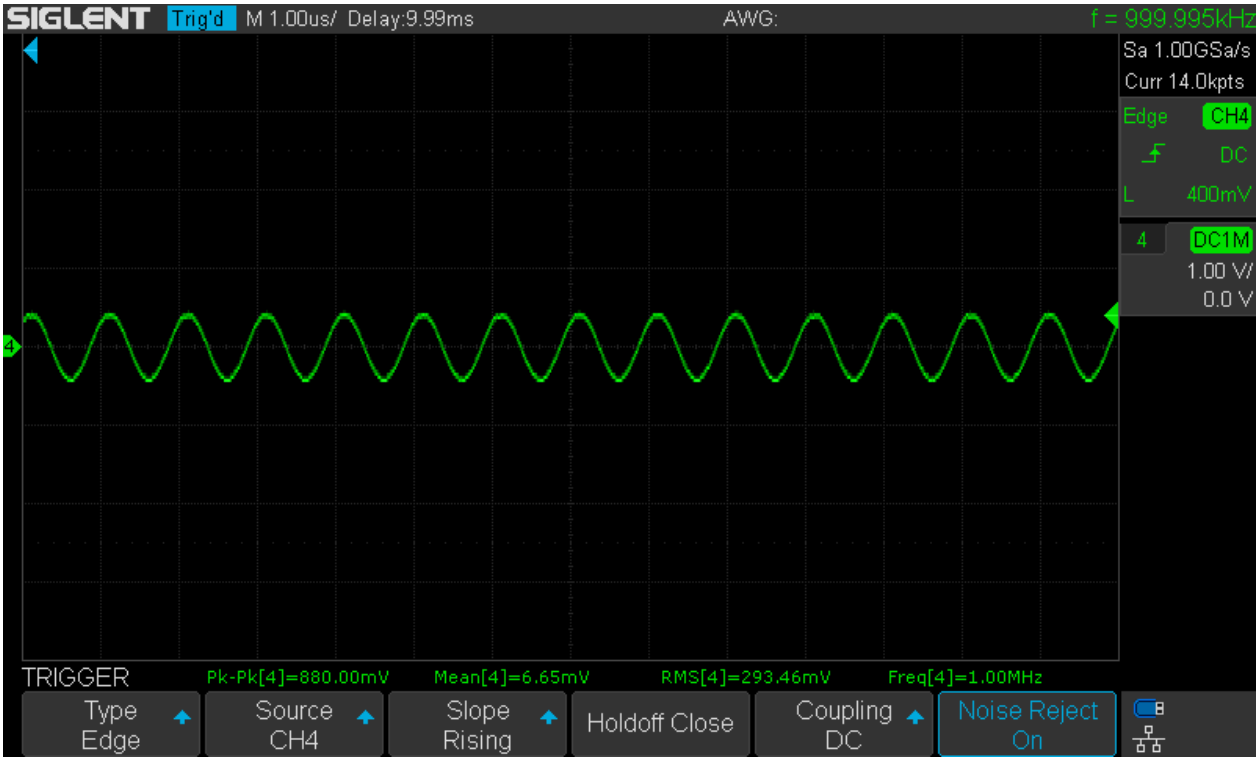
The following screenshots demonstrate all four settings listed above.



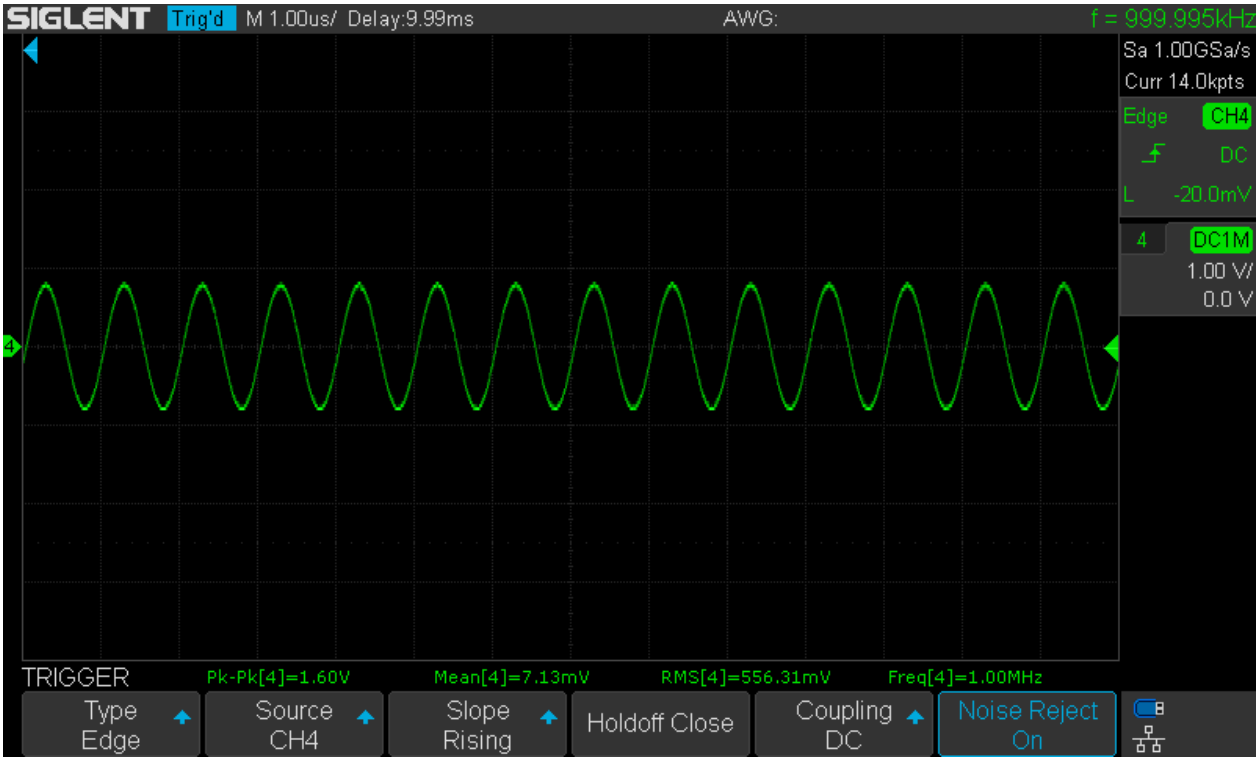
SDS1104X-E_Trig_32%_ML



SDS1104X-E_Trig_60%_AL



SDS1104X-E_Trig_NR_88%_ML



SDS1104X-E_Trig_NR_160%_AL

Trigger Stability

Another interesting aspect of a trigger system is stability, i.e. the absence of excessive jitter, as well as synchronous triggering on all channels so that accurate time measurements are possible.

Let's start with the stability of the trigger point which actually means stability of the trigger level and in case of a digital trigger – as it is implemented here – also the proper interpolation of samples to determine the exact trigger point.

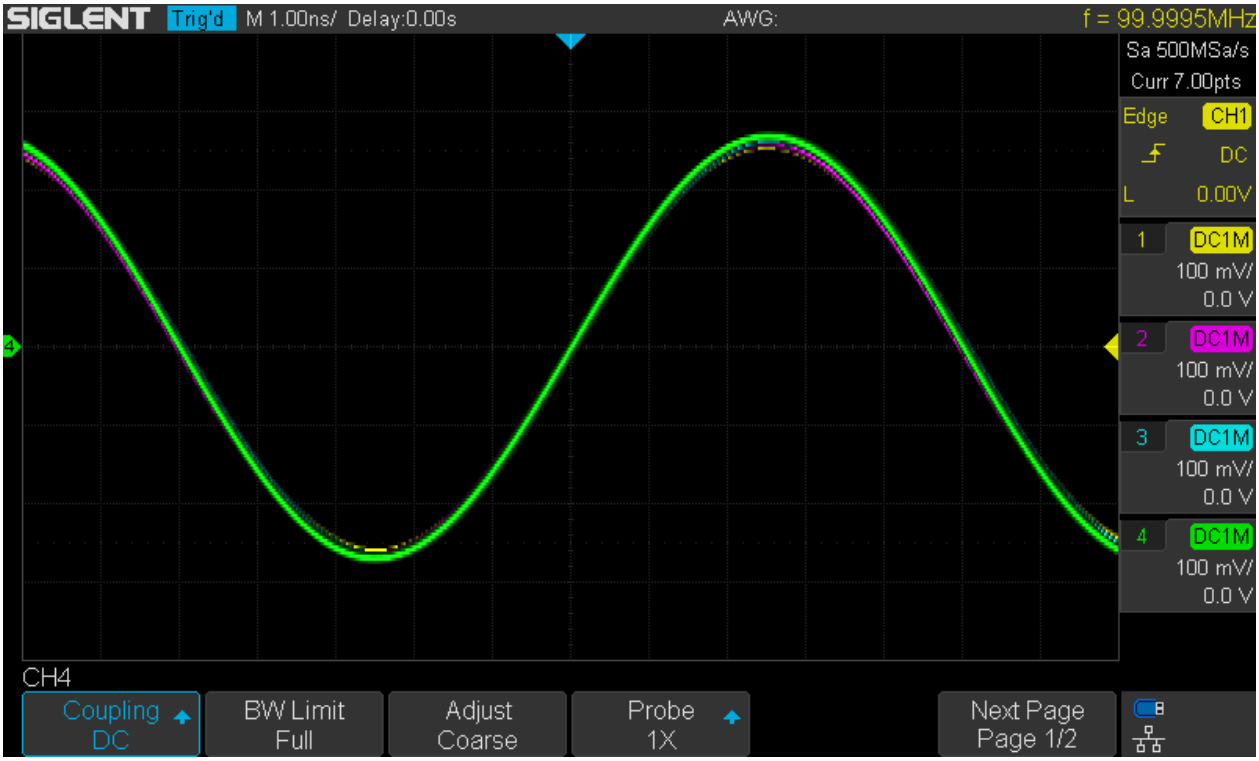
A very accurate and stable 100MHz signal derived from an OCXO is fed into channel 4 of the SDS1104X-E and the fastest timebase of 1ns/div is used together with its associated maximum time delay of 9.99µs. Generally, the trigger delay is limited to 9999 times the time base setting. Persistence mode with 10s decay time has been enabled in order to make any signal variation clearly visible on the screenshot.

Please note that the actual trigger point is far outside the visible screen area by some 714 screen widths and the delay has been fine adjusted so that the falling edge is exactly at the centre of the screen, while the original trigger is on the rising edge.

As can be seen, there is not the slightest sign of jitter in either direction. The signal trace is just rock-stable and automatic measurement has even accurately determined the signal frequency.

SDS1104X-E_Trig_1ns_D10us

The second test looks at the skew between channels, which of course should ideally be nonexistent, i.e. zero. A stable 100MHz signal was used to feed all four channels in parallel through a 4-way power splitter and 4 coaxial RG58 cables with the exact same length of 1 meter. No external input termination was used for the scope channels this time, as I didn't have four identical ones at hand. I certainly do not recommend feeding signals into the scope this way, let alone at high frequencies like this, but for this particular test it is acceptable and even so the result leaves nothing to be desired.



SDS1104X-E_Trig_1ns_Skew

At the fastest timebase of 1ns/div, we can see the small differences in the frequency response of the individual channels, as the amplitudes are slightly different, but there is barely any skew visible between channels. In any case, possible deviations are well below ±100ps and I don't have the means to guarantee that all four signals actually arrive at the scope inputs at the exact same picosecond anyway. However, this is a very pleasing result.

Triggering Noisy Signals

Triggering isn't always straightforward; complex signals, crosstalk and noise often require additional measures to get a stable signal trace that can be properly analyzed.

First decision we have to make is the proper trigger coupling. As has been stated earlier, this means limiting the trigger bandwidth in a way that suits the application. For this purpose, LF and HF reject trigger coupling are most important.

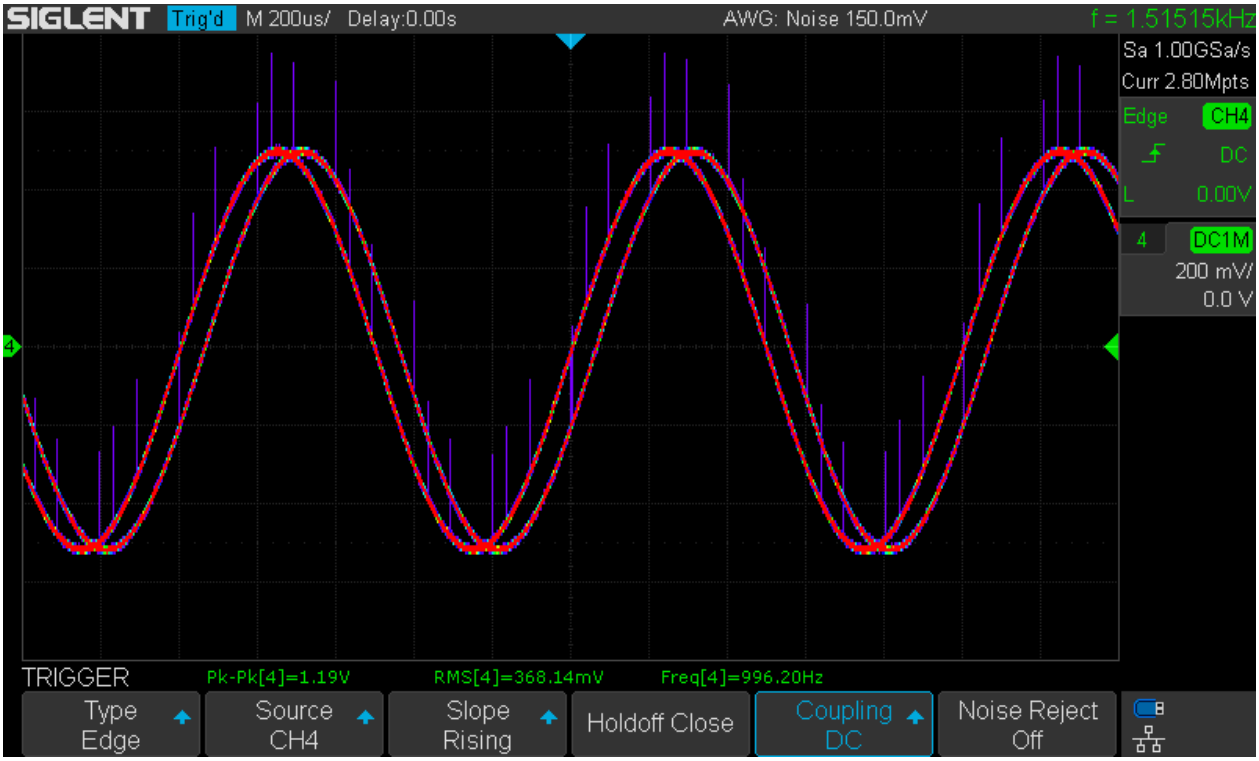
- LFRJ (LF-Reject): High-pass filter with a corner frequency of about 2MHz
- HFRJ (HF-Reject): Low-pass filter with a corner frequency of about 2.2MHz

Another means for aiding stable triggering on complex signals with a known period is the Hold-off time. This simply inhibits triggering for a certain customizable time window.

Finally, the trigger level hysteresis is a critical parameter as well. A higher hysteresis can help to deal with fuzzy signals, but this also reduces trigger sensitivity significantly. Consequently, the Siglent SDS1104X-E provides two levels of trigger hysteresis, which can be toggled by means of the [**Noise Reject**] menu button.

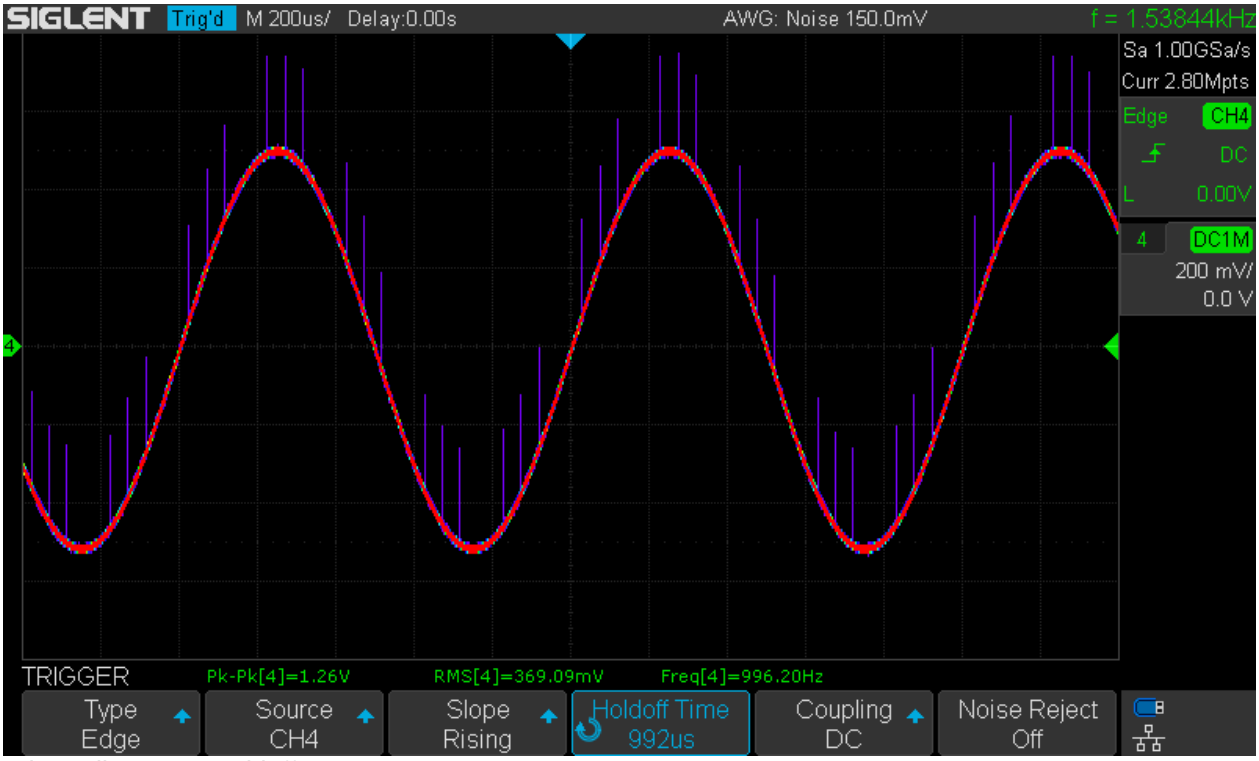
Now let's have a look at some practical examples for the use of trigger coupling, noise reject and hold-off to help with the triggering in certain situations.

First a 1kHz sine wave with random spikes superimposed. Trigger conditions are ambiguous and regular DC coupled edge trigger initially does not yield a stable waveform display.



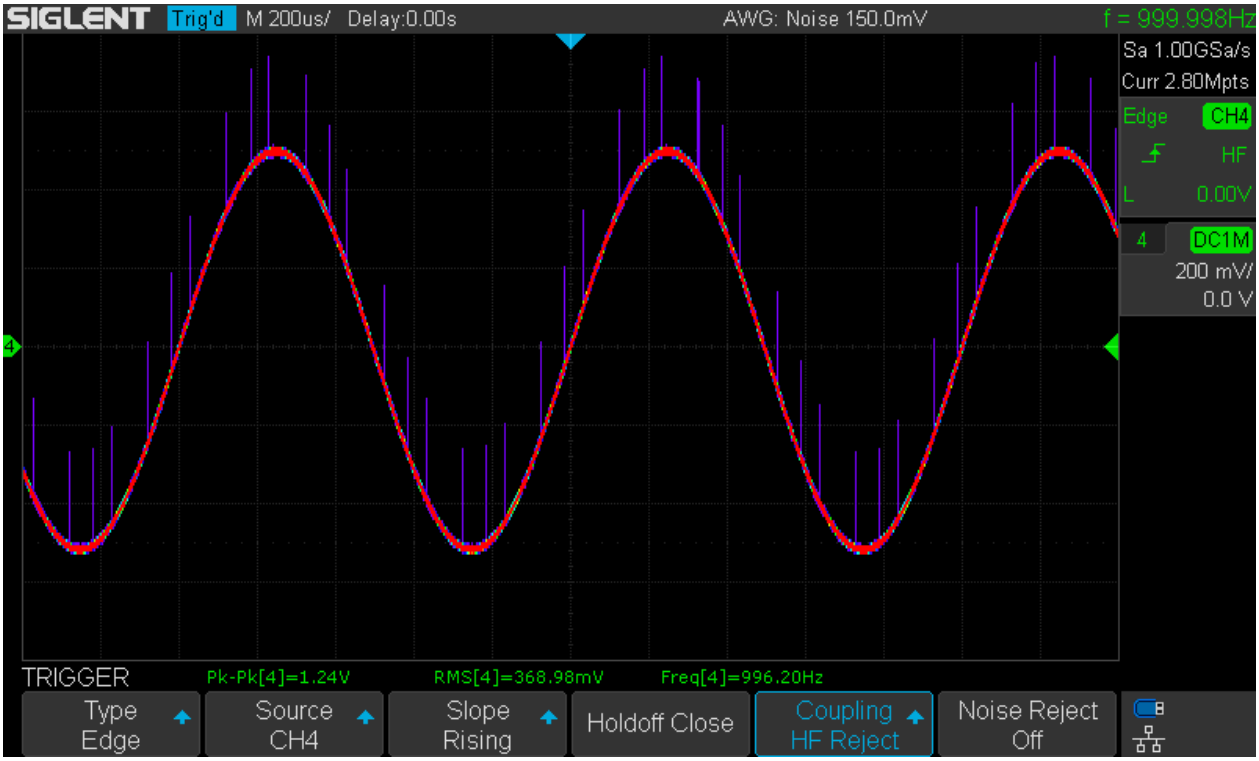
Trig_Spikes_DC

A hold-off time slightly less than the signal period works a treat.



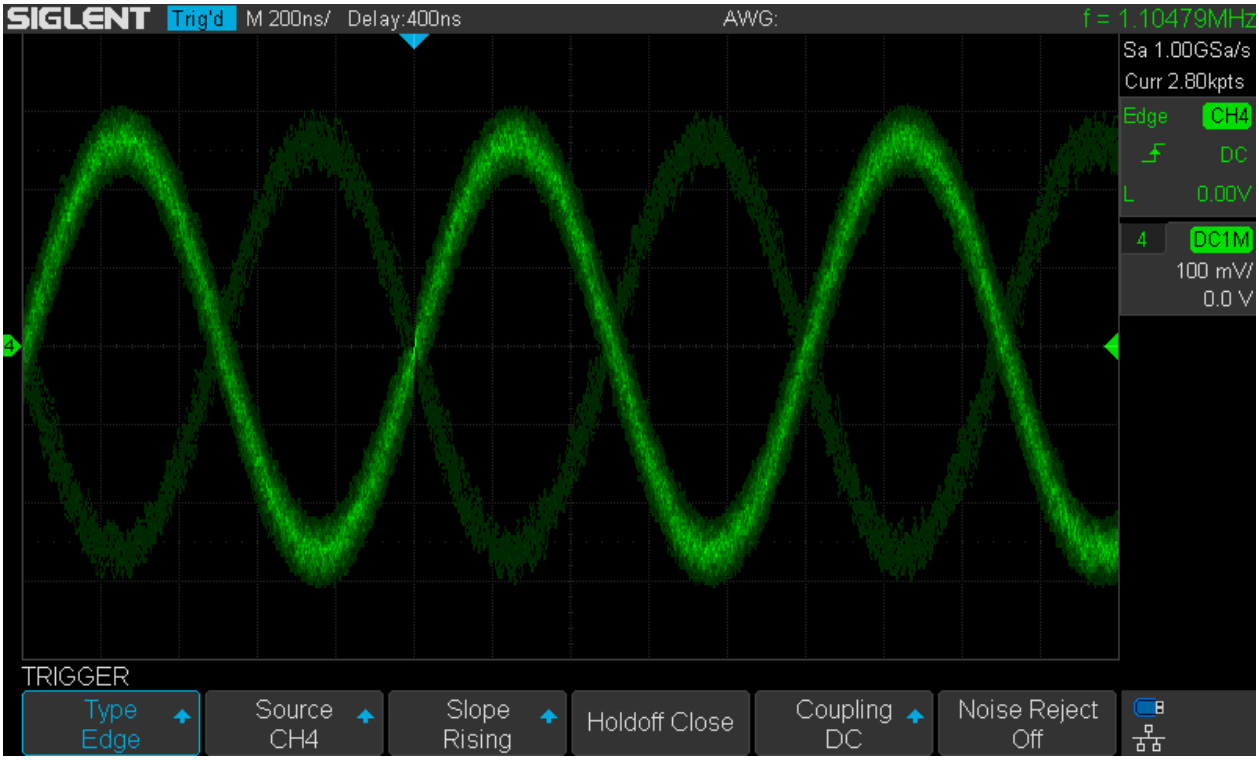
Trig_Spikes_DC_Holdoff

Alternatively, we can use HF-Reject trigger coupling (to ignore the spikes) to get a stable trigger.



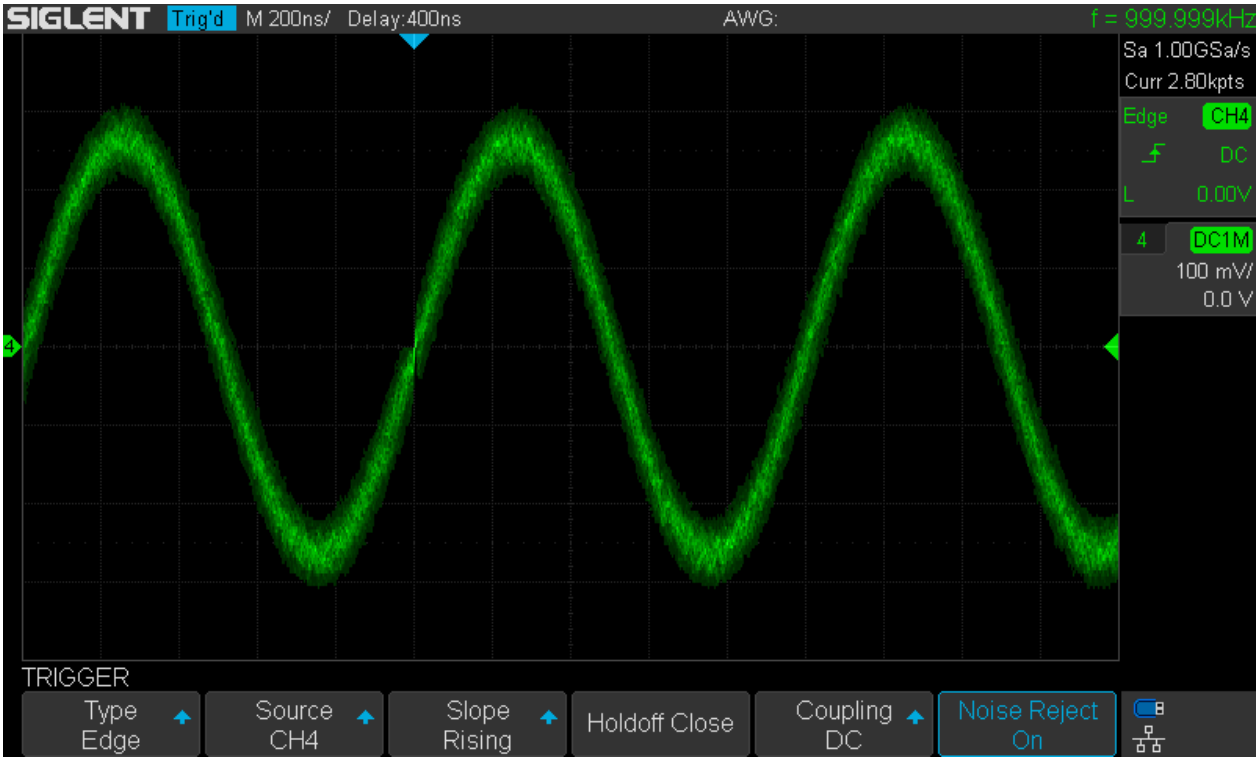
Trig_Spikes_HF_Reject

A similar example shows a very noisy 1MHz sine wave. Standard DC trigger does not work well at all.



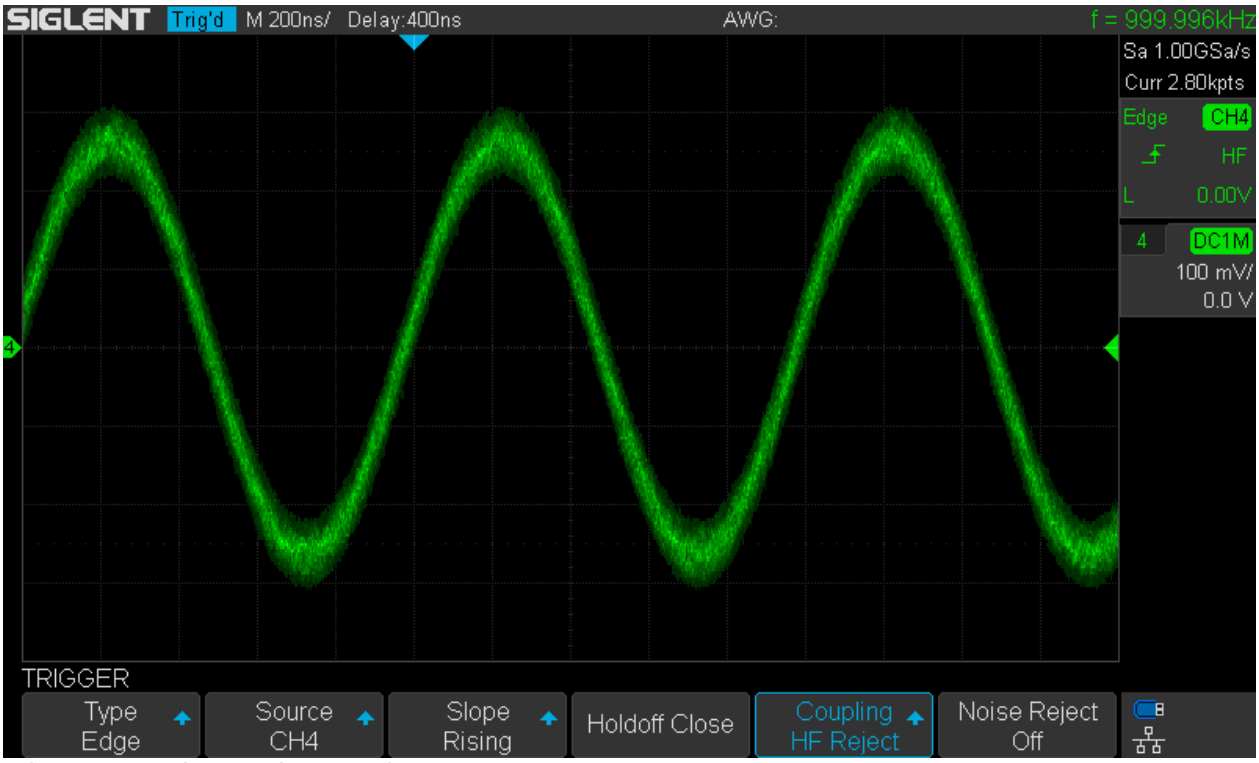
Trig_1MHz_Noise_-20dB_DC

Noise Reject does exactly what its name implies – problem solved.



Trig_1MHz_Noise_-20dB_DC_NR

Once again, HF-Reject works just as well.



Trig_1MHz_Noise_-20dB_HFRej

Trigger Types

The SDS1104X-E comes with a fairly comprehensive set of advanced trigger types, each of them having a bunch of options, hence not all combinations could be tested. The following sections demonstrate the application of advanced triggers on complex signals, where it would be very difficult or even impossible to just use the default edge trigger.

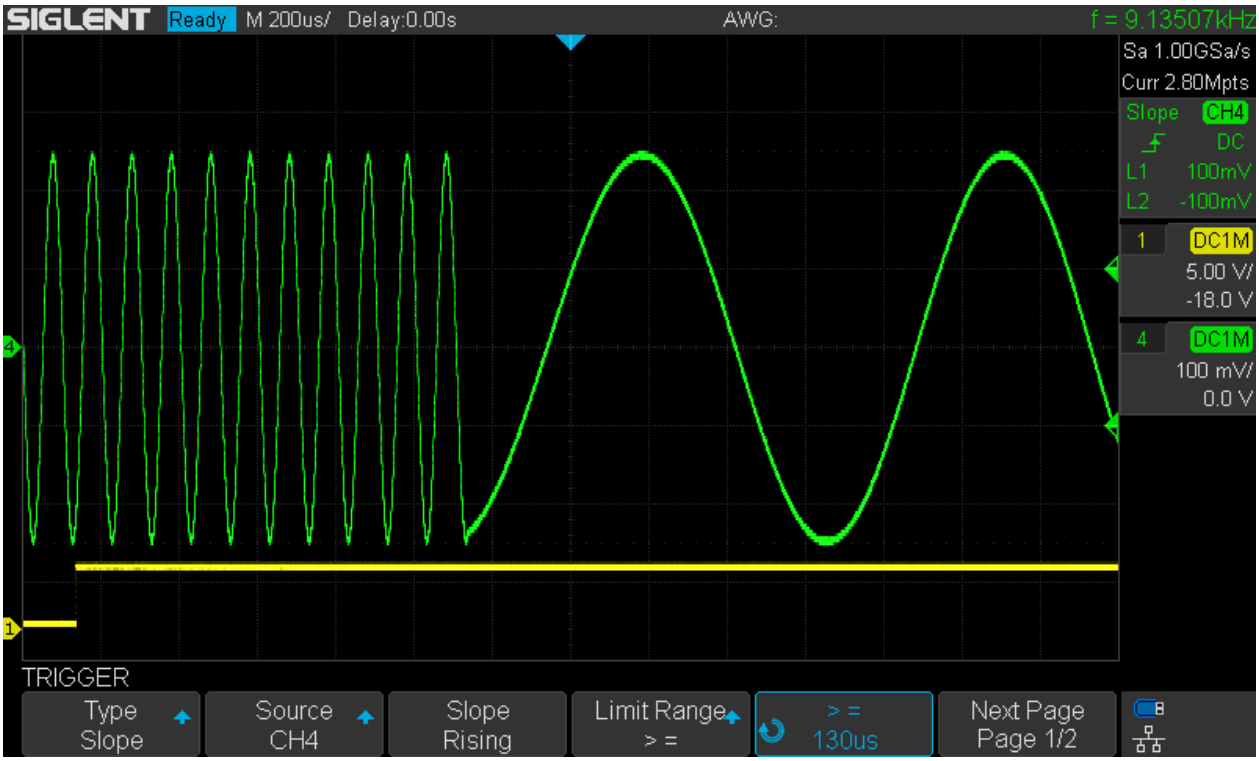
Edge

Edge trigger is used throughout this review, even with its additional features like Hold-off and Noise Reject in order to deal with difficult signals, so no need to introduce it here again.

Slope

We have to define two threshold levels which are used to measure the slope (transition time) of an edge. Options are rising or falling slope, transition time greater or less than a customizable reference value, transition time in- or outside a customizable reference window and Noise Reject.

The example shows the zoomed view of a swept sine 500mVpp, 1kHz to 10kHz, where we want to trigger exactly on the transition from 10kHz to 1kHz, or more precisely, on the rising edge of the very first sine period at 1kHz. This is impossible with edge trigger, but a rising slope trigger set for transition times >=130µs within a ±100mV window does the trick.

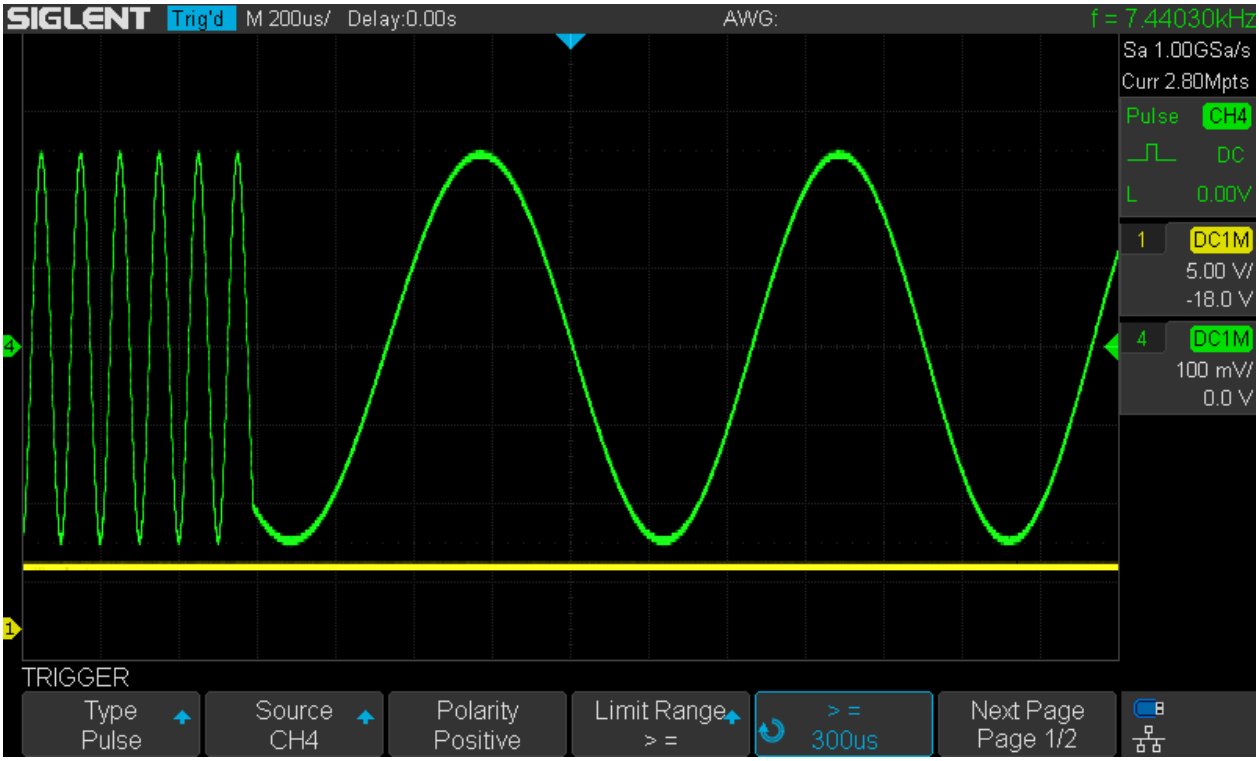


Trig_Slope

Pulse

Triggers on a certain pulse width range. Options are positive or negative pulse, pulse width greater or less than a customizable reference time, pulse width in- or outside a customizable reference time window and Noise Reject.

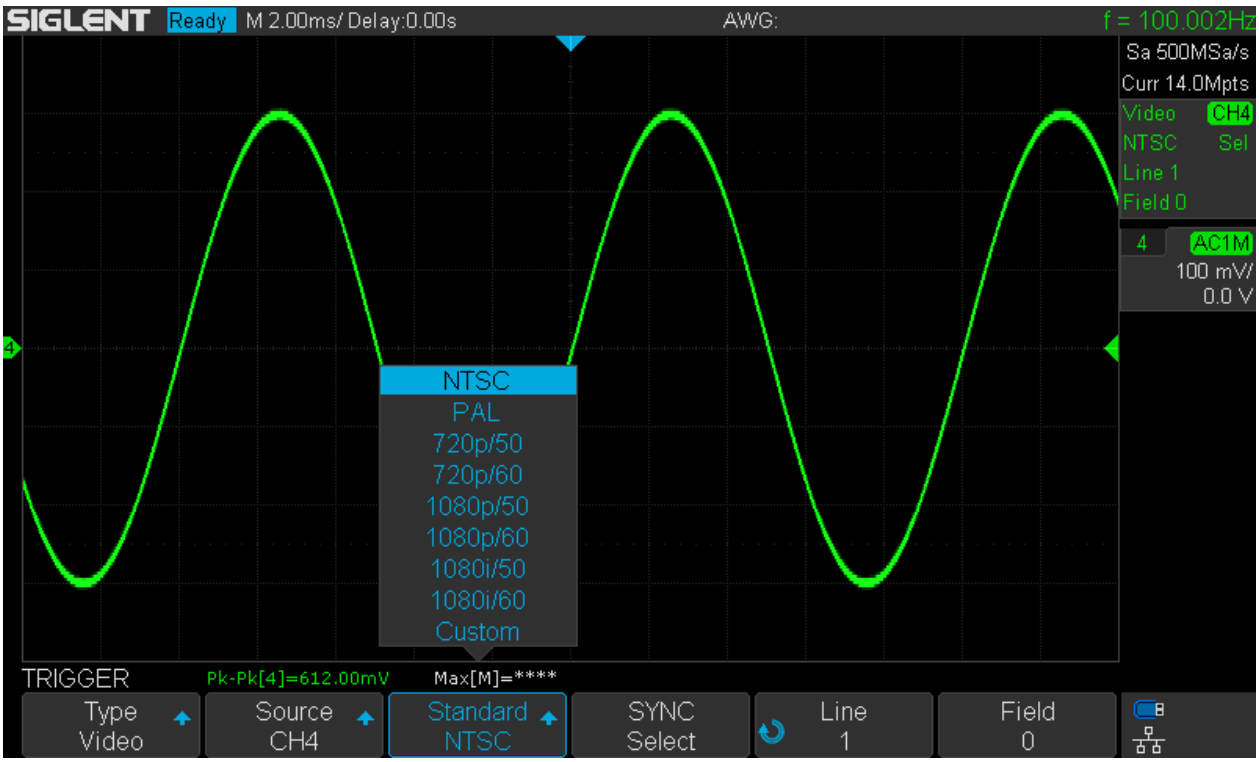
The example shows the zoomed view of a swept sine 500mVpp, 1kHz to 10kHz, where we want to trigger on the falling edge of the very first sine period at 1kHz. This is impossible with edge trigger, but a positive pulse trigger set for pulse widths >=300µs works quite nicely.



Trig_Pulse

Video

Video trigger is not often needed nowadays and I don't have any trigger source for that. It looks pretty comprehensive though. Please refer to the user manual for more details.

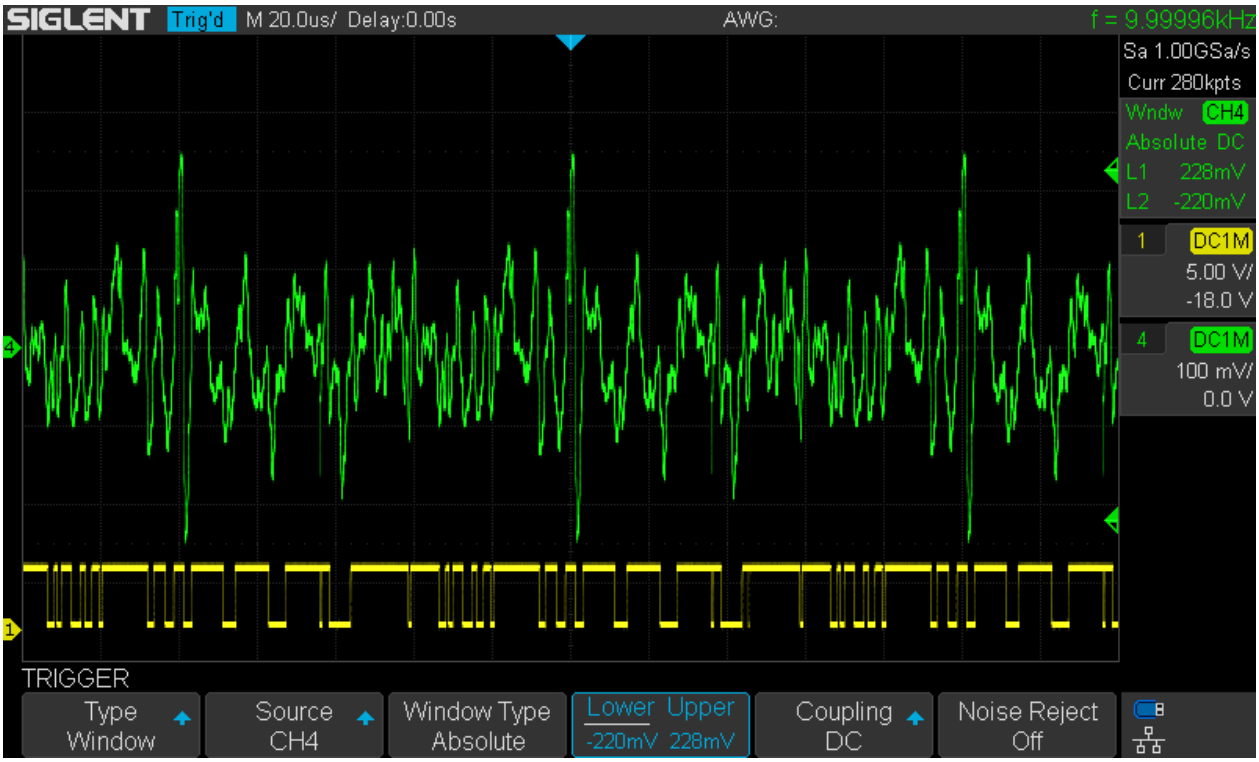


Trig_Video

Window

We have to specify two threshold levels to form a window that defines the valid signal amplitude range. Options are absolute or relative window definition and Noise Reject.

The example shows a random signal 500mVpp at 10kHz, where we want to trigger on the prominent peaks. This is perfectly possible with edge trigger for one polarity, but a window trigger set for a valid range of -220mV/+228mV works for positive and negative spikes at the same time.

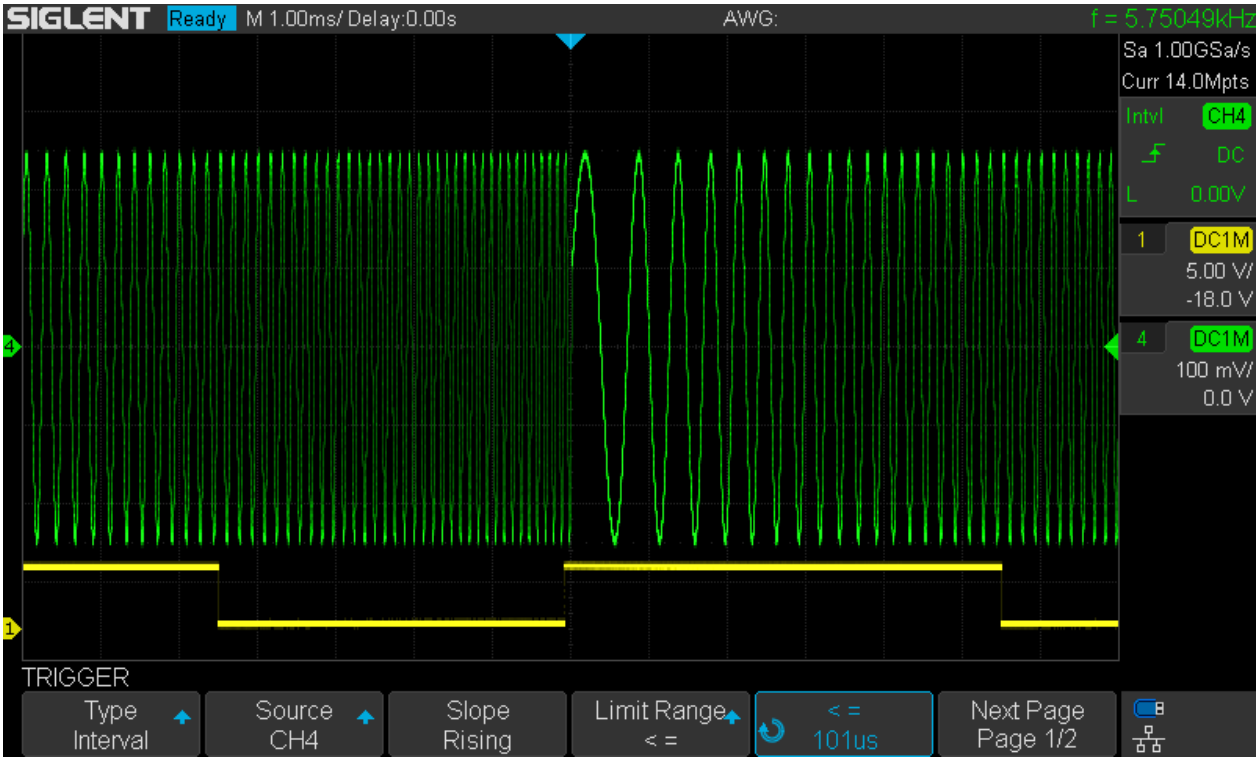


Trig_Window

Interval

This triggers on a certain signal period range. Options are rising or falling slope, signal period greater or less than a customizable reference time, signal period in- or outside a customizable reference time window and Noise Reject.

The example shows a swept sine 500mVpp, 1kHz to 10kHz, where we want to trigger exactly on the transition from 10kHz to 1kHz. This is impossible with edge trigger, but a rising slope interval trigger set for periods <=101µs does an almost perfect job.

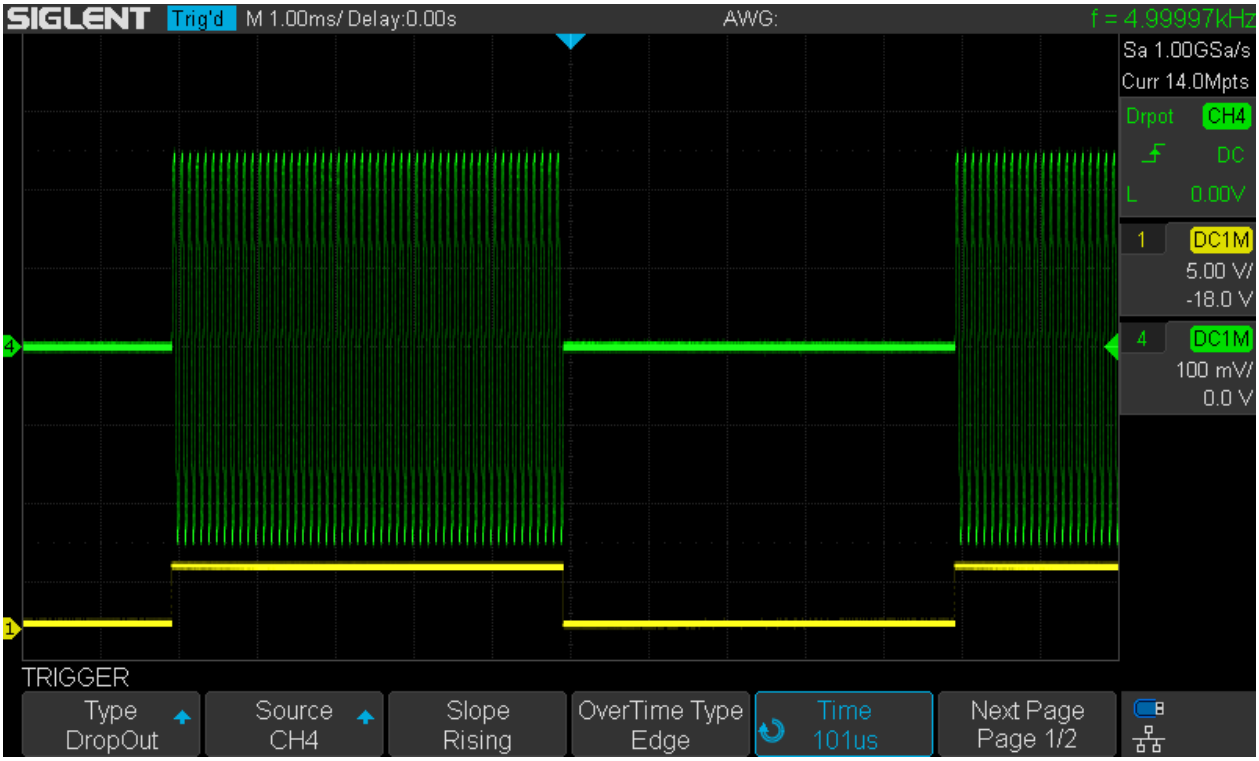


Trig_Interval

DropOut

Also known as Timeout trigger, and that's exactly what it does; it triggers when there is no signal transition within a specified time span. Options are rising or falling slope, timeout condition defined by edge or signal state and Noise Reject.

The example shows a 500mVpp 10kHz burst signal, where we want to trigger on the end of the burst packet. This is impossible with edge trigger, but a rising slope DropOut trigger set for a timeout >=101µs does just that.

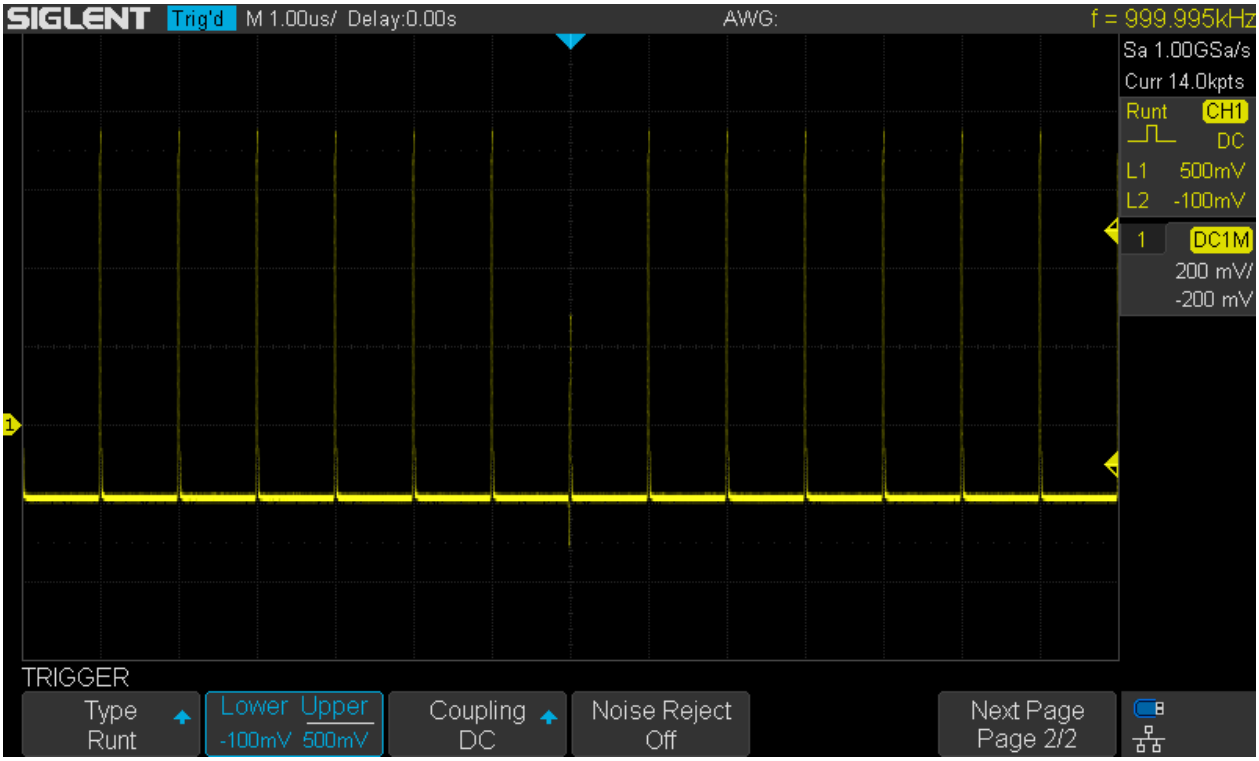


Trig_DropOut

Runt

We have to define two threshold levels. The trigger occurs whenever the signal crosses one of the thresholds twice without crossing the other one within a specified time window. In other words, runt pulse triggering can be limited to a certain pulse width range. Options are positive or negative polarity, pulse width greater or less than a customizable reference time, pulse width in- or outside a customizable reference time window and Noise Reject.

The example shows a pulse train 1Vpp at 1MHz that contains some runt pulses we want to trigger on. This is not possible with edge trigger, but a Runt trigger with the thresholds set at -100mV/+500mV works just fine.

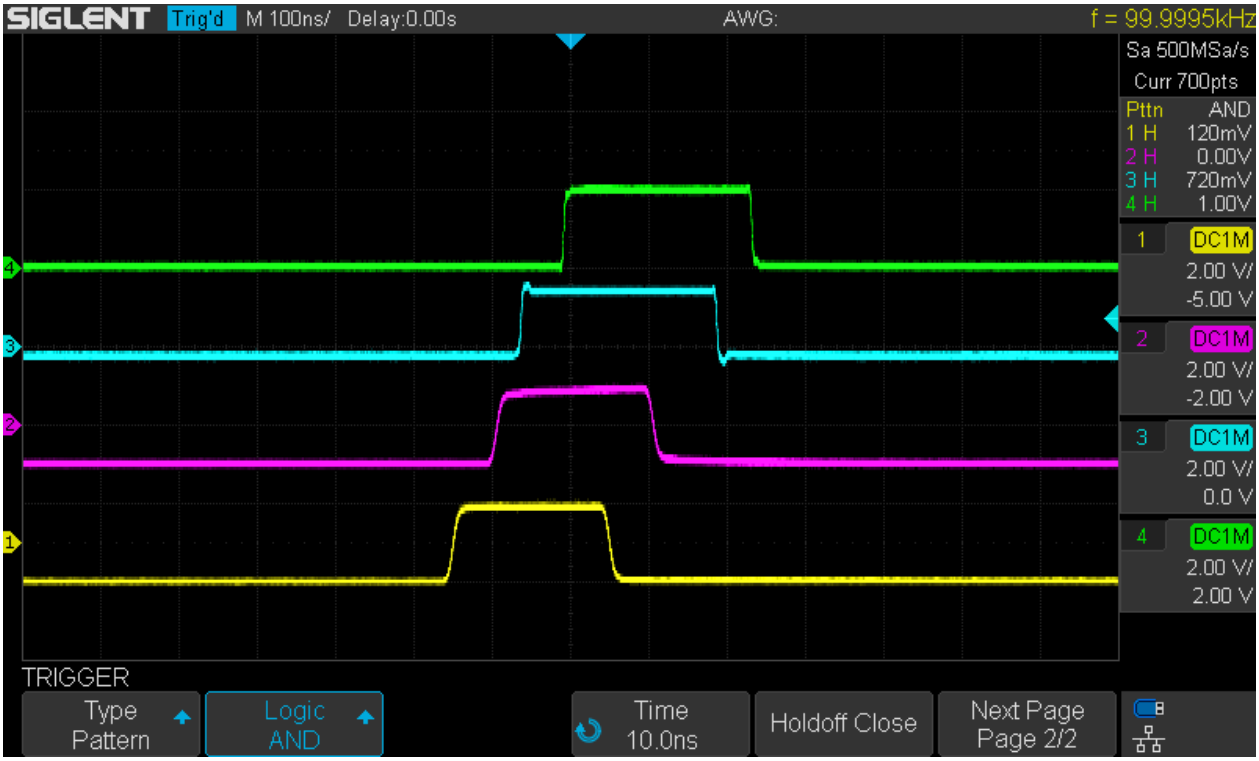


Trig_Runt

Pattern

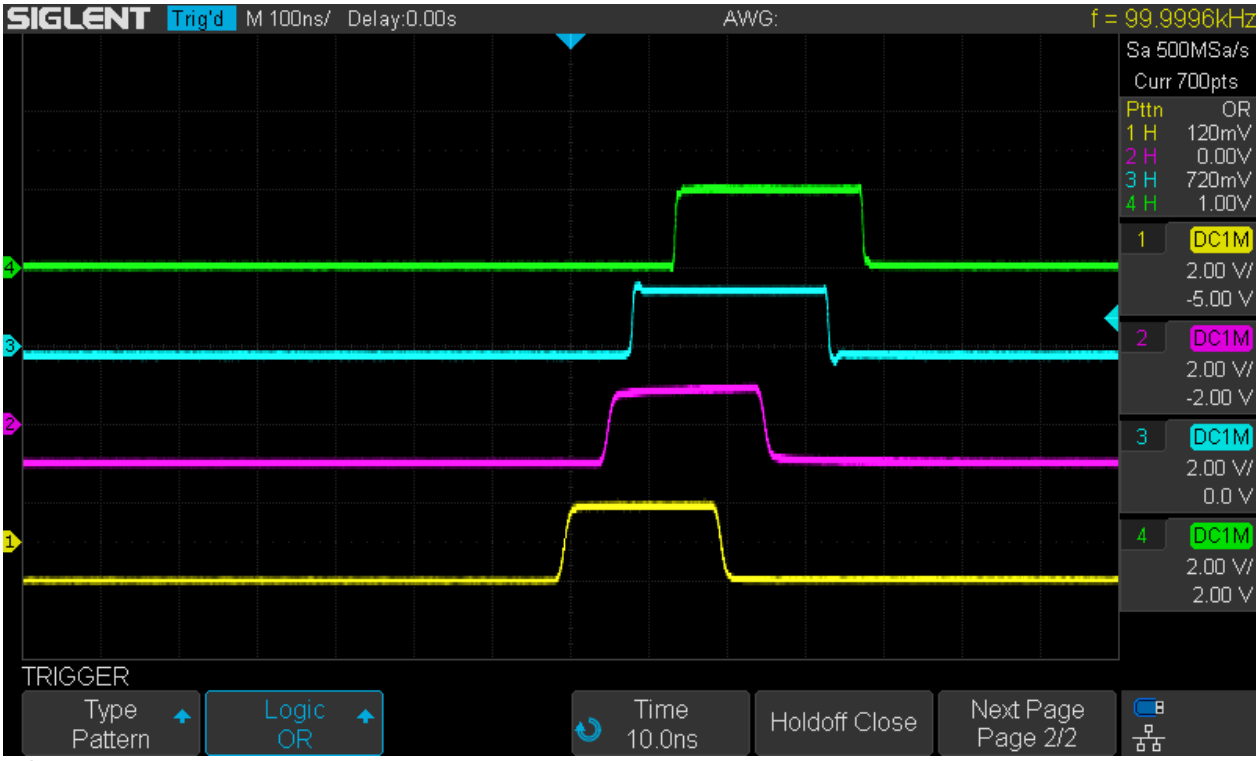
Pattern trigger gives us the opportunity to define a simple logic AND, OR, NAND or NOR operation for triggering on up to four channels at the same time. Options are a separate threshold value for each channel, a minimum duration for the condition to become valid, and Hold-off.

The first example shows four pulse trains 2Vpp at 100kHz with some increasing delay from channel 1 to channel 4. We want to trigger on the last rising edge of all four channels when they remain above their respective thresholds for at least 10ns. Pattern trigger logic AND operation is used.



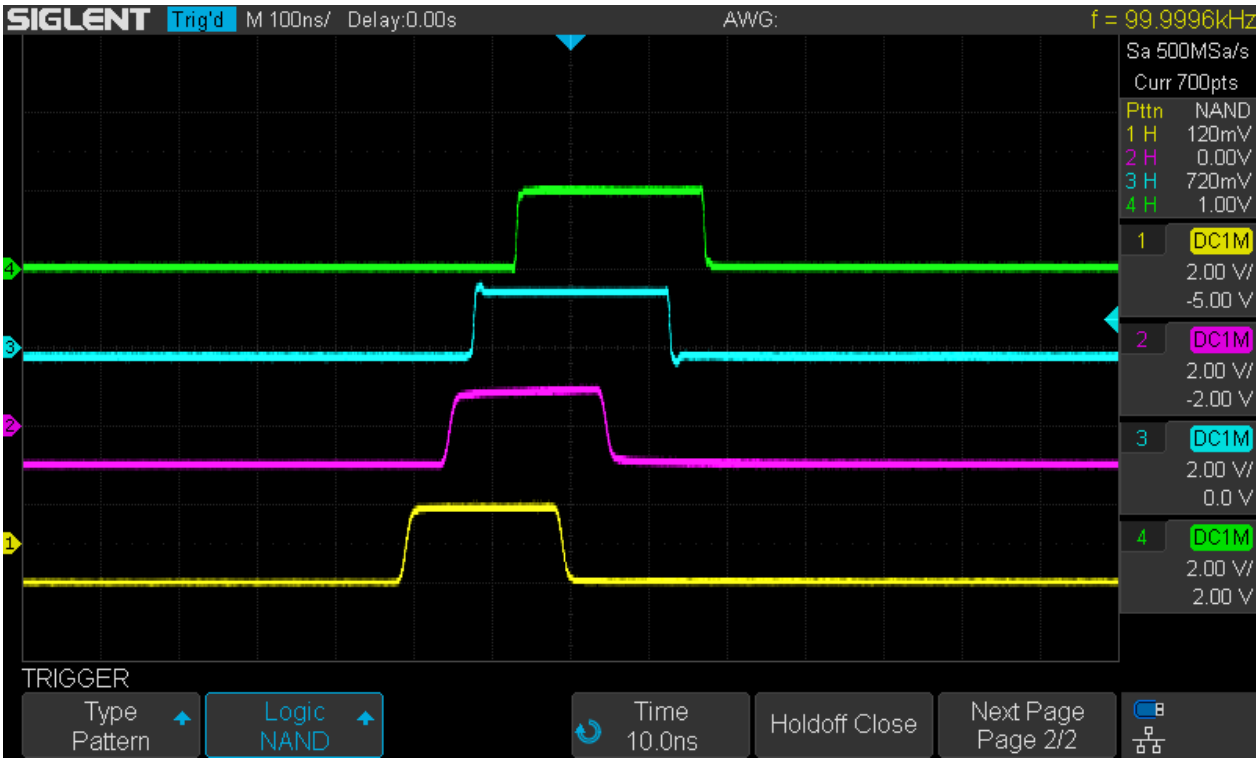
Trig_Pattern_AND

The second example shows four pulse trains 2Vpp at 100kHz with some increasing delay from channel1 to channel 4. We want to trigger on the first rising edge of all four channels when any channel remains above its respective threshold for at least 10ns. Pattern trigger logic OR operation is used.



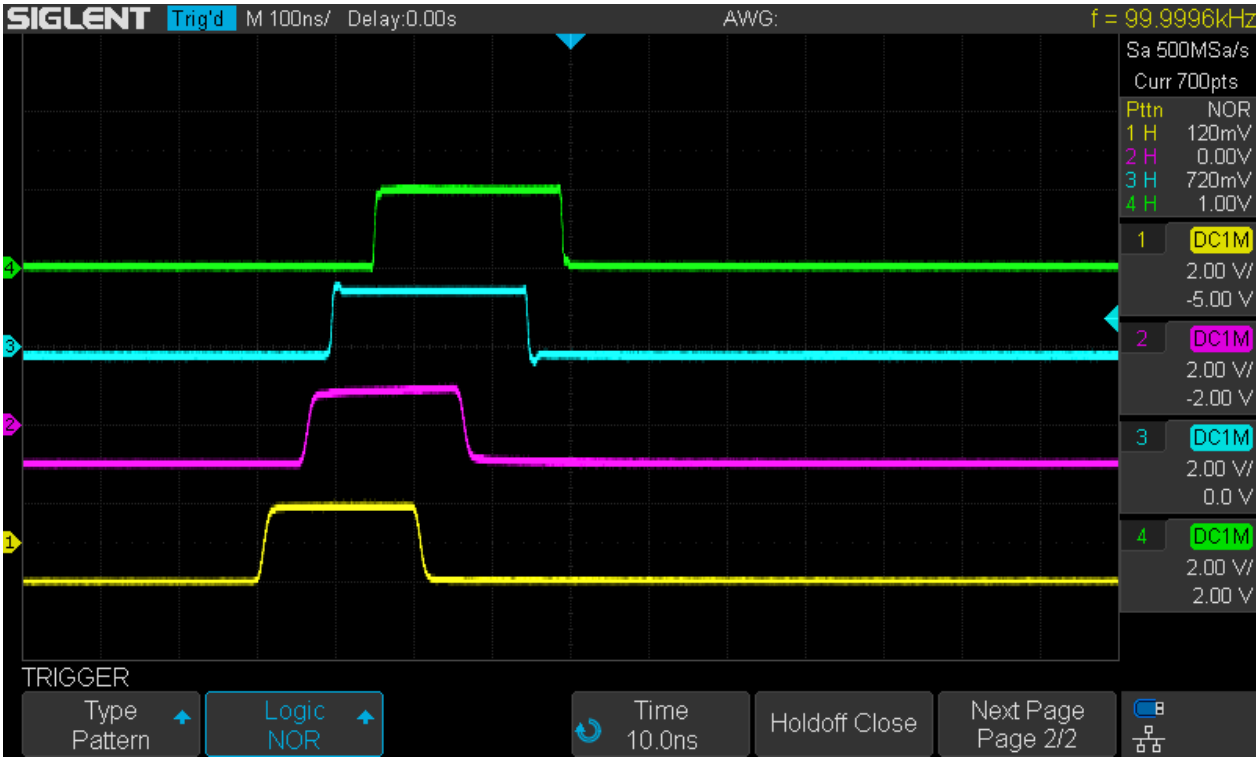
Trig_Pattern_OR

The third example shows four pulse trains 2Vpp at 100kHz with some increasing delay from channel1 to channel 4. We want to trigger on the first falling edge of all four channels when any channel remains below its respective threshold for at least 10ns. Pattern trigger logic NAND operation is used.



Trig_Pattern_NAND

The fourth example shows four pulse trains 2Vpp at 100kHz with some increasing delay from channel1 to channel 4. We want to trigger on the last falling edge of all four channels when they remain below their respective thresholds for at least 10ns. Pattern trigger logic NOR operation is used.



Trig_Pattern_NOR

Serial

Serial trigger will be dealt with later in this review, together with the serial decoders.

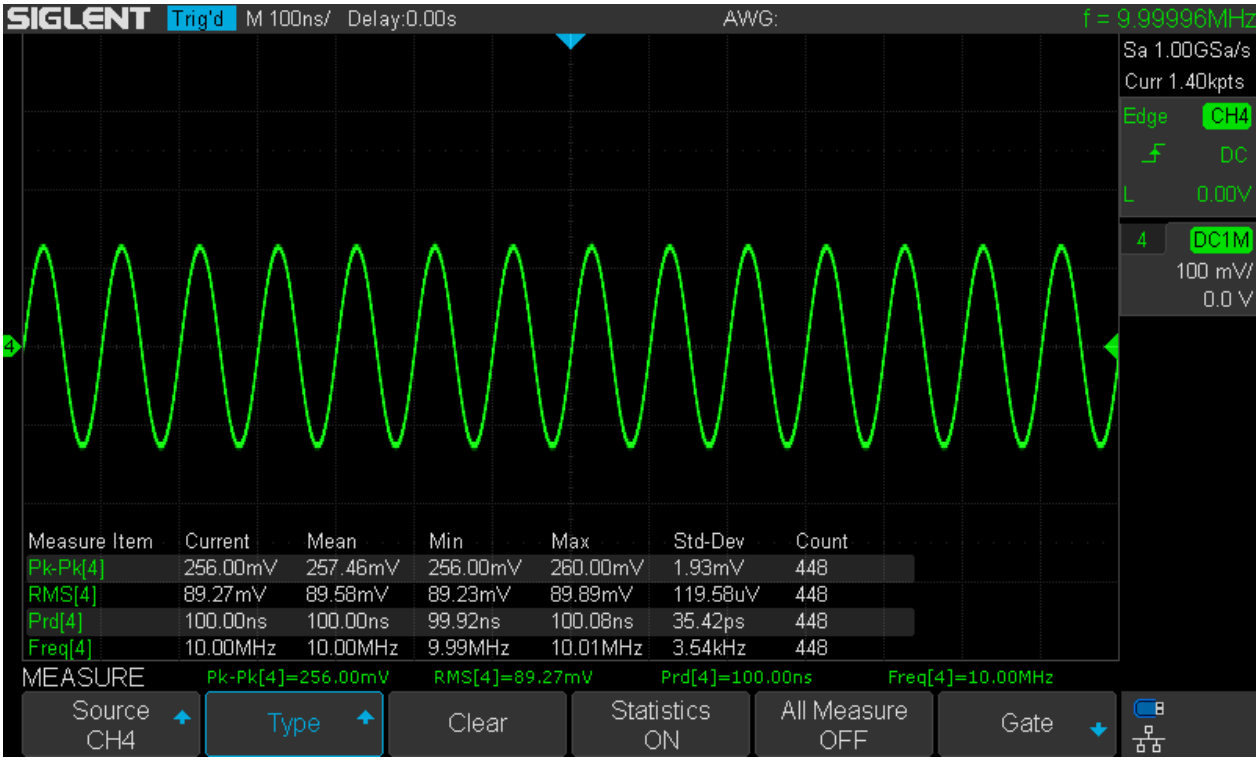
Trigger Frequency Counter

There are tasks where precise frequency measurements to at least 6 digits are essential. It is utterly convenient if a scope can do this, as it's usually just the signals we are monitoring with the oscilloscope where we also want to know their exact frequency. Back in the days of analog scopes, some offered a Y-output for the amplified signal of the current trigger channel and a frequency counter could be connected there – this was the ideal solution. Using the frequency counter alone was far less convenient, because it doesn't have a versatile input amplifier like the scope does and it is most desirable to be able to watch the waveform during the frequency measurement and vice versa without having to connect two instruments at the same test point.

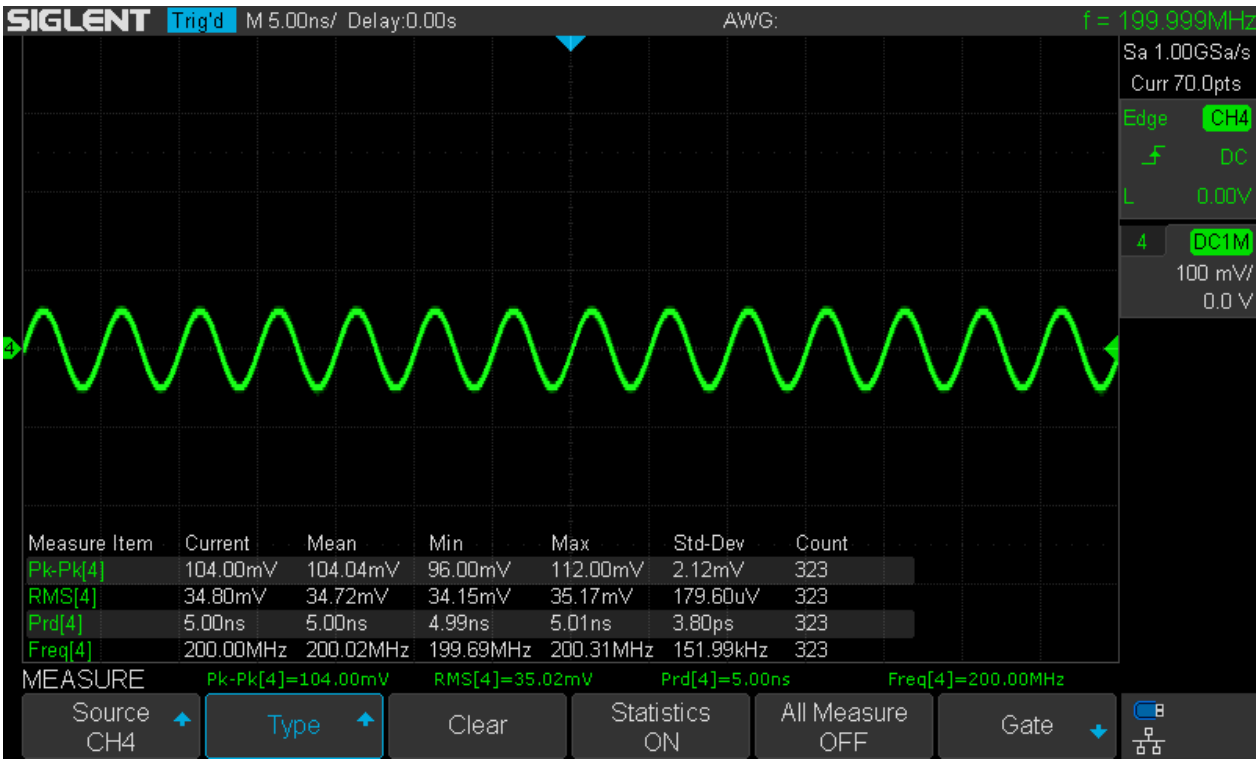
Nowadays, most scopes don't have a Y-output and there are only very few exceptions. Unfortunately, Siglent X-series aren't among them. So we have to stick with the second best thing, i.e. the integrated trigger frequency counter. For the Siglent X-series, it has 6 digits resolution, but it still cannot replace a real frequency counter, yet might be good enough for most of the less demanding tasks.

To illustrate this, let's start with an accurate 10MHz input signal, derived from the OCXO. The automatic measurement with its limited 4-digit resolution does a nice job by displaying 10.00MHz. The trigger frequency counter has 6 digits resolution, but that reveals the cheap TCXO in the low cost SDS1104X-E, because the frequency reading is off by 4ppm. That's still not too bad and can certainly compete with several cheap frequency counters that don't have a high stability option (quartz oven) fitted.

The next screenshot shows the same test with a 200MHz signal, which vastly exceeds the bandwidth of the SDS1104X-E. Yet it is nicely displayed (at a significantly reduced amplitude) and the automatic measurement is spot-on with now 5 digits, whereas the trigger frequency counter reads low again by some 5ppm. Nevertheless a very respectable result!



SDS1104X-E_CNT_10MHz

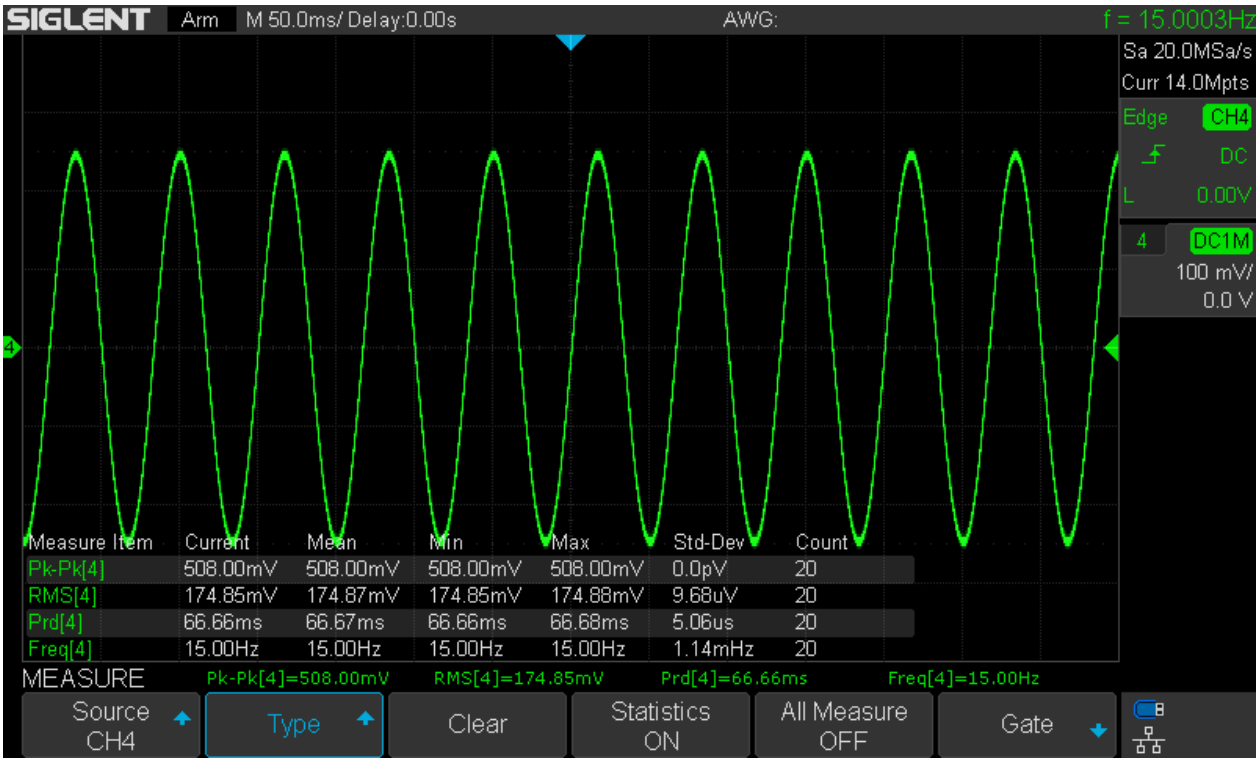


SDS1104X-E_CNT_200MHz

The last screenshot demonstrates a low frequency measurement. Since the trigger frequency counter does not work below 10Hz, the input signal was set to 15Hz to ensure smooth operation.

The readout of the trigger frequency counter doesn't look bad at all and an error of just 20ppm certainly appears acceptable. But the screenshot doesn't tell the whole story, as the frequency display isn't very stable and hardly more than 4 digits are actually usable at frequencies that low.

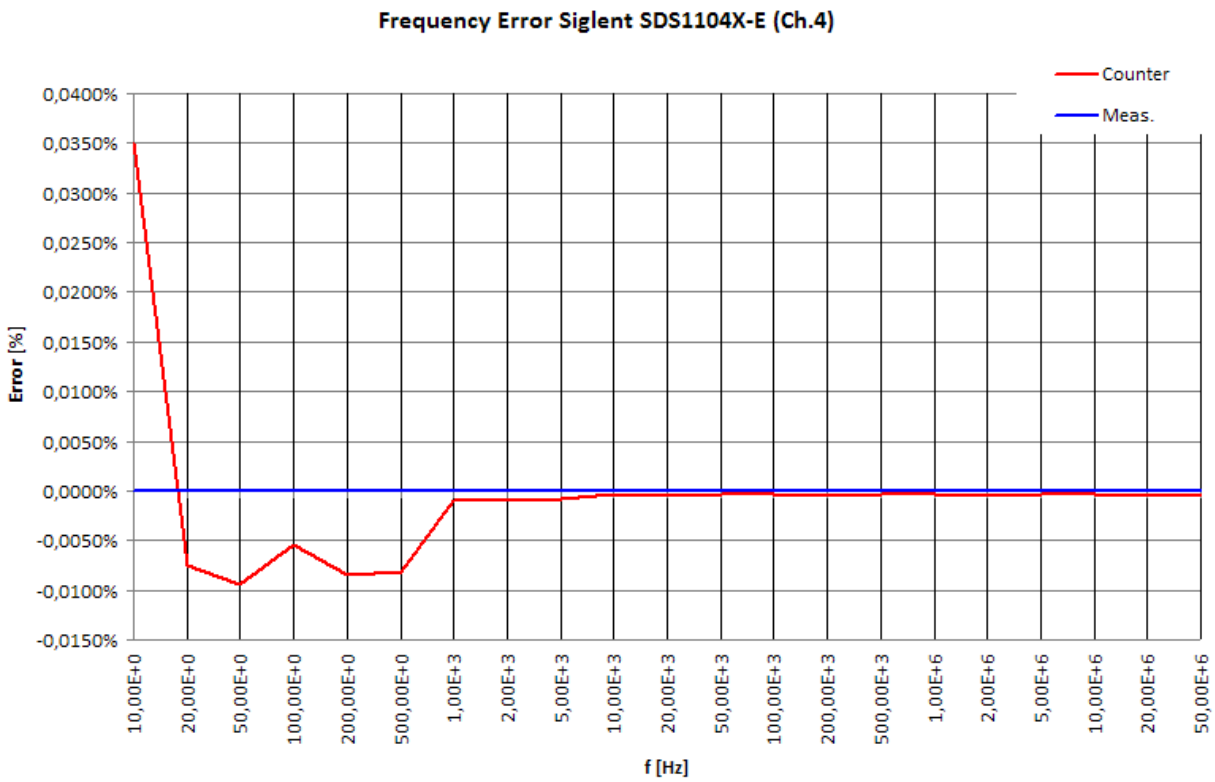
Automatic measurements on the other hand are spot-on again – at only 4 digits resolution, that is.



SDS1104X-E_CNT_15Hz

The situation improves quickly as the frequency goes up and at 1kHz nearly full accuracy and stability can be obtained. The graph below shows the frequency measurement error of both the counter and the automatic measurement for a number of input frequencies from 10Hz up to 50MHz. As can be seen from the results, the error reaches a first minimum at 10Khz and does not exceed 5ppm for any frequencies above. The 2nd screenshot above has already confirmed that for a 200MHz input signal.

It should be noticed, that this graph shows the maximum error, i.e. the maximum deviation from the actual frequency when the display fluctuates because of poor stability at low frequencies. In practical use we can watch the trigger frequency counter for a short while and do some averaging in our heads in order to obtain much more accurate results than what the graph below indicates.



Counter Accuracy Graph

Trigger Rate

By now, the hype of high waveform update rates seems to have calmed down a bit; yet the simple truth remains: a DSO with reasonably fast trigger rate is a joy to use and provides useful additional information.

The actual waveform update rate is affected by so many things like number and combination of active analog and digital channels, display options and background tasks like automatic measurements. Because of the sheer number of combinations on a 4 channel MSO, tests have been limited to one and two channels in one group and all combinations of interpolation x or sin(x)/x and display mode Dots or Vectors. The MSO option was not yet available at the time of testing anyway.

The test results are summarized in the first table below. All measurements have been made with maximum memory length of 14Mpts per channel group enabled.

As it turns out, reconstruction has no effect on the trigger rate, but display mode does. This is not actually a problem, since Dots mode is fine for most applications and consequently Vector mode is rarely needed.

When the 2 channels are activated in different channel groups, the single channel waveform update rate is halved. Speed wise, this is often a better option than using two channels of the same group.

Automatic measurements may decrease the waveform update rate up to -30%.

The second table shows the result for all four acquisition modes with and without Fast Acquisition enabled exemplarily at the two timebase settings with the fastest waveform update rates, i.e. 5ns/div and 50ns/div. It turns out, that Average and Eres are hardware accelerated and only moderately slower than Normal and Peak Detect. Without hardware acceleration, they are even slightly faster.

Trigger: Ch.1		Measurements: OFF							
		Waveforms / second (Dots)				Waveforms / second (Vectors)			
		Ch. 1 only		Ch. 1 + 2		Ch. 1 only		Ch. 1 + 2	
Timebase	In Freq.	x	sin(x)/x	x	sin(x)/x	x	sin(x)/x	x	sin(x)/x
1 ns	50 MHz	6090	6090		3760		6085		3347
2 ns	50 MHz	9844	9844		3049		9400		3000
5 ns	20 MHz	34215	34215		12966		15486		7339
10 ns	10 MHz	12891	12891		17254		12143		8032
20 ns	5 MHz		13430		6470		12544		6247
50 ns	2 MHz		107694		20577		20948		8528
100 ns	1 MHz		19115		36542		18221		10462
200 ns	500 kHz		13362		9049		13089		8751
500 ns	200 kHz		8875		5900		8850		5875
1 µs	100 kHz		7288		3842		7288		3842
2 µs	50 kHz		5082		2851				
5 µs	20 kHz		2280		1512				
10 µs	10 kHz		1289		818				
20 µs	10 kHz		694		471				
50 µs	10 kHz		297		198				
100 µs	10 kHz		148		99				
200 µs	10 kHz		74		49				
500 µs	10 kHz		12,4		16,5				
1 ms	10 kHz		12,4		9,9				

Dots	sin(x)/x	Normal		Peak Detect		Average 16		Eres 2Bit	
Timebase	In Freq.	Acqu. Fast	Acqu. Slow	Acqu. Fast	Acqu. Slow	Acqu. Fast	Acqu. Slow	Acqu. Fast	Acqu. Slow
5 ns	20 MHz	34250	24,8	34250	24,8	27300	29,7	27300	29,7
50 ns	2 MHz	107694	24,8	107694	24,8	85982	29,7	85982	29,7

Trigger Rate

Another interesting – and even more important – aspect is the blind time of the DSO acquisition. This is simply expressed as the percentage of the total acquisition time where no sample data is collected.

For example, at a timebase of 100µs/div in single channel mode we get 148 waveforms per second. Since the screen is 14 divisions wide, a single waveform equals 1.4ms, hence 148 waveforms are equivalent to 148 x 1.4ms = 207.2ms. This in turn means that during one second, only 207.2ms worth of data is actually captured. This is just 20.72% of the total time, thus resulting in 79.28% blind time.

So even for a fast DSO with fairly high waveform update rates, the blind time is still substantial. The table below shows the blind time for all the previous measurements. Please note that even the fast waveform update rate of 107k at 50ns/div still leaves a blind time of 92.461%. While this might appear frustrating, there are still slower scopes around and if we look at the waveform update rate, it should immediately become obvious that for a DSO with twice as many waveforms per second at a certain time base also the “non-blind time” is doubled. It should also be noticed, that even if waveform update rates were identical, a scope with more horizontal divisions will have less blind time.

Trigger: Ch.1		Measurements: OFF							
		Waveforms / second (Dots)				Waveforms / second (Vectors)			
		Ch. 1 only		Ch. 1 + 2		Ch. 1 only		Ch. 1 + 2	
Timebase [s/div]	In Freq. [Hz]	Wfm/s	BT [%]	Wfm/s	BT [%]	Wfm/s	BT [%]	Wfm/s	BT [%]
1E-9	50E+6	6090	99,991%	3760	99,995%	6085	99,991%	3347	99,995%
2E-9	50E+6	9844	99,972%	3049	99,991%	9400	99,974%	3000	99,992%
5E-9	20E+6	34215	99,760%	12966	99,909%	15486	99,892%	7339	99,949%
10E-9	10E+6	12891	99,820%	17254	99,758%	12143	99,830%	8032	99,888%
20E-9	5E+6	13430	99,624%	6470	99,819%	12544	99,649%	6247	99,825%
50E-9	2E+6	107694	92,461%	20577	98,560%	20948	98,534%	8528	99,403%
100E-9	1E+6	19115	97,324%	36542	94,884%	18221	97,449%	10462	98,535%
200E-9	500E+3	13362	96,259%	9049	97,466%	13089	96,335%	8751	97,550%
500E-9	200E+3	8875	93,788%	5900	95,870%	8850	93,805%	5875	95,888%
1E-6	100E+3	7288	89,797%	3842	94,621%	7288	89,797%	3842	94,621%
2E-6	50E+3	5082	85,770%	2851	92,017%				
5E-6	20E+3	2280	84,040%	1512	89,416%				
10E-6	10E+3	1289	81,954%	818	88,548%				
20E-6	10E+3	694	80,568%	471	86,812%				
50E-6	10E+3	297	79,210%	198	86,140%				
100E-6	10E+3	148	79,280%	99	86,140%				
200E-6	10E+3	74	79,280%	49	86,280%				
500E-6	10E+3	12,4	91,320%	16,5	88,450%				
1E-3	10E+3	12,4	82,640%	9,9	86,140%				

Blind Time

Measurements

Traditionally, oscilloscopes were not expected to be useful for highly accurate precision measurements. Back in the days when the screen graticule was the only means for measuring signal parameters, the accuracy of reading alone was worse than 1% full scale - this has slightly improved with digital cursor readout and changed completely with automatic measurements. Now the reading accuracy is not an issue anymore, and the time accuracy (X-axis in Y-t mode) has improved significantly by several orders of magnitude with digital scopes, where the timing is determined by a digital clock coming from a crystal rather than a free-running RC oscillator that used to produce the ramp for horizontal deflection of the CRT. Even the amplitude accuracy (Y-axis) can be quite good in modern high resolution oscilloscopes with analog to digital converters using more than the traditional 8 bits.

The Siglent SDS1104X-E uses a reasonable stable TCXO to generate the clock of which the horizontal timebase is derived. More important is the effective sample rate, which causes an uncertainty of at least ±1ns at 1GSa/s and quite obviously the uncertainty becomes correspondingly higher as the sample rate decreases by enabling both channels in a group and/or lowering memory depth and/or timebase.

Automatic Measurements

Back in the days of analog oscilloscopes, the screen grid was pretty much the only aid for measurements. Characterizing a signal used to be a time consuming and error prone task, yet some measurements like RMS required additional equipment, at least for non-textbook waveforms. Nowadays we can utilize a bunch of automatic measurements, which can make life so much easier – as long as they work reliably and provide reasonably accurate results. This is to be examined in the following sections.

Time Resolution

Accurate time measurements shouldn't be a problem for a modern DSO that has its sample clock derived from a TCXO (Temperature Compensated Quartz Oscillator). Yet there are many DSOs which use just the screen buffer for calculating the measurement results. This is fast and easy to implement, but requires zooming into the waveform in order to get meaningful measurement results, just like with an ancient analog oscilloscope. In contrast to this, the Siglent X-E series DSOs use the full acquisition memory to calculate the measurements – and this is still fast, thanks to the high processing power.

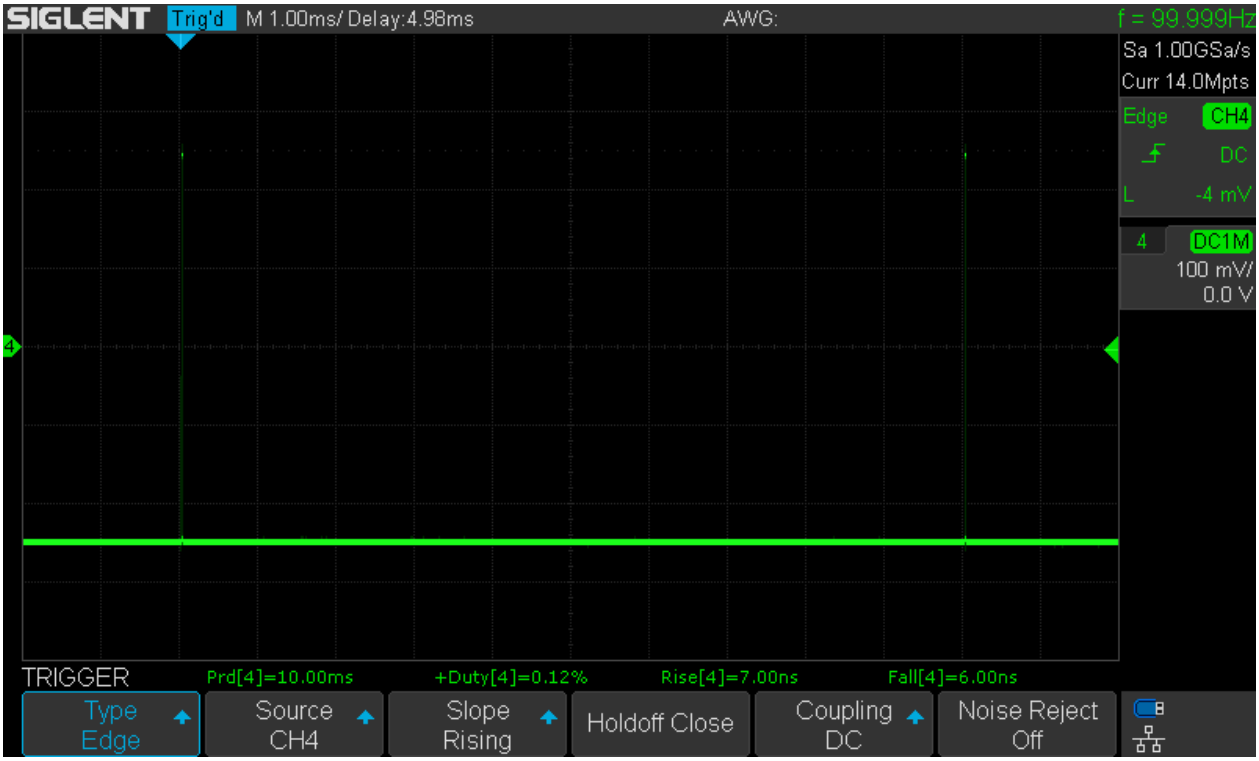
Consider a pulse train with fast transitions, maybe also narrow pulse widths, but relatively slow repetition rate, like a high resolution PWM signal. If we want to see the repetition rate, duty cycle and transition times at a glance, a scope that only uses the screen buffer just won't be up to the task.

Here we have a 100Hz PWM signal with a duty cycle of 90%, displayed at 1ms/div in order to see one full period:

SDS1104X-E_PWM_100Hz_90%_M1ms

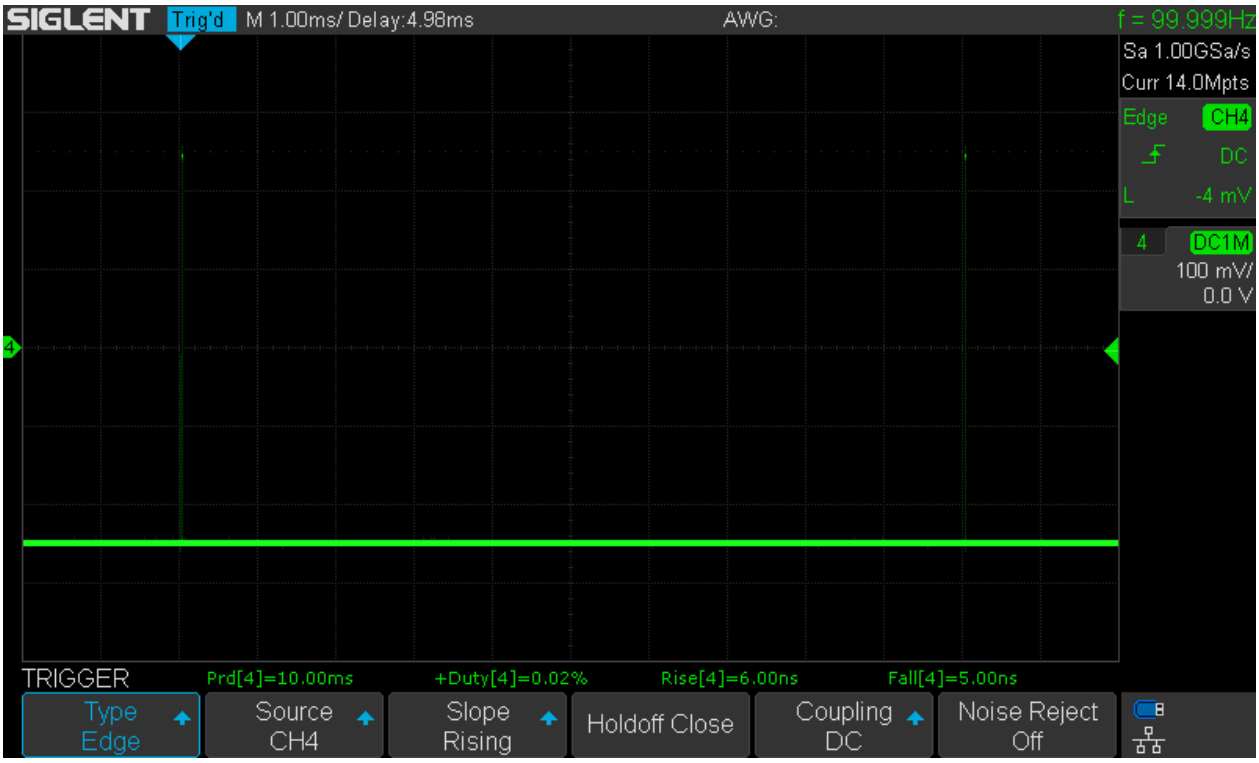
In the screenshot above, we can clearly see the 100Hz repetition rate – the trigger frequency counter is off by 0.001Hz, but automatic period measurement is spot-on, showing 10.00ms. We can also accurately measure the duty cycle of 90.00% and despite the slow timebase of 1ms/div, we still get a reasonably close measurement of the transition times in the realm of single-digit nanoseconds.

For a high resolution PWM, we might use more than 8 bits and therefore be able to adjust the duty cycle in steps smaller than 0.4%. The following example shows such a situation with a duty cycle of 0.12%. The duty cycle measurement is still spot-on, but is somewhat compromised by its limited resolution. As it is now, it can handle just about 13 bits – with only one digit more, it could be used for 16 bits PWM as well. This is one of several instances, where the display resolution of the automatic time measurements should be enhanced and I hope Siglent will improve this eventually.



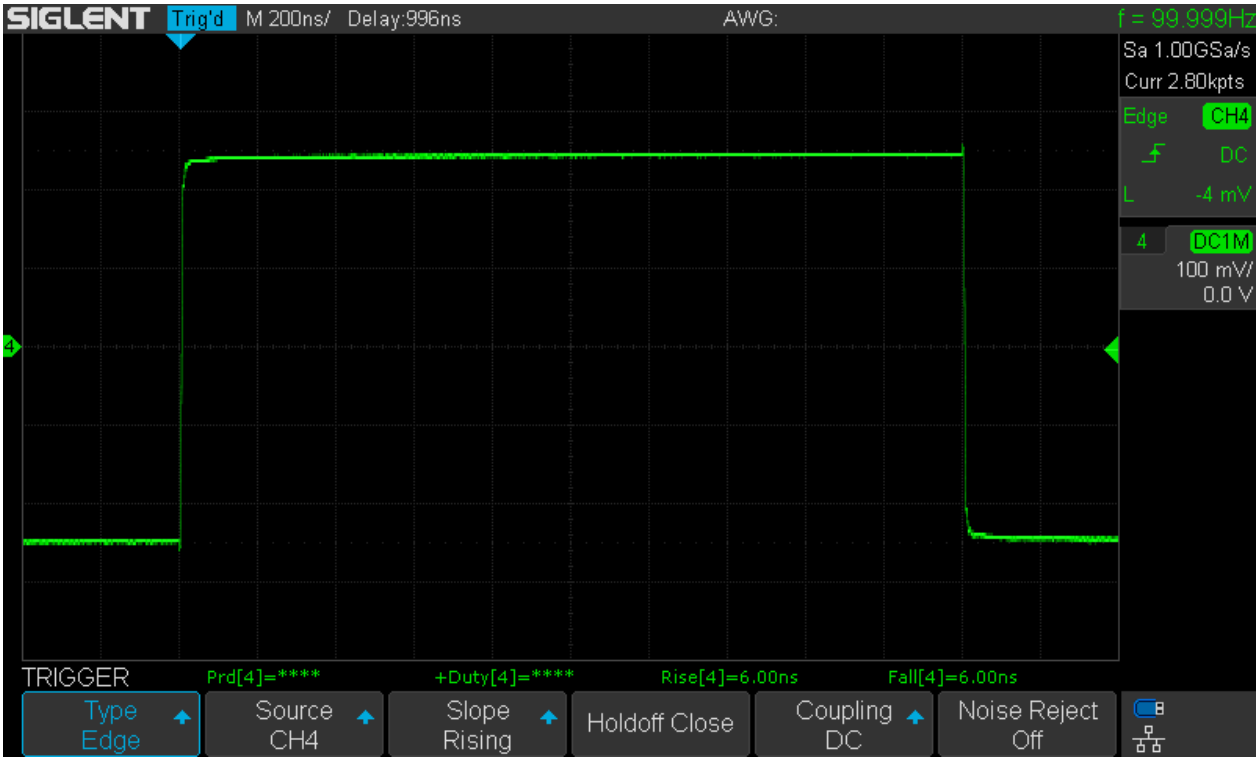
SDS1104X-E_PWM_100Hz_0.12%_M1ms

Now a duty cycle of only 0.02%:



SDS1104X-E_PWM_100Hz_0.02%_M1ms

Finally, let's zoom into the waveform 5000 times (200ns/div) to see just the pulse:

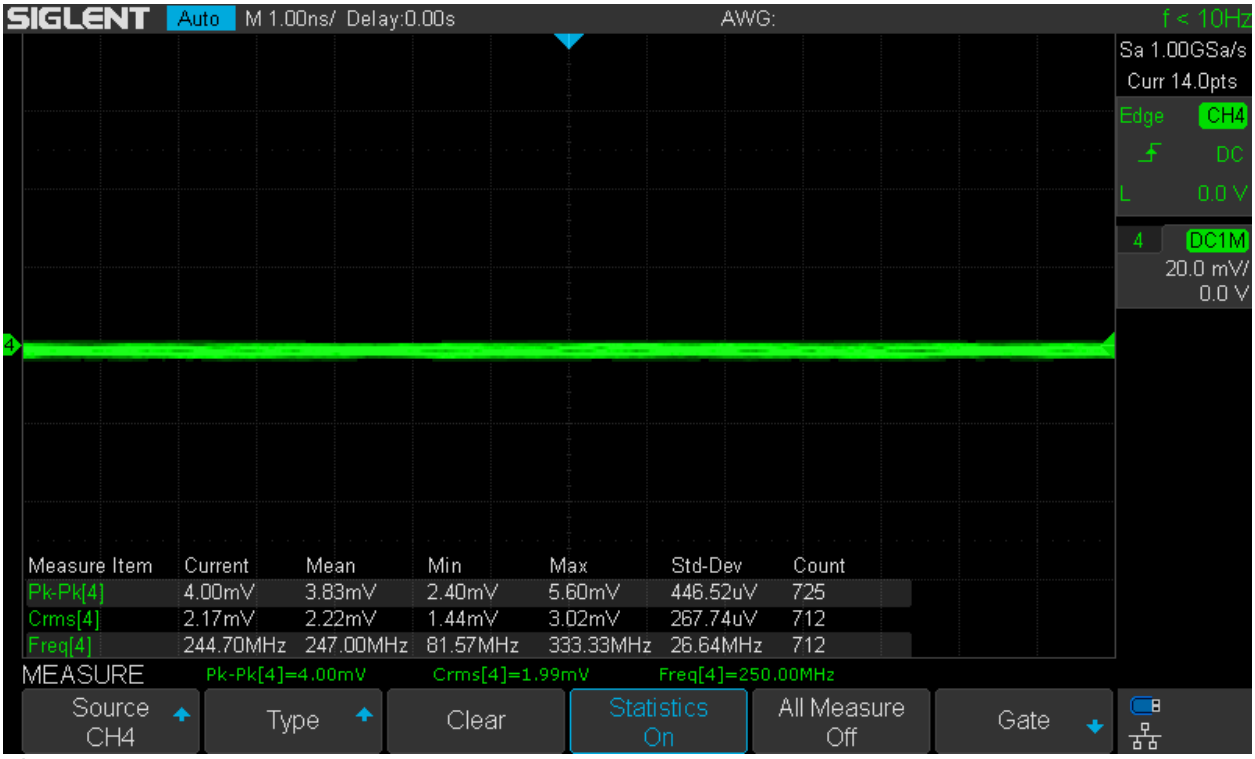


SDS1104X-E_PWM_100Hz_0.02%_M200ns

The trigger frequency counter still displays the (almost) correct frequency, but automatic measurements for period and duty cycle cannot provide any results, as they need at least one full signal period to work. Transition times still haven't changed, so there was no need to zoom anyway.

Sensitivity

Not a real test case, just a rather curious observation that should be documented here. A 250MHz sine is far above the capabilities of the 100MHz SDS1104X-E and even with the amplitude of 150mVpp it is just adding a little noise to the un-triggered trace line. Consequently, the trigger frequency counter shows nothing, but the automatic measurements still work and the frequency measurement is not far off.



Sine_250MHz_150mVpp_Stat

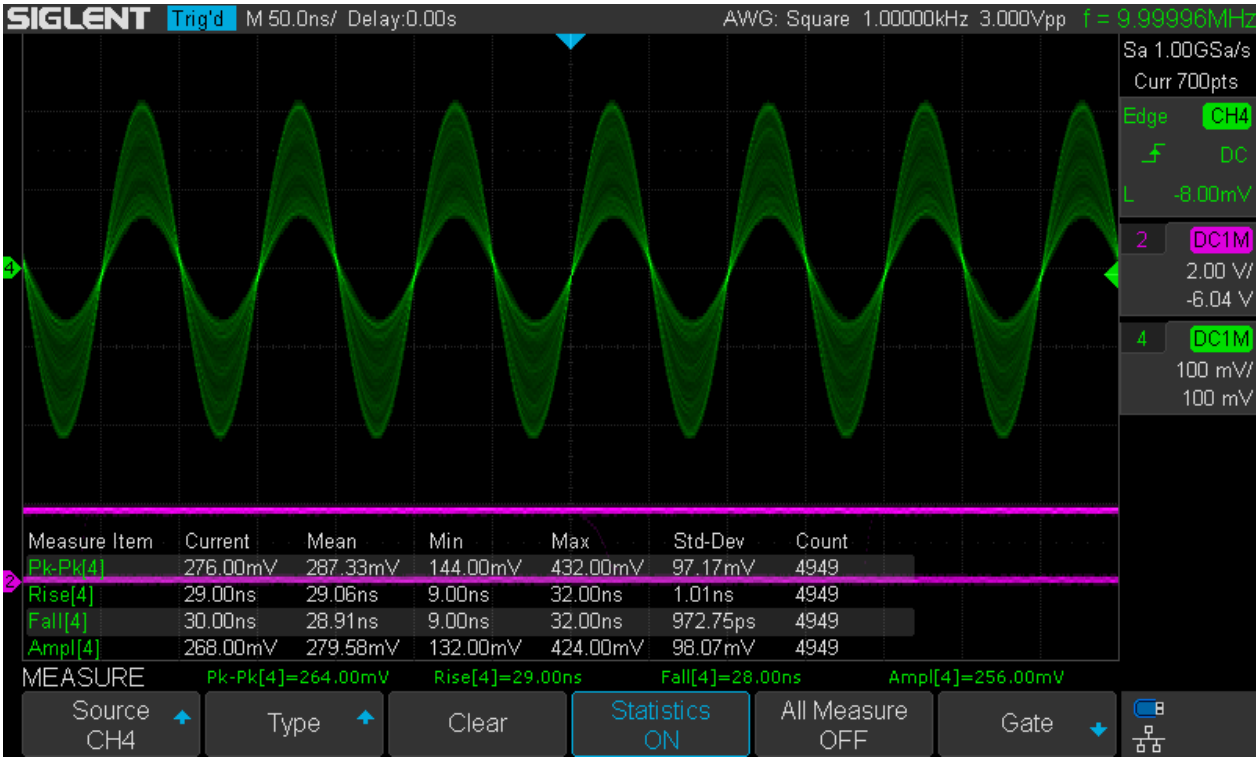
Speed

We don't only want measurements to have high time resolution and good accuracy, they should also be fast. Speed matters, especially for dynamic signals. The SDS1104X-E certainly delivers in this regard, see the following examples.

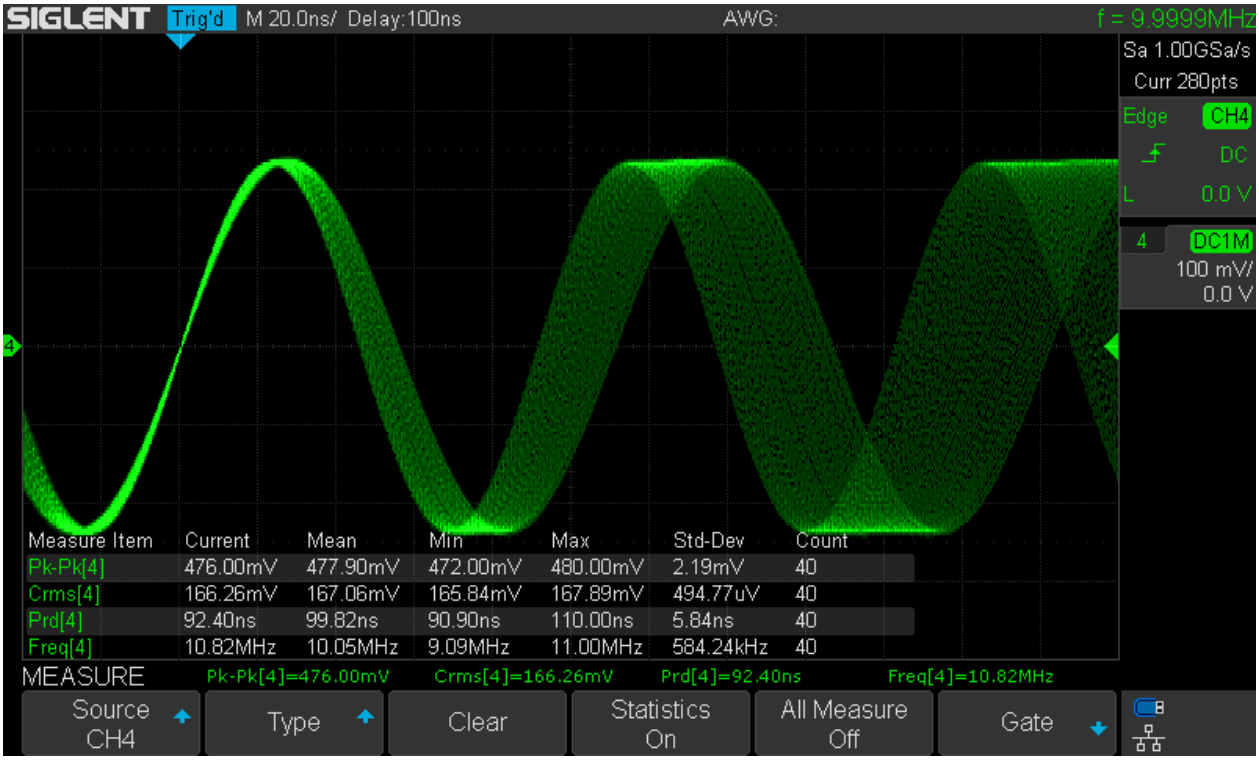
The screenshots below show two modulated waveforms.

- First a 10MHz carrier, 50% amplitude modulated with 1kHz.
- Second a 10MHz carrier, with 1kHz frequency modulation and 1MHz peak deviation.

When looking at the min/max values in the measurement statistics, the modulation parameters can be easily confirmed. So measurements are indeed fast enough to characterize even modulated signals with good accuracy within acceptable time. Especially for FM, accuracy is impressive even after only 40 measurements (or less than 3 seconds).



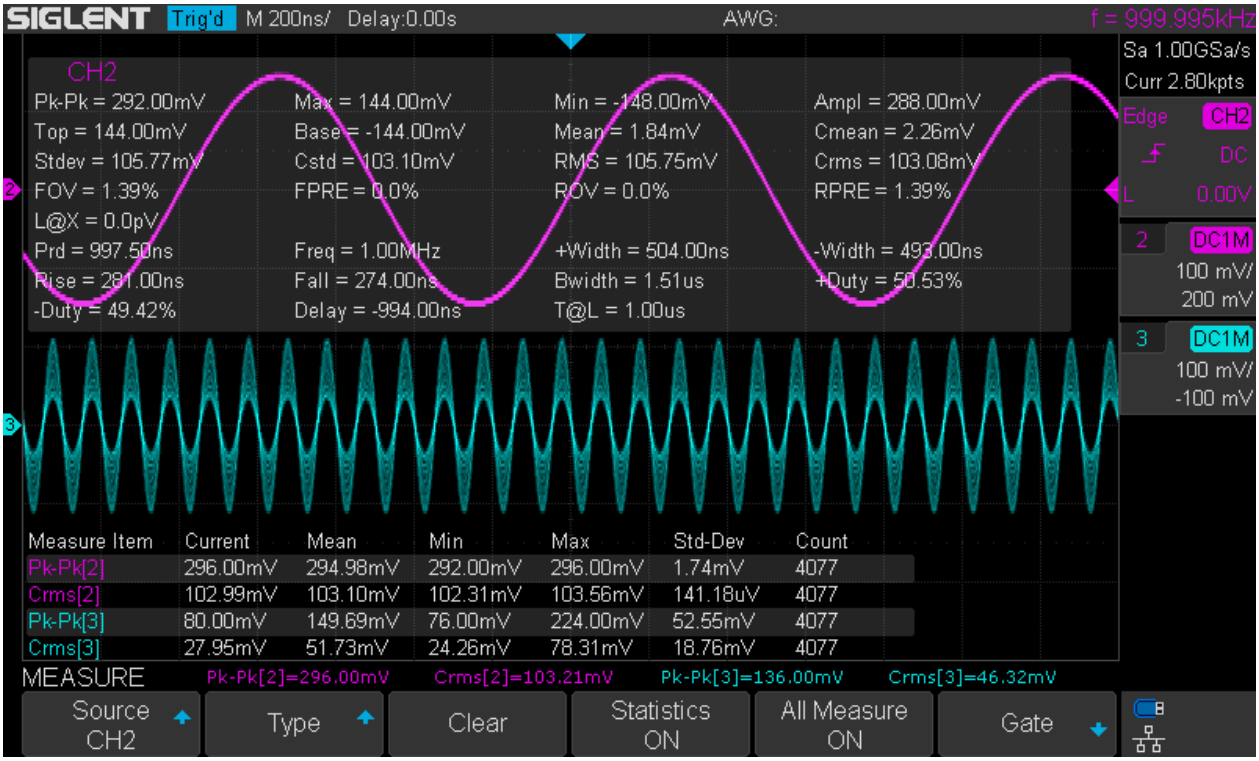
Sine_10MHz_AM1kHz_50%_M50ns_Dots



Sine_10MHz_FM1kHz_D1MHz_M20ns_Dots

Standard Measurements

Generally, we can have a maximum of 4 simultaneous measurements displayed at the bottom of the screen and one more in the statistics. Sometimes this is not enough, so we are given the opportunity to display all standard measurements together at the same time by selecting “All Measure ON”.



SDS1104X-E_all_measure

This is of course only possible for a single channel at a time, yet we can have all measurements for one channel plus 4 more for other channels at the bottom of the screen. The example above shows all measurements for channel 2 and the bottom line holds two Ch.2 measurements (which is of course redundant in this case, but we can have statistics for these), plus two Ch.3 measurements.

The big question remains, how accurate all these measurements are. There are no explicit specifications for that. In principle, we can use the specifications for amplitude accuracy as well as the sample clock as a basis for estimating the expected accuracy. I have done this exactly and run a number of tests in order to verify the accuracy of all measurements, one sample of such a test protocol is shown in the table below.

The resulting Error field is color coded:

Error is less than ¼ of what was to be expected.
Error is less than what was to be expected.
Error is more than what was to be expected.
Error is more than two times what was to be expected.

As can be seen, all measurements are pretty accurate except for L@X, which is a bit of a mystery anyway. According to the manual, this should be “The voltage value of the trigger point”, which in my understanding has to be pretty close to the trigger level. On the other hand, this wouldn’t make much sense since we certainly know the trigger level anyway and need no measurement for that. So it’s most likely a translation issue and/or lack of a clear description what this measurement really does and my expectations were just wrong.

Base Freq [Hz]	10,00E+3	Period [s]	100,00E-6
Base Ampl [V]	1,00E+0	Duty [%]	10,00%
Base Width [s]	10,00E-6	Base [-]	990,00E-3
Pulsewidth [s]	1,00E-6	Pulse [-]	10,00E-3
Pulseheight [V]	-100,00E-3	Duty [%]	1,00%
Transition [s]	12,00E-9	Duty+ [%]	10,00%
Pulsepos. [0/1]	1	Duty- [%]	89,00%
Samplerate (Hz)	1,00E+9		
Trigger [V]	128,00E-3		
Timebase [s]	50,00E-6	Periods [-]	6
Gain [V/div]	200,00E-3		
Gain Error	3,00%		
Offset Error [V]	2,00E-3		

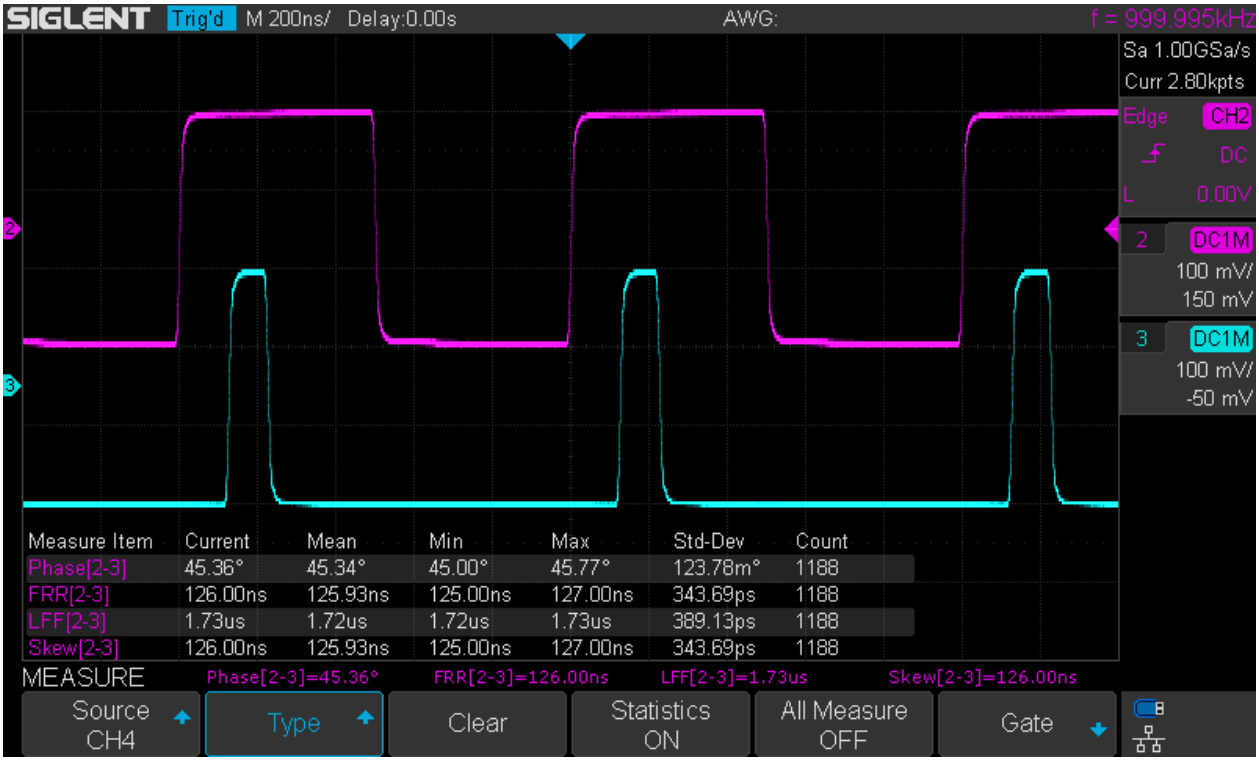
Trigger on falling Edge!

Measurement	Expected	Measured	Error [%]	Error Limit [%]
Pk-Pk [V]	1,10E+0	1,11E+0	0,91%	3,00%
Max [V]	500,00E-3	496,00E-3	-0,80%	3,40%
Min [V]	-600,00E-3	-616,00E-3	2,67%	3,33%
Ampl [V]	1,00E+0	1,01E+0	1,00%	3,00%
Top [V]	500,00E-3	496,00E-3	-0,80%	3,40%
Base [V]	-500,00E-3	-512,00E-3	2,40%	3,40%
Mean [V]	-401,00E-3	-409,00E-3	2,00%	3,50%
Cmean [V]	-401,00E-3	-409,00E-3	2,00%	3,50%
Stdev [V]	305,78E-3	301,90E-3	-1,27%	3,65%
Cstd [V]	305,78E-3	301,90E-3	-1,27%	3,65%
RMS [V]	501,10E-3	511,50E-3	2,08%	3,40%
Crms [V]	501,10E-3	511,50E-3	2,08%	3,40%
FOV [%]	0	0,79	0,79%	3,00%
FPRE [%]	0	0	0,00%	3,00%
ROV [%]	0	0	0,00%	3,00%
RPRE [%]	10	10,32	0,32%	3,00%
L_At_X [V]	128,00E-3	64,00E-3	-50,00%	4,56%
Prd [s]	100,00E-6	100,00E-6	0,00%	0,01%
Frequ [Hz]	10,00E+3	10,00E+3	0,00%	0,01%
Pos_Width [s]	10,00E-6	10,00E-6	0,00%	0,01%
Neg_Width [s]	90,00E-6	90,00E-6	0,00%	0,01%
Rise [s]	12,00E-9	13,00E-9	8,33%	8,34%
Fall [s]	12,00E-9	13,00E-9	8,33%	8,34%
Bwidth [s]	610,00E-6	610,00E-6	0,00%	0,01%
Pos_Duty [%]	10,00%	10,00%	0,00%	0,01%
Neg_Duty [%]	90,00%	90,00%	0,00%	0,01%
Delay [s]	-310,00E-6	-310,00E-6	0,00%	0,01%
T_At_L [s]	290,00E-6	290,00E-6	0,00%	0,01%

SDS1104X-E_RUS-RPRE

Channel Delay Measurements

There is a second group of measurements, summarizing all the possible time measurements between different channels. Unfortunately, there is no option to display them all at once, so the screenshot below gives just an example of 4 selected measurements.



SDS1104X-E_Ch_Delay

A Freq [Hz]	1,00E+6	A Period [s]	1,00E-6
A Duty [%]	50,00%	A Pulse [s]	500,00E-9
B Duty [%]	10,00%	B Pulse [s]	100,00E-9
B Phase [°]	-45,00	B Delay [s]	125,00E-9
Samplerate (Hz)	1,00E+9		
Timebase [s]	200,00E-9	Periods [-]	2

Trigger on falling Edge!

Measurement	Expected	Measured	Error [%]	Error Limit [%]
Phase [°]	45,00	45,33	0,73%	1,61%
FRR [s]	125,000E-9	125,920E-9	0,74%	1,61%
FRF [s]	225,000E-9	225,940E-9	0,42%	0,90%
FFR [s]	625,000E-9	625,550E-9	0,09%	0,33%
FFF [s]	725,000E-9	726,000E-9	0,14%	0,29%
LRR [s]	2,125E-6	2,130E-6	0,24%	0,10%
LRF [s]	2,225E-6	2,230E-6	0,22%	0,10%
LFR [s]	1,625E-6	1,630E-6	0,31%	0,13%
LFF [s]	1,725E-6	1,730E-6	0,29%	0,13%
Skew [s]	125,000E-9	126,370E-9	1,10%	1,61%

SDS1104X-E_Ch_Delay_Accuracy

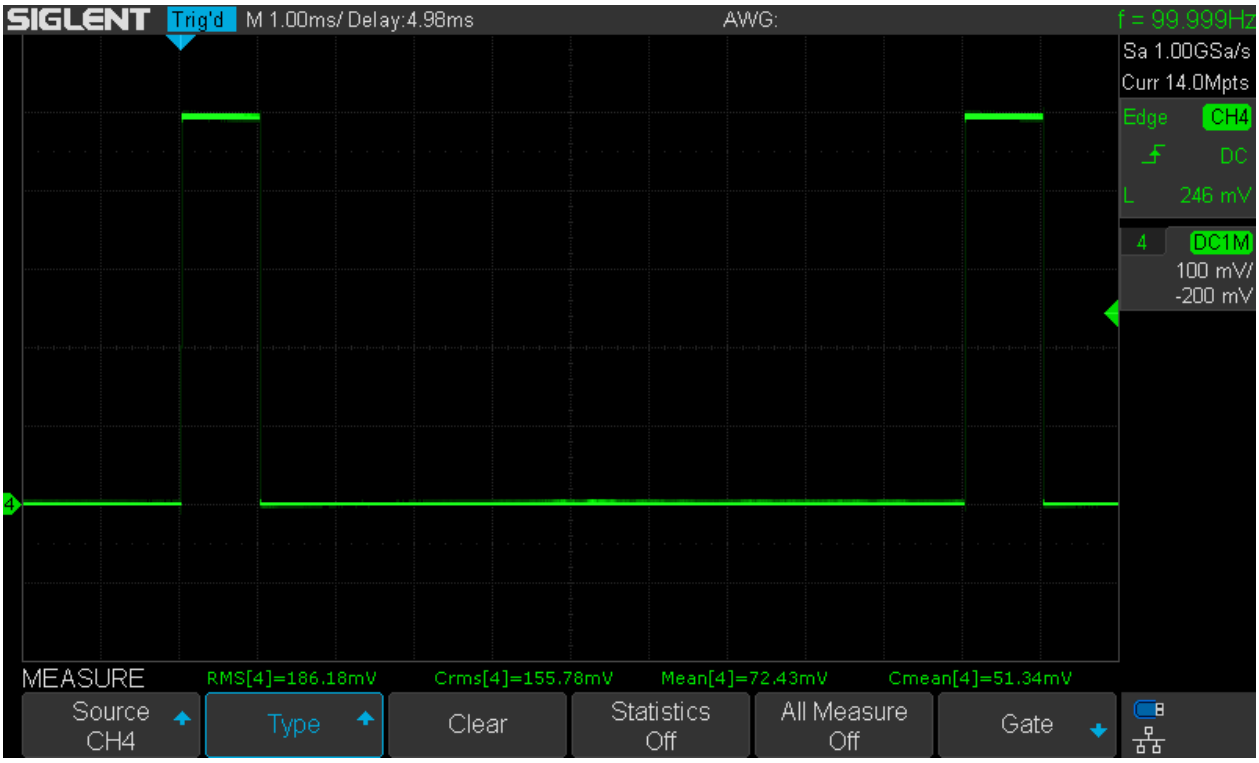
Again, the expected error limits and accuracies for all possible measurements have been determined, checked and summarized in the table above. Since this is all about time measurements, we could expect a very high accuracy, so it might be all the more surprising to find some error results highlighted in red. But it's not a faulty measurement, just insufficient resolution together with some rounding, which makes a basically correct result look bad. For some weird reason, measurements are limited to 3 digits in this area, thus turning e.g. 2.125µs into 2.13µs and making the error much bigger than necessary. One more time I do hope Siglent will take notice and improve that in a future firmware release.

Gated Measurements

Sometimes we don't want to measure over the whole record length and need a means to limit measurements to just a part of it. This is called a gated measurements and the Siglent SDS1104X-E provides dedicated X-Cursors for this.

The most popular example for gating is the integrating amplitude measurements, such as RMS and Mean. They can only work correctly on integer multiples of the signal period. Since the screen width (= record length) often does not meet this criterion, the automatic measurements already include a special set for these cases: it's the Cycle measurements Cycle Mean, Cycle Stdev and Cycle RMS, so we don't need to set up a gate for these common tasks.

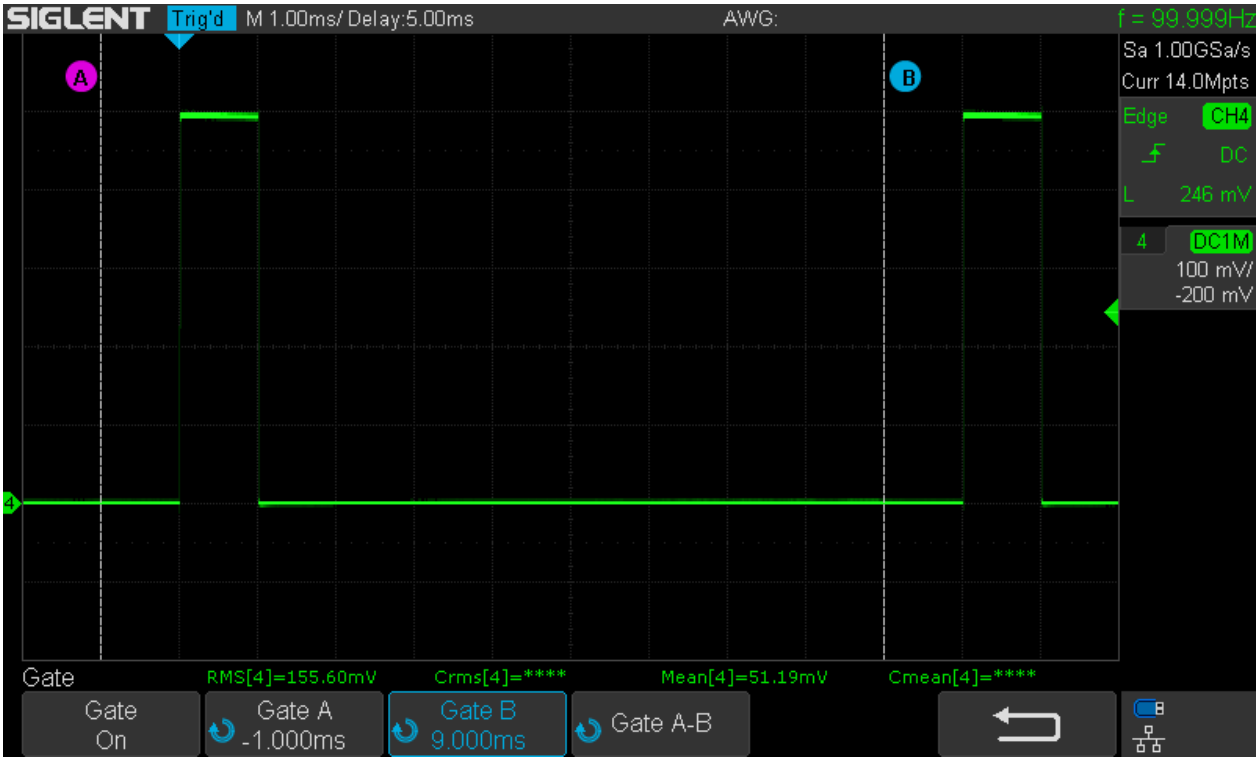
The next example uses the PWM signal with 10% duty cycle again and at 1ms/div the record length is 140% of the signal period, which certainly isn't an integer multiple. So there's no surprise that the Cycle measurements yield different results compared to the normal ones:



SDS1104X-E_PWM_100Hz_10%_NoGate

As was to be expected, the error would be huge if we forget to use the Cycle measurements. 186.18mV RMS vs. 155.78mV Cycle RMS and 72.43mV Mean vs. 51.34mV Cycle Mean.

Since this chapter is about gated measurements, let's see if we can get the correct results with normal measurements when using a properly defined gate. The RMS and Mean measurements are now pretty close to Crms and Cmean from the previous screenshot:



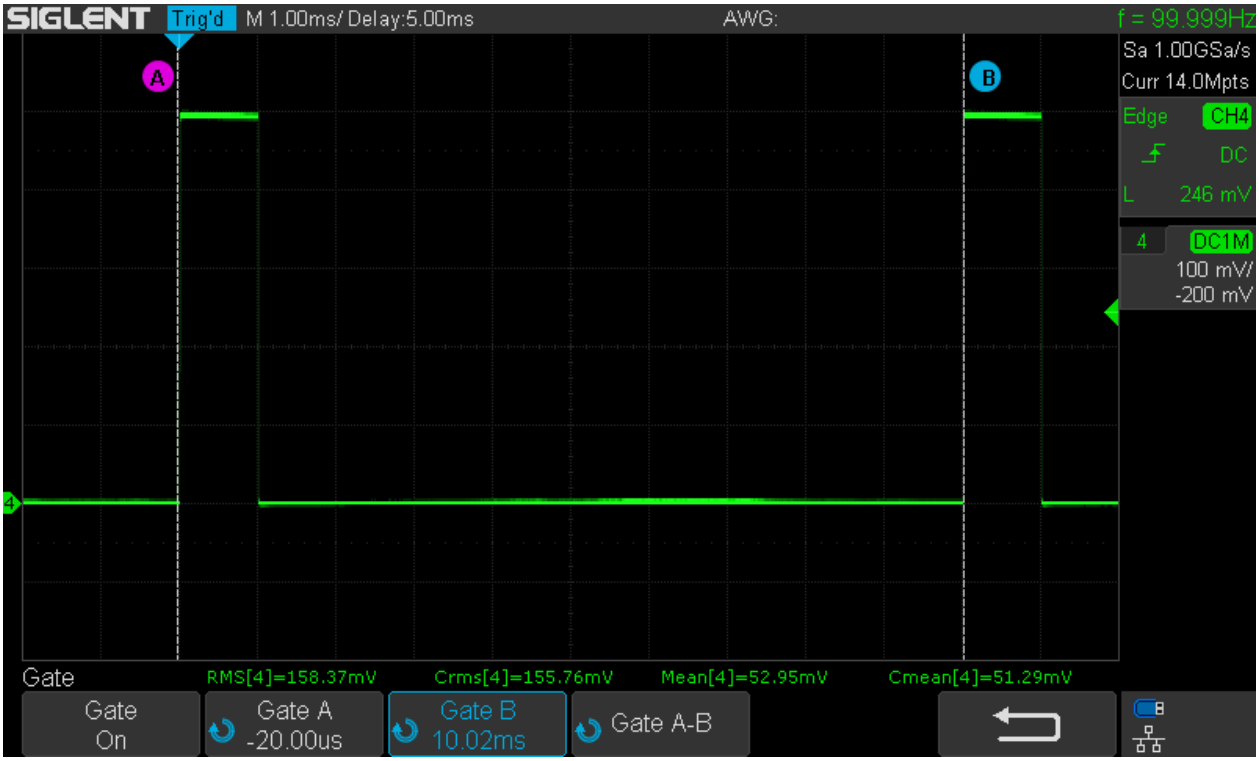
SDS1104X-E_PWM_100Hz_10%_Gate10ms

Why don't we define the gate from rising edge to rising edge?

Well, a proper cycle should include exactly one rising and one falling edge. We should avoid setting the gate border at a fast signal transition, at least when using time bases so slow that we cannot actually see the transition, because it's nothing but a vertical line. For accurate measurements, the signal level needs to be the same at both gate borders and we cannot guarantee this for transitions in the realm of single-digit nanoseconds by setting a gate with 20µs resolution like in the example above.

Currently, we don't get any results from the dedicated Cycle measurements, because the gate limits the view and makes it impossible to detect two rising (or falling) edges, which would be essential for recognizing a full signal cycle. That's not a problem here, as Cycle measurements don't make sense anyway – after all we have set the gate for exactly one cycle just to make the Cycle measurements superfluous.

Just for fun, we can still have both types of measurements at the same time by making the gate a tad wider and shift it a bit, so that it includes two rising edges. This will introduce a small error for the normal measurements (because this now sees a little bit more than just one cycle), but lets us compare both approaches within one screenshot. As can bee seen, RMS and Crms as well as Mean and Cmean measurements are now very similar as was to be expected.



SDS1104X-E_PWM_100Hz_10%_Gate10.04ms

Cursors

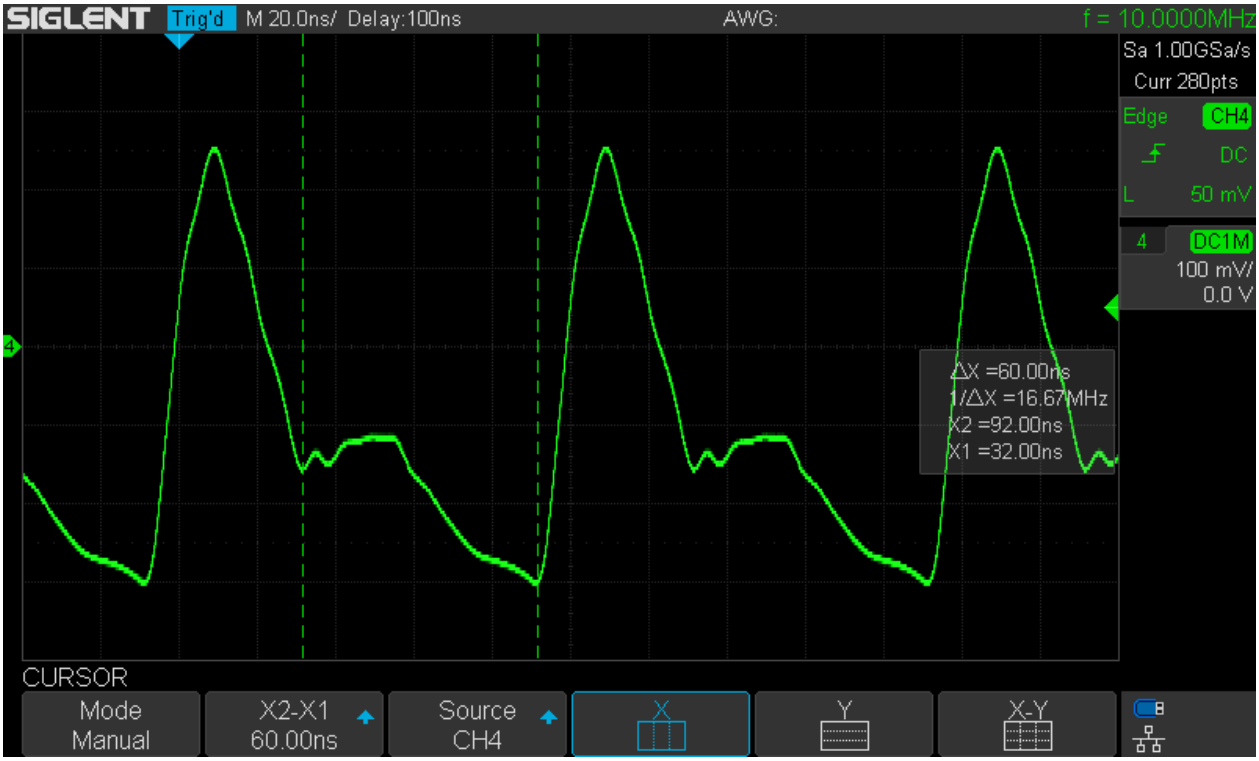
The main applications for Cursor measurements are:

- No automatic measurement available for a particular measurement task.
- Visual highlighting of a particular measurement.
- Measuring and comparing time and level at an arbitrary point of the displayed waveform(s).

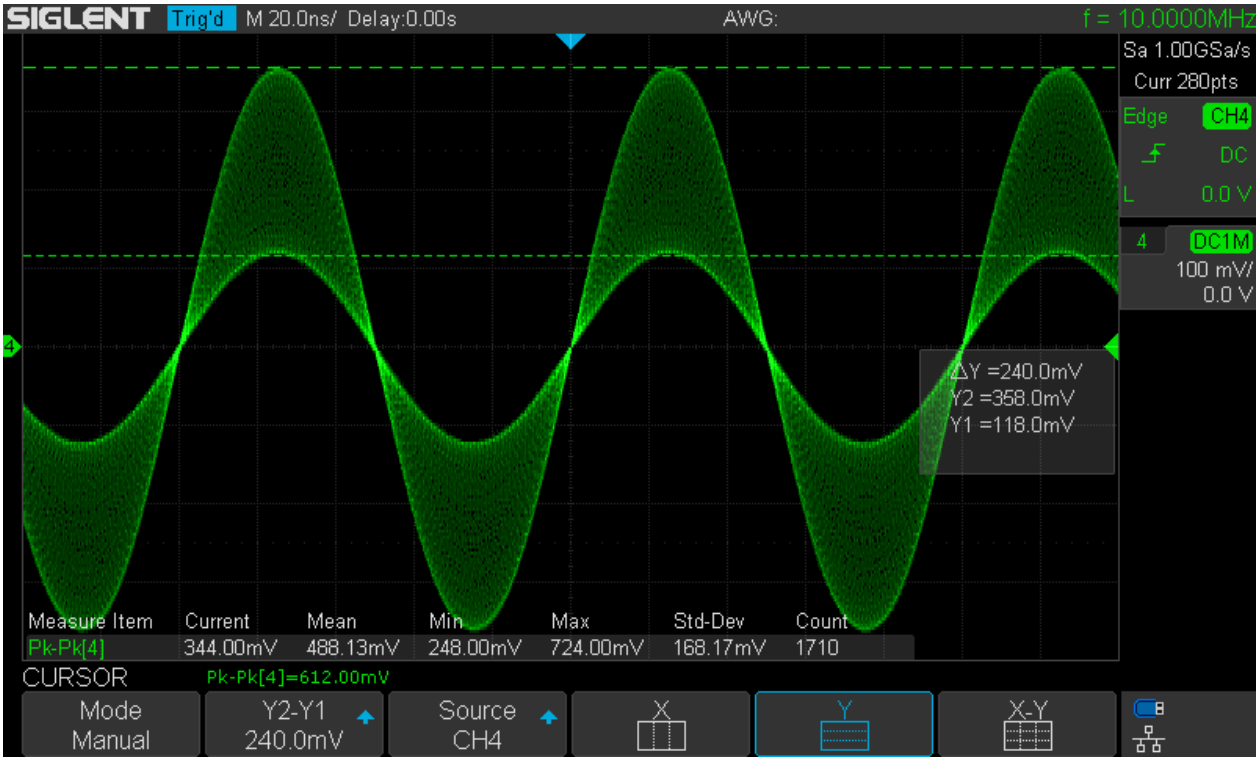
Manual

Manual cursor measurements are pretty straightforward, so they are dealt with here just for the sake of completeness.

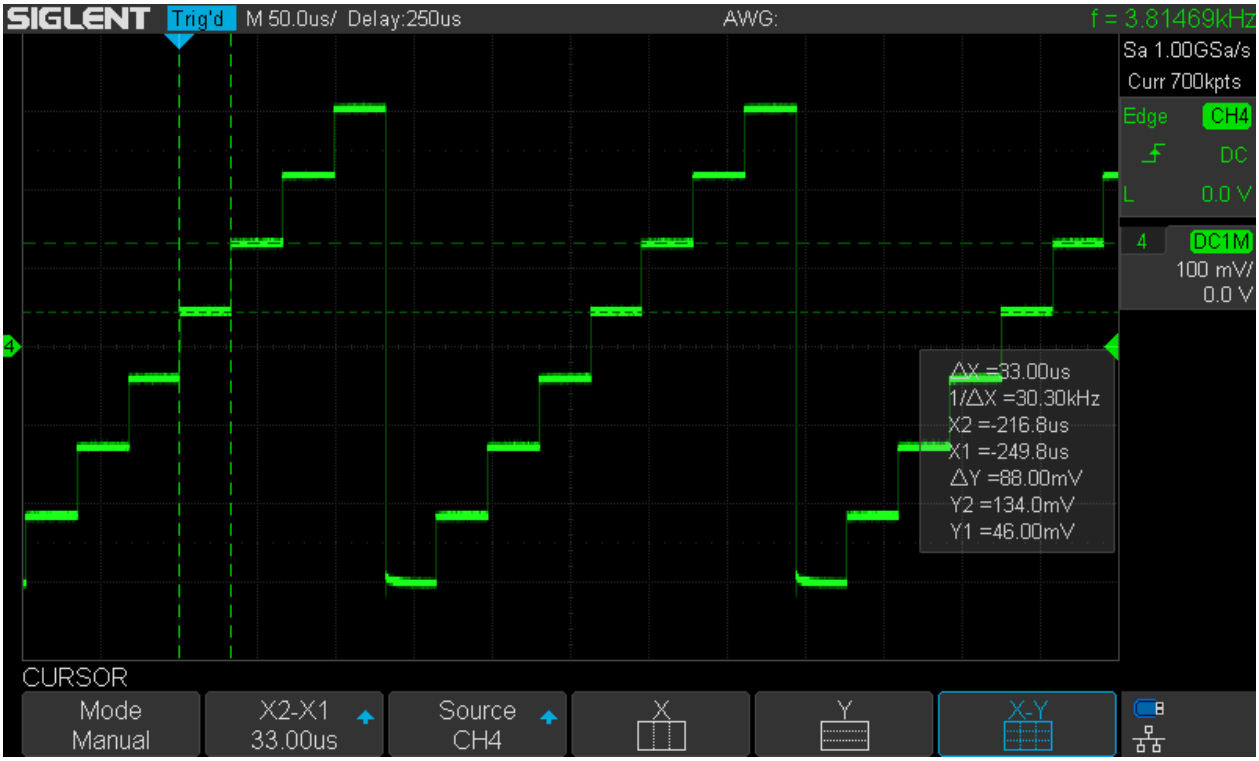
The following screenshots demonstrate the use of X, Y and X/Y cursors for measuring some signal properties where no corresponding automatic measurements exist. Furthermore, the use of cursors can make a screenshot more descriptive, as it becomes immediately clear what exactly has been measured.



Cursors_Man_X



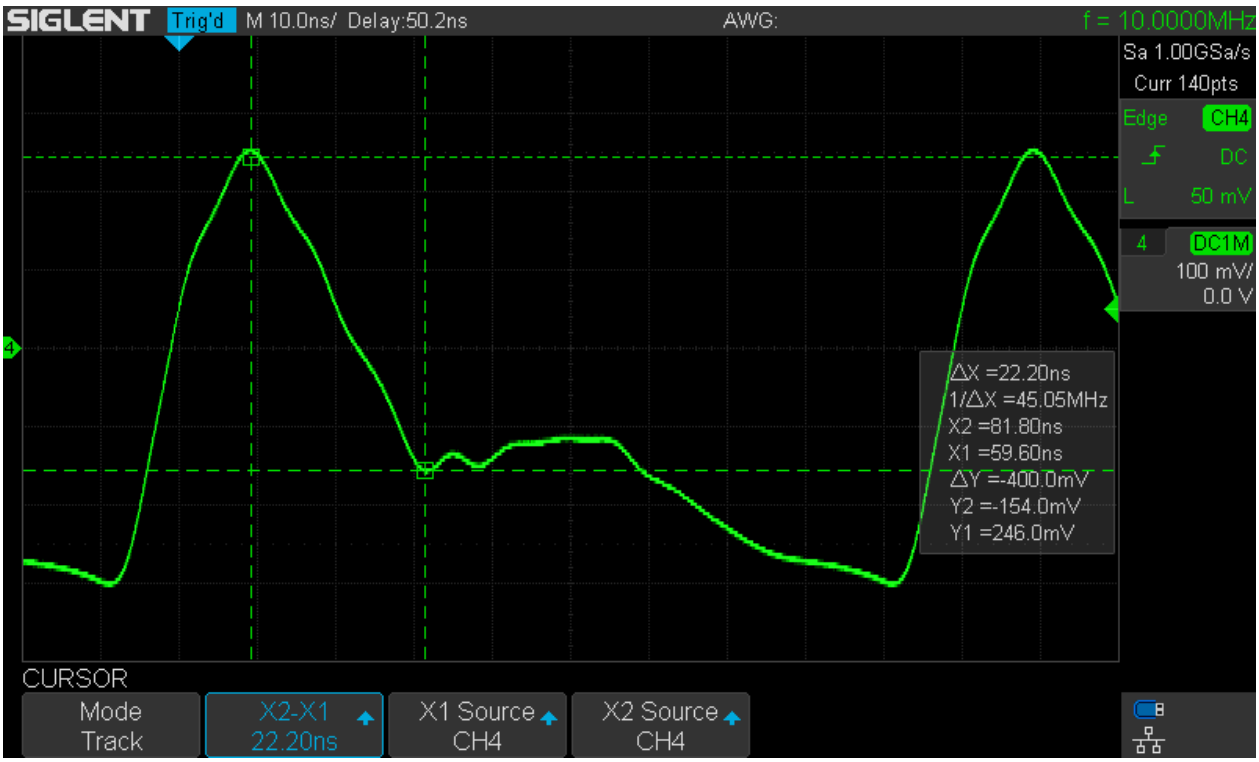
Cursors_Man_Y



Cursors_Man_XY

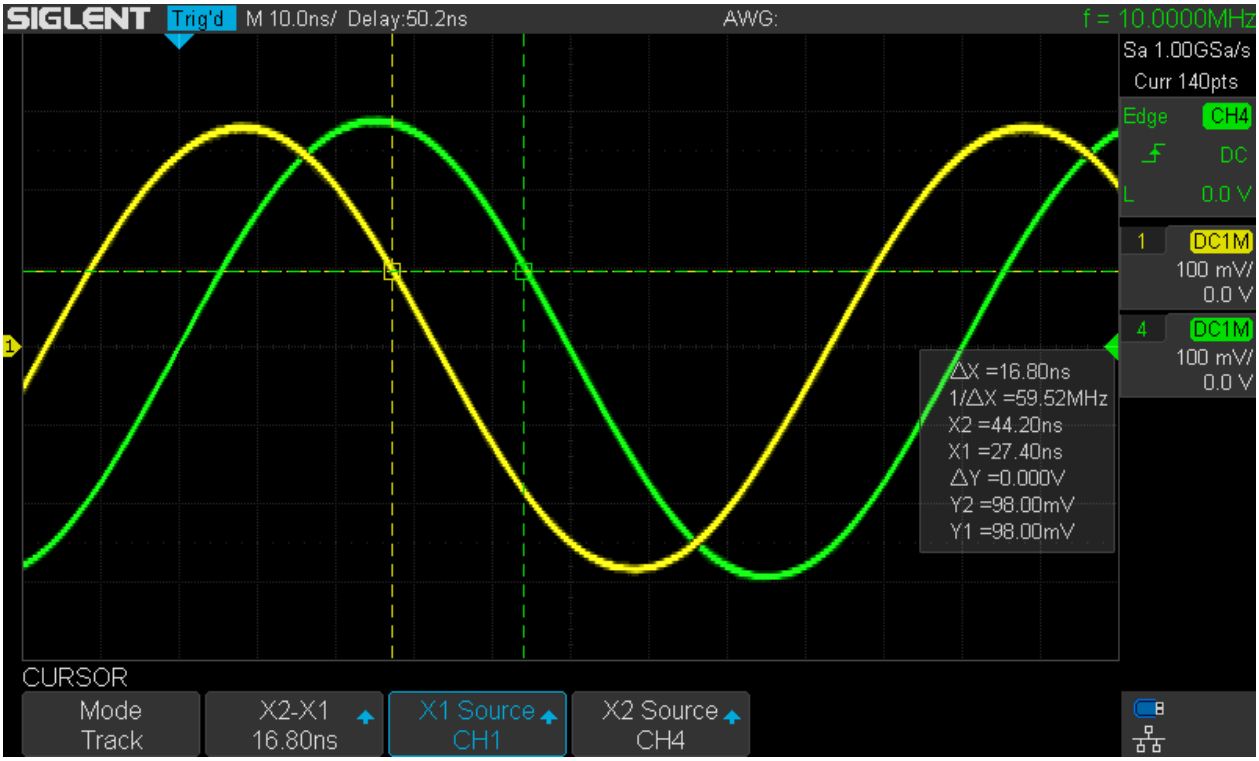
Tracking

Tracking cursors let us view the time and level at any point of the displayed waveform. At the same time, this can be compared to any other point, even on a different waveform on a different channel.

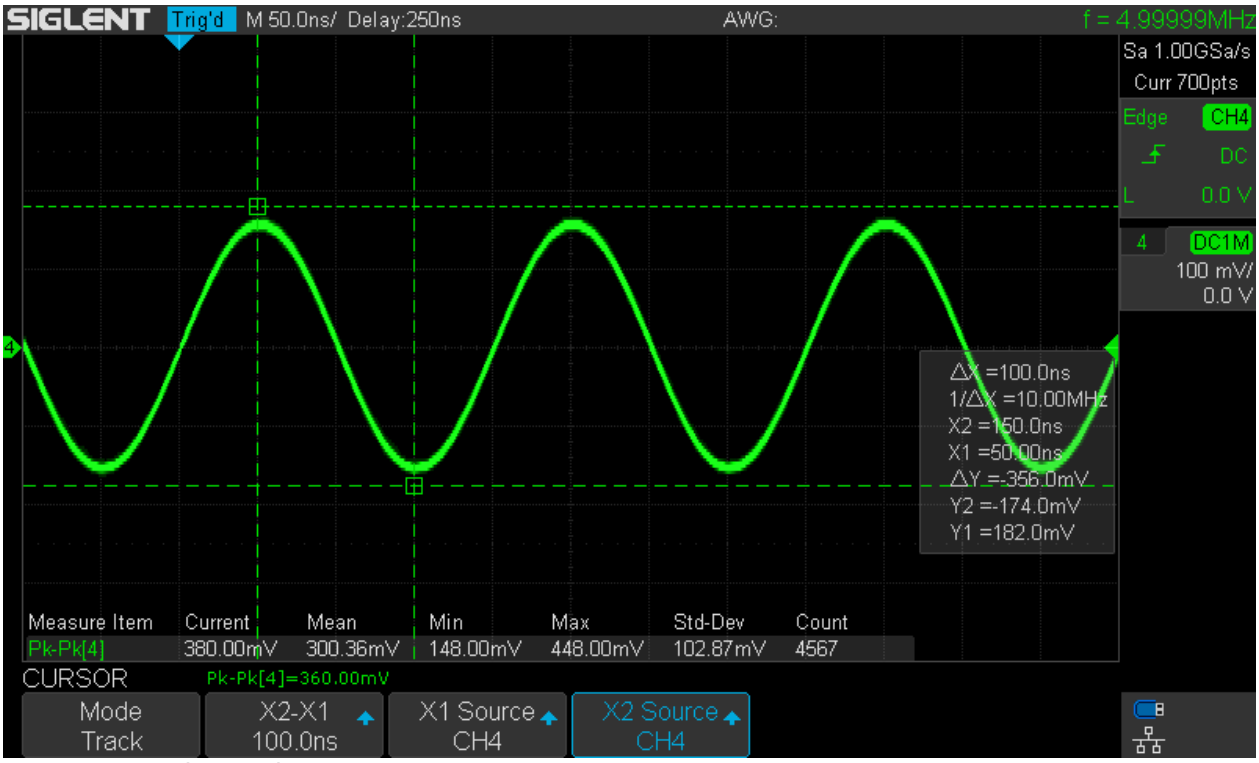


Cursors_Aut_Ch4

Measurement across channels is demonstrated by the example below, were we are just measuring the time delay and by adjusting the cursors so that the level difference is zero we ensure an accurate measurement.



Cursors_Aut_Ch14



Cursors_Aut_Ch4_Mod_0.5Hz

The last example demonstrates tracking cursors on a dynamic signal – a 50% amplitude modulated waveform in this case. Modulation frequency is quite low at 0.5Hz, still the cursors lag behind a bit. Plotting the cursors on the screen is not at all optimized for speed, even though the actual measurements in the cursor box update much faster than that; so it's really just the cursor display being rather slow.

Advanced Functions

Math

The math functions (other than FFT) are not exactly a strong point of the SDS1104X-E. They are pretty basic and I have to admit that I personally only use a few of them: Add, Subtract, Multiply, Integral and FFT. Other than that, I don't consider single operator math functions particularly useful, no matter how many of them were available.

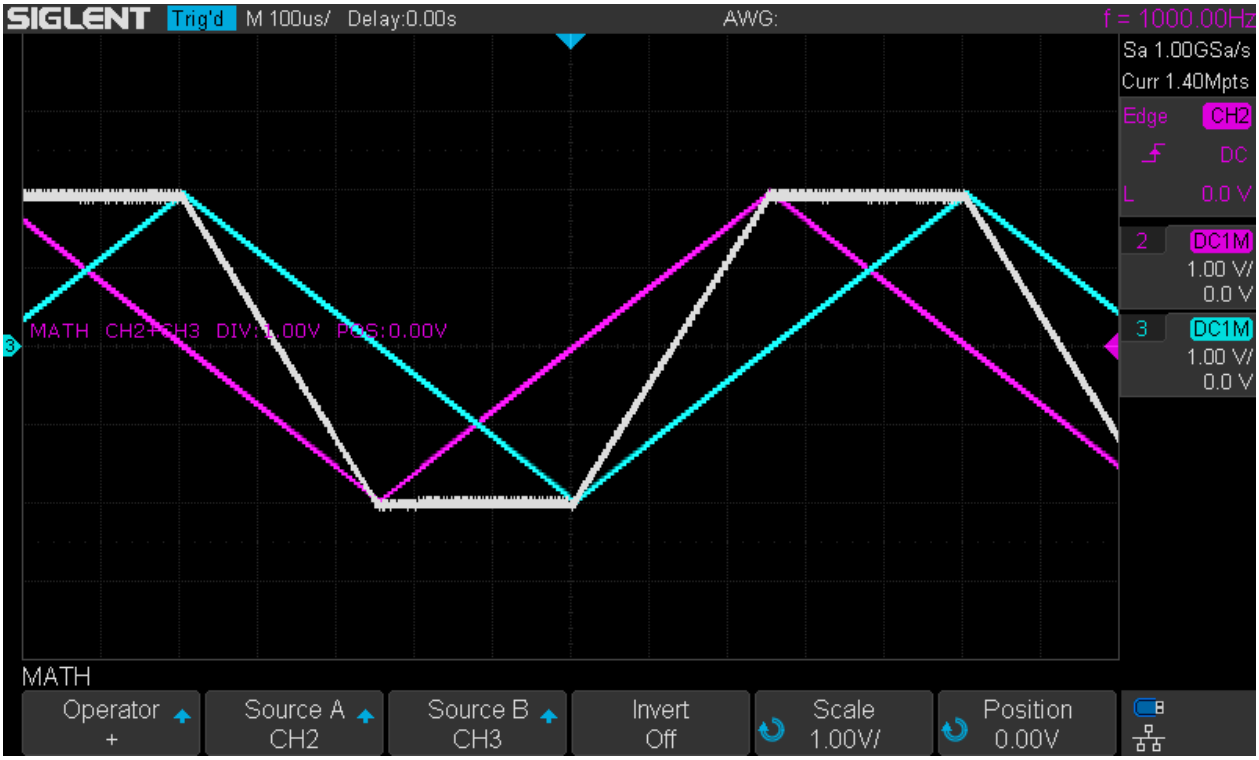
Whenever I have a need for math channels, then I might use more than one of them at the same time, want to be able to enter complex formulas and expect heavy support through a comprehensive library of math functions. None of these is available in the SDS1104X-E and this situation is not helped by the rather loveless implementation of most math operators, which often yields rather ugly, grainy and noisy results.

Add & Subtract

This is something even old analog scopes could do. In fact, it was only *Add* and a subtraction was accomplished by inverting the 2nd channel. There was also no dedicated math channel but the original traces were simply replaced by the sum of the two input channels.

Like most other DSOs, the Siglent SDS1104X-E has a dedicated math channel that is displayed together with the original input channels. The math channel has a white trace which doesn't look particularly pretty, as it is fat and grainy in most situations.

First is the sum of channels 2 + 3.
Second is the sum of channels 2 + inverted 3, to mimic the subtraction on an analog scope.
Third is the difference of channels 2 – 3, which should give the exact same result – and it certainly does.



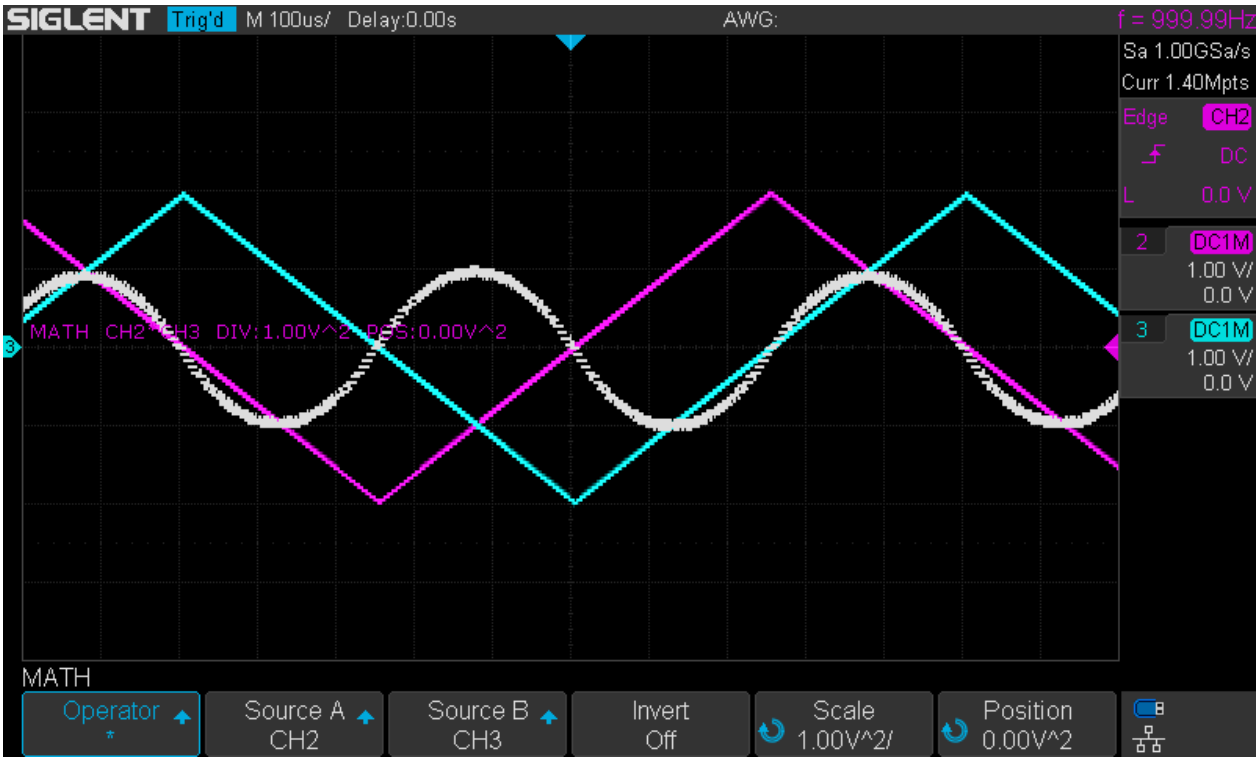
SDS1104X-E_Add

SDS1104X-E_Add_InvB

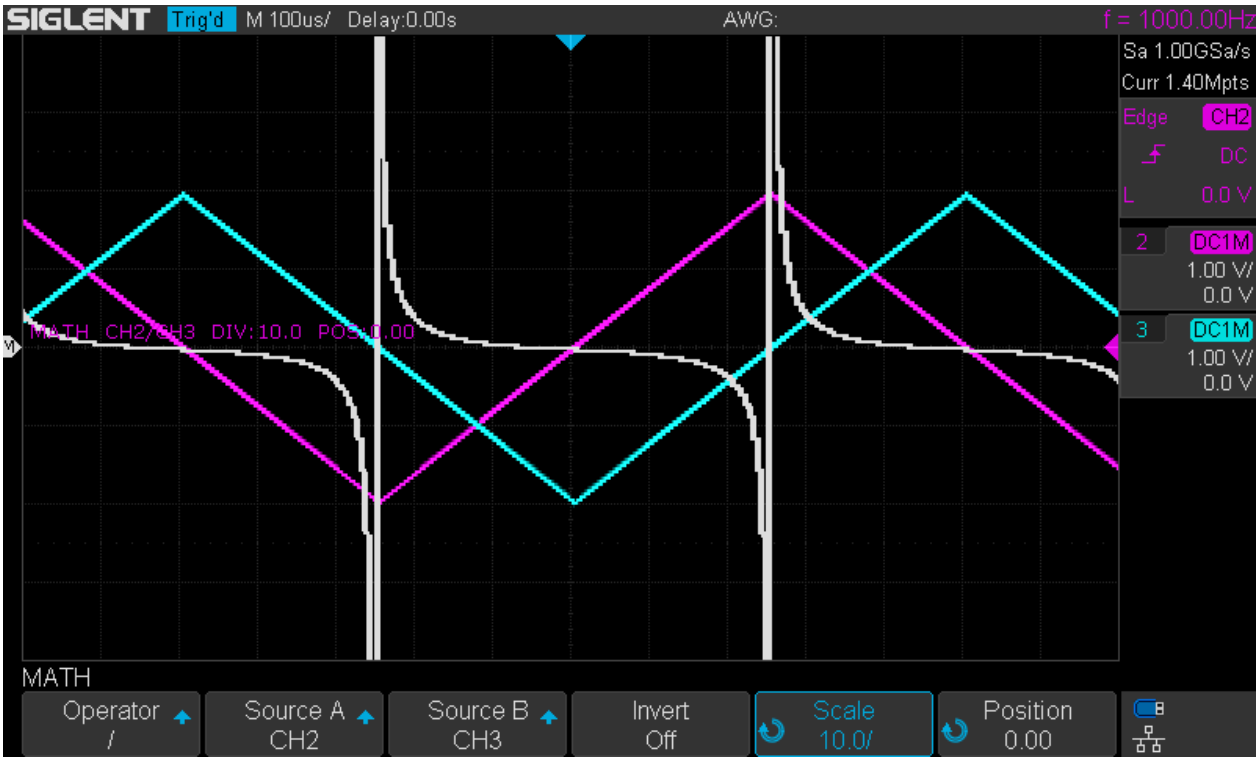
SDS1104X-E_Sub

Multiply & Divide

This could be used for signal gating as well as generating sum and difference frequencies – or producing a sinusoidal waveform out of two triangles, as in the example below.



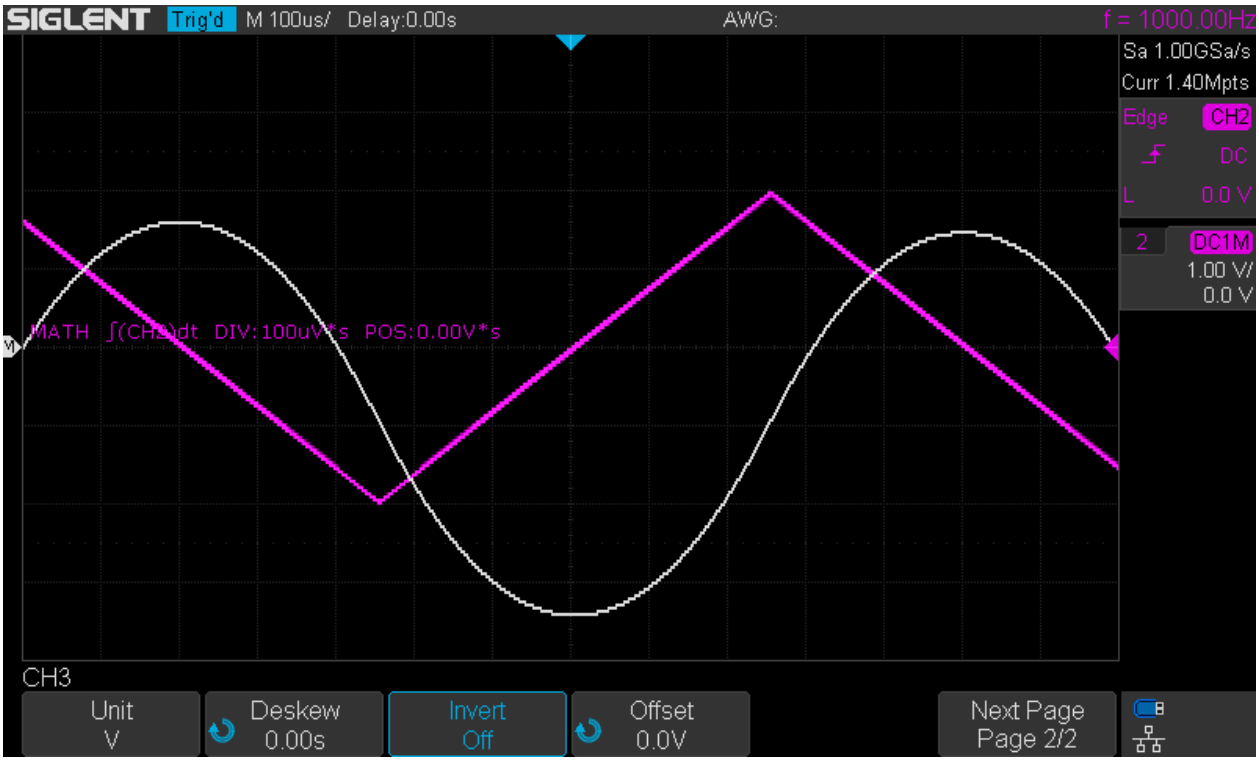
SDS1104X-E_Mul



SDS1104X-E_Div

The division as shown above looks reasonably nice and at least the scope is not thrown off track when the denominator becomes zero.

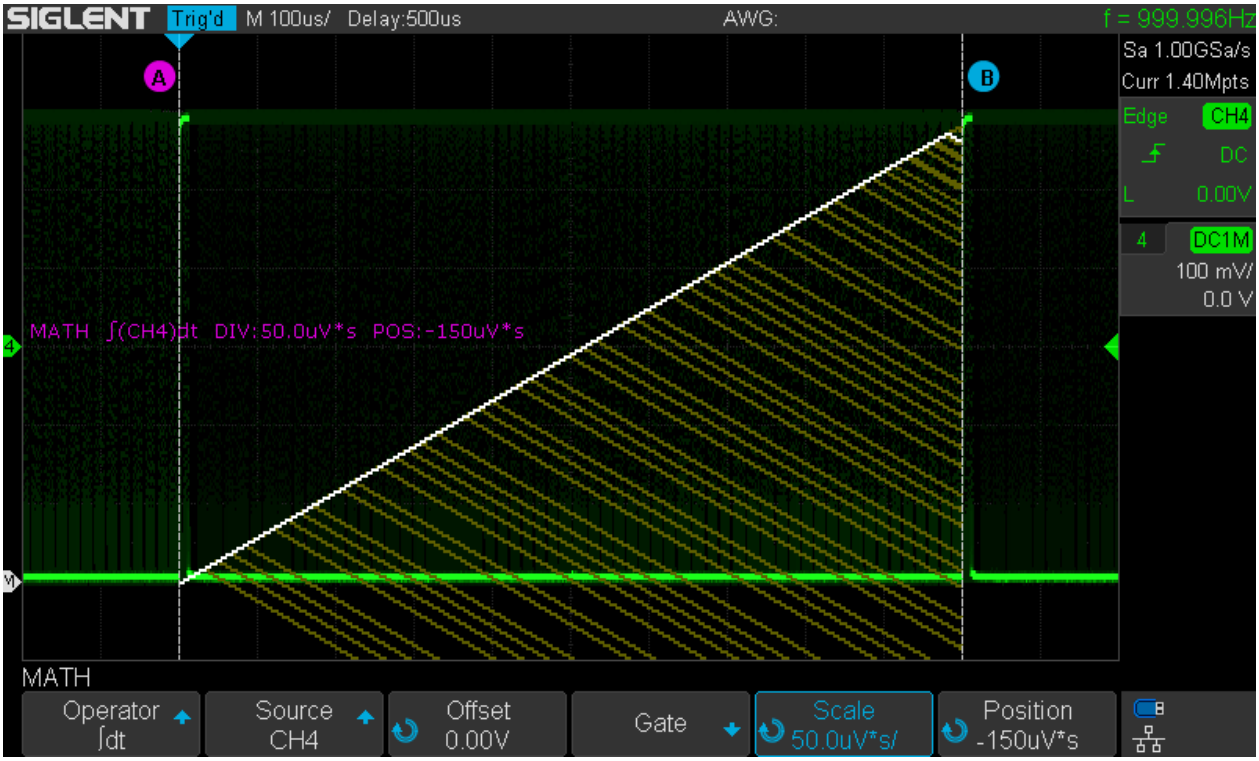
Integral



SDS1104X-E_Int

The integral function shown in the example above works very well by attenuating the harmonics of the triangle wave and thus turning it into a sinusoidal waveform.

The integral function is somewhat special in that it has an implicit gate function to calculate a definite integral. This can be used to display the output of a PWM signal.



Integrate_1ms_persistence_10s

In the example above, a PWM signal has been swept between 0 and 100% duty cycle and the gate for the integration has been defined for the PWM period. 10s persistence has been used in an attempt to visualize the dynamic signal in a static screenshot, which has been taken just before the 100% duty cycle had been reached. The white trace can be interpreted as the output voltage as a function of the duty cycle. The yellow lines are just previous math traces still visible because of the persistence.

Differential

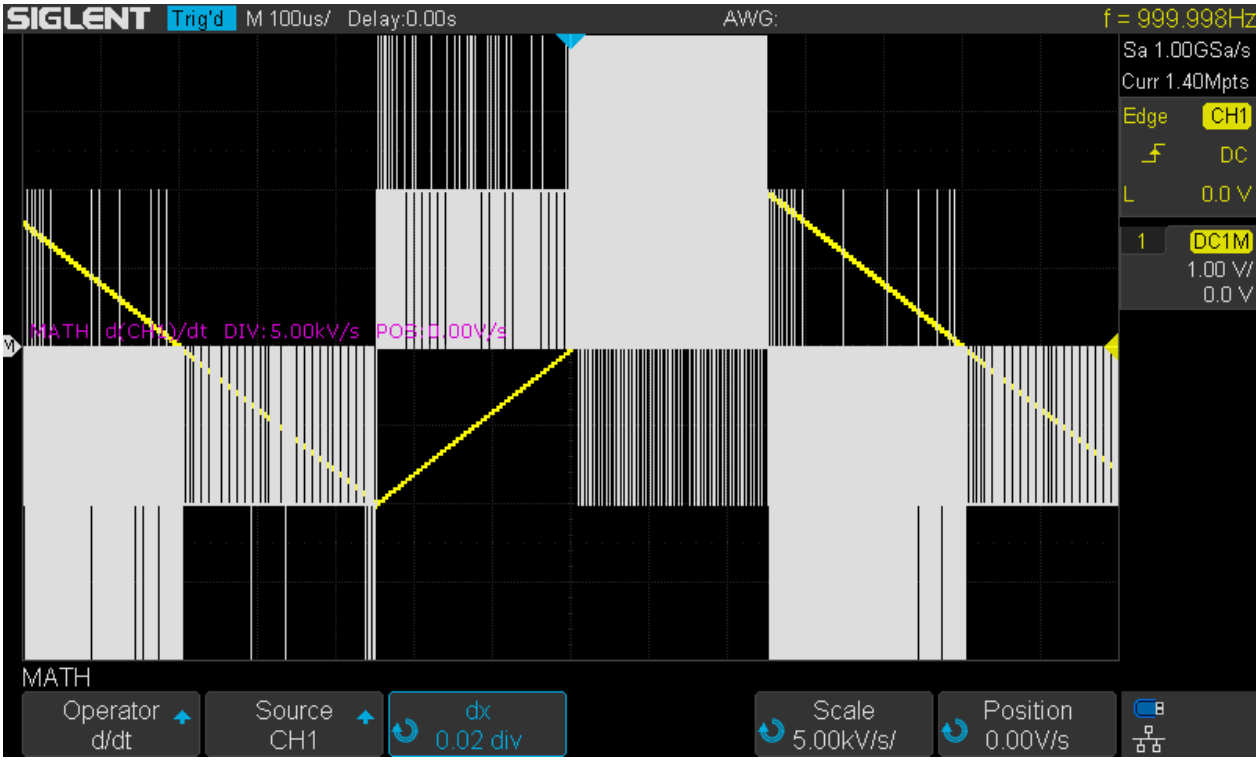
This is a particularly cumbersome math function that just cannot work well on an 8 bit system. It has a parameter *dx*, which determines the interval used for the difference computation. For an accurate and detailed result, we would want *dx* to be as small as possible and the lowest value that can be set is 0.02 div. With this setting, we can get totally useless results, as we will see later.

Why is it so? Let's do some simple math.

The screen is 800x480 pixels, with 700x400 dedicated to the trace area. Since there are 14 horizontal divisions, one division is 700/14 = 50 pixels wide. The lowest *dx* parameter would thus be 0.02 x 50 = 1, which means the difference between two adjacent samples is calculated. This will work well for fast transitions, but not for a triangle wave, which is about the worst we can throw at the differentiate function.

For the amplitude, the trace area on the screen shows 200LSB of the ADC, and there are 8 vertical divisions, hence we get 200/8 = 25 LSB per division.

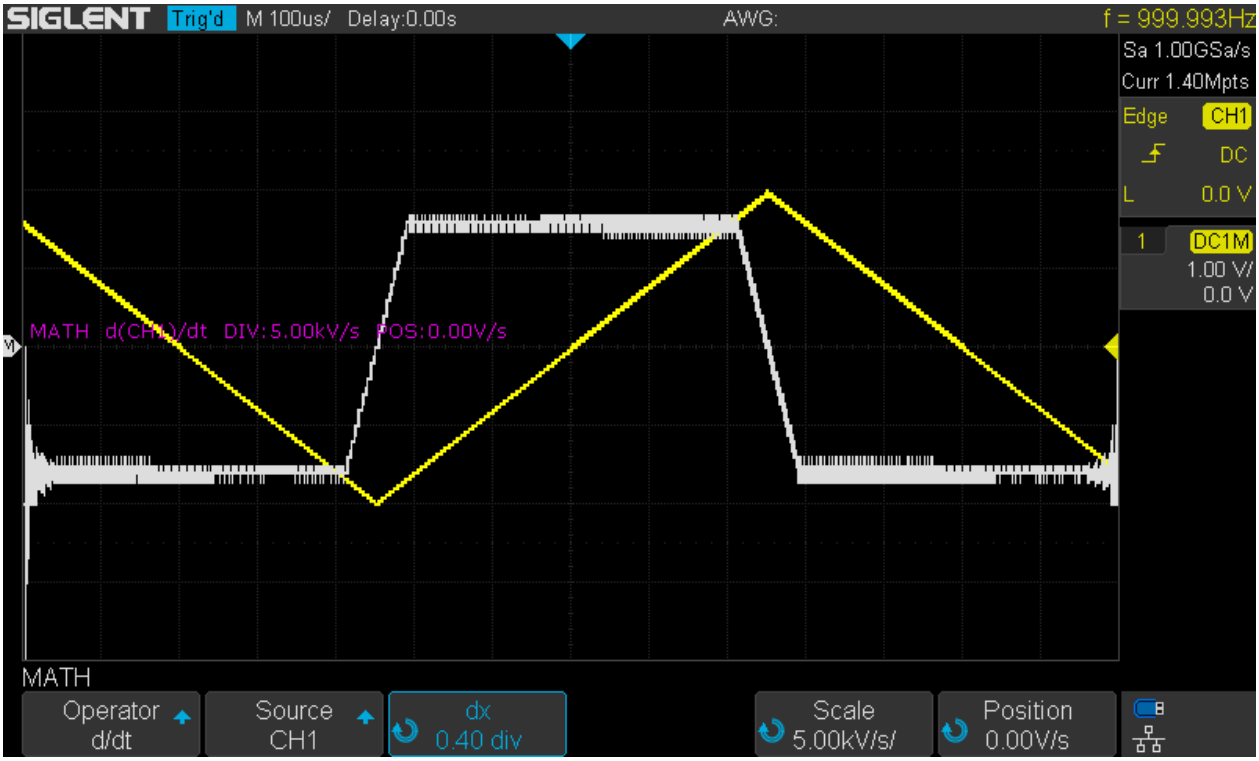
Let's have a look at a triangle (symmetrical ramp) with 4Vpp and 1kHz. A triangle should turn into a square when differentiated. One ramp (up or down) takes 500µs or 5 divisions at 100µs/div timebase and the amplitude is 4 divisions at 1V/div vertical gain. For a slope of 1 as with the triangle, one horizontal pixel (equivalent to *dx*=0.02) corresponds to half a LSB in vertical direction. This quite obviously cannot work and even *dx*=0.04 means just one LSB per time interval, where we are down in the DNL (differential nonlinearity) and noise. So no wonder that low *dx* parameter settings don't work with slow ramps.



SDS1104X-E_Dif_Ramp_Avg16_2%

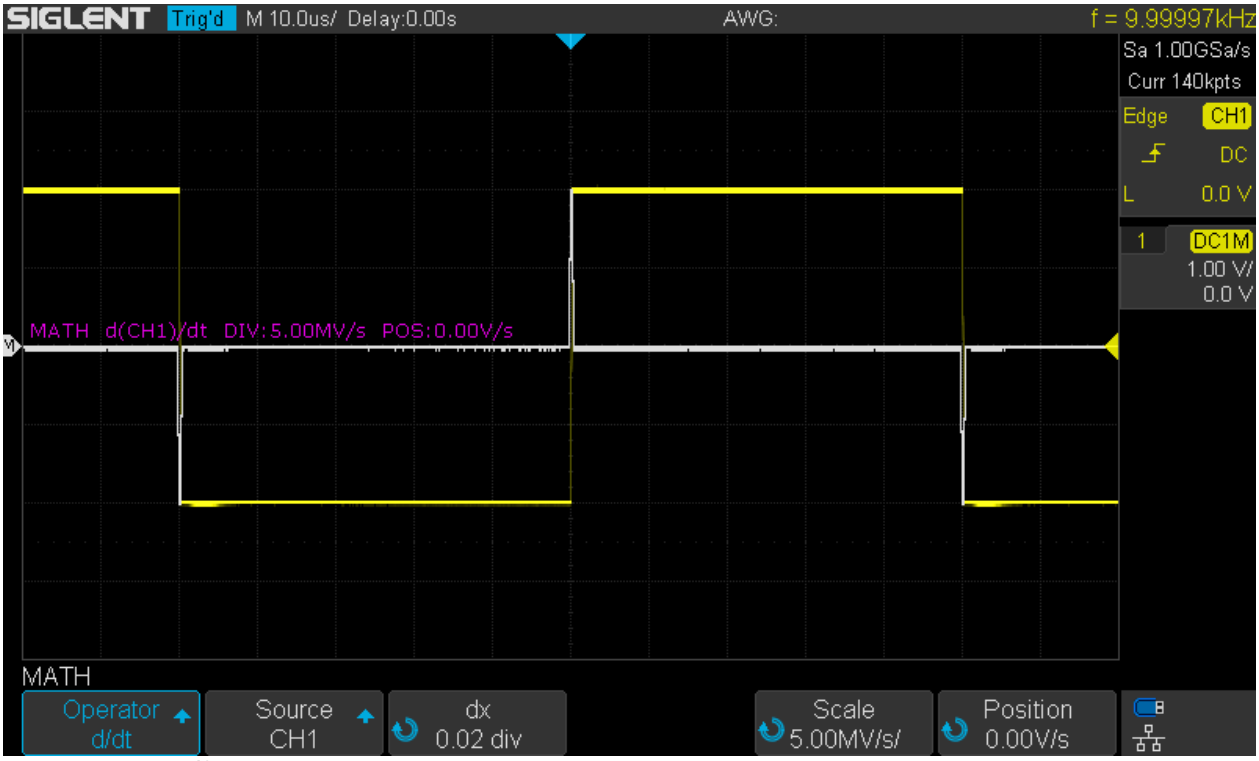
SDS1104X-E_Dif_Ramp_Avg16_4%

Even though the noise gets better with increasing time intervals, the results are still not very usable. With the maximum value 0.4 for *dx,* we get very slow transitions for the resulting rectangle, yet the math trace is fat and noisy.



SDS1104X-E_Dif_Ramp_Avg16_40%

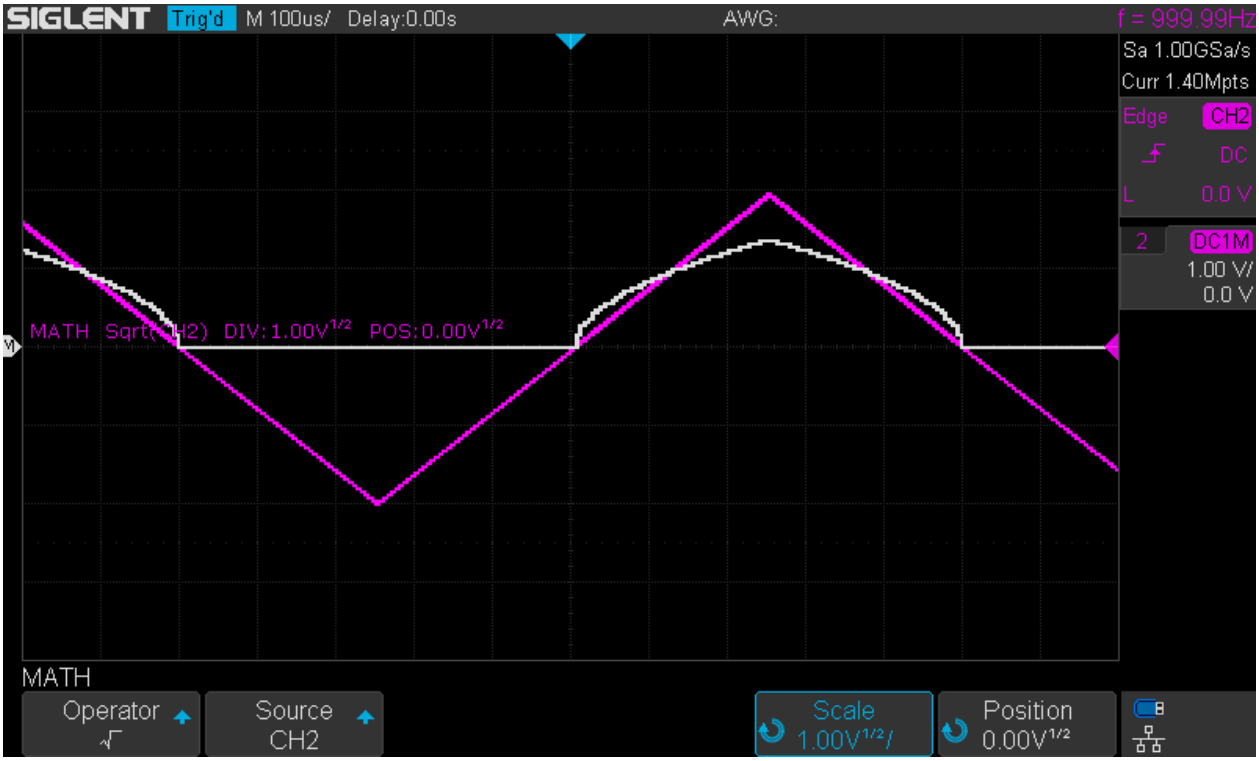
A square wave with fast transitions on the other hand is a perfect candidate for the differentiate function, and particularly so with the lowest possible *dx* parameter of 0.02.



SDS1104X-E_Dif_Square_2%

Square Root

Might be useful to convert some sensor signal, that represents power, back into voltage/current.



SDS1104X-E_Sqrt

Poor Men's Differential Probing

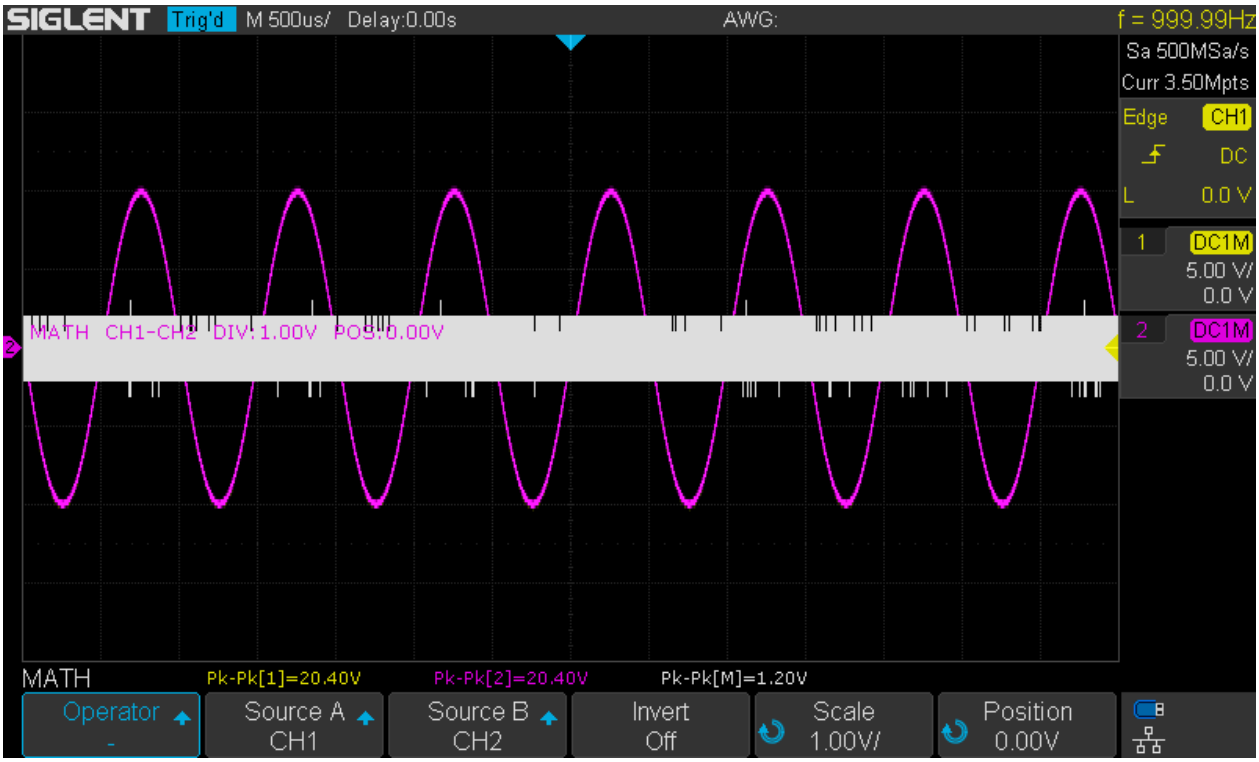
With analog scopes, we were able to combine two regular (single ended, ground referenced) channels into one differential channel. This was done by adding both channels with the 2nd channel inverted, whose gain had to be fine tuned in order to give a maximum of common mode rejection. Of course, this solution was far from ideal and sensitivity as well as common mode rejection were rather limited, especially at higher frequencies, which made it hard to get meaningful results when common mode voltages were high compared to the differential signal.

Digital scopes generally seem to have pretty much lost that functionality, even though the preconditions appear good at first glance. After all we have a separate math channel that has its own individual gain and position, which should be ideal for viewing small signals in presence of large common mode voltages. But the problems start already when we attempt to fine-adjust the channel gain in order to cancel out the common mode signal as much as possible; this is just not possible. In fact, the math functions completely ignore the channel gain settings except for the input attenuator, which just cannot be ignored. This seems to be convenient at first, as it makes the math channel somewhat independent from the regular channel settings, but makes it impossible to fine-tune the gain in order to get it equal across the channels.

Furthermore, the individual gain for the math channel doesn't actually help, as it can only be a software zoom and an 8-bit resolution is not up to the task, particularly when the math gain is set higher than the channel gain.

The screenshot below demonstrates the result of two identical signals fed into channels 1 and 2 at 5V/div and a difference math channel is set to a 5 times higher gain at 1V/div. The result is just an approximately 1 division wide bar. Common mode rejection can be estimated from the amplitude measurements and would be 1.2V/20.4V = 0.0588 ~ -24.6dB, which is certainly not sufficient for differential measurements.

As a conclusion, poor men's differential probing doesn't work with this scope and the same is most likely true for the majority of other DSOs as well. For tasks that require floating measurements and/or differential probing, there really is no way around getting the appropriate physical probes.



SDS1104X-E_PoorMen_Diff_Probing

FFT

Now we've finally arrived at one of the highlights of the SDS1kX-E series, the 1Mpts Fast Fourier Transform, which turns the scope into some sort of spectrum analyzer. This can be found in the math *Operator* menu, yet it deserves a dedicated chapter simply because it's almost something like an instrument within the instrument.

Thankfully, Siglent have implemented a more spectrum analyzer oriented user interface instead of treating the spectrum plot just like an ordinary Y-t signal trace. This means a big plus in terms of usability.

Even though there are basically no automatic measurements, markers or analysis applications, which is in contrast to modern spectrum analyzers and higher end DSOs, this does not prevent us from getting all the answers we could possibly expect when using an 8-bit DSO for analyzing a signal in the frequency domain. We should not forget that there have been times where even dedicated SAs had barely anything but a manual marker to determine the approximate frequency at any point of the spectrum – and yet engineers got their job done.

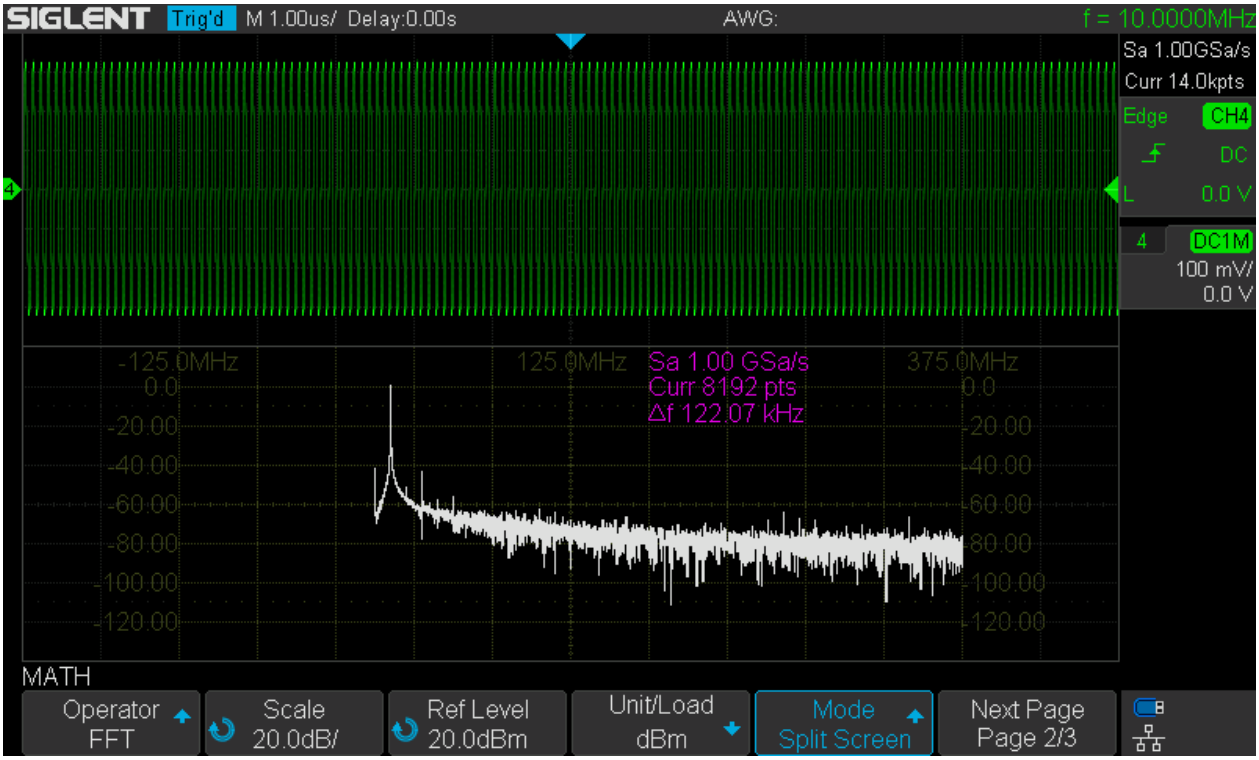
Generally speaking, we should not get too obsessed with features and gadgets, but concentrate more on the quality of measurements. All the bells and whistles won't serve us anyway, unless we get low distortion and noise, accurate levels, a reasonable high spurious-free dynamic range and adequate frequency resolution. These are parameters that really matter for any SA and make the difference between a useful tool or just another "me too" feature.

Because of this, the FFT will be evaluated just like the base functionality on any real spectrum analyzer, so this is going to be a rather lengthy review.

Basic Operation

This section covers some of the fundamentals of the user interface that need to be understood as a basis for the following chapters.

FFT is enabled by pressing the [**Math**] button on the front panel and then selecting “FFT” from the *Operator* menu. This will show the FFT window with some random settings and hopefully in Split Screen mode. If not, go to page 2 of the *FFT* menu and set this mode there.



FFT_first_open

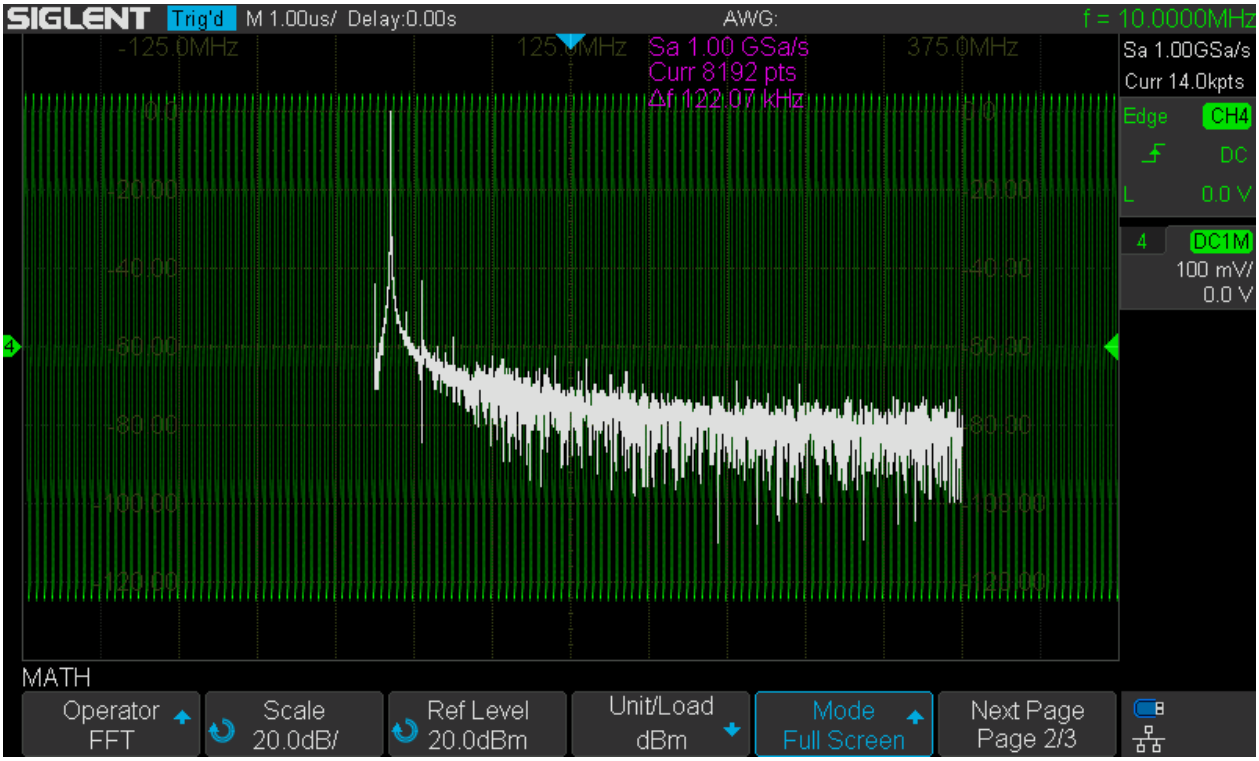
Screen Modes

This is the very first thing to know; there are three different ways to display the FFT:

- Split Screen
- Full Screen
- Exclusive

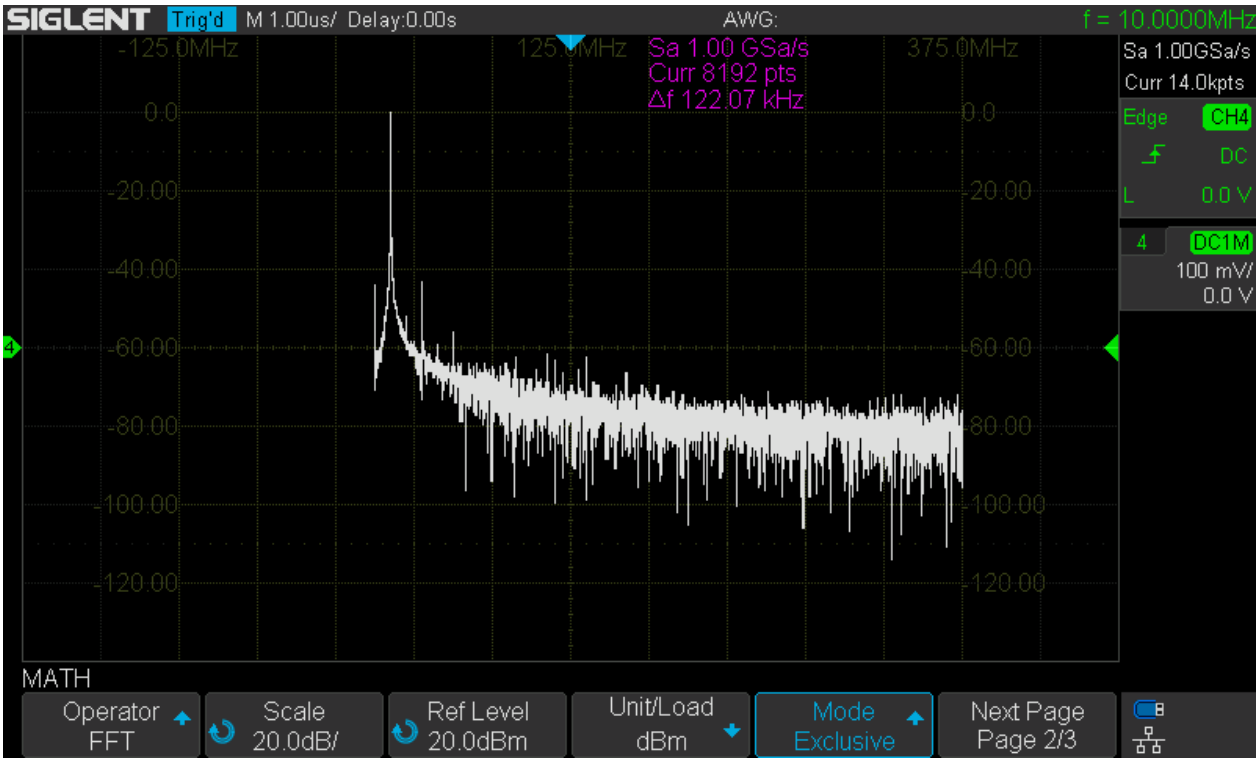
Split Screen as has been shown in the previous screenshot is most useful for setting up the measurement, as it allows the observation of the input signal in the time- and frequency-domain in parallel. This is important, because as a general rule we want the signal to nearly fill the screen height in order to maximize the dynamic range. On the other hand, we need to avoid clipping of the input signal and saturating the ADC by all means, because this will lead to distortions and in turn generate lots of unwanted harmonics and spurious signals. Consequently, the previous screenshot shows a proper setup of the vertical gain and/or output of the signal source and this has to be observed during setup.

Full Screen could serve the same purpose as split screen, but now both frequency and time domain are stacked on top of each other which obscures the grid and its labels and might be distracting in general. This is also why I personally don’t like this mode very much and rather use Split Screen whenever I need to see the Y-t display.



FFT_Mode_Full_Screen

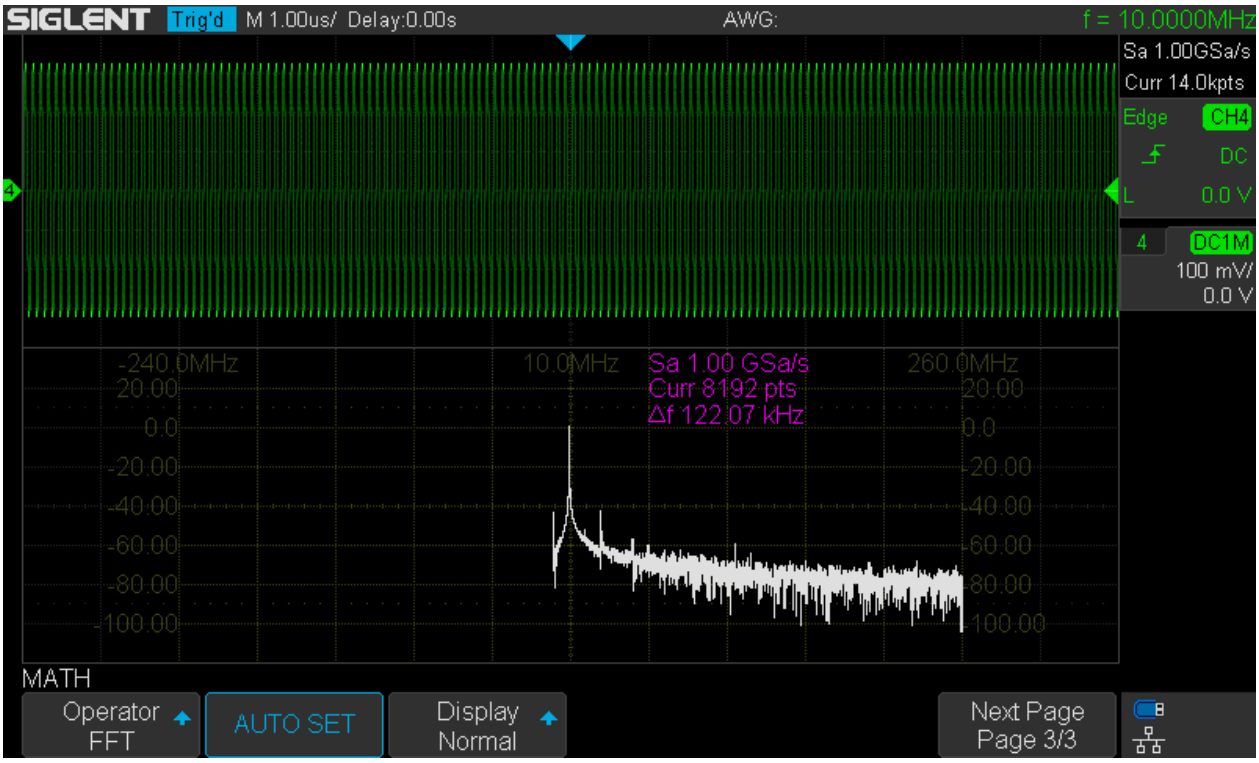
Exclusive shows the FFT exclusively and makes the instrument look more like a dedicated spectrum analyzer. Use this mode once the measurement is properly set up and the input signal amplitude is more or less stationary.



FFT_Mode_Exclusive

AUTO SET

This can be found on page 3 of the *FFT* menu and generally does a lousy job. It sets the center frequency according to the strongest signal, thus could be used as a *starting point* for narrowband signal analysis. Other than that, it's best to stay away from the AUTO SET function and there is no way to avoid setting up the instrument manually as described later in this document.



FFT_Auto_Set

Units

This submenu can be found on page 2 of the *FFT* menu. The units used for amplitude measurements, i.e. the labeling for the Y-axis of the grid, are selected here. The same submenu also allows the specification of the external load impedance. This is required for the dBm units, which specify a power level instead of voltage/current.

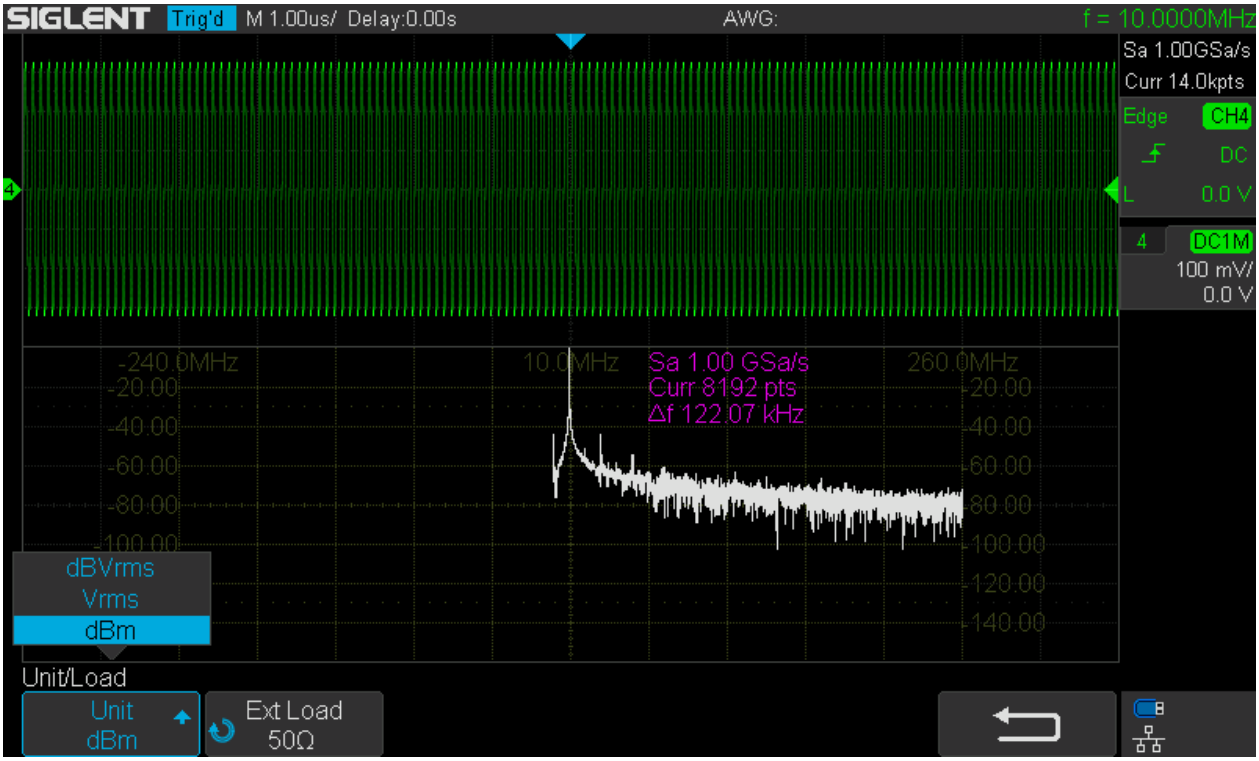
For general laboratory use, we would universally have an external 50Ω pass-through terminator, hence a 50Ω load impedance. The same is true for most RF applications, even though there are exceptions especially for the antenna inputs of domestic broadcast receivers. 75Ω is the standard for video signals, as well as 600Ω for professional Audio. For Audio in general, the use of dBm is not common but might be useful for characterizing the output of power amplifiers, where it comes in handy that the Siglent FFT allows us to specify load impedances down to 1Ω.

dBVrms is the relative voltage level expressed in decibel with reference to 1Vrms. This is the right setting for (unspecified) high impedance source/load configurations.

Vrms is the absolute voltage, which also means that we get a linear Y-axis that severely limits the dynamic range that can be displayed. I really don't see much use for that.

dBm is the relative power level expressed in decibel with reference to 1mW into the specified load impedance. This is the preferred unit for general laboratory use as well as RF applications.

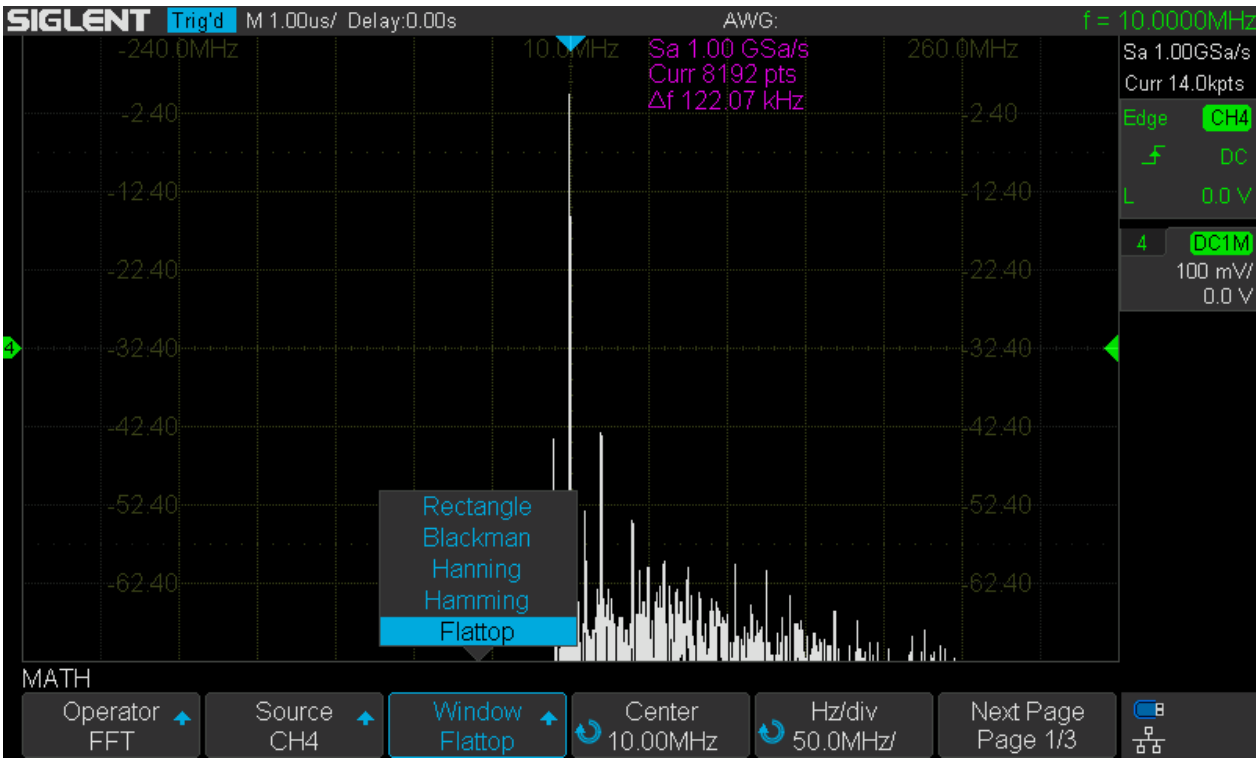
The screenshot below shows how the FFT is set up for dBm units and 50Ω external load. At the same time, an external 50Ω pass-through terminator has been fitted for channel 4.



FFT_Units

Window

Another important setting is the window function on page 1 of the *FFT* menu.



FFT_Windows

This is roughly equivalent to selecting the properties of the final IF filters in a real SA. Of course, the latter do not offer such a choice, apart from the selectable IF bandwidth. But FFT works differently than a swept spectrum analyzer; we cannot set the analysis bandwidth directly and the “final IF filter” is just some signal processing on a limited set of sample data (record) gathered within a certain time interval, where the actual input signal is not zero outside this interval. This causes computation errors and produces artifacts. This is why the window function is required to do some pre-conditioning on the sample data and it comes down to providing a suitable compromise between bandwidth, amplitude accuracy, filter shape and selectivity including leakage (side lobe suppression). The simplest window is the rectangle which has the narrowest -3dB bandwidth but very poor properties otherwise. Historically a number of window functions exist that provide a better compromise at the expense of an increased -3dB bandwidth. Some window functions have been particularly optimized for a certain property and these are also the most beneficial ones for practical use – in my book, at least.

The table below gives an overview of the window functions available in the SDS1104X-E. Note that some windows have parameters (Hanning in this selection) and I do not know what Siglent has actually implemented, so the table can only show a span of properties for the usual range of these parameters. Also note that “Bins” refers to what is displayed as “Δf” (frequency step) within the FFT window.

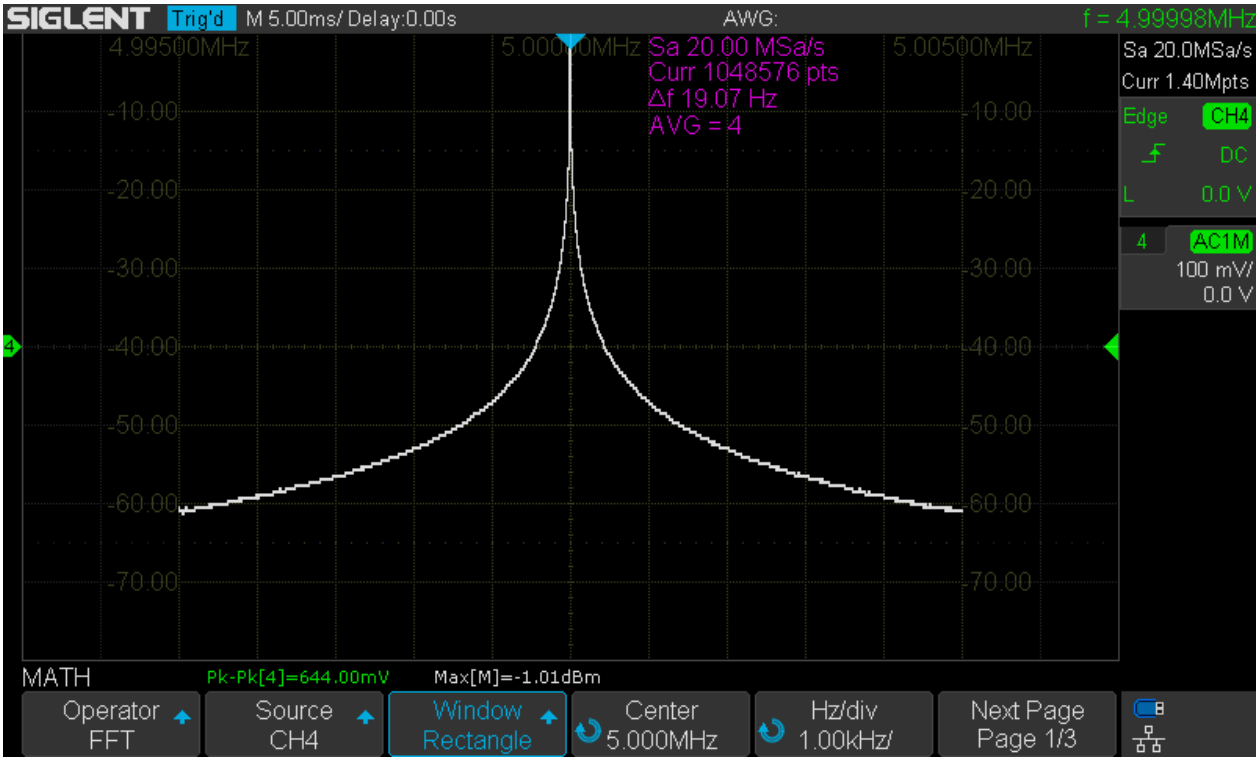
Window	**-3dB BW (Bins)**	**Max. Side Lobe [dB]**	**Side Lobe roll-off [dB/Oct.]**	**Remark**
Rectangle	0.89	-13.2	6	Min. 3dB bandwidth, used for short transients
Blackman	1.68	-58	18	High side lobe suppression, used for audio
Hanning	1.20 ~ 1.86	-23 ~ -47	12 ~ 30	Used for audio and vibration measurement
Hamming	1.30	-41.9	6	Used for speech analysis
Flattop	2.94	-44	6	Negligible pass band ripple, used in spectrum analyzers and for calibration

Many more window functions do exist and I personally would love to have Gaussian and particularly Blackman-Harris available, but that’s not really a problem, especially not for an 8-bit system. For the time being I have the following recommendation for anyone who just wants to use the FFT right away without striving for an university degree in digital signal processing (RBW = Resolution Bandwidth):

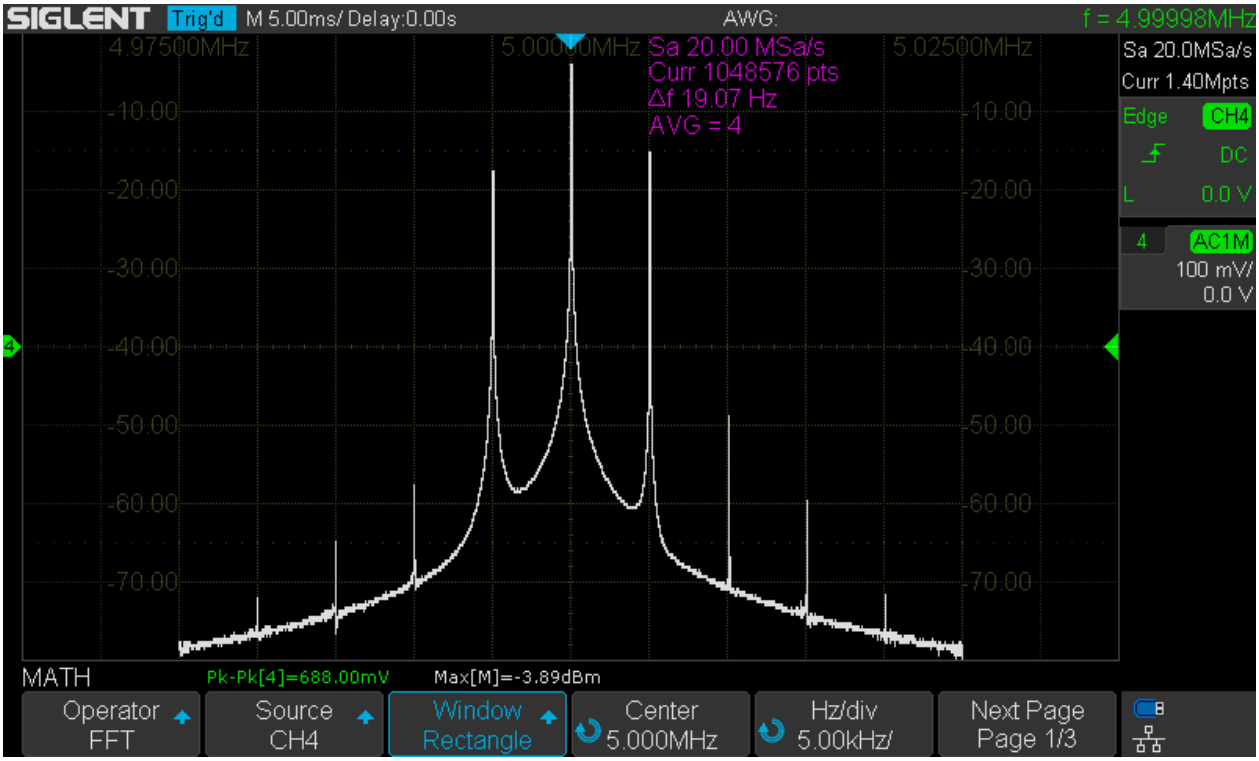
- Use Flat-Top whenever best amplitude accuracy is important. RBW ~ 3 x Δf
- Use Blackman otherwise, especially when a high dynamic range is desired. RBW ~ 1.7 x Δf

The following screenshots show all available window functions in two situations:

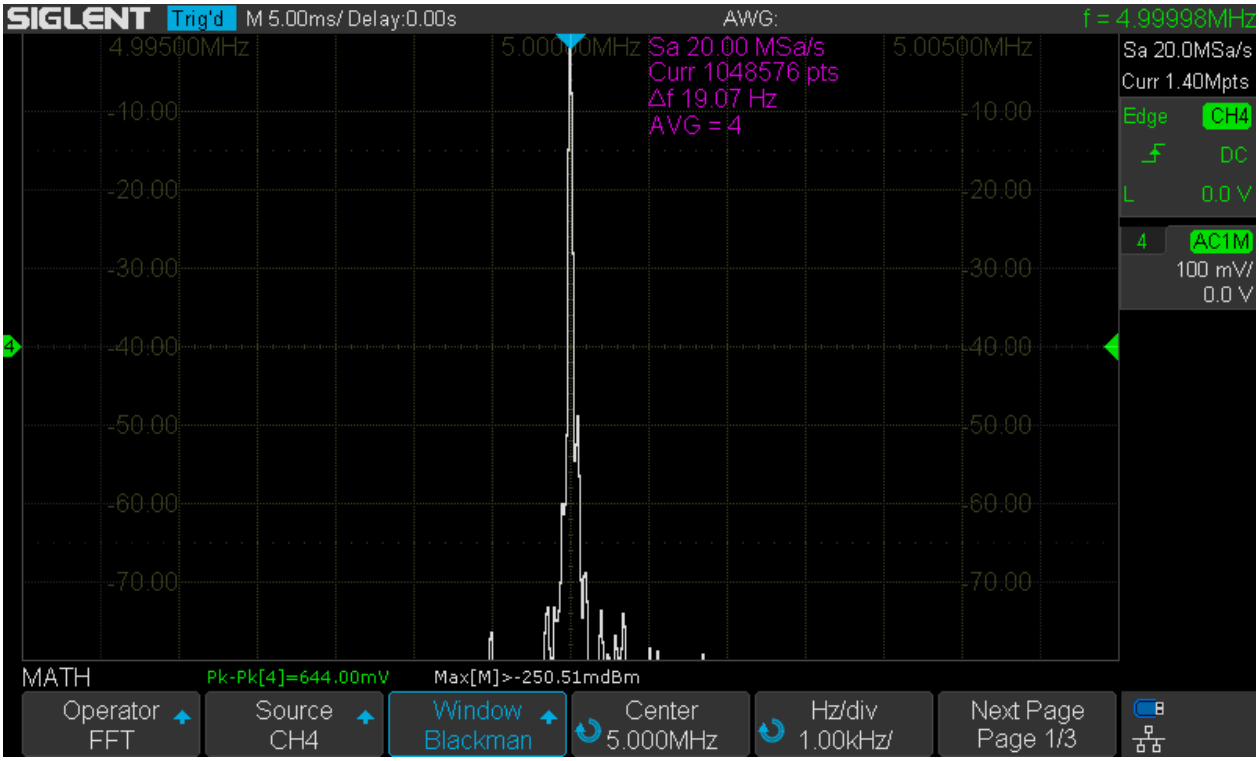
1. Sine, 5MHz, 0dBm, displayed at 1kHz/div to show the selectivity and filter shape
2. Sine, 5MHz, -3dBm and 50% AM with 5kHz, displayed at 5kHz/div



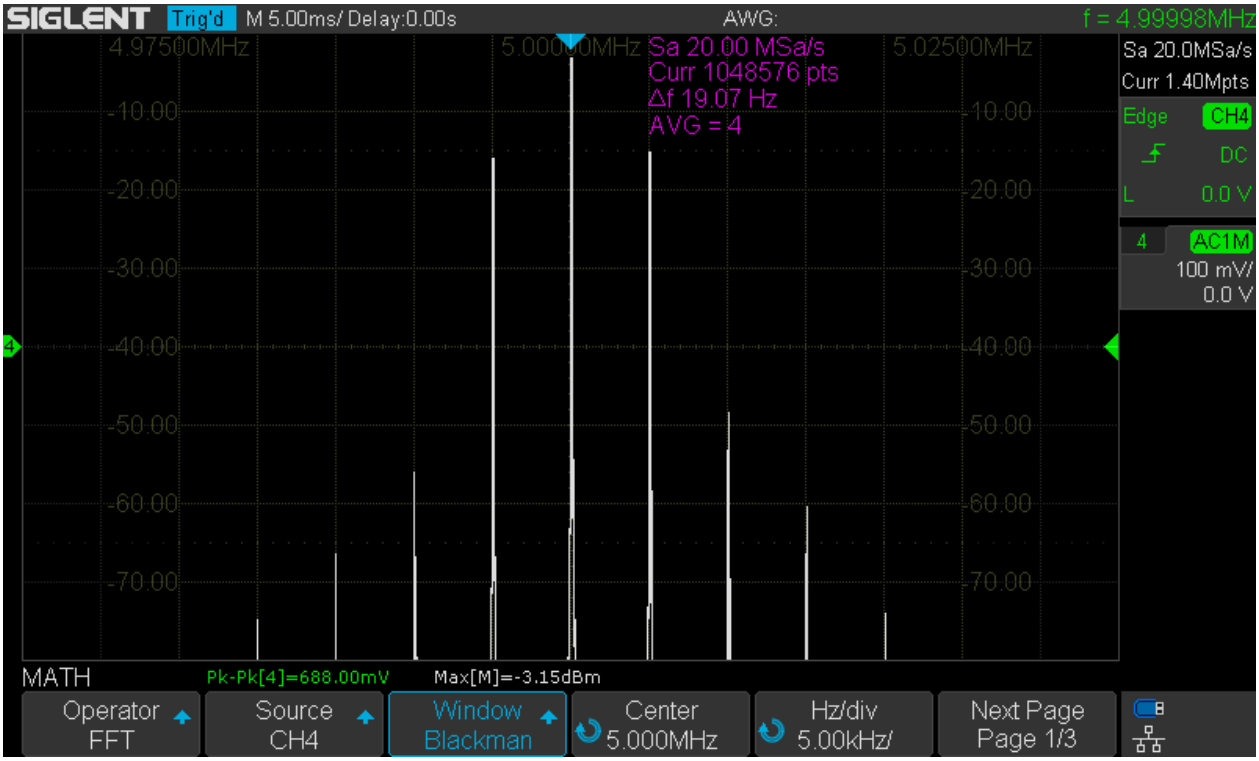
SDS1104X-E_FFT_Rectangle



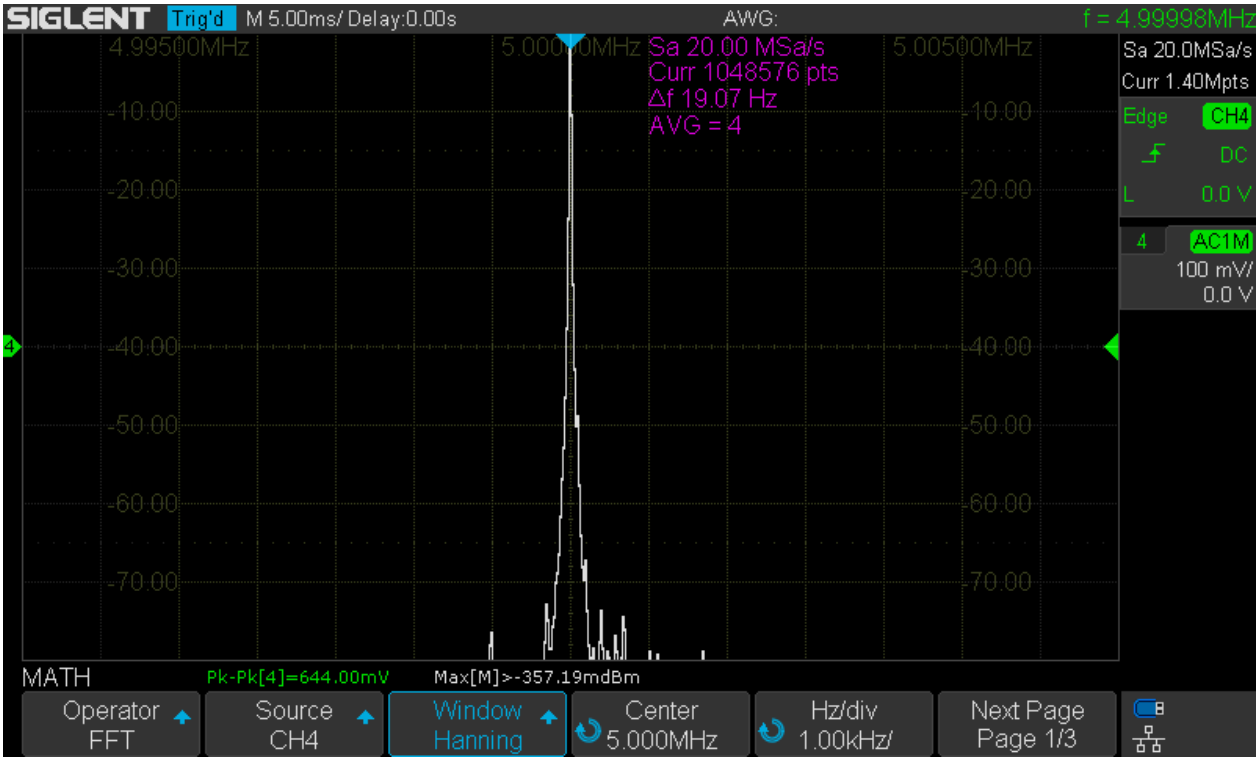
SDS1104X-E_FFT_Mod_Rectangle



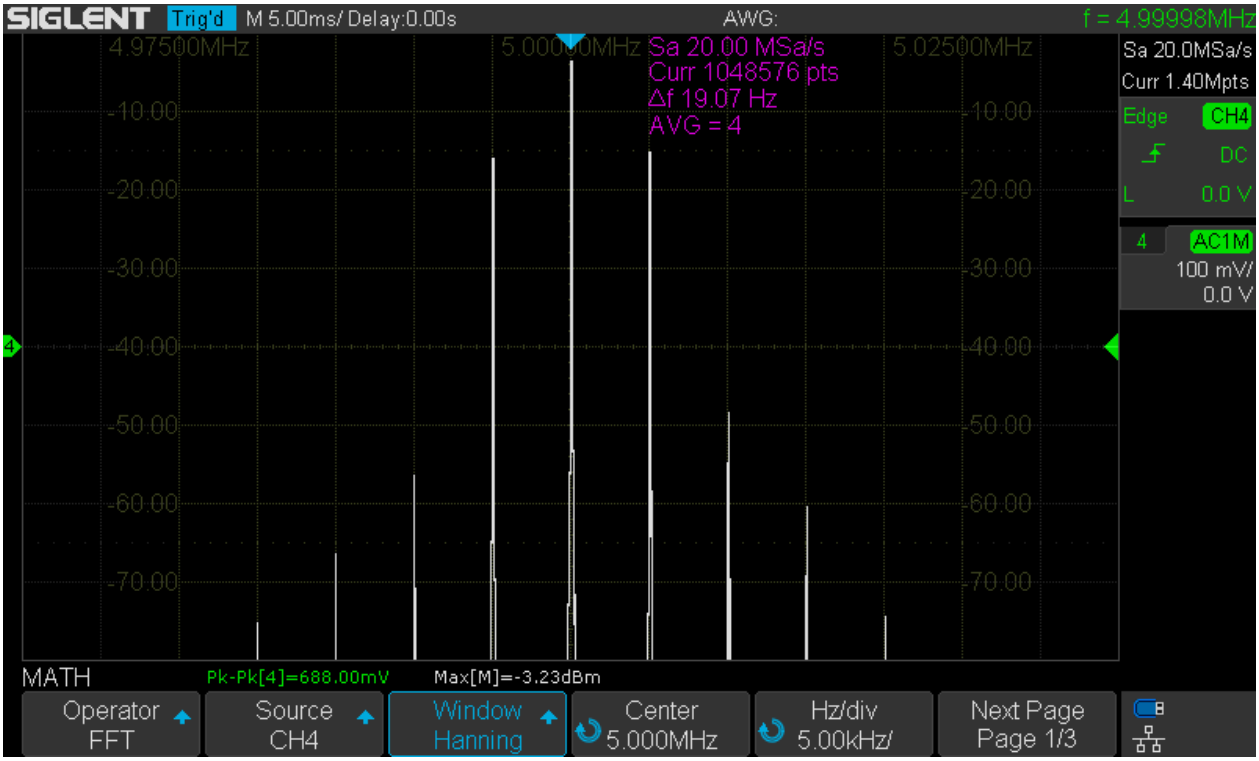
SDS1104X-E_FFT_Blackman



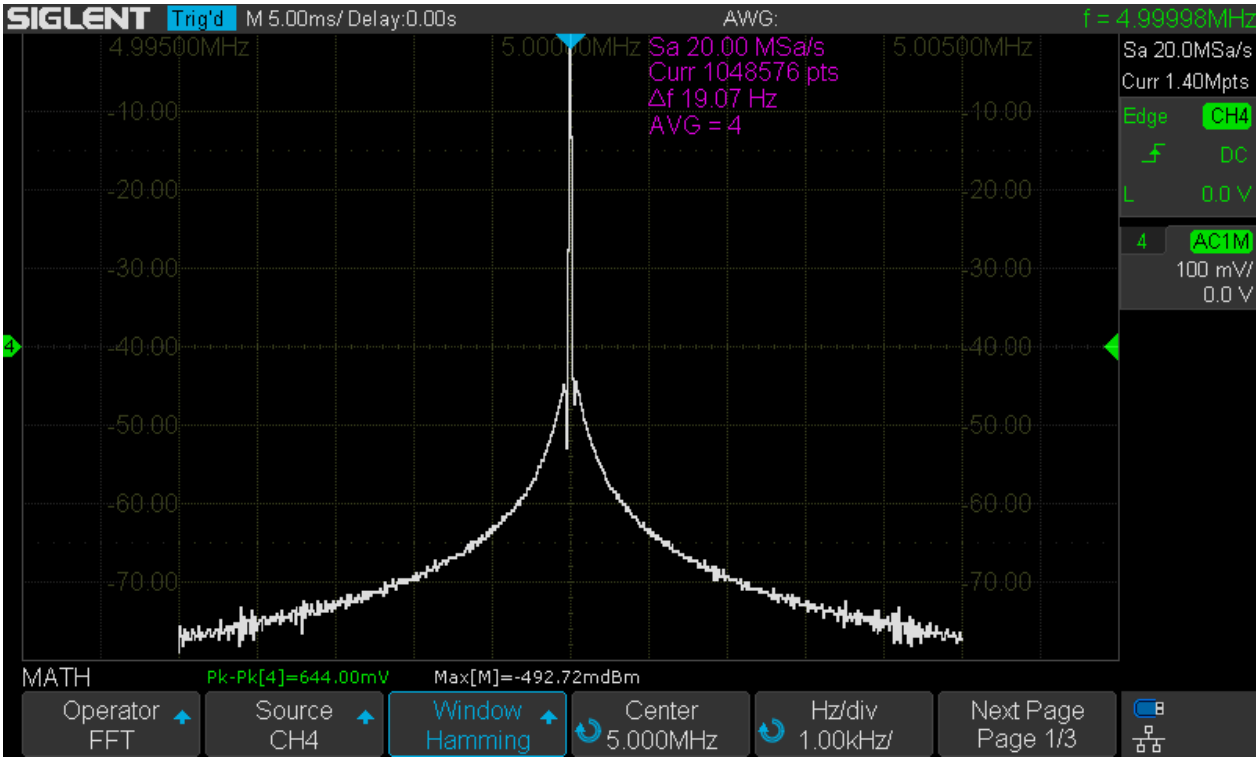
SDS1104X-E_FFT_Mod_Blackman



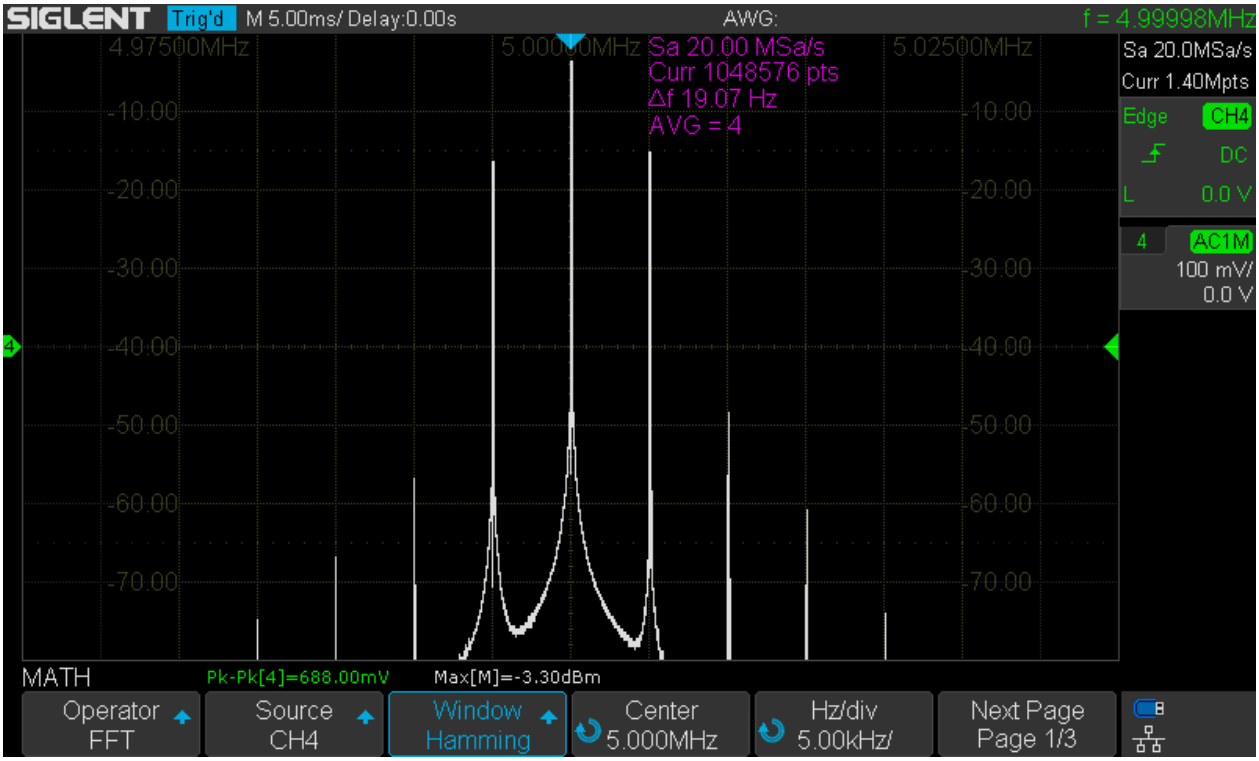
SDS1104X-E_FFT_Hanning



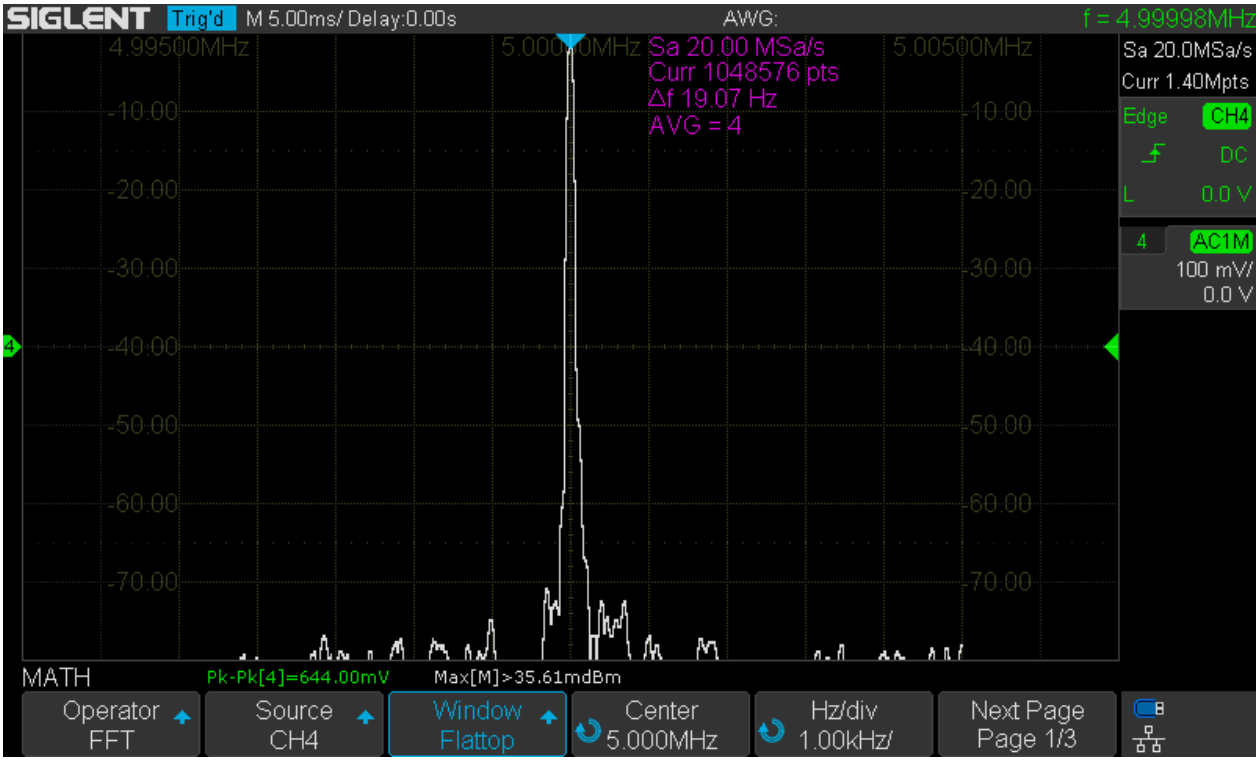
SDS1104X-E_FFT_Mod_Hanning



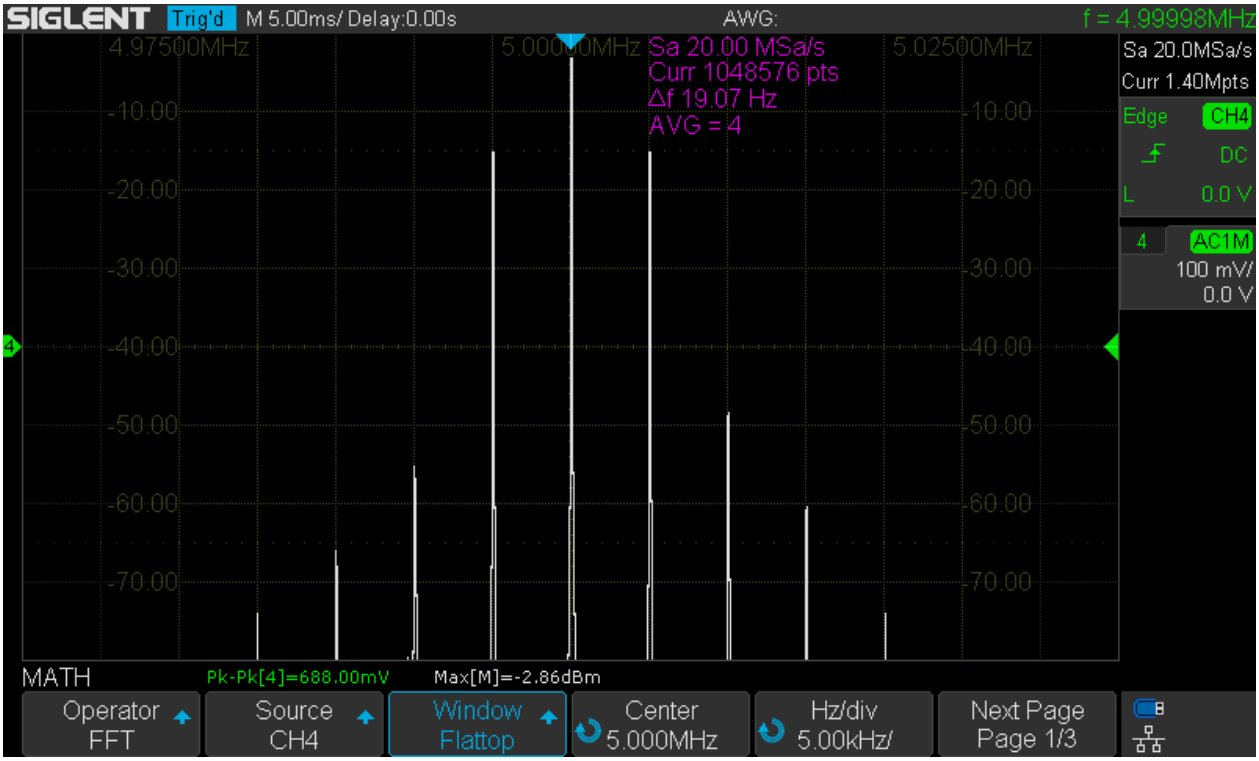
SDS1104X-E_FFT_Hamming



SDS1104X-E_FFT_Mod_Hamming



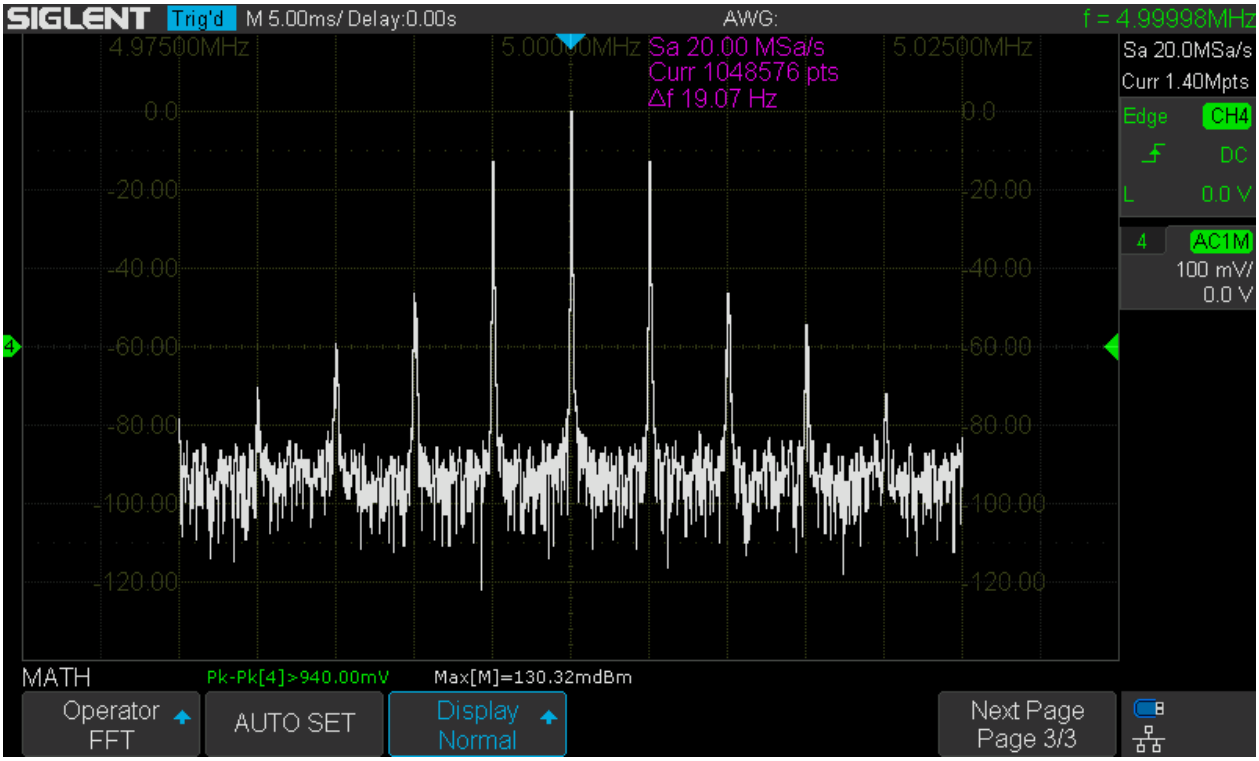
SDS1104X-E_FFT_Flattop



SDS1104X-E_FFT_Mod_Flattop

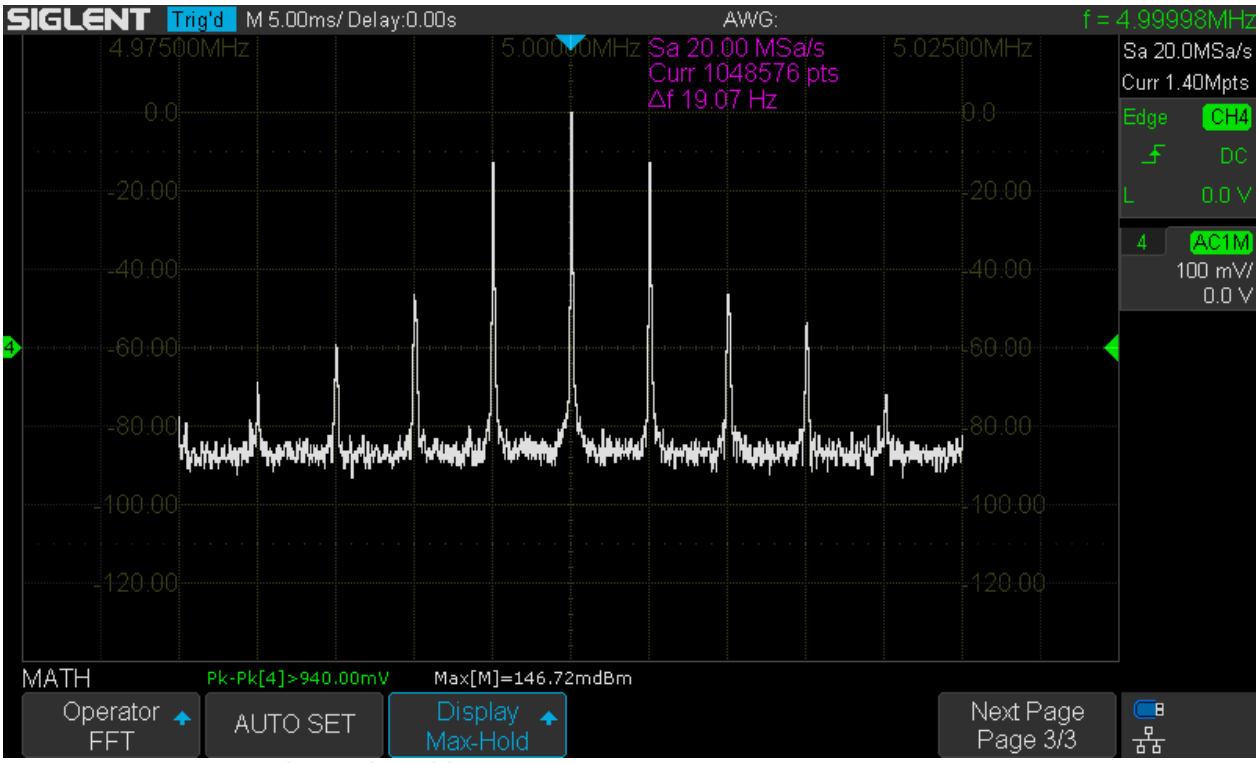
Display Modes

Normal mode will reflect any input signal change instantly and completely on the following FFT trace.



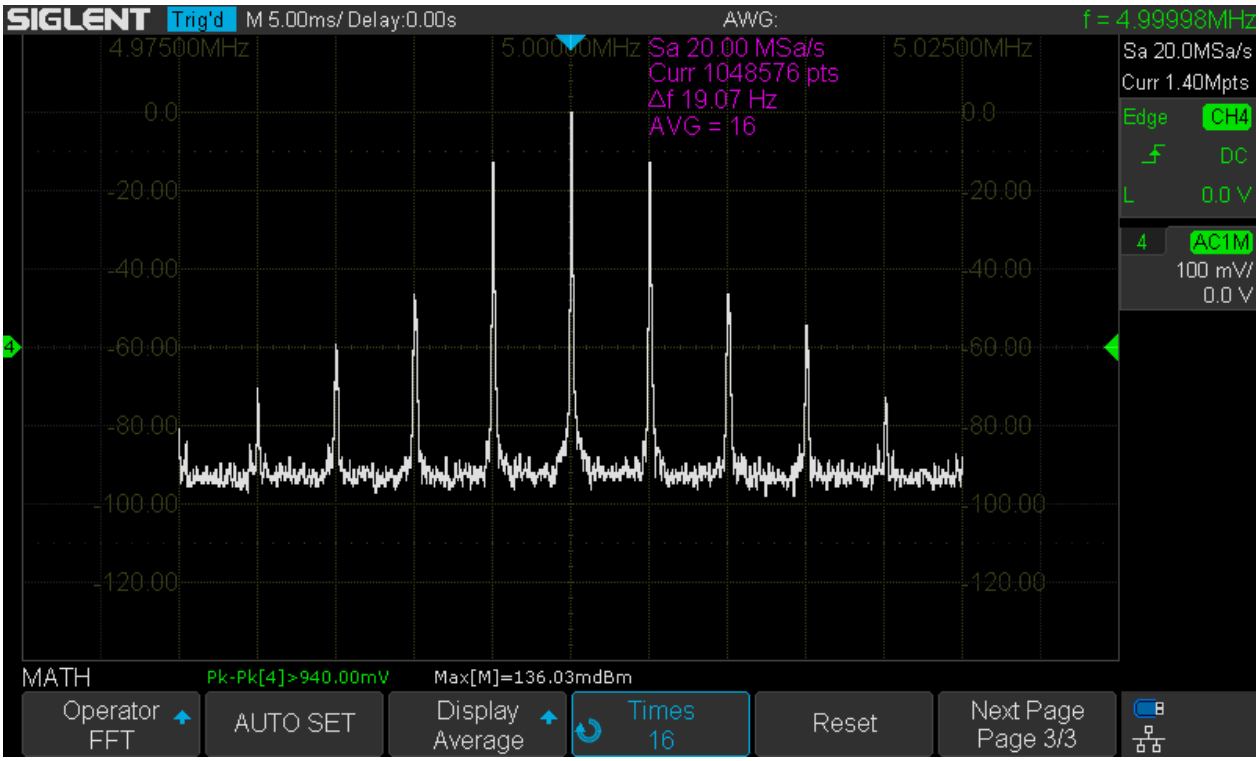
SDS1104X-E_FFT_Mode_Normal

Max-Hold only keeps the long-term maximum values for each frequency bin.



SDS1104X-E_FFT_Mode_Peak_Hold

Average builds the sliding average over the number of records specified by the *Times* soft menu item, which can be set to any value between 4 and 1024.



SDS1104X-E_FFT_Mode_Average16

FFT-Bandwidth and RBW

This is quite different to a real SA. There is no menu for the resolution bandwidth and also no direct setting for the FFT-bandwidth, even though we have a soft menu item for the horizontal scale in Hz/div, which ultimately specifies the visible span. But this is just for zooming into a longer FFT trace and for best speed and lowest RBW we need to make sure that no high zoom factor is required to get the display we want. The following rules apply:

- The analysis bandwidth (FFT-BW) is always half the sample rate.
- The frequency step (Δf) is the sample rate divided by the number of FFT points.
- The resolution bandwidth (RBW) is the frequency step multiplied with a factor specific for the window function in use.
- The number of FFT points depends on the record length, which in turn increases with slower timebase settings, but is ultimately limited by the maximum memory set in the *Acquire* menu.

The following table shows the FFT bandwidth (column *BW*) and frequency step (delta frequency, column *df*) for all possible memory depths as specified in the *Acquire* menu and for timebase settings from 10ns/div up to 1s/div for both channels within a channel group enabled. Keep in mind that the frequency step has to be multiplied by the factor specific for the applied window function in order to know the -3dB resolution bandwidth. That factor can be found in the column *-3dB BW* of the table that summarizes the available window functions in the *Window* section.

SDS1104X-E Dual Channel FFT Bandwidth / FFT Frequency Step								
	7k		70k		700k		7M	
Timebase	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]
10ns	250,00E+6	7,8E+6	250,00E+6	7,8E+6	250,00E+6	7,8E+6	250,00E+6	7,8E+6
20ns	250,00E+6	3,9E+6	250,00E+6	3,9E+6	250,00E+6	3,9E+6	250,00E+6	3,9E+6
50ns	250,00E+6	2,0E+6	250,00E+6	2,0E+6	250,00E+6	2,0E+6	250,00E+6	2,0E+6
100ns	250,00E+6	976,6E+3	250,00E+6	976,6E+3	250,00E+6	976,6E+3	250,00E+6	976,6E+3
200ns	250,00E+6	488,3E+3	250,00E+6	488,3E+3	250,00E+6	488,3E+3	250,00E+6	488,3E+3
500ns	250,00E+6	244,1E+3	250,00E+6	244,1E+3	250,00E+6	244,1E+3	250,00E+6	244,1E+3
1µs	250,00E+6	122,1E+3	250,00E+6	122,1E+3	250,00E+6	122,1E+3	250,00E+6	122,1E+3
2µs	125,00E+6	61,0E+3	250,00E+6	61,0E+3	250,00E+6	61,0E+3	250,00E+6	61,0E+3
5µs	50,00E+6	24,4E+3	250,00E+6	15,3E+3	250,00E+6	15,3E+3	250,00E+6	15,3E+3
10µs	25,00E+6	12,2E+3	250,00E+6	7,6E+3	250,00E+6	7,6E+3	250,00E+6	7,6E+3
20µs	12,50E+6	6,1E+3	125,00E+6	3,8E+3	250,00E+6	3,8E+3	250,00E+6	3,8E+3
50µs	5,00E+6	2,4E+3	50,00E+6	1,5E+3	250,00E+6	1,9E+3	250,00E+6	1,9E+3
100µs	2,50E+6	1,2E+3	25,00E+6	762,9E+0	250,00E+6	953,7E+0	250,00E+6	953,7E+0
200µs	1,25E+6	610,4E+0	12,50E+6	381,5E+0	125,00E+6	476,8E+0	250,00E+6	476,8E+0
500µs	500,00E+3	244,1E+0	5,00E+6	152,6E+0	50,00E+6	190,7E+0	100,00E+6	190,7E+0
1ms	250,00E+3	122,1E+0	2,50E+6	76,3E+0	25,00E+6	95,4E+0	50,00E+6	95,4E+0
2ms	125,00E+3	61,0E+0	1,25E+6	38,1E+0	12,50E+6	47,7E+0	25,00E+6	47,7E+0
5ms	50,00E+3	24,4E+0	500,00E+3	15,3E+0	5,00E+6	19,1E+0	10,00E+6	19,1E+0
10ms	25,00E+3	12,2E+0	250,00E+3	7,6E+0	2,50E+6	9,5E+0	5,00E+6	9,5E+0
20ms	12,50E+3	6,1E+0	125,00E+3	3,8E+0	1,25E+6	4,8E+0	2,50E+6	4,8E+0
50ms	5,00E+3	2,4E+0	50,00E+3	1,5E+0	500,00E+3	1,9E+0	1,00E+6	1,9E+0
100ms	2,50E+3	1,2E+0	25,00E+3	762,9E-3	250,00E+3	953,7E-3	500,00E+3	953,7E-3
200ms	1,25E+3	610,4E-3	12,50E+3	381,5E-3	125,00E+3	476,8E-3	250,00E+3	476,8E-3
500ms	500,00E+0	244,1E-3	5,00E+3	152,6E-3	50,00E+3	190,7E-3	100,00E+3	190,7E-3
1s	250,00E+0	122,1E-3	2,50E+3	76,3E-3	25,00E+3	95,4E-3	50,00E+3	95,4E-3

SDS1104X-E_2CH_FFT_BW_Step

Why did I limit the timebase range to 10ns – 1s? The scope can certainly do a much wider range? The lower limit is 10ns/div just because this already means a FFT length of only 64 points in dual channel mode. This is about the limit for any useful FFT and there is certainly no advantage whatsoever going any lower. It is different for the upper limit and there is basically nothing wrong with timebase settings slower than 1s/div. This would allow us to get extremely narrow frequency steps and resolution bandwidths in turn. But then, even with just 1s/div, one single acquisition already takes a healthy 14 seconds and frequency steps down to some 0.1Hz are obtained. I felt this should cover the vast majority of use cases.

If only one channel in a group is enabled, the max. sample rate as well as memory depth will be doubled. During my experiments, I got the impression that this operating mode generates slightly less spurious signals. The table for single channel (interleaved) mode is shown below.

SDS1104X-E Single Channel FFT Bandwidth / FFT Frequency Step								
	14k		140k		1.4M		14M	
Timebase	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]	BW [Hz]	df [Hz]
10ns	500,00E+6	7,8E+6	500,00E+6	7,8E+6	500,00E+6	7,8E+6	500,00E+6	7,8E+6
20ns	500,00E+6	3,9E+6	500,00E+6	3,9E+6	500,00E+6	3,9E+6	500,00E+6	3,9E+6
50ns	500,00E+6	2,0E+6	500,00E+6	2,0E+6	500,00E+6	2,0E+6	500,00E+6	2,0E+6
100ns	500,00E+6	976,6E+3	500,00E+6	976,6E+3	500,00E+6	976,6E+3	500,00E+6	976,6E+3
200ns	500,00E+6	488,3E+3	500,00E+6	488,3E+3	500,00E+6	488,3E+3	500,00E+6	488,3E+3
500ns	500,00E+6	244,1E+3	500,00E+6	244,1E+3	500,00E+6	244,1E+3	500,00E+6	244,1E+3
1µs	500,00E+6	122,1E+3	500,00E+6	122,1E+3	500,00E+6	122,1E+3	500,00E+6	122,1E+3
2µs	250,00E+6	61,0E+3	500,00E+6	61,0E+3	500,00E+6	61,0E+3	500,00E+6	61,0E+3
5µs	100,00E+6	24,4E+3	500,00E+6	15,3E+3	500,00E+6	15,3E+3	500,00E+6	15,3E+3
10µs	50,00E+6	12,2E+3	500,00E+6	7,6E+3	500,00E+6	7,6E+3	500,00E+6	7,6E+3
20µs	25,00E+6	6,1E+3	250,00E+6	3,8E+3	500,00E+6	3,8E+3	500,00E+6	3,8E+3
50µs	10,00E+6	2,4E+3	100,00E+6	1,5E+3	500,00E+6	1,9E+3	500,00E+6	1,9E+3
100µs	5,00E+6	1,2E+3	50,00E+6	762,9E+0	500,00E+6	953,7E+0	500,00E+6	953,7E+0
200µs	2,50E+6	610,4E+0	25,00E+6	381,5E+0	250,00E+6	476,8E+0	250,00E+6	476,8E+0
500µs	1,00E+6	244,1E+0	10,00E+6	152,6E+0	100,00E+6	190,7E+0	100,00E+6	190,7E+0
1ms	500,00E+3	122,1E+0	5,00E+6	76,3E+0	50,00E+6	95,4E+0	50,00E+6	95,4E+0
2ms	250,00E+3	61,0E+0	2,50E+6	38,1E+0	25,00E+6	47,7E+0	25,00E+6	47,7E+0
5ms	100,00E+3	24,4E+0	1,00E+6	15,3E+0	10,00E+6	19,1E+0	10,00E+6	19,1E+0
10ms	50,00E+3	12,2E+0	500,00E+3	7,6E+0	5,00E+6	9,5E+0	5,00E+6	9,5E+0
20ms	25,00E+3	6,1E+0	250,00E+3	3,8E+0	2,50E+6	4,8E+0	2,50E+6	4,8E+0
50ms	10,00E+3	2,4E+0	100,00E+3	1,5E+0	1,00E+6	1,9E+0	1,00E+6	1,9E+0
100ms	5,00E+3	1,2E+0	50,00E+3	762,9E-3	500,00E+3	953,7E-3	500,00E+3	953,7E-3
200ms	2,50E+3	610,4E-3	25,00E+3	381,5E-3	250,00E+3	476,8E-3	250,00E+3	476,8E-3
500ms	1,00E+3	244,1E-3	10,00E+3	152,6E-3	100,00E+3	190,7E-3	100,00E+3	190,7E-3
1s	500,00E+0	122,1E-3	5,00E+3	76,3E-3	50,00E+3	95,4E-3	50,00E+3	95,4E-3

SDS1104X-E_1CH_FFT_BW_Step

IMPORTANT: Please note that these tables are not guaranteed to be entirely correct as they just contain calculation results and not collected data from the real scope – it would have been rather time consuming to actually try all these combinations on the instrument. The sample rate in the SDS1kX-E might not always be in accordance with my calculations, yet most results should be correct and the tables good enough for determining the appropriate settings for a certain bandwidth / frequency step combination.

For even greater convenience, I've prepared a set of tables that show all appropriate settings for any available combination of analysis bandwidth & frequency step. To use these tables, proceed as follows:

1. Look at all table entries that show the desired analysis bandwidth in column **BW [Hz]**.
2. Pick the entry with the desired frequency step in column **df [Hz]**. Note that there are two such columns, one for dual channel (individual) and another one for single channel (interleaved) configuration.
3. Use the associated entries for timebase **TB [s]** and Max. memory depth **Mem [Pts]** to configure the scope accordingly.
4. The **FFT [Pts]** entry is there just as additional information about the actual FFT length.

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
500,00E+6					7,8E+6	128	10ns	≥14k
500,00E+6					3,9E+6	256	20ns	≥14k
500,00E+6					2,0E+6	512	50ns	≥14k
500,00E+6					976,6E+3	1024	100ns	≥14k
500,00E+6					488,3E+3	2048	200ns	≥14k
500,00E+6					244,1E+3	4096	500ns	≥14k
500,00E+6					122,1E+3	8192	1µs	≥14k
500,00E+6					61,0E+3	16384	2µs	≥140k
500,00E+6					15,3E+3	65536	5µs	≥140k
500,00E+6					7,6E+3	131072	10µs	≥140k
500,00E+6					3,8E+3	262144	20µs	≥1.4M
500,00E+6					1,9E+3	524288	50µs	≥1.4M
500,00E+6					953,7E+0	1048576	100µs	≥1.4M

SDS1104X-E_FFT_Setup_500MHz

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
250,00E+6	7,8E+6	64	10ns	≥7k				
250,00E+6	3,9E+6	128	20ns	≥7k				
250,00E+6	2,0E+6	256	50ns	≥7k				
250,00E+6	976,6E+3	512	100ns	≥7k				
250,00E+6	488,3E+3	1024	200ns	≥7k				
250,00E+6	244,1E+3	2048	500ns	≥7k				
250,00E+6	122,1E+3	4096	1µs	≥7k				
250,00E+6	61,0E+3	8192	2µs	≥70k	61,0E+3	8192	2µs	14k
250,00E+6	15,3E+3	32768	5µs	≥70k				
250,00E+6	7,6E+3	65536	10µs	≥70k				
250,00E+6	3,8E+3	131072	20µs	≥700k	3,8E+3	131072	20µs	140k
250,00E+6	1,9E+3	262144	50µs	≥700k				
250,00E+6	953,7E+0	524288	100µs	≥700k				
250,00E+6	476,8E+0	1048576	200µs	7M	476,8E+0	1048576	200µs	1.4M
125,00E+6	61,0E+3	4096	2µs	7k				
125,00E+6	3,8E+3	65536	20µs	70k				
125,00E+6	476,8E+0	524288	200µs	700k				

SDS1104X-E_FFT_Setup_125-250MHz

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
100,00E+6					24,4E+3	8192	5µs	14k
100,00E+6					1,5E+3	131072	50µs	140k
100,00E+6	190,7E+0	1048576	500µs	7M	190,7E+0	1048576	500µs	1.4M
50,00E+6	24,4E+3	4096	5µs	7k	12,2E+3	8192	10µs	14k
50,00E+6	1,5E+3	65536	50µs	70k	762,9E+0	131072	100µs	140k
50,00E+6	190,7E+0	524288	500µs	700k	95,4E+0	1048576	1ms	1.4M
50,00E+6	95,4E+0	1048576	1ms	7M				
25,00E+6	12,2E+3	4096	10µs	7k	6,1E+3	8192	20µs	14k
25,00E+6	762,9E+0	65536	100µs	70k	381,5E+0	131072	200µs	140k
25,00E+6	95,4E+0	524288	1ms	700k	47,7E+0	1048576	2ms	1.4M
25,00E+6	47,7E+0	1048576	2ms	7M				
12,50E+6	6,1E+3	4096	20µs	7k				
12,50E+6	381,5E+0	65536	200µs	70k				
12,50E+6	47,7E+0	524288	2ms	700k				

SDS1104X-E_FFT_Setup_12.5-100MHz

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
10,00E+6					2,4E+3	8192	50µs	14k
10,00E+6					152,6E+0	131072	500µs	140k
10,00E+6	19,1E+0	1048576	5ms	7M	19,1E+0	1048576	5ms	1.4M
5,00E+6	2,4E+3	4096	50µs	7k	1,2E+3	8192	100µs	14k
5,00E+6	152,6E+0	65536	500µs	70k	76,3E+0	131072	1ms	140k
5,00E+6	19,1E+0	524288	5ms	700k	9,5E+0	1048576	10ms	1.4M
5,00E+6	9,5E+0	1048576	10ms	7M				
2,50E+6	1,2E+3	4096	100µs	7k	610,4E+0	8192	200µs	14k
2,50E+6	76,3E+0	65536	1ms	70k	38,1E+0	131072	2ms	140k
2,50E+6	9,5E+0	524288	10ms	700k	4,8E+0	1048576	20ms	1.4M
2,50E+6	4,8E+0	1048576	20ms	7M				
1,25E+6	610,4E+0	4096	200µs	7k				
1,25E+6	38,1E+0	65536	2ms	70k				
1,25E+6	4,8E+0	524288	20ms	700k				

SDS1104X-E_FFT_Setup_1.25-10MHz

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
1,00E+6					244,1E+0	8192	500µs	14k
1,00E+6					15,3E+0	131072	5ms	140k
1,00E+6	1,9E+0	1048576	50ms	7M	1,9E+0	1048576	50ms	1.4M
500,00E+3	244,1E+0	4096	500µs	7k	122,1E+0	8192	1ms	14k
500,00E+3	15,3E+0	65536	5ms	70k	7,6E+0	131072	10ms	140k
500,00E+3	1,9E+0	524288	50ms	700k	953,7E-3	1048576	100ms	1.4M
500,00E+3	953,7E-3	1048576	100ms	7M				
250,00E+3	122,1E+0	4096	1ms	7k	61,0E+0	8192	2ms	14k
250,00E+3	7,6E+0	65536	10ms	70k	3,8E+0	131072	20ms	140k
250,00E+3	953,7E-3	524288	100ms	700k	476,8E-3	1048576	200ms	1.4M
250,00E+3	476,8E-3	1048576	200ms	7M				
125,00E+3	61,0E+0	4096	2ms	7k				
125,00E+3	3,8E+0	65536	20ms	70k				
125,00E+3	476,8E-3	524288	200ms	700k				

SDS1104X-E_FFT_Setup_125kHz-1MHz

	Dual Ch. (max. 500MSa/s)				Single Ch. (max. 1GSa/s)			
BW [Hz]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]	df [Hz]	FFT [Pts]	TB [s]	Mem [Pts]
100,00E+3					24,4E+0	8192	5ms	14k
100,00E+3					1,5E+0	131072	50ms	140k
100,00E+3	190,7E-3	1048576	500ms	7M	190,7E-3	1048576	500ms	1.4M
50,00E+3	24,4E+0	4096	5ms	7k	12,2E+0	8192	10ms	14k
50,00E+3	1,5E+0	65536	50ms	70k	762,9E-3	131072	100ms	140k
50,00E+3	190,7E-3	524288	500ms	700k	95,4E-3	1048576	1s	1.4M
50,00E+3	95,4E-3	1048576	1s	7M				
25,00E+3	12,2E+0	4096	10ms	7k	6,1E+0	8192	20ms	14k
25,00E+3	762,9E-3	65536	100ms	70k	381,5E-3	131072	200ms	140k
25,00E+3	95,4E-3	524288	1s	700k				
12,50E+3	6,1E+0	4096	20ms	7k				
12,50E+3	381,5E-3	65536	200ms	70k				
10,00E+3					2,4E+0	8192	50ms	14k
10,00E+3					152,6E-3	131072	500ms	140k

SDS1104X-E_FFT_Setup_10-100kHz

As an example, let's assume we want to perform an analysis in the audio range up to 20kHz, which would then be our desired analysis bandwidth. In the last table SDS1104X-E_FFT_Setup_10-100kHz we cannot find 20kHz, so we pick the next higher value, that is 25kHz. For this bandwidth, we get a total of five choices: three for dual channel configuration with frequency steps of 12.2Hz, 0.763Hz and 0.095Hz as well as another two for single channel configuration with frequency steps of 6.1Hz and 0.382Hz. If we use the Blackman window (recommended), we have to multiply the frequency step by 1.7 in order to get the effective -3dB resolution bandwidth. With single channel configuration, 200ms/div and 140k max. memory depth we get a frequency step of 0.3815Hz and the resolution bandwidth will be about 0.65Hz. Acquisition time will be 200ms x 14 = 2.8s, hence FFT update rate will be slow no matter how powerful the signal processing might be. Dealing with low frequencies and narrow frequency steps just takes its toll, there's no way to get around this.

Setting up an FFT Measurement

Even from the best FFT implementation, we can only expect good results as long as the scope has been set up properly for that specific task. How many so called "reviews" have we seen where FFT has been

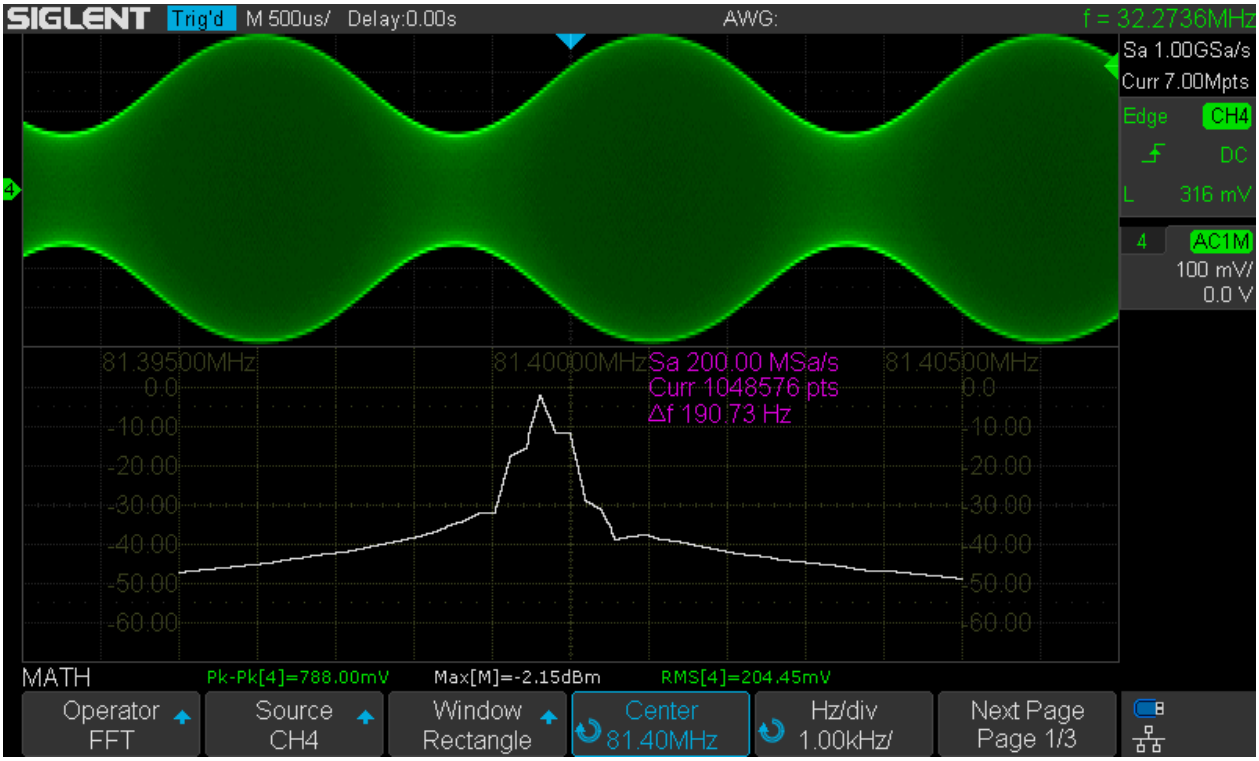
engaged and then some scope settings have been randomly altered just to get some halfway plausible but actually rather meaningless FFT graph, which was then either praised or criticized? Of course we can get away with some quick & dirty setup if we just want to get a quick overview, but for optimal speed, frequency resolution and dynamic range, we need to put a little more effort into a proper setup, which has quite different requirements compared to the usual Y-t view. Below there is a complete checklist how to properly set up the DSO for analysis in the frequency domain:

1. Set acquisition mode to normal. Use average only for a good reason and stay away from Eres. Avoid Peak Detect under all circumstances and without any exception!
2. Use edge trigger in auto mode to make sure signal acquisition doesn't stop even when the signal amplitude drops below the trigger sensitivity. FFT doesn't absolutely require a stable trigger by the way, but it certainly doesn't hurt, especially for narrowband analysis.
3. Determine the lower bandwidth limit for the FFT analysis. If it is >3Hz, use AC-coupling for the input channel to ensure maximum dynamic range even with large DC offsets and/or high input sensitivities.
4. Determine the upper bandwidth limit for the FFT analysis. In order to avoid aliasing artifacts, this should not only cover the desired analysis bandwidth, but include the highest expected input frequency. In general, it's best to start with a higher upper bandwidth limit and reduce it only after it has been confirmed that there is no significant signal content above the desired final limit.
5. Choose the frequency step size, which would be about half the required resolution bandwidth.
6. Find an appropriate set of horizontal timebase and max. memory depth settings by means of the tables provided in the *FFT-Bandwidth and RBW* section earlier in this document and setup the scope accordingly. Be aware that the desired resolution bandwidth might not be achievable due to the limited choice of sample rates and memory depths and/or the maximum FFT length of 1Mpts.
7. Engage FFT mode, select the correct source channel and start with Split Screen mode.
8. Set the vertical gain so that the peak amplitude of the input signal is between ±2 to ±4 divisions.
9. Set the FFT center frequency to the arithmetic mean between lower and upper bandwidth limit.
10. Set the FFT frequency scale so that the desired analysis bandwidth is displayed on the screen.
11. Set the desired level units and make sure the external load impedance matches reality whenever working with power levels, i.e. dBm.
12. Set the reference level and vertical scale so that the FFT amplitude range of interest makes best use of the available space.
13. Setup automatic peak-peak (and maybe RMS) measurement for the input channel, as well as Max for the math channel. During frequency domain analysis, especially in Exclusive mode, keep an eye on the Vpp measurement for the input channel to make sure no overload occurs.
14. Select an appropriate window function; refer to the *Window* section earlier in this document.

Hint: stay in Split Screen mode until the amplitude setup is finished and the levels are reasonably stable, then switch to Exclusive mode. By keeping an eye on the peak to peak measurement of the input signal, you can still detect an overload condition instantly; the scope indicates that by displaying **>** instead of **=** in front of the measurement value, e.g. Pk-Pk[4]**>**796.00mV instead of Pk-Pk[4]**=**640.00mV.

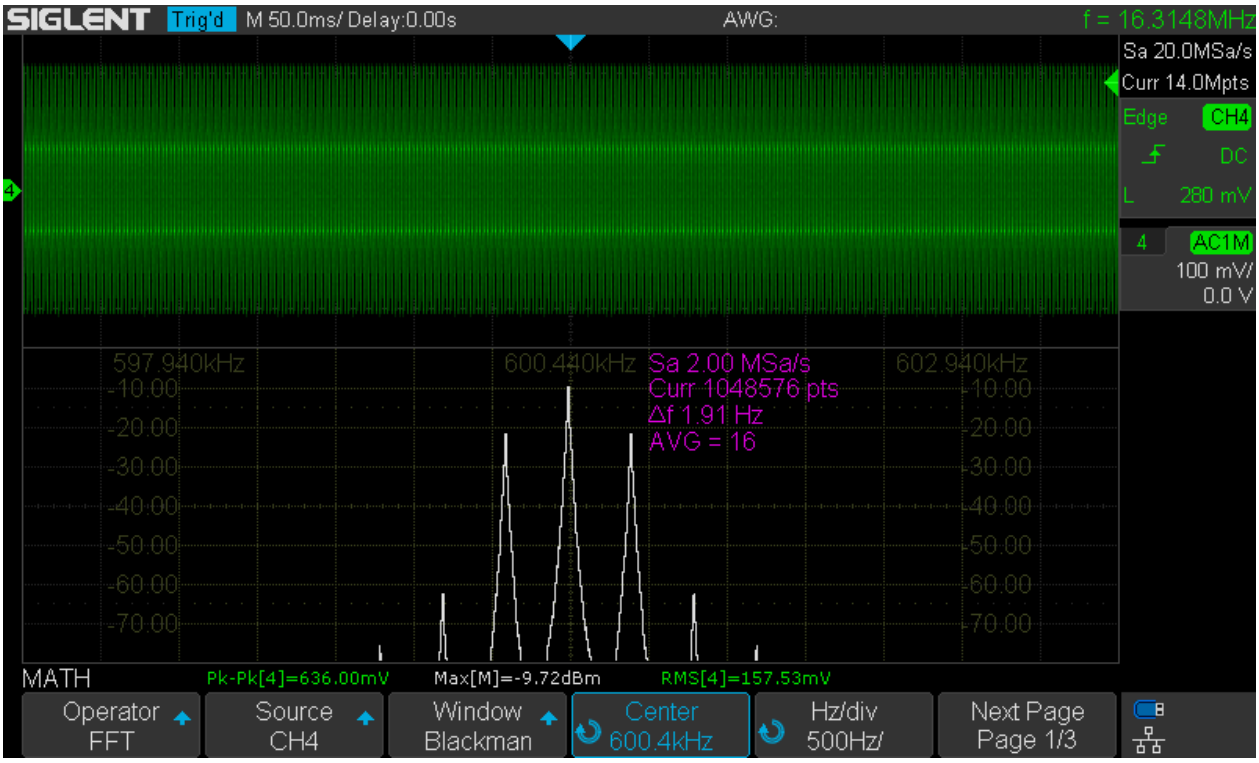
Looking at IF Signals

Sometimes we want more resolution bandwidth than a 1Mpts FFT can provide. Assume a professional HF communications receiver that has an IF of 81.4MHz, where we want to analyze a 400Hz AM signal within the IF signal chain. We could try to do that by setting up a 100MHz analysis bandwidth and using maximum FFT length. According to table SDS1104X-E_FFT_Setup_12.5-100MHz this can be done by setting the timebase to 500µs/div and max. memory depth to (at least) 1.4Mpts in single channel (interleaved) mode. This results in a frequency step of 190.7Hz, which is just too wide for properly analyzing sidebands that are only 400Hz away from the carrier.



SDS1104X-E_FFT_AM_400Hz_81.4MHz

Even with the rectangle window (which gives the most narrow resolution bandwidth), we don't get enough frequency resolution to distinguish the sidebands from the carrier. We can try something different though:



SDS1104X-E_FFT_AM_400Hz_81.4MHz_US

Since an IF signal is bandwidth limited (hence no risk of aliasing), we can downconvert it just by undersampling. Any ADC acts as a mixer, thus producing a spectrum of ±*n* x *fi* ± *m* x *fs*, where *fi* is the

input frequency and *fs* is the sample clock, whereas *n* and *m* are just integers running from 0 to (theoretically) infinity. During normal operation, we don't want to see any mixer products, which is perfectly possible as long as the input signal and all its harmonics don't exceed *fs*/2 and the output of the ADC has a brick-wall filter (then in the digital domain of course) that removes everything above *fs*/2. But in some circumstances, we can make use of a certain high-order mixer product, just as in this example, where the effective FFT sample rate is only 2MSa/s, which is obviously much too low for an 81.4MHz signal.

According to the formula given above, we are aiming at the mixer product for -1 x *fi* + 41 x *fs*, which is

41 x 2MHz – 1 x 81.4MHz = 82MHz – 81.4MHz = 600kHz;

Now if we set the center frequency to 600kHz, we get the carrier at 81.4MHz and can also clearly see the sidebands 400Hz apart from it. 16x Averaging has been used in order to get a clean and stable display.

There are several drawbacks though:

- With this mixing scheme, we get the result in reverse frequency position, i.e. the upper sideband appears below the carrier and vice versa.
- Mixing with the 41st harmonic of the sample clock introduces also 41 times more phase noise and jitter and it clearly shows in the FFT plot. Just look at the filter shape of the individual spectral lines.
- Amplitude accuracy is pretty much gone, as a harmonic mixing process cannot be as efficient as the fundamental one, hence we get about 4.6dB attenuation. Note that I had to reduce the input level by 3dB compared to the first screenshot in order to avoid input overloading, so 3dB of the total difference are caused by that and not by the measurement error due to harmonic mixing.

So this is not an ideal application, just some emergency measure to get a result – following the motto "a compromised measurement is better than nothing at all …"

Performance Test

Up to now, we have just explored the possibilities to properly set up an optimal FFT analysis for various tasks, but we have not checked the actual performance of the FFT implementation in the SDS1104X-E. We can often hear opinions suggesting that the FFT length is the most important factor and if this were true, the Siglent SDS1kX-E series with 1Mpts max. FFT length would be hard to beat. Actually, FFT length only determines the frequency resolution and noise. Yet for the majority of tasks, like characterizing an unknown signal and interference hunting, we don't really need an extremely high frequency resolution. Likewise, for the vast majority of measurements with a DSO, noise isn't the limiting factor either. Consequently, a long FFT is nice to have, but most of the time a high spurious-free dynamic range and low harmonic and intermodulation distortions would be much more important. For this, an 8-bit sampling system sets tight limits with about 49dB dynamic range, which cannot be changed by increasing the FFT length or any other averaging techniques. We can indeed get more than that in certain scenarios and an e.g. 70dB dynamic could easily be demonstrated with a special setup, but this is not universally true and ironically fails just for the majority of real world applications. So while we should not expect any wonders, the FFT in this scope is still a very powerful implementation and very useful tool within the constraints of the 8-bit acquisition system.

Amplitude Accuracy

Analyzing a signal in the frequency domain is about exploring its spectral components with their amplitudes and maybe also relative phases. Well, we obviously don't get any phase information and this is quite common in DSO-FFT implementations as well as scalar spectrum analyzers. But we do expect decent amplitude accuracy within the dynamic range of the 8-bit system, at least when using the Flattop window. For this test, a 50MHz sine from a DDS function generator is fed into channel 4 of the scope via a precision step attenuator and this combination is capable of providing a fairly accurate amplitude range of more than 120dB. Let's start with 0dBm using the low noise gain of 100mV/div and an analysis bandwidth of 100MHz so we can see all the spurious signals generated by the scope itself.

SDS1104X-E_FFT_100mV_Amp_0dBm

The signal strongest spur at 75MHz is -62dBm, hinting on a spurious free dynamic range of ~62dB.

SDS1104X-E_FFT_100mV_Amp_-10dBm

With a signal level of -10dBm, the spurious signals at 25 and 75MHz have dropped by nearly 10dB as well, which indicates they are directly input signal related. In contrast, the spur at 62MHz is even stronger now, so it's only loosely related to the input signal level. The next screenshot shows a -20dBm signal:

SDS1104X-E_FFT_100mV_Amp_-20dBm

The spurs at 25 and 75MHz have dropped even further, while the one at 62MHz remains fairly constant.

SDS1104X-E_FFT_100mV_Amp_-30dBm

At -30dBm the spur at 75MHz has completely vanished, but the other two remain at -75dBm.

SDS1104X-E_FFT_100mV_Amp_-40dBm

At -40dBm, there are only a couple ADC LSBs used anymore and linearity gets worse. A new maximum is reached at 62MHz and several other spurs have emerged.

SDS1104X-E_FFT_100mV_Amp_-50dBm

Up to this point, the Max measurement on the FFT has been pretty accurate, but now it catches on the DC component and has therefore become meaningless. Other than that, the signal level is displayed as some

-48dBm, so we're starting to lose accuracy as we're finally approaching the limit of the 8-bit dynamic range.

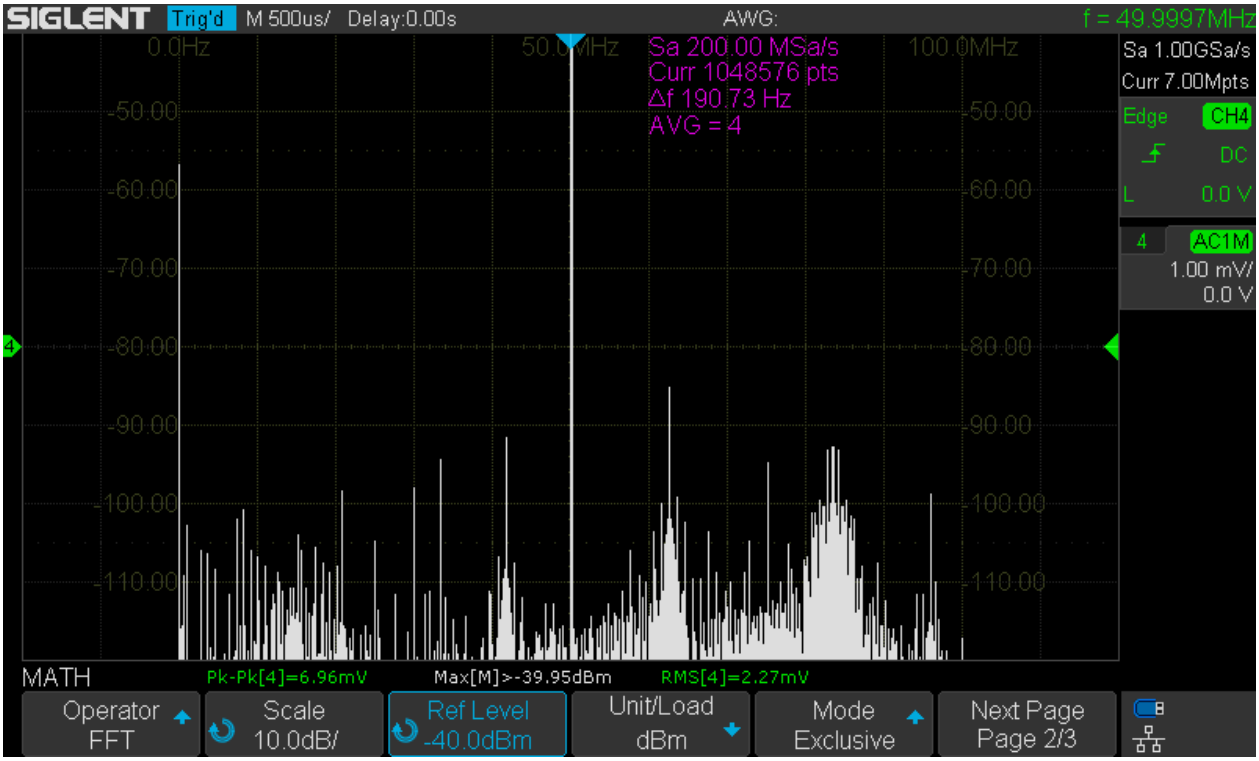
SDS1104X-E_FFT_100mV_Amp_-60dBm

At -60dBm, the spectral line for the signal drops dramatically, showing some -73dBm. Even though there is no visible noise and only one substantial spur, the range below -50dBm is simply unusable for single signal measurements due to the constraints of the 8-bit sampling system. So don't make the mistake to think you have 80dB+ dynamic range, just because the noise floor appears so low and only few spurious signals are visible – which by the way comes as no surprise in this scenario, because all harmonics related to the 50MHz signal are outside the analysis bandwidth.

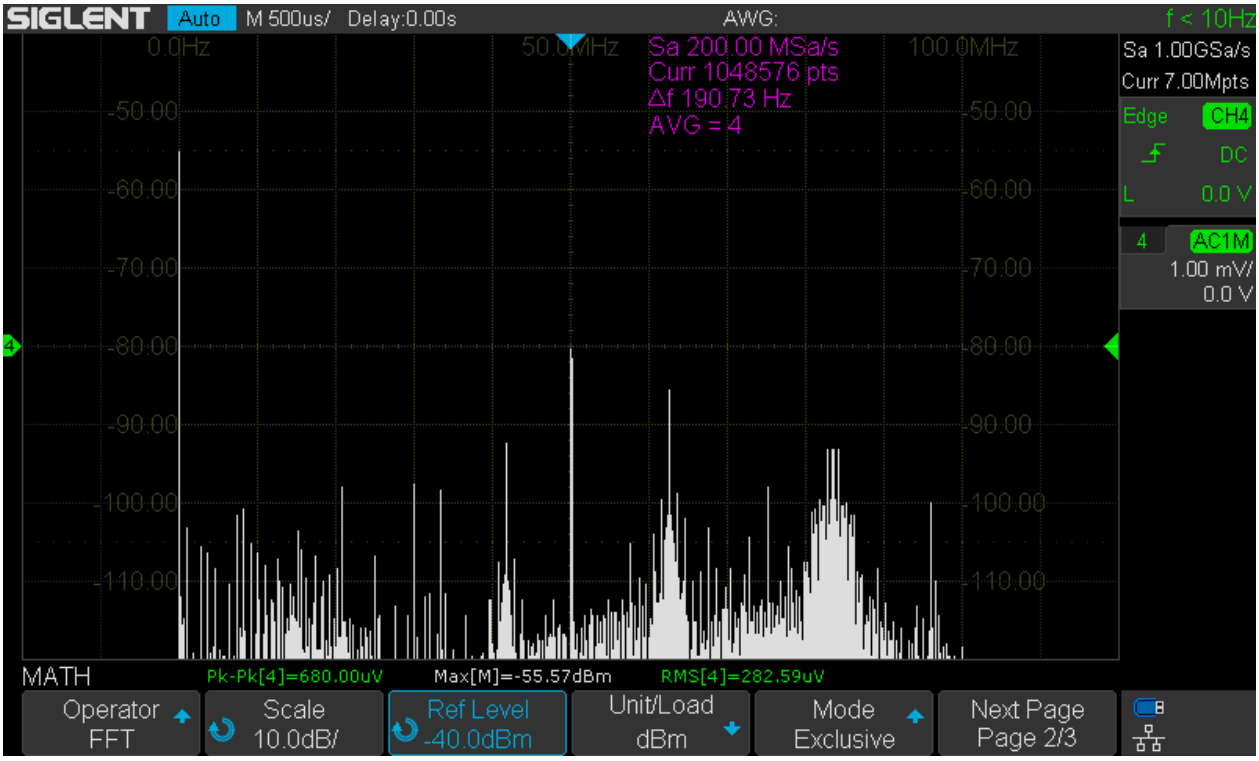
One might wonder how low we can go with the amplitude measurements. Up to now, we could demonstrate that it works fairly well down to -50dBm, which is just 707µVrms or 2mVpp. This is almost as sensitive as most AF or RF Millivolt meters with the added benefit of the selective measurement that ignores harmonics as well as most of the noise. But we can do better…

Up to this point we have been using 100mV/div gain throughout this test and could demonstrate accurate measurements down below one mV. By increasing the channel gain to 1mV/div there should be a boost in sensitivity by 40dB and we can hope to measure signal levels down to -90dBm, equivalent to 7µVrms!

The screenshot below shows the 50MHz signal at -40dBm. Due to the high channel gain, the noise level is down below -110dBm, but there are lots of spurious signals, not input signal related, just electrical noise and interference from inside the scope itself. The level of this is pretty low and there is certainly no reason to complain about spurs below -80dBm (<22µVrms), especially not for a cheap entry level DSO like the SDS1104X-E. Yet this ultimately limits the ability to measure unknown weak signals to about 10µVrms whereas for known signals like in this test we would be able to go even lower and the proposed -90dBm should be realistic.

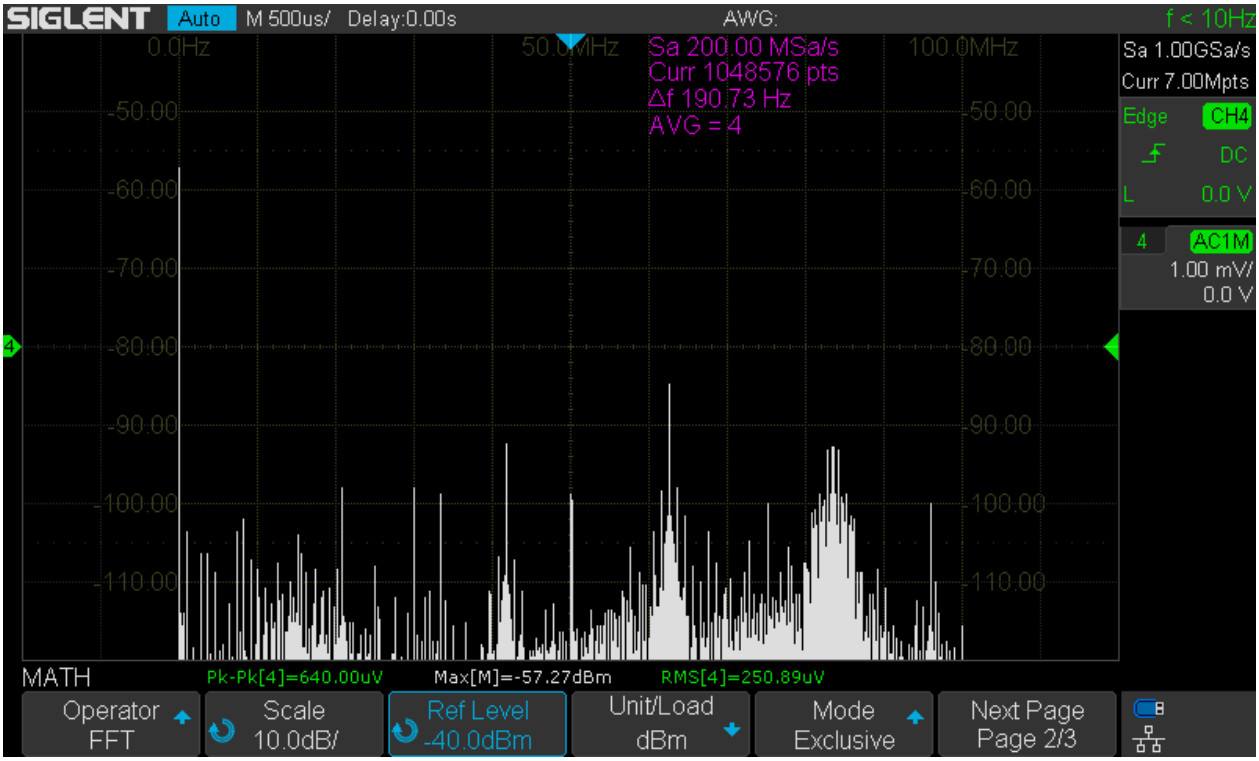


SDS1104X-E_FFT_1mV_Amp_-40dBm

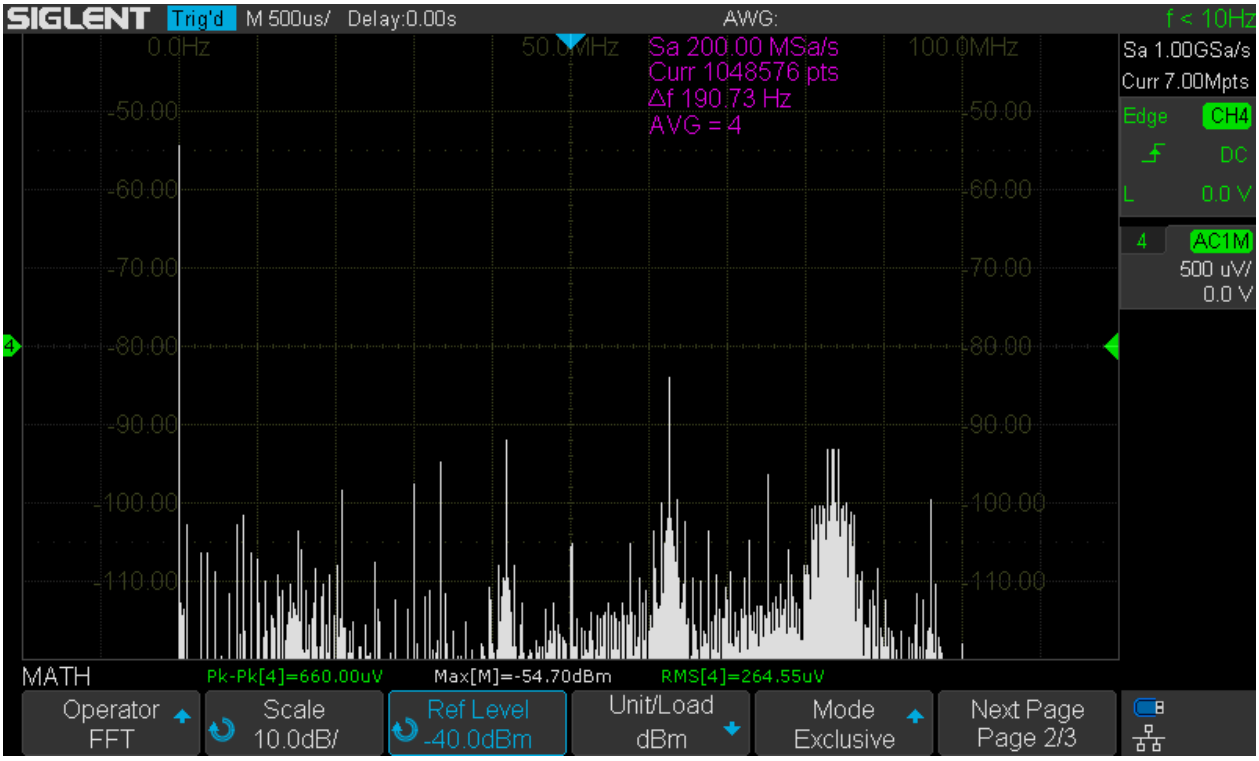


SDS1104X-E_FFT_1mV_Amp_-80dBm

At -80dBm as shown above, the automatic Max measurement has become useless again because of the DC component, but the spectral line for the signal still shows a pretty good level accuracy. As expected, all the spurs remain unchanged as they are all generated internally by the scope itself.



SDS1104X-E_FFT_1mV_Amp_-100dBm



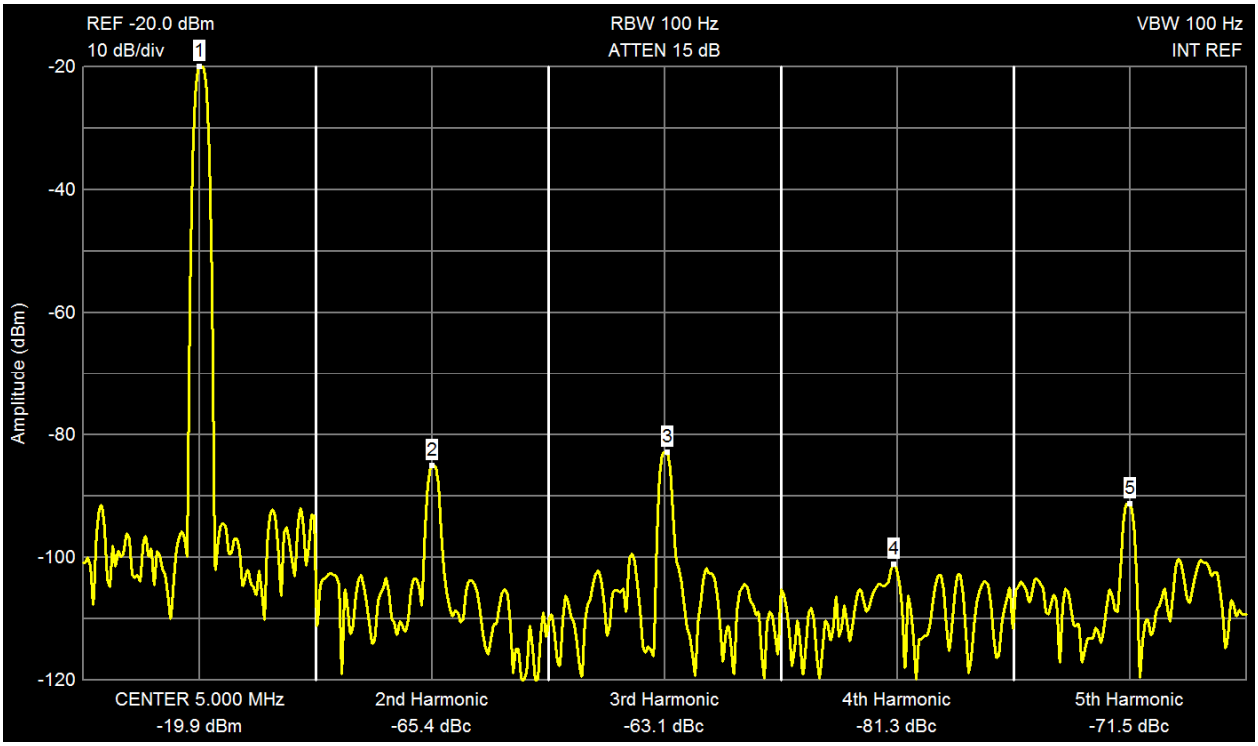
SDS1104X-E_FFT_500uV_Amp_-110dBm

As can be seen in the screenshots above, we can even measure the level of a known signal at -100dBm, that is 2,24µVrms or 6,32µVpp. This is the absolute limit though and the test fails at -110dBm, where the magnitude of the displayed spectral line is off by several dB even with the maximum channel gain of 500µV/div.

Harmonic Distortion

Another common task for frequency domain analysis is measuring the harmonics of a signal. Quite obviously, the harmonic distortion generated within the analyzer is a limiting factor for such measurements, hence it should be fairly low. Once again we'll have to face the limitations of an 8-bit system and cannot expect any better than about -46dB, which would be equivalent to 0.5% THD.

I've checked the harmonic distortion of the SDS1104X-E for several frequencies from 1MHz to 30MHz. Of course this test requires a low distortion sine wave, so the quality of the signal source has to be verified beforehand. This is done by means of a "proper" SA utilizing its "Harmonic Viewer" application. Below is a screenshot exemplarily showing the result for the 5MHz test signal.



THD_Ref_5MHz

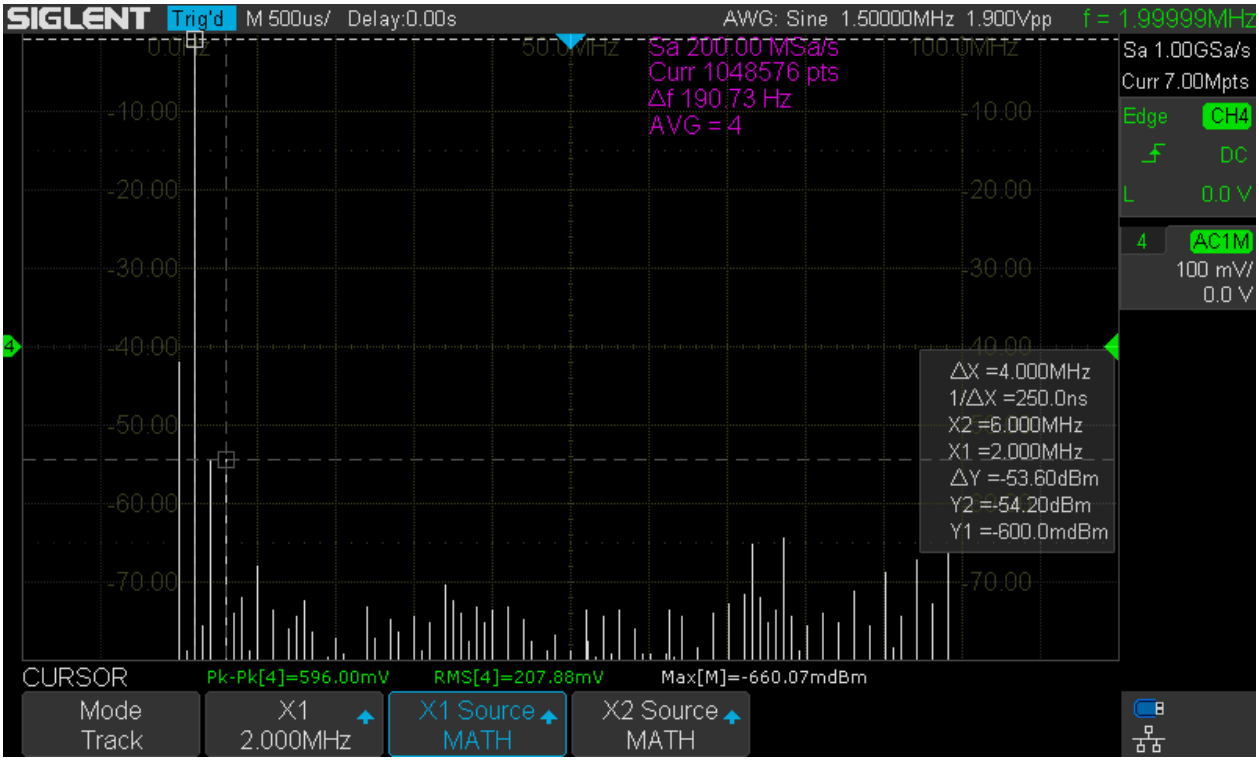
All the harmonics stay well below -60dBc at that frequency, which should be good enough for characterizing an 8-bit system.

Now let's see how the SDS1104X-E performs. The following screenshots demonstrate the harmonic distortion at 2, 5, 10 and 30MHz. Higher frequencies have not been tested, because the important 3rd harmonic would be outside the bandwidth of this scope. 1MHz has been tested as well, but the result was practically identical to the 2MHz test, so it has been omitted here.

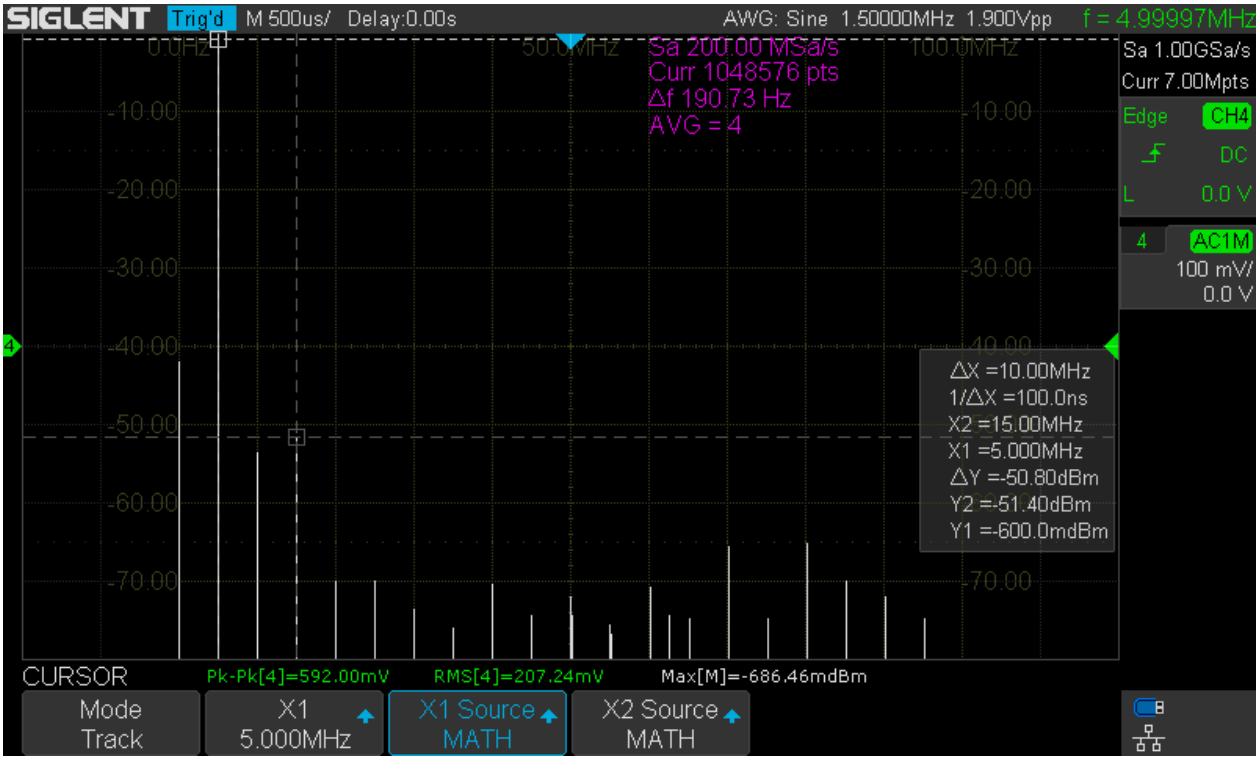
No automatic measurements for the harmonics are available, but tracking cursors on the math trace are a near perfect tool to both measure and highlight (in the screenshots) the strongest harmonic.

As can be seen, the strongest harmonic is the 3rd and starting at about -53dBc at 2MHz it slowly degrades to -47dBc at 30MHz. All in all this is a fair bit better than expected and hints on a very good ADC linearity, i.e. its INL has to be better than 1LSB, even at high frequencies. This test would also reveal any problems within the frontend, but there is nothing to complain either. Please note the relatively low level of higher order harmonics and spurs!

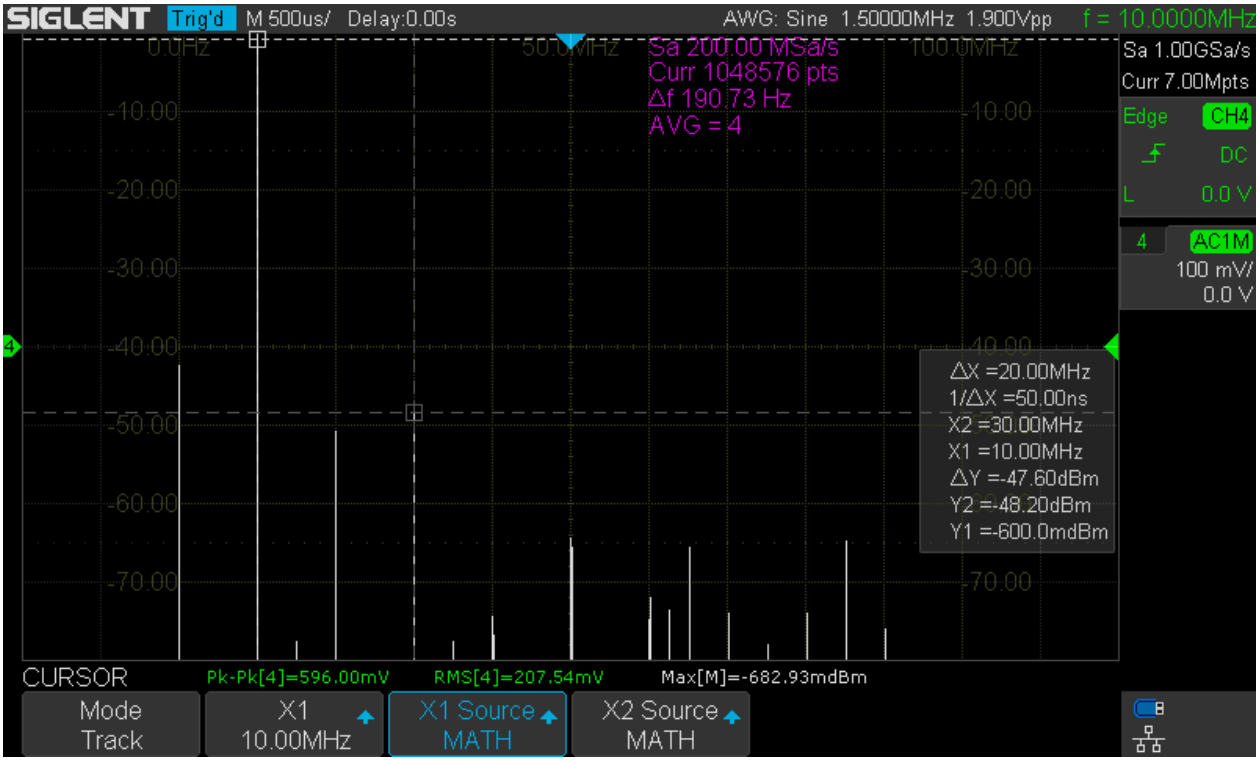
The conclusion can only be, while this scope certainly isn't up to the task of characterizing low distortion sine waves, high fidelity audio gear or high performance RF circuits, it still performs pretty good for the majority of undemanding tasks with less strict requirements.



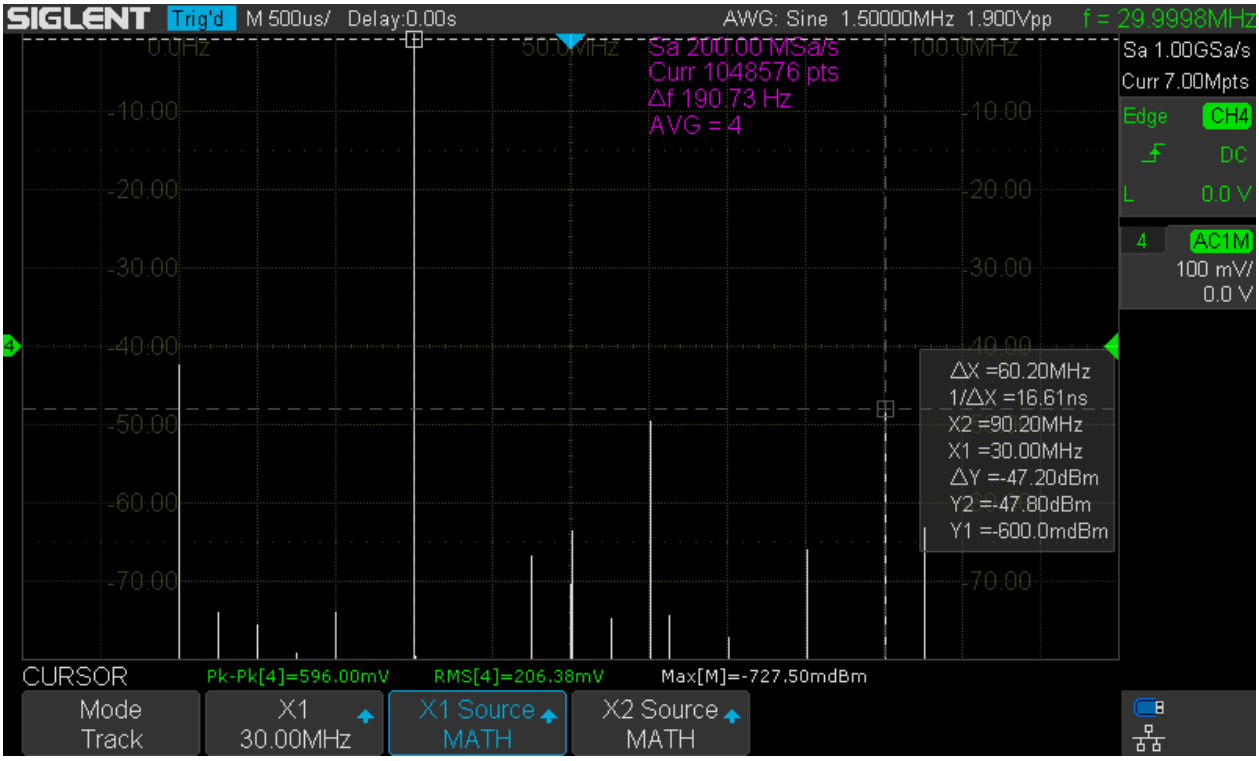
SDS1104X-E_FFT_THD_2MHz



SDS1104X-E_FFT_THD_5MHz



SDS1104X-E_FFT_THD_10MHz



SDS1104X-E_FFT_THD_30MHz

Two Tone Test

The key specification for every spectrum analyzer in terms of practical use for narrowband measurements is its 3rd order dynamic range, usually expressed as IIP3 (3rd order Input Intermodulation Intercept point).

In short it's the ability to measure weak signals in presence of strong ones without generating unwanted 3rd order mixing products, which would appear near the original signals, thus creating a chaotic mix of wanted and unwanted (or better: relevant and irrelevant) signals – with the risk of unwanted signals even totally obscuring the real ones.

This test uses two input signals at 9 and 11MHz, both at a 0dBm level initially. The 9MHz signal is then attenuated step by step in order to check the amplitude accuracy of the measurements. At the same time, the 3rd order mixing products at 7 and 13MHz are measured, so the resulting IMD (intermodulation distortion) can be estimated.

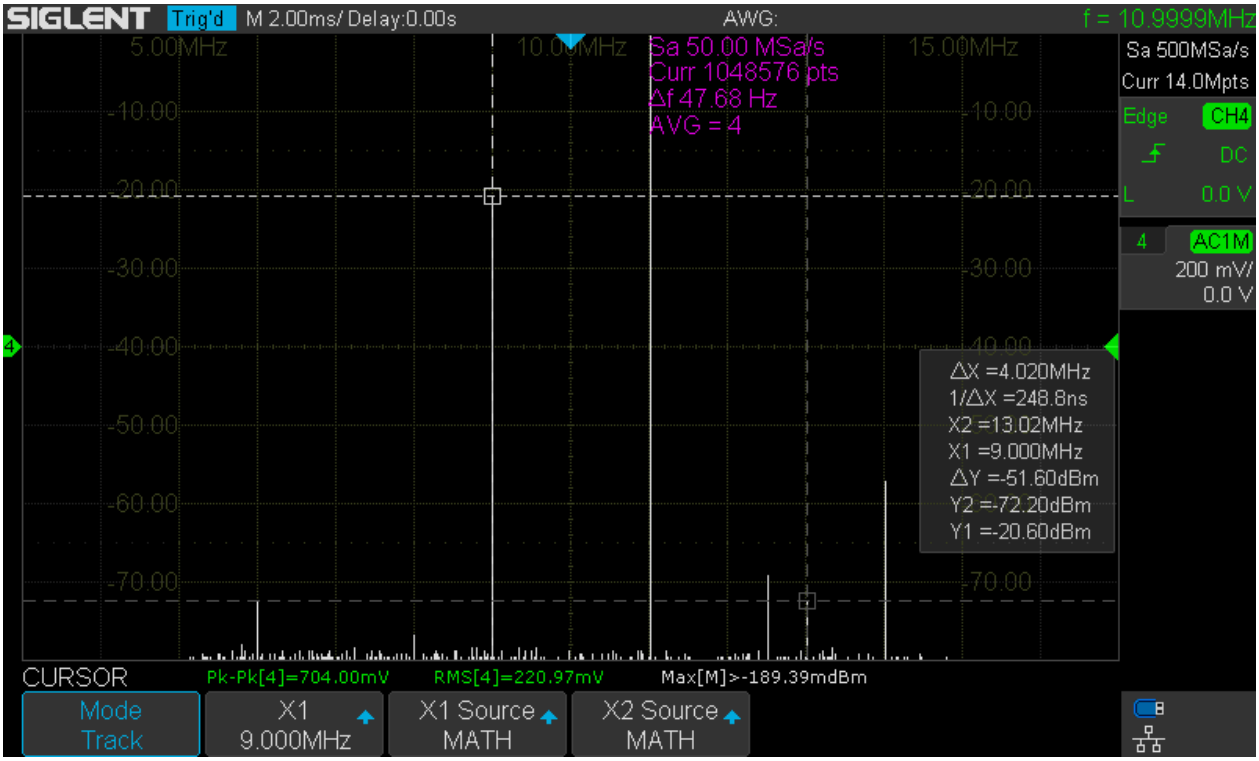
The screenshots below show the results for 0, -20, -30, -60 and -70dBm.

The 0dBm test would be the classical setup for determining the 3rd order intercept point. The unwanted product at 7MHz is weaker at -72dB, but the opposite intermodulation product at 13MHz isn't strong either at only -66dB. This means a 3rd order intermodulation distortion of -66dB at 0dBm input level and the input intercept point would thus be a healthy +33dBm for a channel gain of 200mV/div.

At -20dBm, the intermodulation product at 7MHz vanishes (goes below -80dBm), the same is true for the 13MHz product at -30dBm. That's fairly decent and when looking at the other spurious signals, we can conclude that despite the 8-bit system, we can get at least 60dB dynamic range for narrowband applications like this.

We can also measure levels below -50dBc quite accurately, simply because the 2nd signal stays fixed at 0dBm and serves as a dither for the ADC. As can be seen, measuring -70dBc isn't a problem at all. Just for fun, I've added a measurement at -76dBm level, which still works surprisingly well.

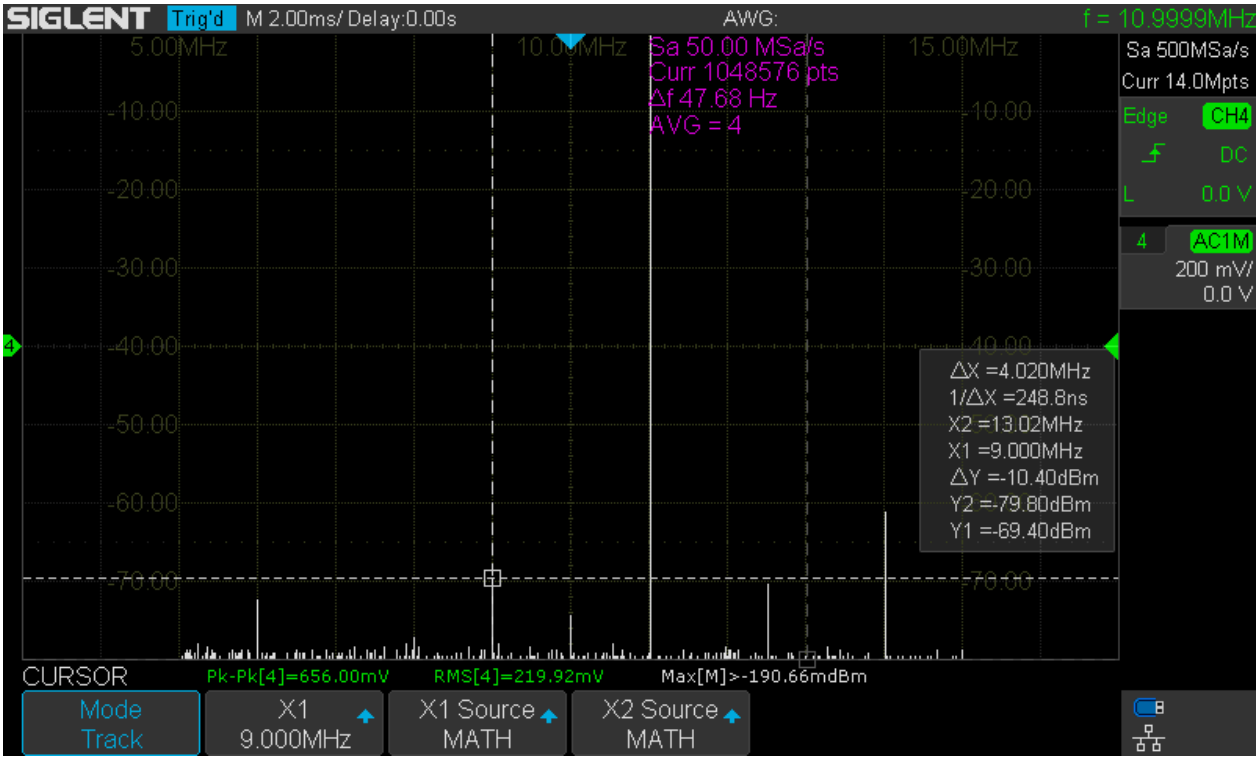
SDS1104X-E_FFT_IM3_0dBm



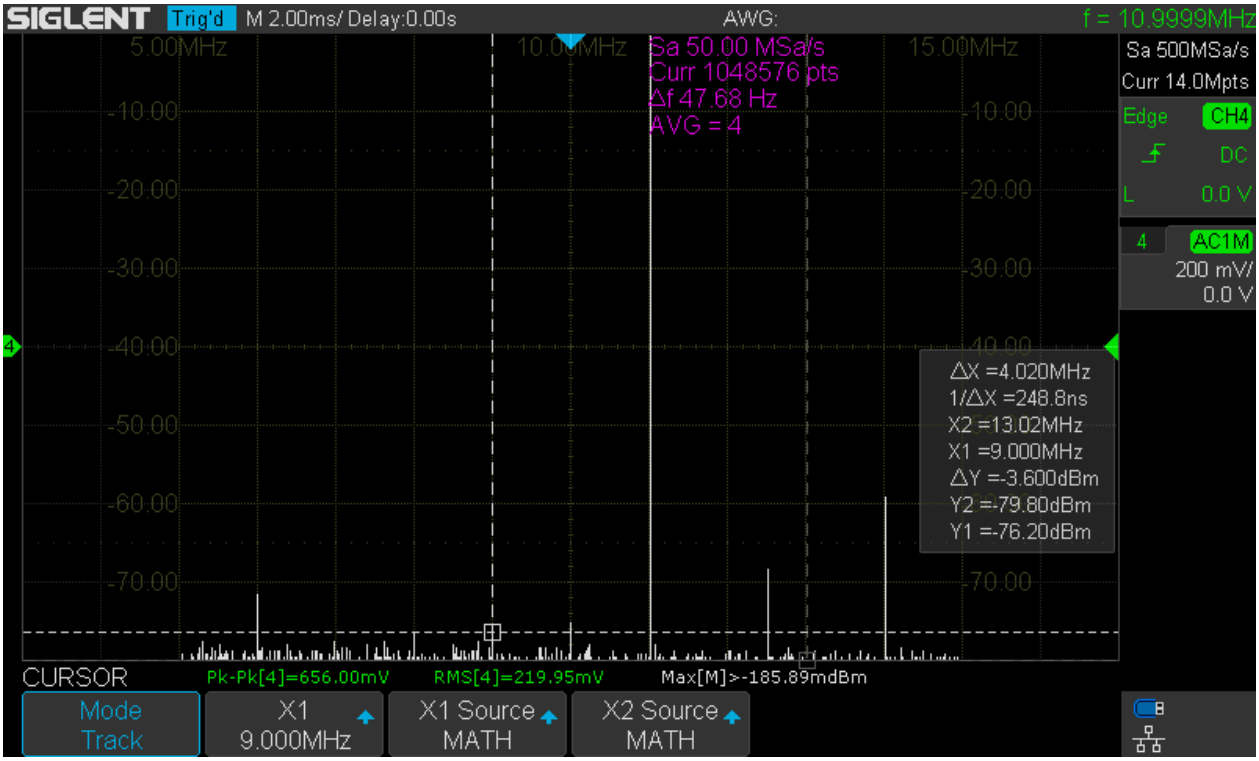
SDS1104X-E_FFT_IM3_-20dBm

SDS1104X-E_FFT_IM3_-30dBm

SDS1104X-E_FFT_IM3_-60dBm



SDS1104X-E_FFT_IM3_-70dBm



SDS1104X-E_FFT_IM3_-76dBm

Application Examples

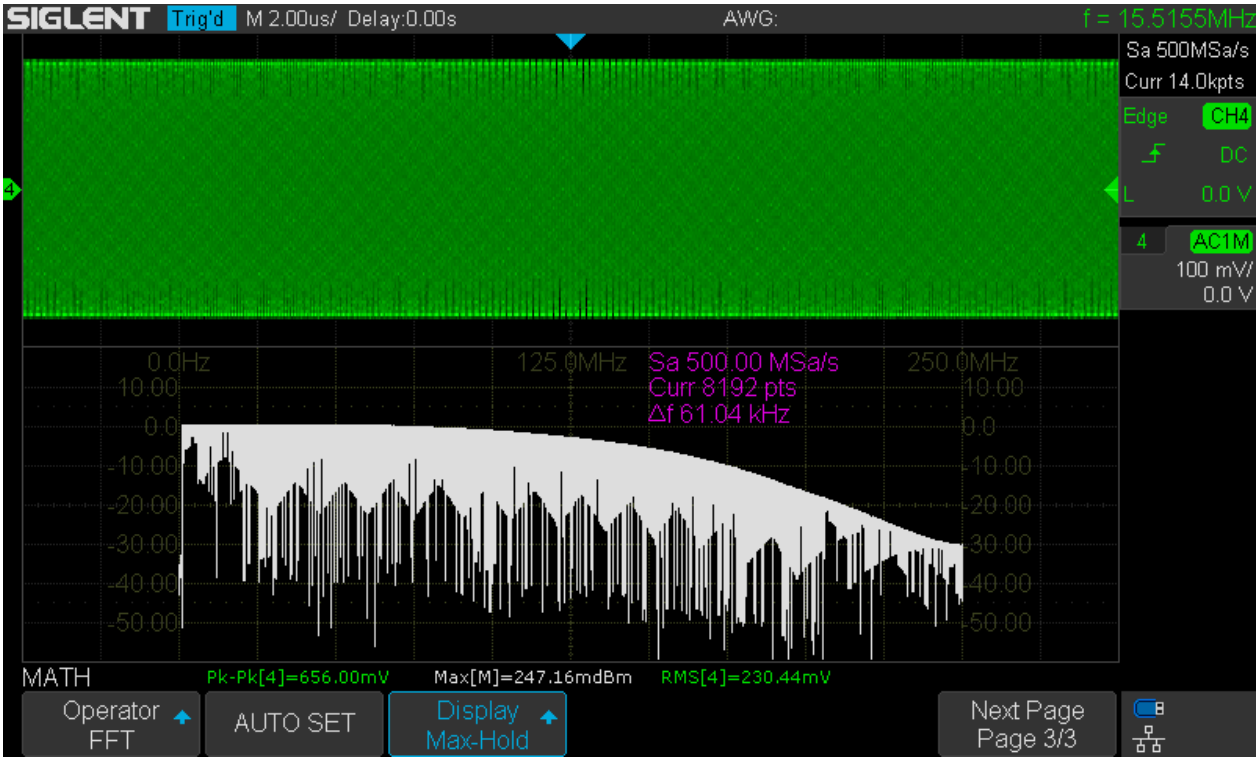
In this chapter we're looking at some more practical examples for using the SDS1104X-E FFT.

Wideband Measurement

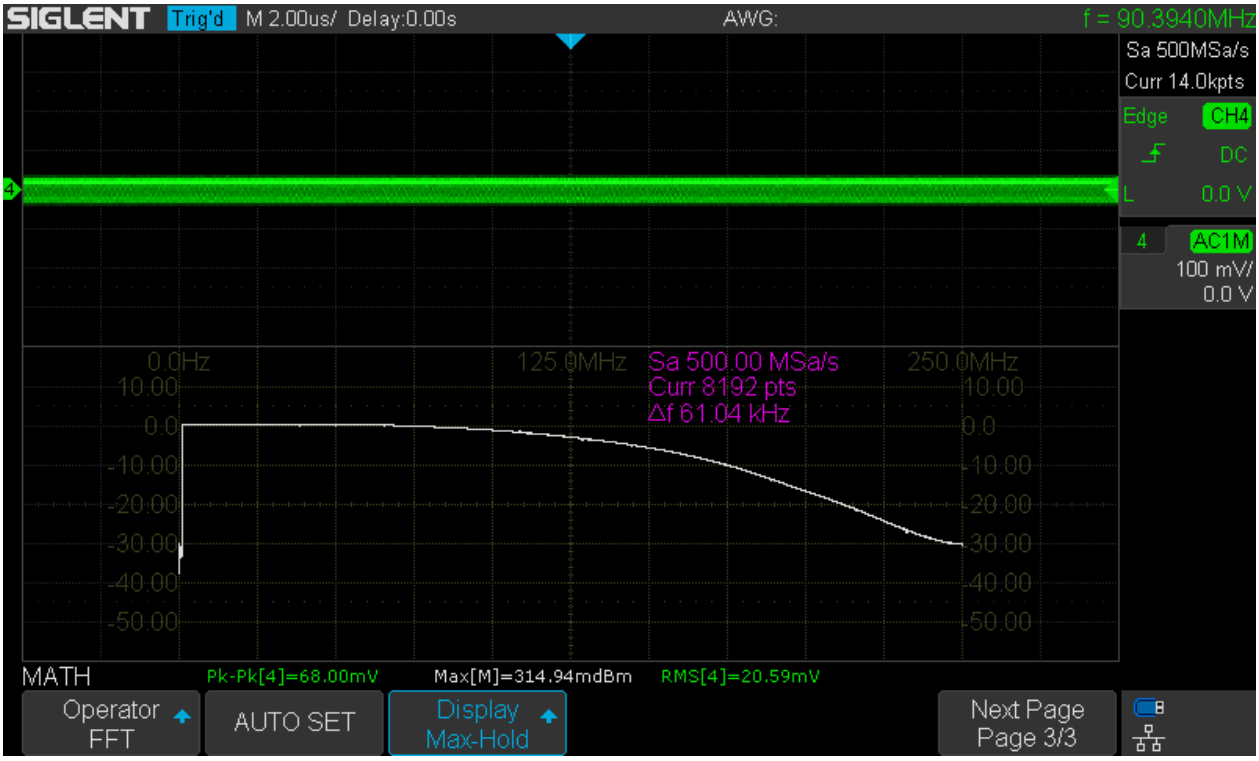
We can plot the frequency response of the SDS1104X-E in the range of 1MHz to 250MHz using its FFT and an external signal source that covers the entire frequency range. There are three possibilities:

1. A wideband white noise source would ideally output a continuous spectrum of all frequencies at the same time. Unfortunately, such noise sources with a completely flat spectrum up to 250MHz aren't that easy to find and if so, they are certainly anything but cheap.
2. A pulse generator that outputs extremely narrow pulses with near zero transition time will create a discrete spectrum with a spectral line at constant amplitude for every *f* x *n*, where *f* is the repetition frequency of the pulse and *n* is any integer value from 1 to infinity. Once again, such a pulse generator is anything but common or cheap – e.g. 3.5ns pulse width and 1ns rise time would only be good up to some 30MHz.
3. A swept sine signal can provide near ideal amplitude accuracy, it just takes quite a while to build up the complete frequency response plot.

I've tried all three methods (using my limited resources), and unsurprisingly the third one yielded the best results by far. A sine wave with 0dBm amplitude, swept from 1MHz to 250MHz within 60 seconds was fed into channel 4 of the SDS1104X-E. The FFT has been setup for 250MHz analysis bandwidth, i.e. 2µs/div timebase and memory depth limited to 14kpts, yielding a frequency step of 61kHz, which is just perfect for this task.



SDS1104X-E_FFT_Sweep_1M-250MHz_init



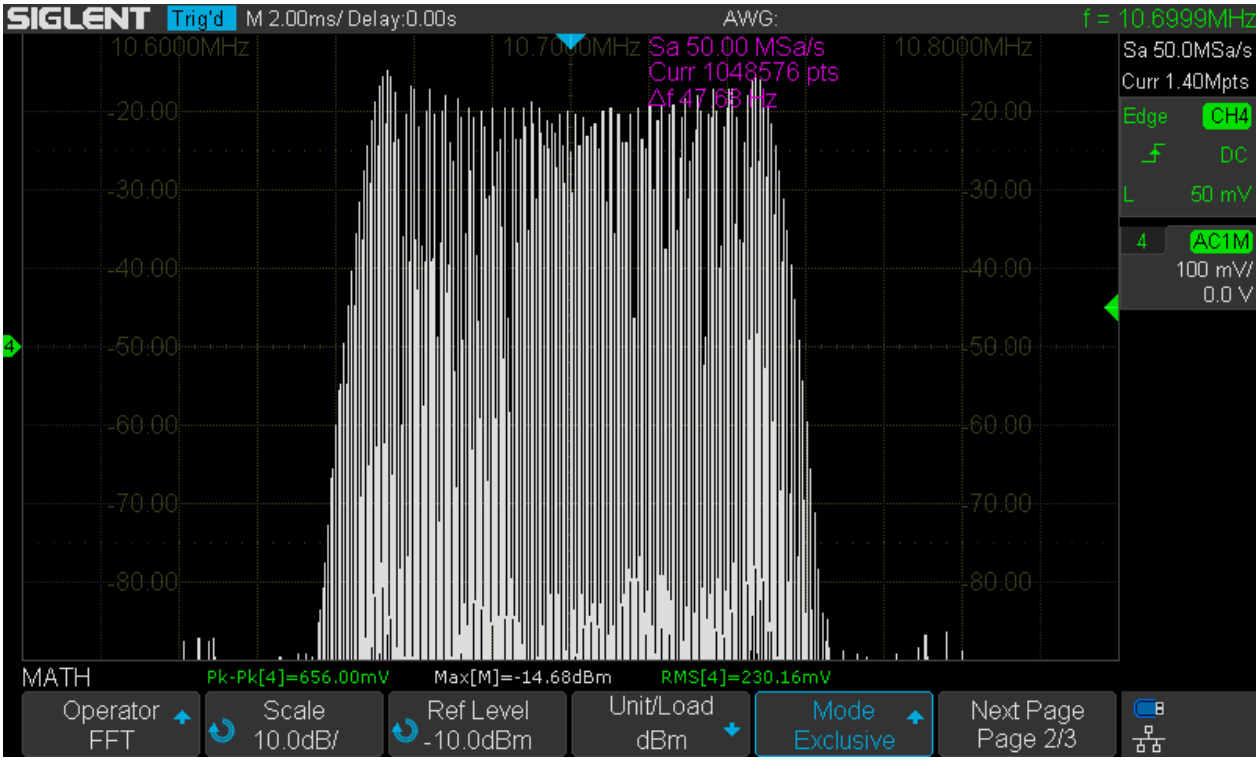
SDS1104X-E_FFT_Sweep_1M-250MHz_final

In order to collect all individual measurements, we use the Max-Hold Display mode. The first screenshot shows the trace after the first scan (after some 60 seconds), whereas the 2nd screenshot demonstrates how a near perfect frequency response graph can be obtained by just waiting several minutes until the maxima for all horizontal pixels have been registered. Of course, the same result could be obtained after

the initial sweep if it were made much slower, but with the faster sweep we get a quick overview and then only need to wait any further if the result actually meets our expectations.

Narrowband Measurement

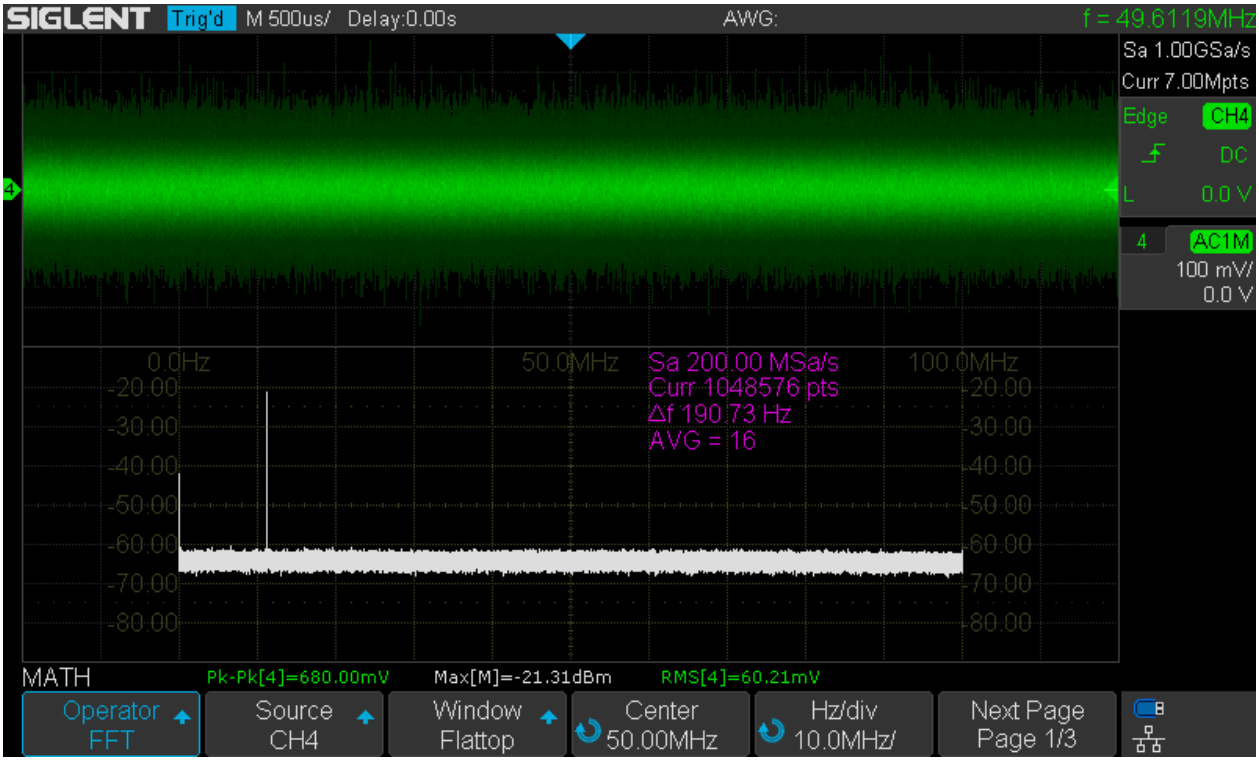
We can look at the IF signal of a domestic FM receiver at 10.7MHz, with a 1kHz frequency modulation and 50kHz max. deviation. For this we choose an analysis bandwidth of 25MHz – 12.5MHz would be even better, but we'd need to enable the 2nd channel in the group (Ch. 3 in this example) since this BW is only available in dual channel mode. We want the best possible frequency resolution, so we choose the longest FFT with 1048576 points, providing a frequency step of 47.7Hz. This leads us to 2ms/div timebase and 1.4Mpts memory depth and we just need to adjust the center frequency to 10.7MHz and set a span of 200kHz by selecting a horizontal scale of 20kHz/div.



SDS1104X-E_FFT_FM_10.7MHz

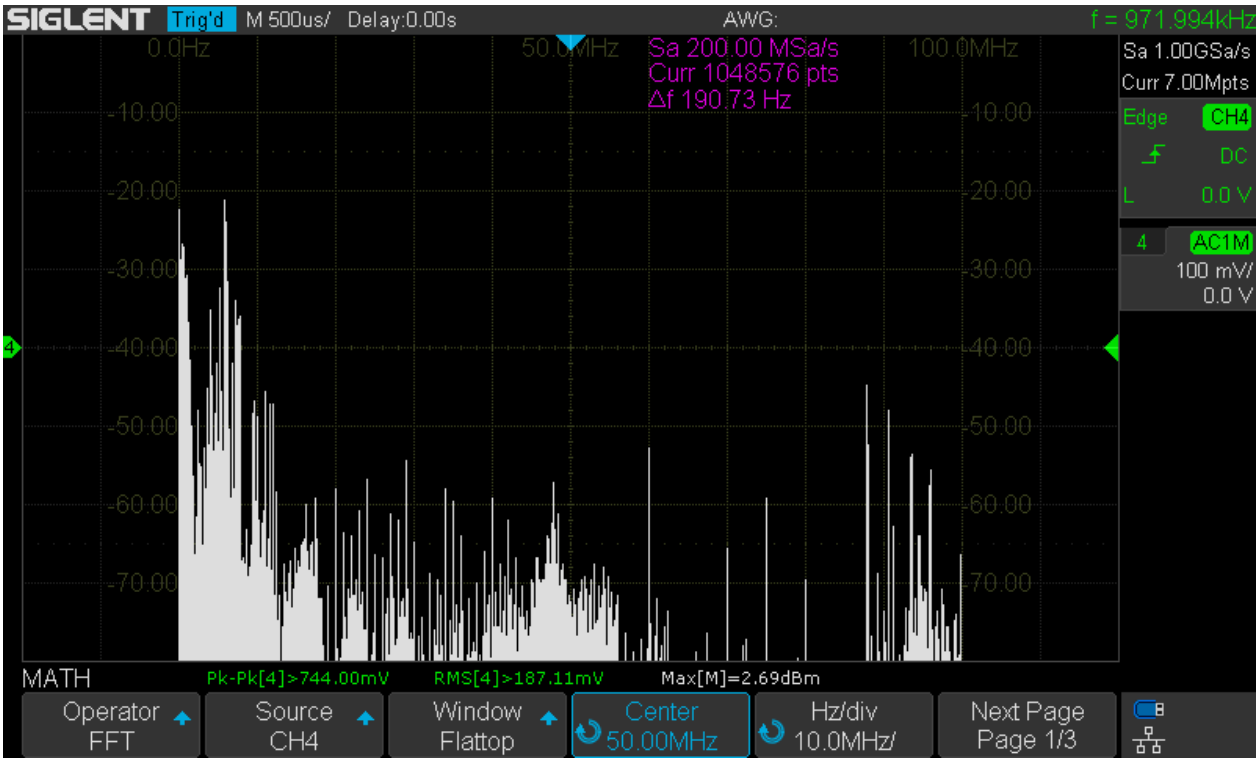
Signal burried in Noise

FFT can help to find weak signals completely buried in noise, hence invisible in the time domain. The example below shows a -20dBm 11.111MHz signal hidden under a 680mVpp noise floor. In the time domain, we can only see the noise and wouldn't even know about the -20dBm signal. The FFT over the full scope bandwidth shows that signal very clearly.



SDS1104X-E_Noise100mV_Sig-20dBm_11111kHz

Broadcast



SDS1104X-E_FFT_Broadcast

The Screenshot above shows the broadcast signals over a 100MHz bandwidth captured with just a piece of wire (about 10m long) as antenna. We can see everything from VLF up to the local VHF radio stations.

Mask Testing

This is usually not a very popular feature – and for a reason; it often comes as an expensive option and has the reputation to be a tool exclusively for production tests. The latter might also explain why its implementation is agonizingly slow on many scopes, just capable of analyzing a couple of acquisitions per second. This might be sufficient for production tests, but makes it almost useless for anything else.

Thankfully, Siglent went a different route as they have understood the potential of a more ambitious approach on mask test (Siglent calls it “Pass/Fail”) and it comes as a standard feature for free. The implementation on the X-series DSOs works pretty much at the same speed as normal acquisition and the maximum fault detection rate is close to the usual trigger rate aka “waveform update speed”. Consequently, mask testing can be a very useful tool for glitch hunting and fault finding, yielding a reasonably high probability for capturing a rare glitch. With one of the more common ridiculously slow implementations, it might take forever until a rare and very brief mask violation finally gets detected.

The table below shows the maximum fault detection rate for a 2MHz input signal at various timebase settings from 1ns/div up to 10µs/div. The table also shows the expected detection rate in percent of the total number of faults as well as the average time for detecting a glitch that has a repetition rate of 1Hz.

SDS1104X-E Pass/Fail			
Time Base [s/div.]	Detection Rate [Hz]	Detection Rate [%]	Detection Time [s²]
1,00E-9	4,80E+3	0,007%	14881,0
2,00E-9	9,18E+3	0,026%	3890,4
5,00E-9	33,78E+3	0,236%	422,9
10,00E-9	12,76E+3	0,179%	559,8
20,00E-9	13,40E+3	0,375%	266,5
50,00E-9	107,60E+3	7,532%	13,3
100,00E-9	18,12E+3	2,537%	39,4
200,00E-9	12,00E+3	3,360%	29,8
500,00E-9	6,79E+3	4,753%	21,0
1,00E-6	4,56E+3	6,384%	15,7
2,00E-6	2,82E+3	7,896%	12,7
5,00E-6	1,31E+3	9,170%	10,9
10,00E-6	694,00E+0	9,716%	10,3

SDS1104X-E_Mask_Test_Detection_Rate

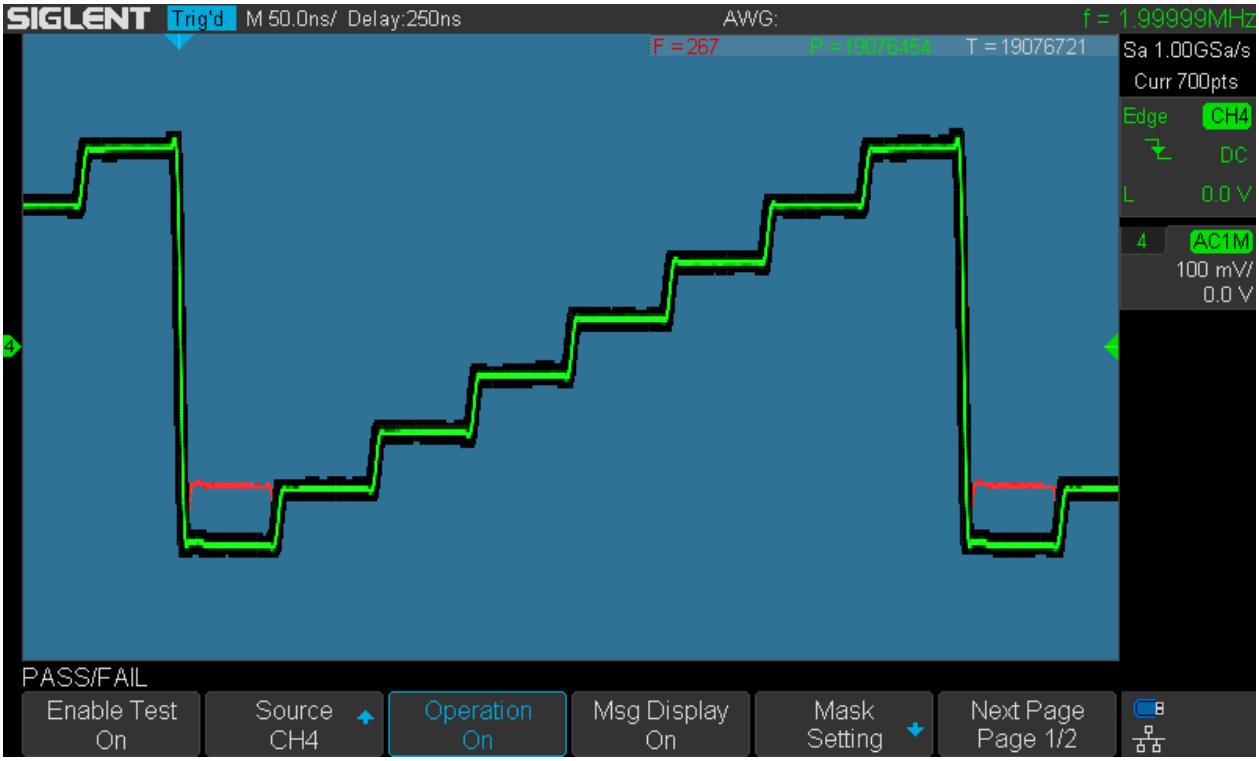
As an example, let’s assume a step-up curve where the bottom step is occasionally missing. The signal repetition rate is 2MHz and the fault occurs at 20Hz, hence an original fault rate of 1:100000 or 0.001%.

At the sweet spot of 50ns/div we get a max. fault detection rate of 107600 acquisitions per second, hence a theoretical probability for capturing the fault of

Fault_detection_rate x *timebase* x *horizontal_divisions* = 107600 x 50e-9 x 14 = 0.07532 = 7.532%;

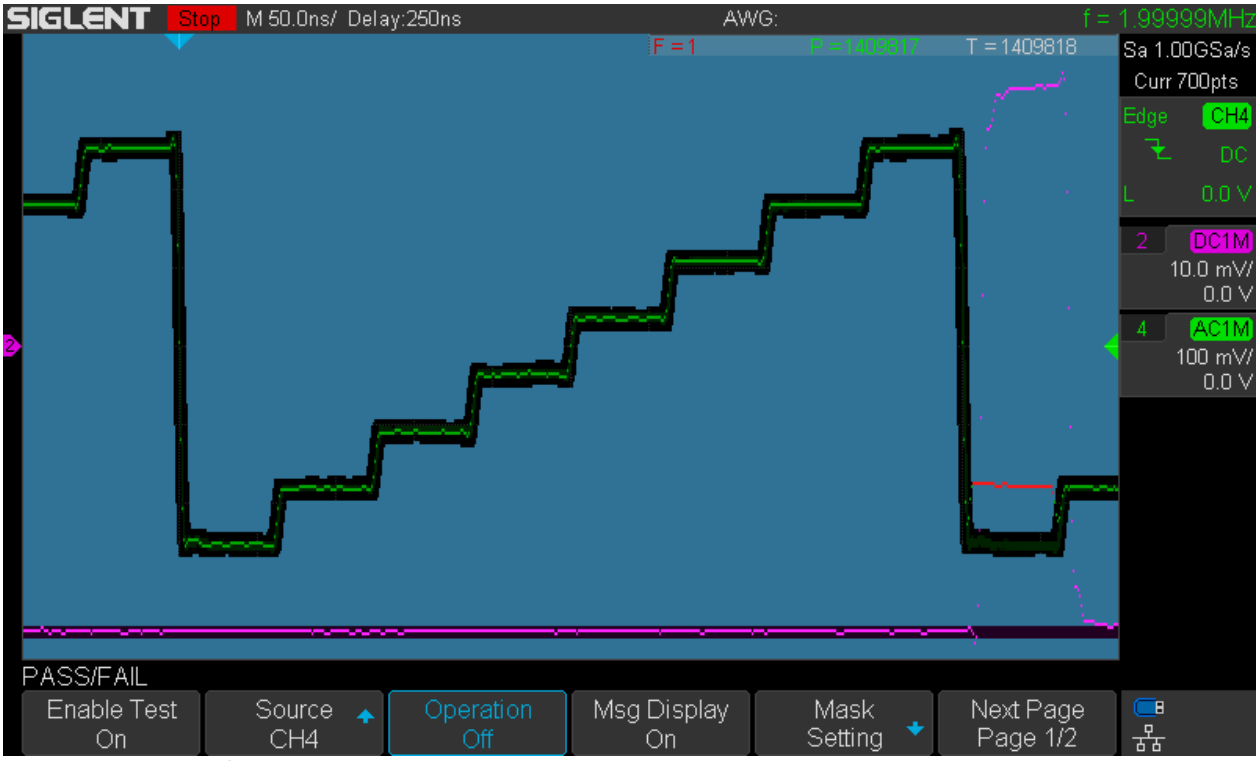
The screenshot below demonstrates this very situation, showing the pass/fail statistics after 177 seconds runtime: a total number of 19076721 (19 million!) tests and 267 fault detections. The actual fault rate was 20Hz, resulting in a total of 20 x 177 = 3540 faults where 267 got detected. 267 / 3540 = 0.0754 = 7.54%; Average detection time was 177s / 267 = 0.663s and for 1Hz fault rate this would be 0.663s x 20 = 13.26s. So these measurements confirm the theoretical numbers given in the table above very nicely.

Don’t forget to turn display persistence on in order to clearly see where the mask violations have occurred. It’s the red traces in the screenshot, indicating that the bottom step gets cropped occasionally. With this information, one could easily find an appropriate (e.g. runt) trigger to capture the glitch itself and look for related signals that might be causing the fault directly or indirectly.



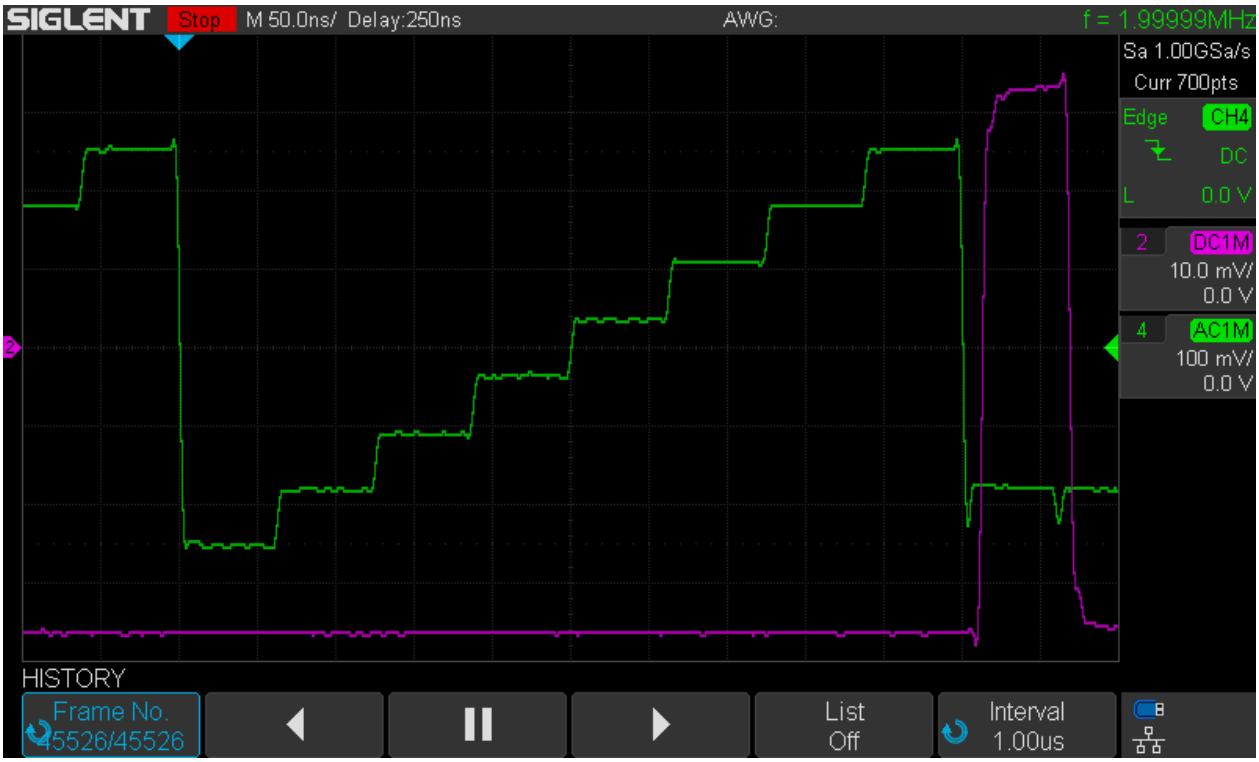
SDS1104X-E_Masktest_50ns_2MHz_20Hz_177s

Another option is enabling the *Stop on Fail* option on page 2 of the *PASS/FAIL* menu. This will stop the acquisition after the first mask violation has been detected, as shown in the screenshot below.

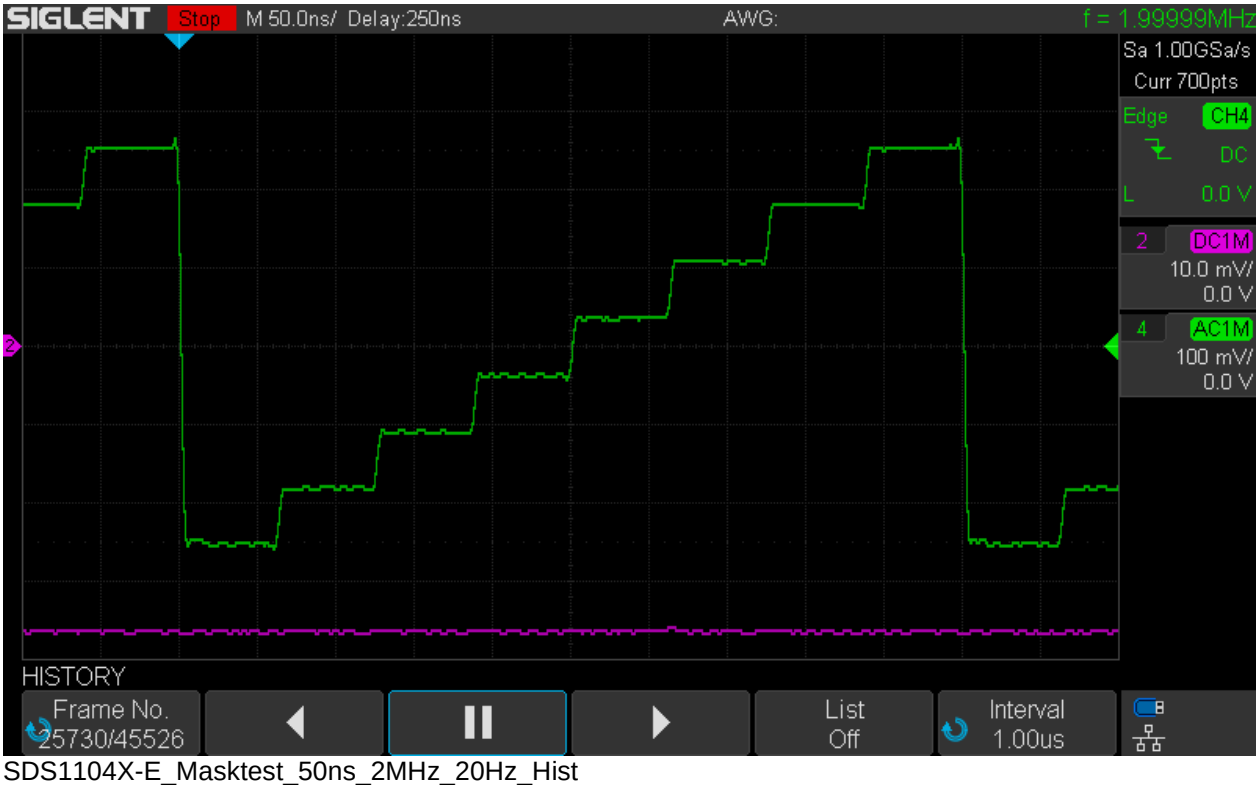


SDS1104X-E_Masktest_50ns_2MHz_20Hz_Stop

Another channel is used to monitor a signal that is suspected to be related to the fault and sure enough it is, as can be seen in this screenshot already thanks to the display persistence. Now we can disable mask test and enter the history:



SDS1104X-E_Masktest_50ns_2MHz_20Hz_Hist_Last



SDS1104X-E_Masktest_50ns_2MHz_20Hz_Hist

The first screenshot shows the last history frame that holds the acquisition with the mask violation. We can clearly see that the signal at channel 2 is closely related to this event. Any other history frame just shows the normal signal without a fault together with the inactive signal on channel 2 and the 2nd screenshot gives an example for this.

You might ask what the advantages of mask testing are compared to just using (infinite) display persistence alone. Well, there is a couple…

- We might have less stationary signals that require wider tolerances on the mask setting. With such an unstable signal, any violation would be harder to spot with just persistence alone, whereas in the mask test, they stick out in red color.
- Mask test provides some statistics if *Msg Display* is turned on. By knowing the number of tests and faults, we can estimate the fault rate, which might give valuable hints on where to look for the culprit.
- We can set mask test to stop after the first occurrence of a mask violation. This allows the close examination of the situation including related signals.

We can calculate the fault rate by dividing the detected faults by the total number of tests. In this example, it would be 267 / 19076721 ~ 14e-6; this is not exactly the true fault rate which would be 20Hz/2MHz = 10e-6, but certainly close enough to make a guess what other signal might be related to the glitch. We would only consider slow signals (or conditions) with a repetition rate <30Hz and could concentrate on observing these. Likewise, it could be a problem in our firmware and we would use a GPIO to generate a signal that allows us to monitor a critical region in the firmware with the scope and see whether the fault consistently occurs when the code in that region is being executed.

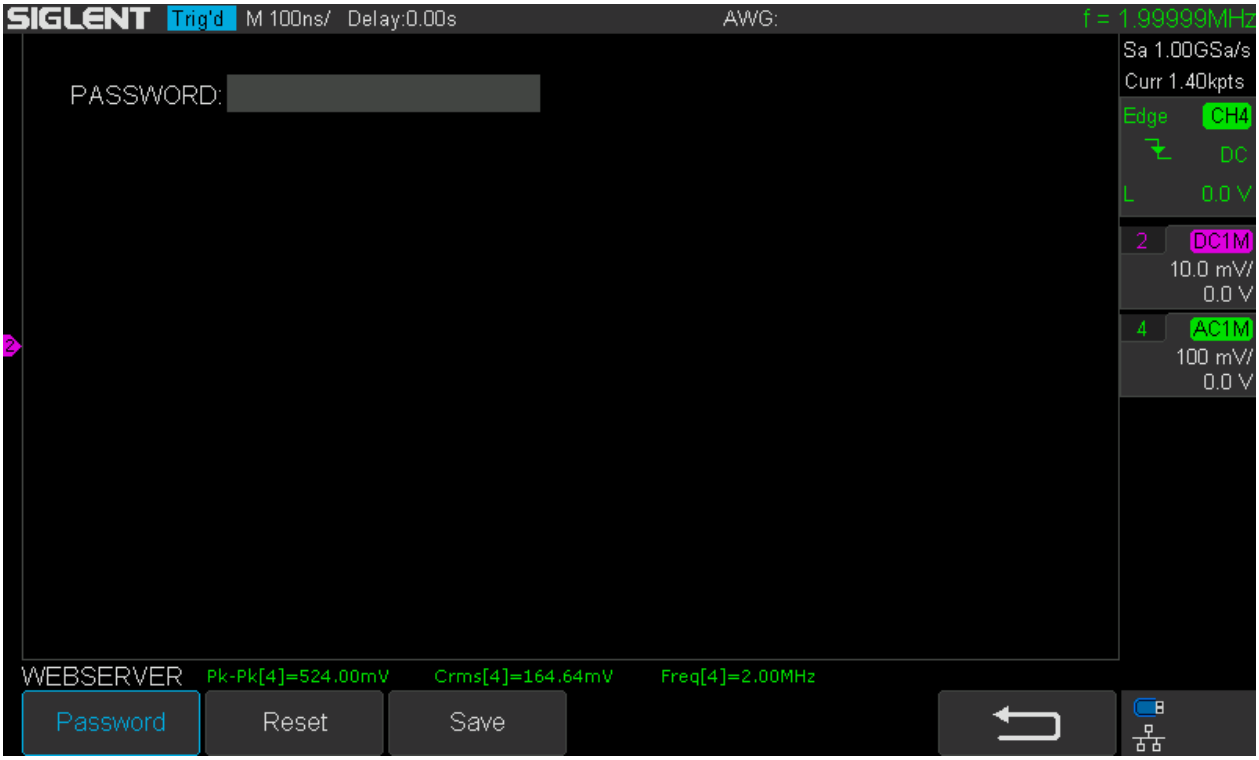
Web Server

For the first time, a Siglent DSO has a built-in web server. It is nothing exciting, and does not show a live view of the DSO screen, so it's by no means a substitute for an USB scope. It just allows remote setup of the most common functions and pulling individual screenshots. A nice first shot nevertheless…

SDS1104X-E_IP_setting

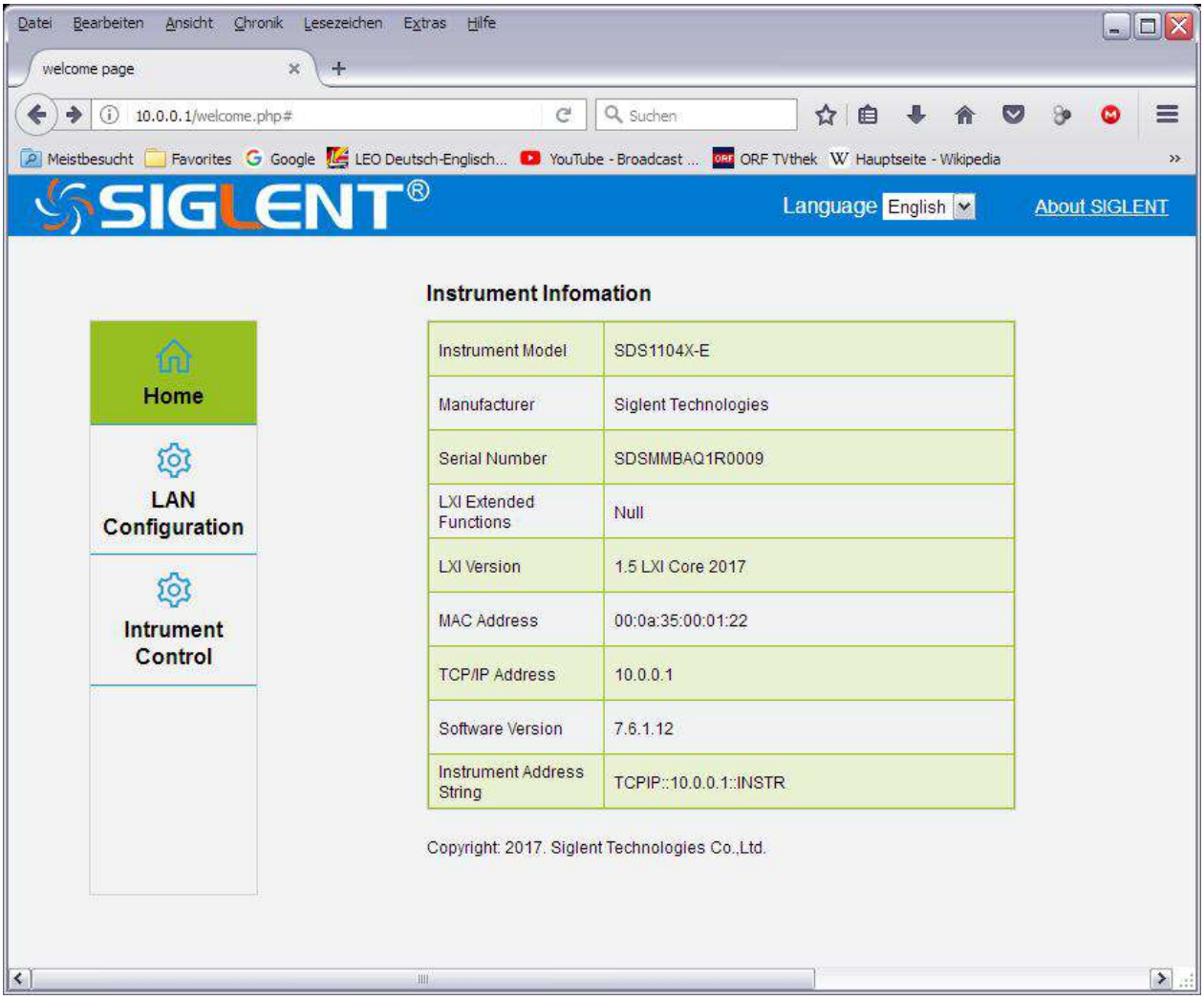
On page 2 of the *Utility* menu, we can set up the IP configuration. This is totally easy as long as a network router with integrated DHCP server is available. All that's needed is enabling DHCP and waiting a moment until the scope has received its individual IP address, see screenshot above. In a simpler network without DHCP server, the IP address, Subnet Mask and Gateway have to be set manually – I have tried that as well and it works as expected.

On page 4 of the *Utility* menu a password for web server access can be set. I haven't bothered to do this, as I hate passwords (which I tend to forget) and my local network isn't accessible from the internet.



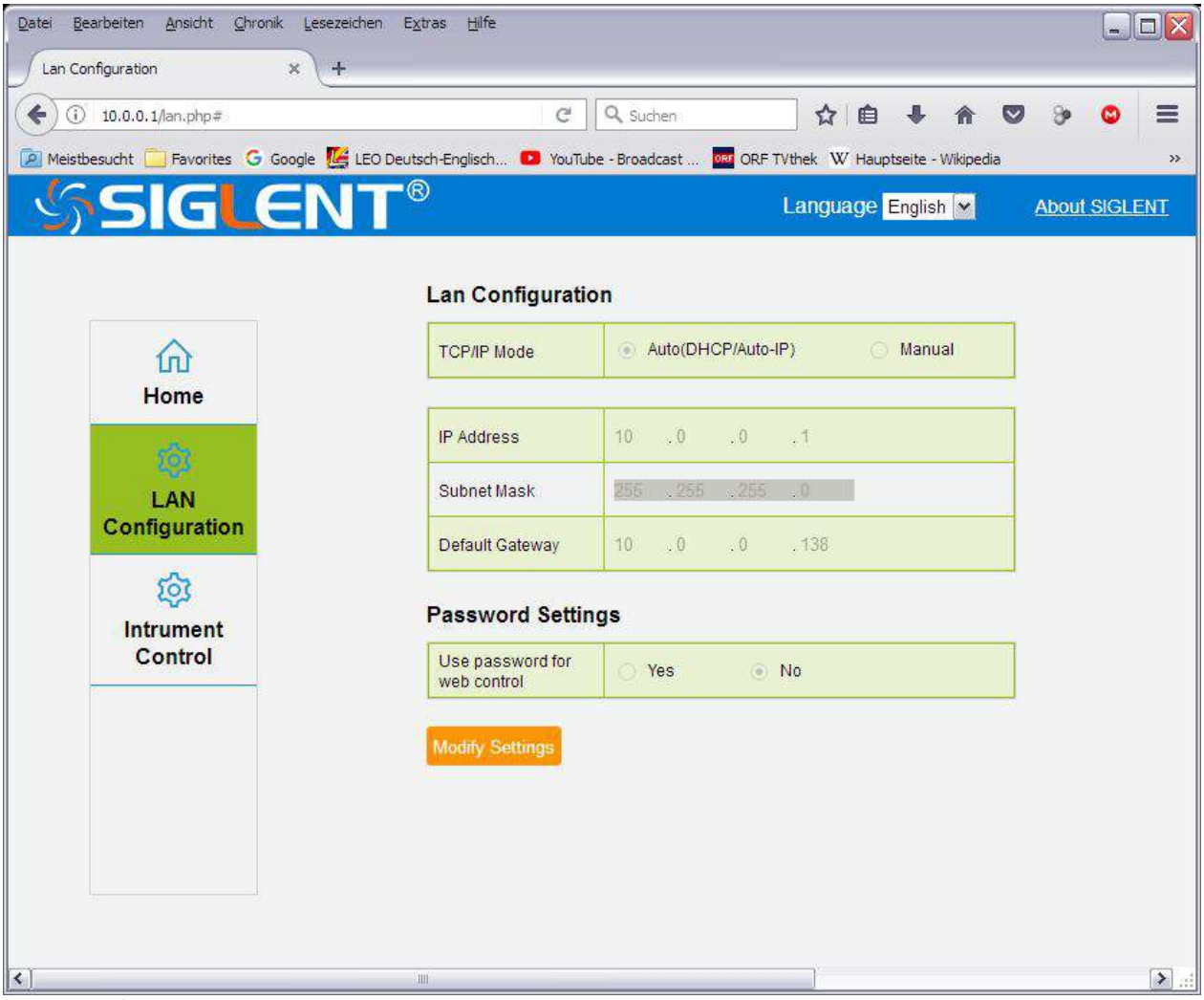
SDS1104X-E_WEB_Password

On the local computer, we just need to open a web browser and type the scope's IP address into the address bar. After hitting [**Return**] the home screen of our web server appears.



WEB_Start

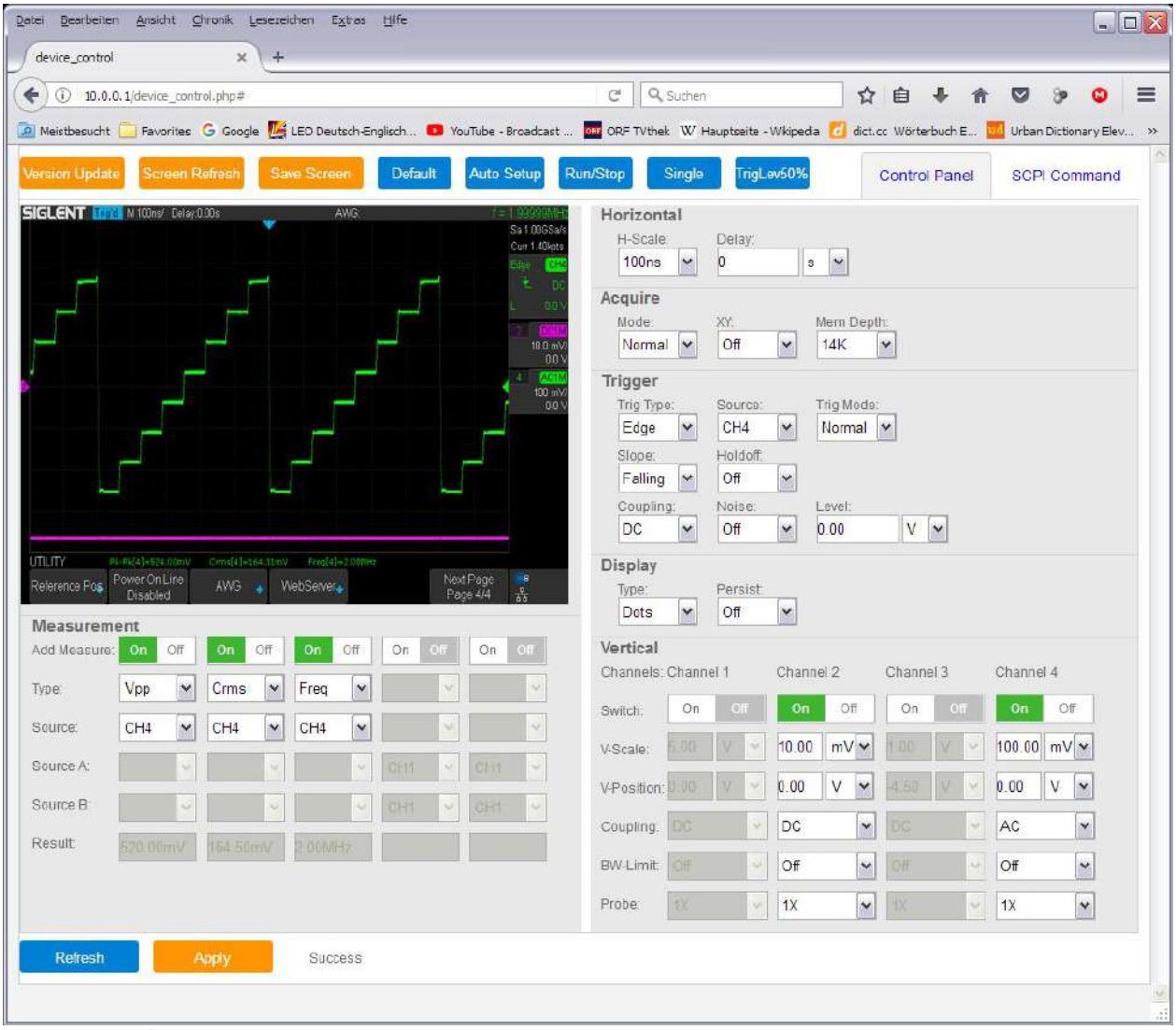
This contains some useful information about the instrument, like model, serial number and firmware version. This screen offers access to the LAN configuration as well as the Instrument Control. Let's have a look at LAN Configuration first.



WEB_Settings

Nothing fancy here and I assume the screenshot is self-explanatory.

So let's move on to the actually interesting part, the Instrument Control.



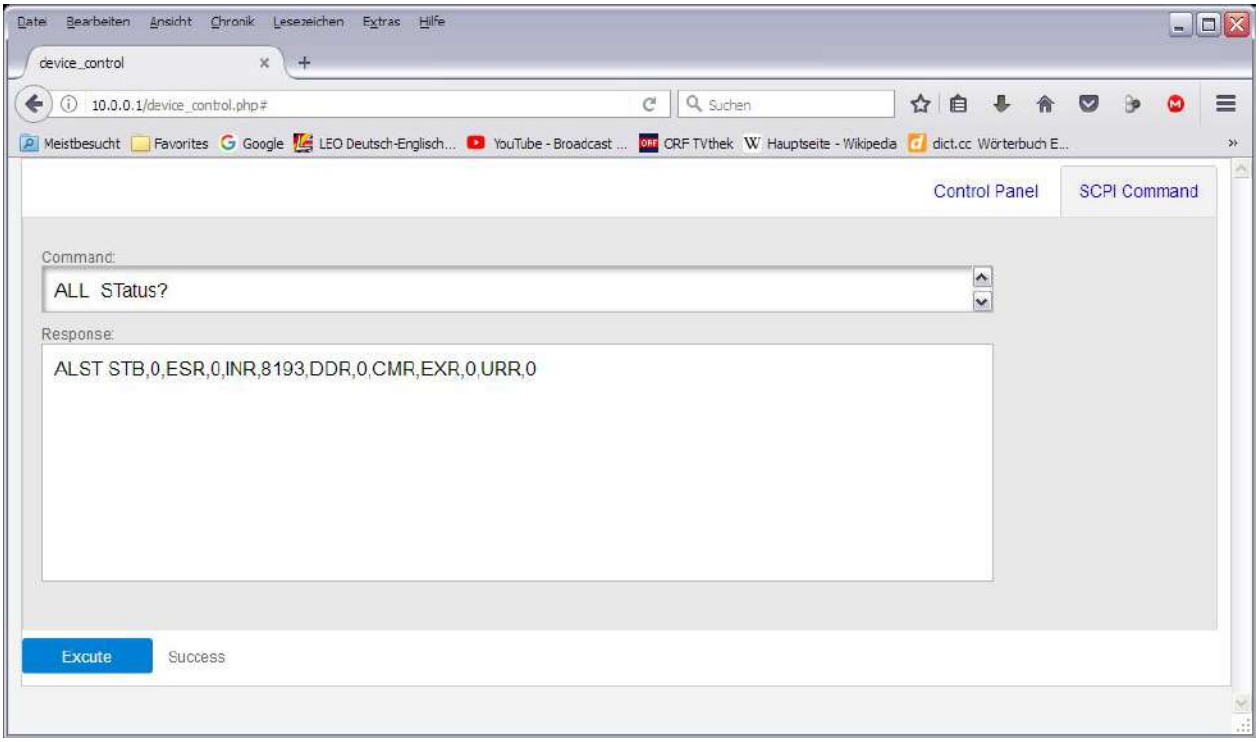
WEB_Control

We get an initial snapshot of the DSO screen and can do a screen refresh anytime we want – it just isn't fast; even on a Gbit network, it takes up to 3s to load a new screenshot, at least on my ancient computer.

We can configure the most basic settings, but even for this, not all options are available. For example, we cannot choose the interpolation method in the Acquire section and the Display section does not even offer Color mode. Likewise, not all trigger types are available. The Vertical section does not provide the Deskew and Invert Options. The available Measurements appear to be fairly complete, but here again, the options for Gate, Statistics and All Measurements are missing. All in all it's a really basic control application that doesn't support any of the more advanced features of the SDS1104X-E. I'm not sure if it really would be sensible to offer much more, as a true remote operation of the scope makes little sense without a live view anyway. I look at the web server more as a demonstration tool to play with remote commands and the most common tasks would be getting screenshots from the scope without the need for an USB flash-drive.

Screenshots can be saved by clicking *Save Screen* and then a bitmap file is stored in the web browser's default download directory. Alternatively, the usual methods for getting an image from a website can be applied, like right-click on the screen image and select "Save graphics as…" from the context menu. In any case, a bitmap file is stored, whereas we want it to be PNG. My preferred method is right-click, select "Copy Graphics" from the context menu, then paste it to an image processing program – I prefer Microsoft Office Picture Manager for simple tasks like this – and then export it as .png file from there.

The Instrument Control page offers yet another nice option; this is the “SCPI Command” tab in the upper right corner. With this we get a terminal window and can talk to the instrument via SCPI without the need for running a terminal program on the computer. Of course this is absolutely basic and mainly serves for playing around with the scope and trying out various commands. The example below shows the response to an ALL_STatus? command.



WEB_SCPI_ALL_ST

Segmented Memory

Often sold as an expensive option (or not available at all), this very convenient feature comes for free with the Siglent X-series scopes. There are two ways to use it: History and Sequence Mode.

First we need to understand how the memory depth setting in the *Acquisition* menu affects segmented memory.

X-Series scopes generally use automatic memory depth selection; the current record length (number of points for a single acquisition) is always determined by the timebase setting and displayed in the top right corner of the screen. At fast timebases, the record length becomes very short and is only a tiny fraction of the available acquisition memory. Yet this memory is not wasted, but gets filled with up to 80000 records, each of them resulting from an individual trigger event.

This is just one of the reasons why there is still a manual memory depth selection in the *Acquisition* menu – it can be useful for slow timebase settings, where we might want to limit the record length in order to increase the number of records that fit into memory, hence can be retrieved in the history later. This is just one example, why we still need the *Mem Depth* setting, and in actual fact it just sets a record length limit.

Let's assume a single channel active at a 1ms/div timebase. With 14Mpts of acquisition memory, we still maintain a 1GSa/s sample rate and the record length is 14Mpts, so it fills the entire memory. Yet when we look at the history, we will find *two* records stored there, so there is actually a total 28Mpts of memory available. Since this is the same for both channel groups in an SDS1104X-E, we could even claim to have a total of 56Mpts of memory, but Siglent thankfully does not take the numbers game that far.

Anyway, we currently have just two memory segments and might want more. So we are able to sacrifice sample rate in favor of an increased number of history records. The table below shows how the record length limit affects sample rate and number of history frames along with the total memory available for a single channel at 1ms/div.

Siglent SDS1104X-E memory depth selection at 1ms/div			
Rec. Limit [Pts]	Sample Rate [Sa/s]	Segs [-]	Total Mem. [Pts]
14M	1G	2	28,00E+6
1.4M	100M	17	23,80E+6
140k	10M	188	26,32E+6
14k	1M	1891	26,47E+6

Record_Length_Limit_1ms

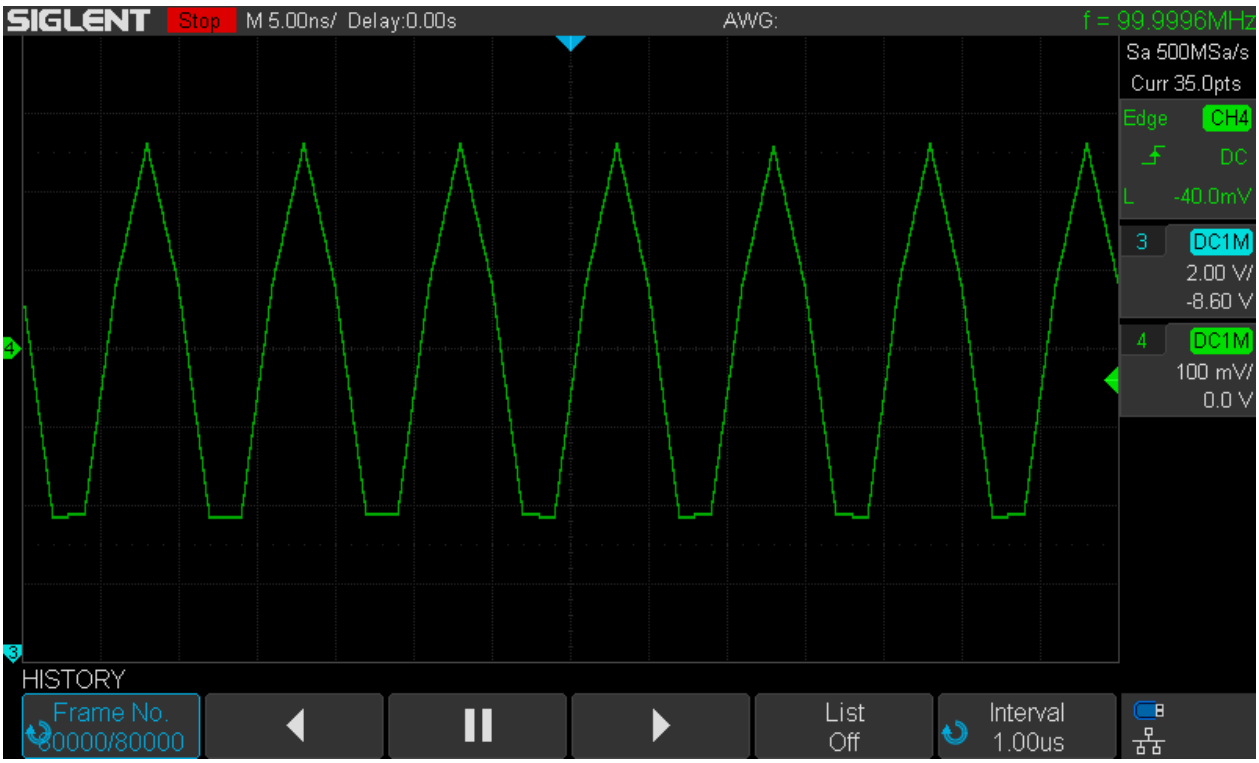
History

As has been already stated, this is not a special mode, but works silently in the background all the time. Consequently, history is available whenever we need it.

Let's examine a 100MHz sine wave once again, which looks rather fuzzy at 500MSa/s when displayed as vectors with simple x interpolation:

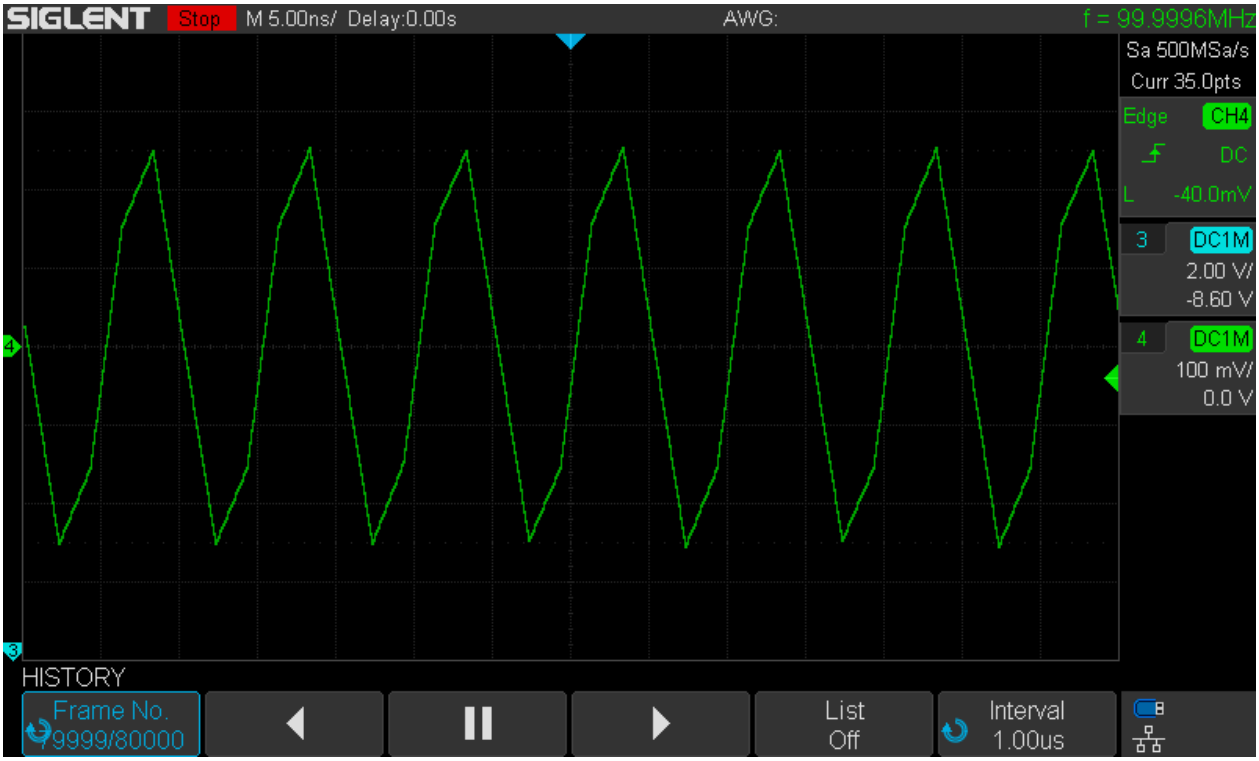
Hist_100MHz_vectors_x_run

We can stop acquisition and enter history mode by pressing the [**History**] button on the front panel.

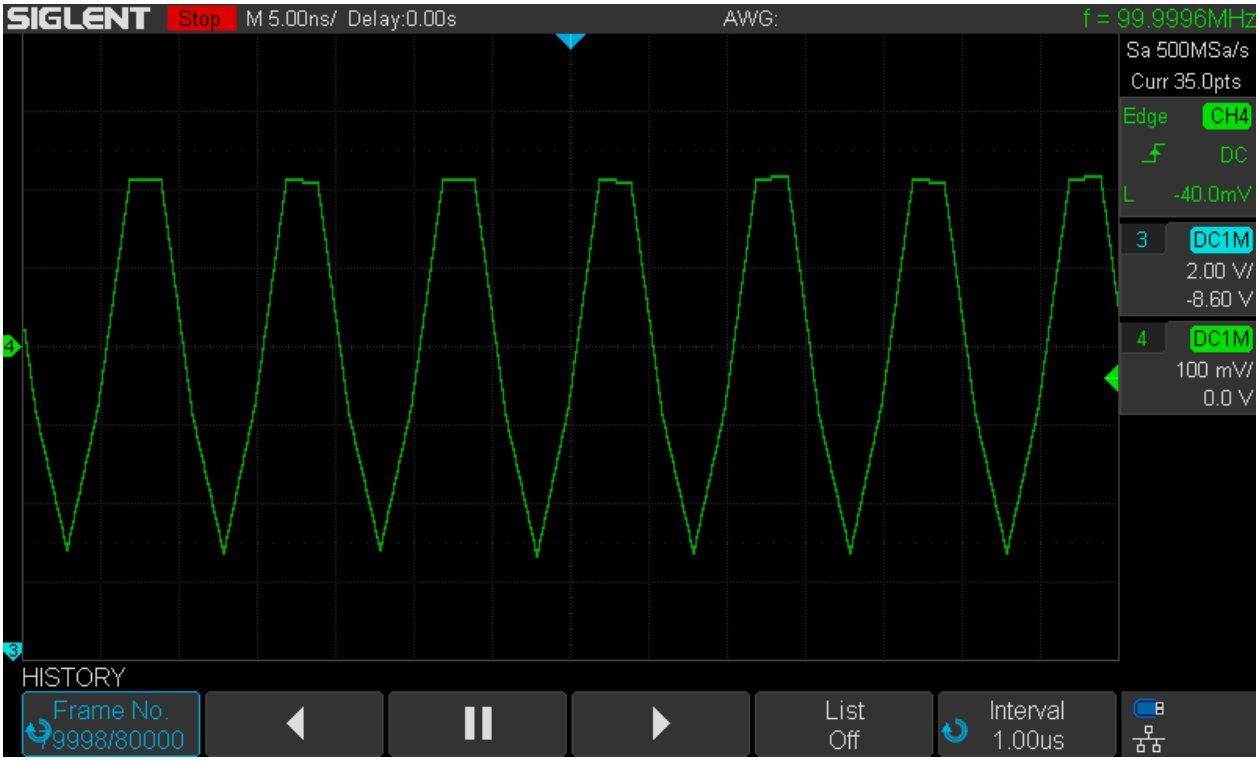


Hist_100MHz_vectors_x_80000

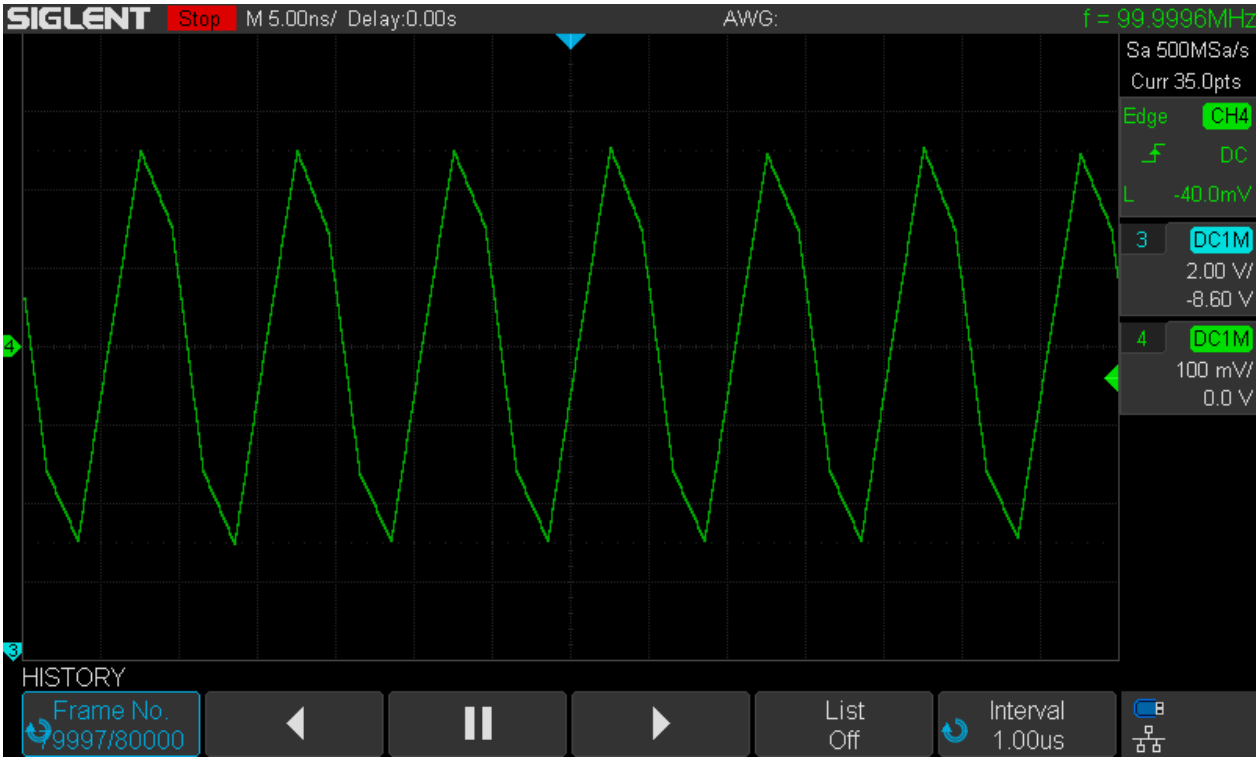
Now we can see that 80000 records (frames) have been stored in the history and we can analyze every single one if we want to. Initially, we're just seeing the last one, but can scroll through the frames in order to see the different variations of the misshaped signal caused by the simple x-interpolation.



Hist_100MHz_vectors_x_79999

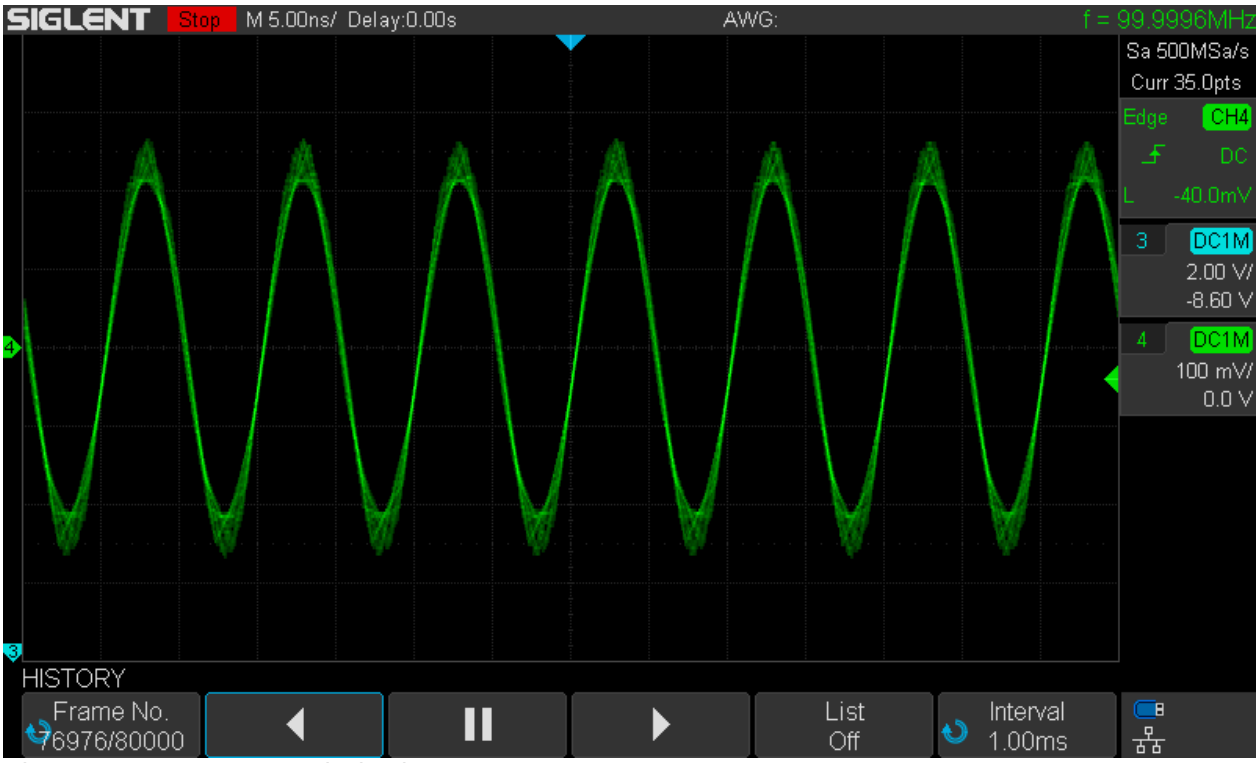


Hist_100MHz_vectors_x_79998



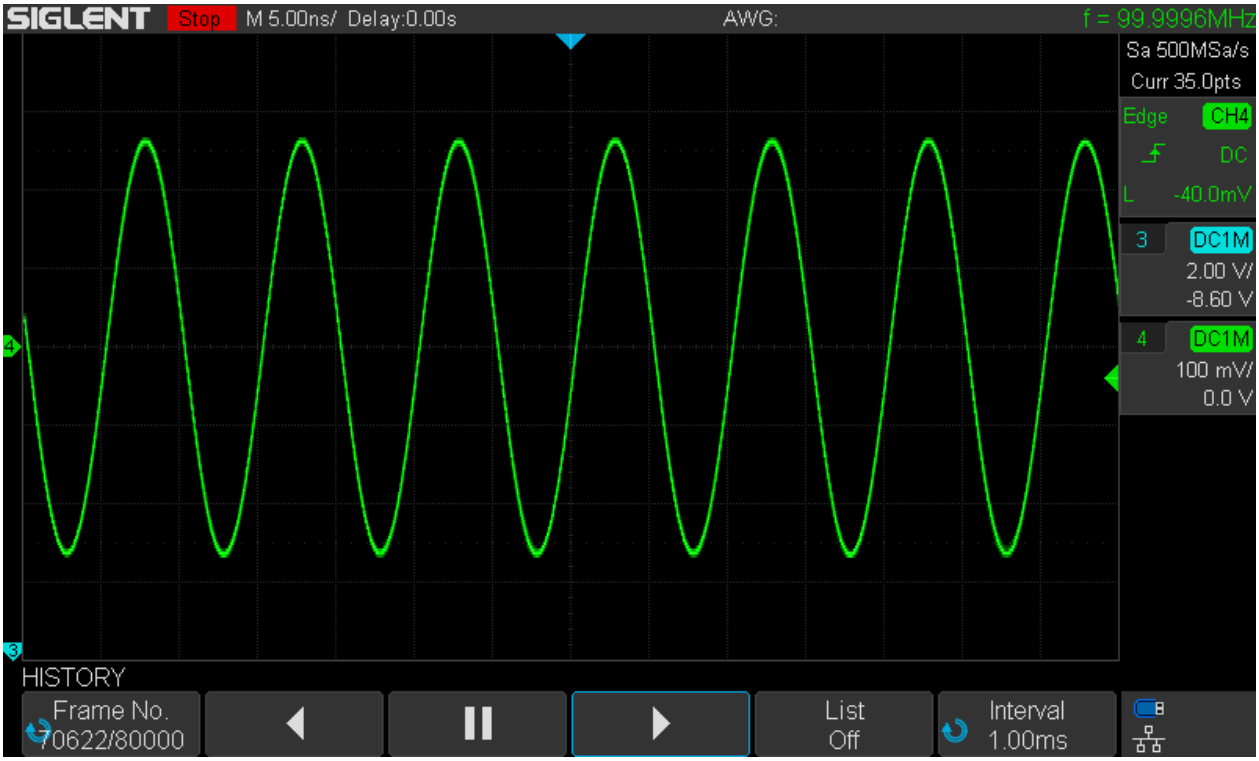
Hist_100MHz_vectors_x_79997

We can also playback the history in both directions at an arbitrary frame rate. We do it backwards using an interval of 1ms (=1000 frames per second) in this example and get the fuzzy trace again:

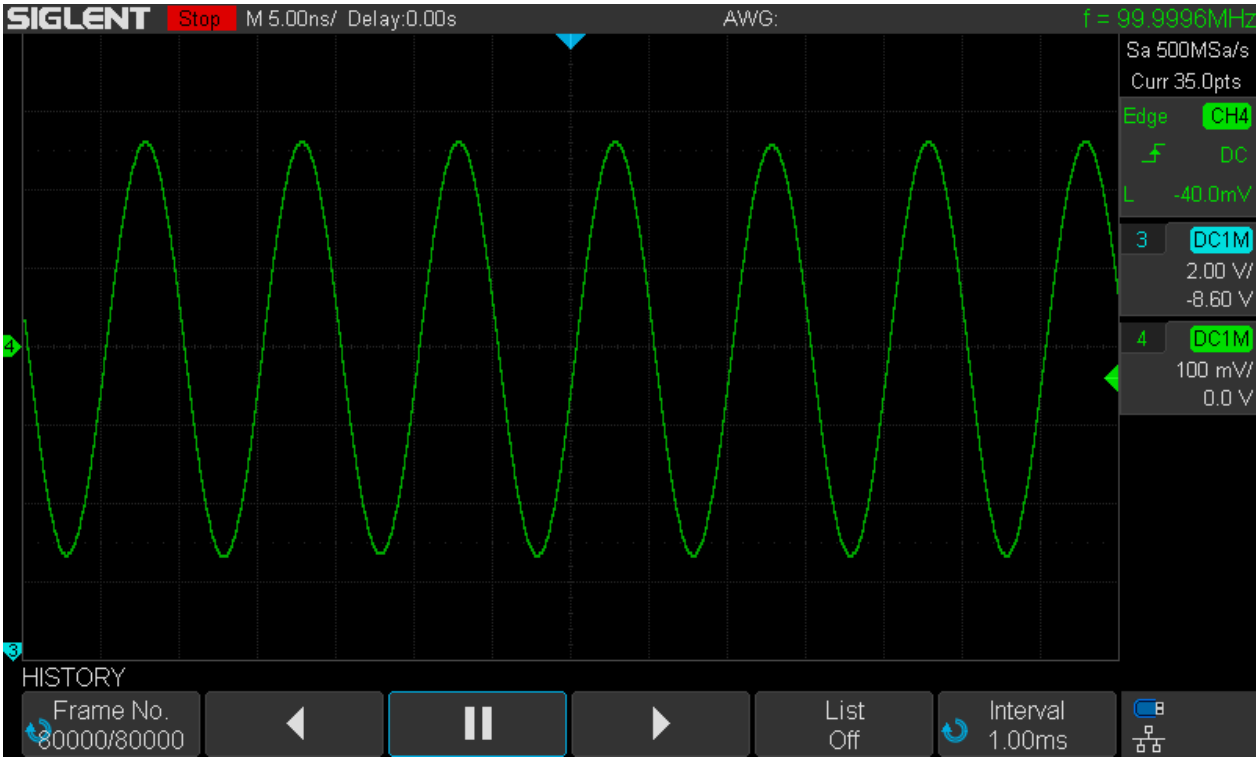


Hist_100MHz_vectors_x_Playback

Even in history mode, we can switch to sin(x)/x reconstruction and get a clear signal trace again during playback (in forward direction this time):



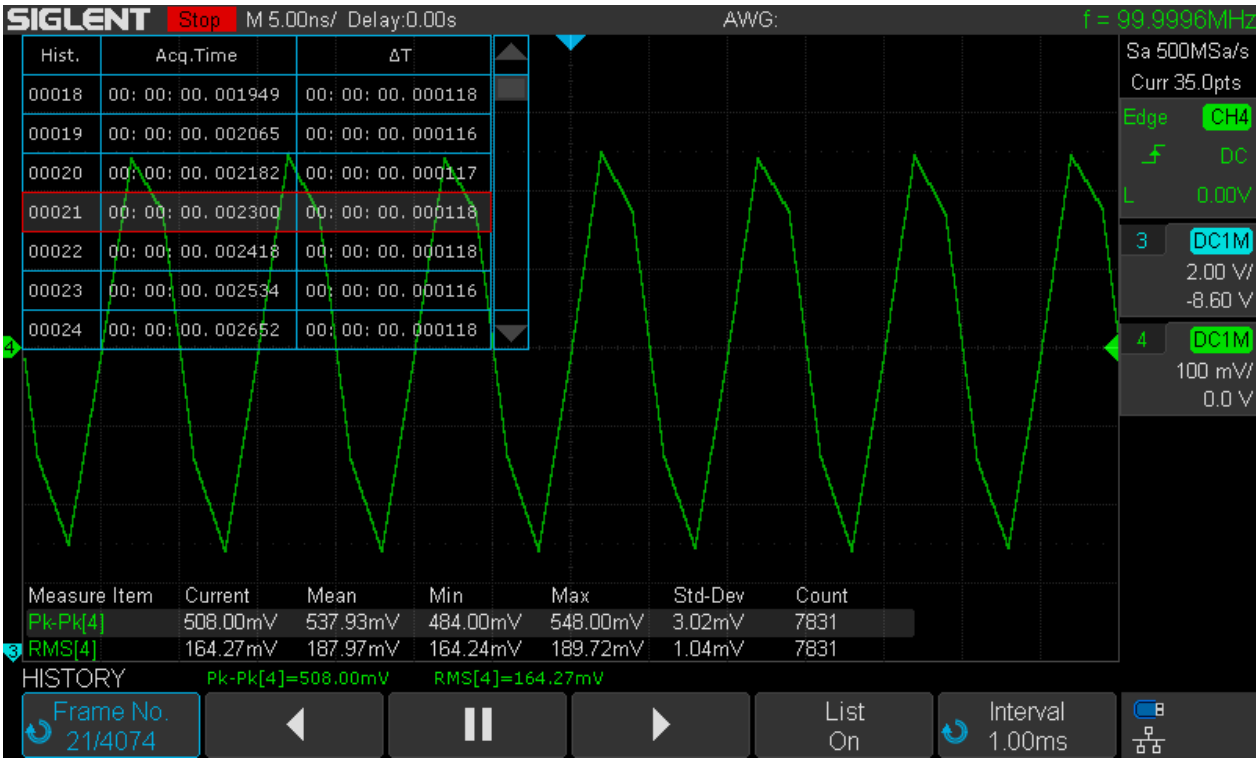
Hist_100MHz_vectors_sinx_Playback



Hist_100MHz_vectors_sinx_80000

Of course we can also look at a single acquisition record (frame) when playback is stopped and with the sin(x)/x reconstruction it looks fine too, as can be seen in the screenshot above.

When viewing the history, we can turn on a list that shows every single record stored in the history, together with a timestamp and a delta time relative to the previous frame. The timestamp format is hh:mm:ss.μμμμμμ, where h=hours, m=minutes, s=seconds, μ=microseconds.



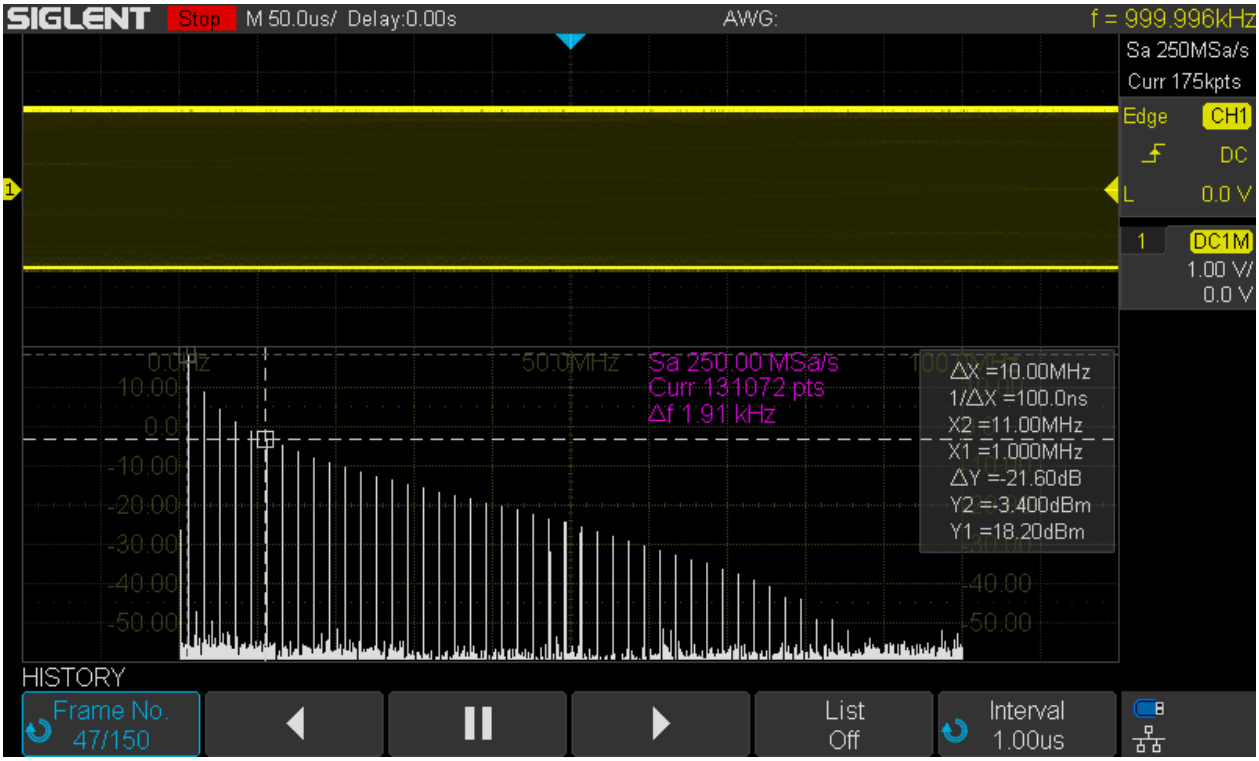
Hist_100MHz_vectors_x_list

We normally use the universal select knob for scrolling through the history, which becomes tedious very quickly with long lists like this. But we have still three options left to speed things up:

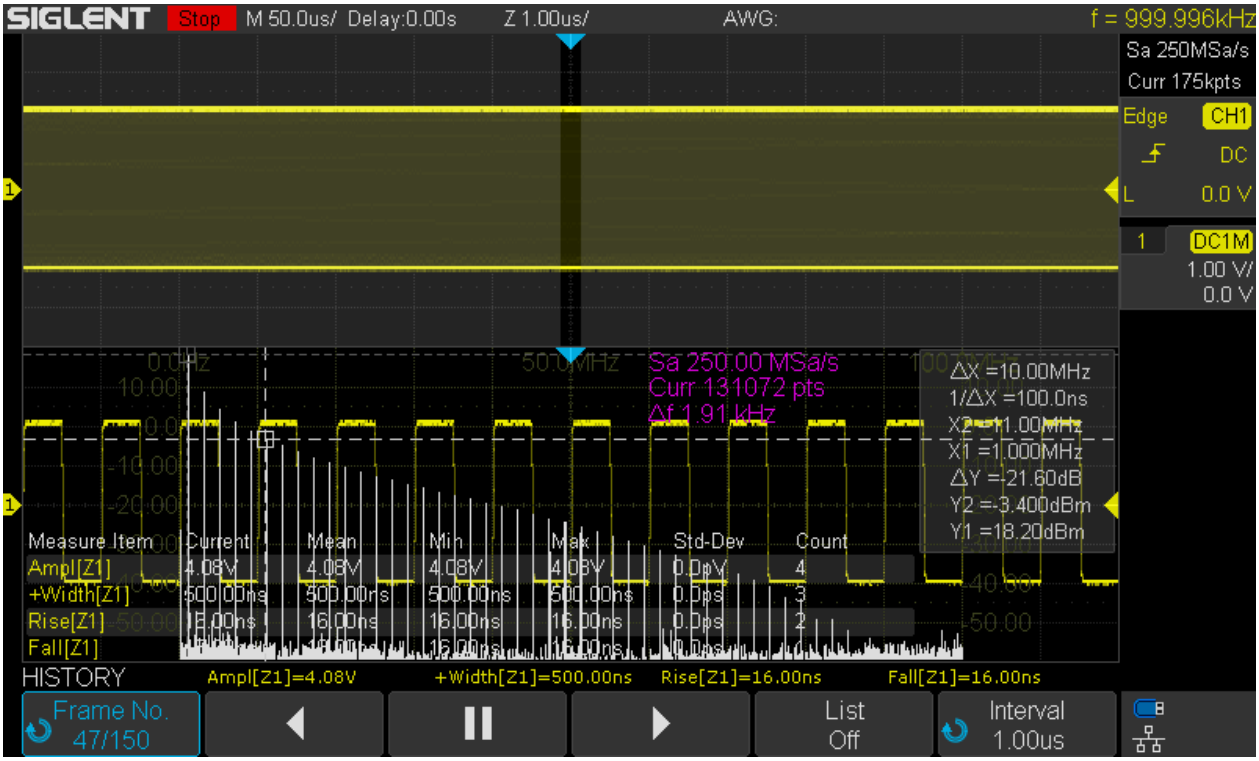
1. Push the universal select knob to display a dialog where we can jump to a certain record instantly by entering the desired frame number.
2. Use the playback function at an appropriate frame rate to emulate a fast forward or rewind like on a tape recorder.
3. Bring up the *Navigation* menu by pressing the [**Navigate**] button on the front panel. When selecting “History Frame” as the navigation type, we can fast forward or rewind as well, this time at three pre-defined discrete speeds that are simply selected by pressing the back or forward direction buttons multiple times.

The real beauty of history is that we can use almost all available tools to analyze any individual frame. In the following example, there are 150 history frames of a 10MHz square wave. We can pick any frame – No. 47 for example – and use math functions like FFT and cursor measurements.

We can go even further and enter zoom mode within the history and add automatic measurements – even though with all these tools active at the same time, the screen looks rather busy.



Hist_Square_10MHz_FFT_Cursors

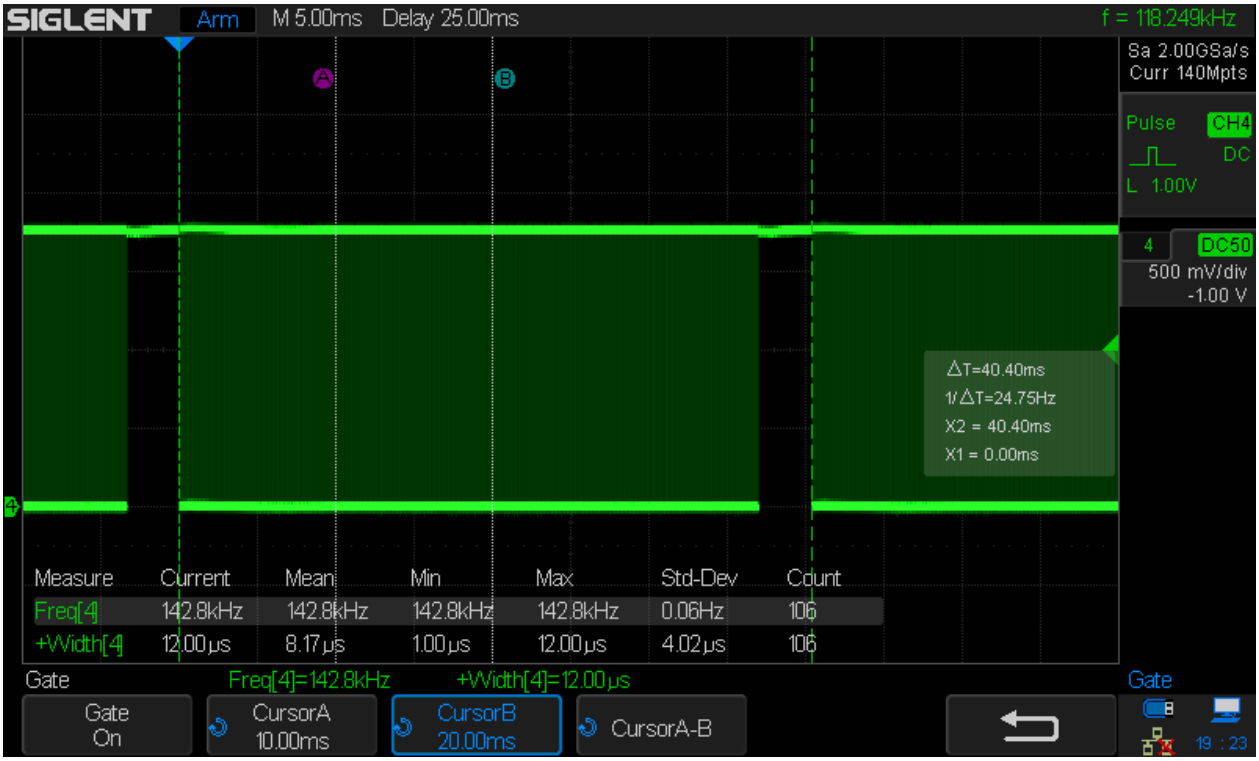


Hist_Square_10MHz_Zoom_FFT_Cursors_Meas

Sequence Mode

Sequence mode is located in the *Acquisition* menu and it is different to the ordinary acquisition modes in that it captures the specified number of segments (=records) as fast as possible and then displays all the data at once. In normal mode, data is displayed at the screen refresh rate, which is approximately 25Hz at 50ns/div. So we can still see all the acquired data in sequence mode, just at a much slower screen update (but much higher acquisition) rate.

Let's have a look at the trigger output of the SDS1104X-E with another DSO (Siglent SDS2304X):



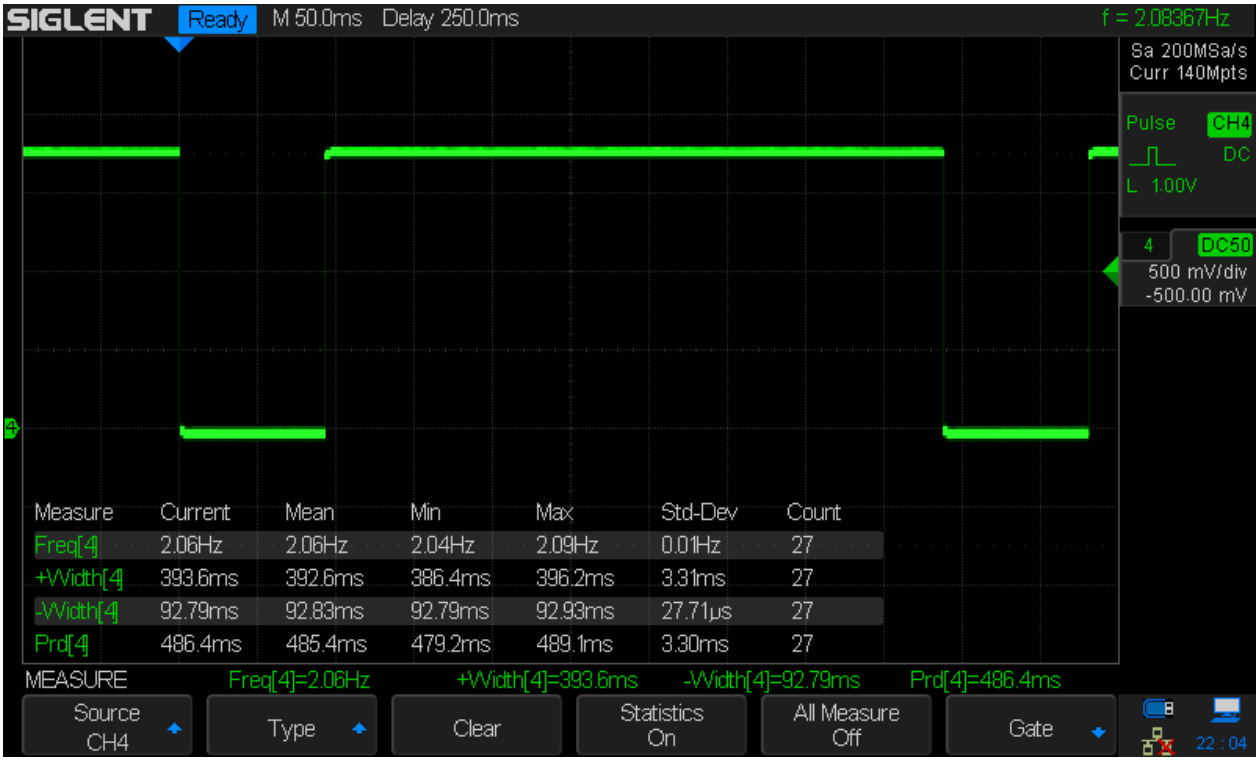
SDS1104X-E_TRIG_Std_50ns

For normal acquisition at 50ns/div timebase the waveform update rate is about 118kWfms/s (yes, this has been improved with the 7.6.1.20 firmware) and the display is updated every 40.40ms, which corresponds to a screen update rate of 24.75Hz.

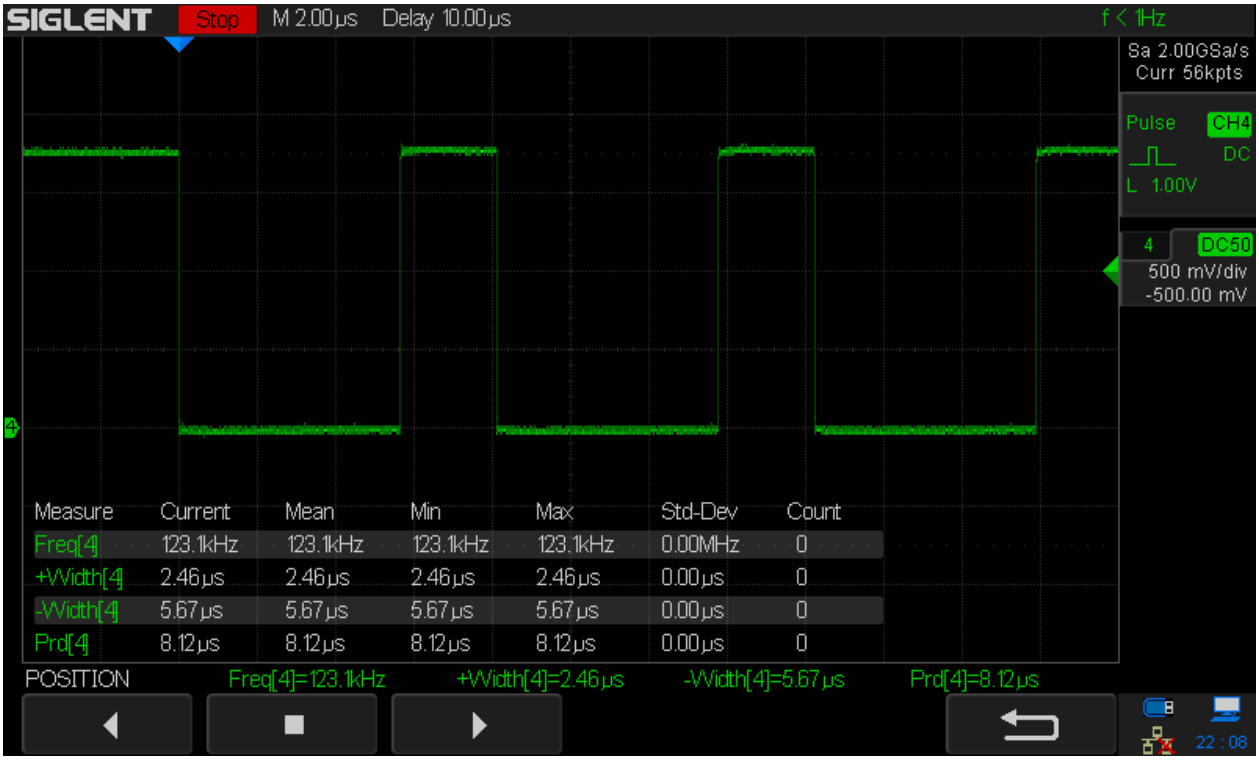
Now let's have a look at sequence mode. The sweet spot setting is with a timebase of 50ns/div and dots display, where the waveform update rate is measured as nearly 488k (depending on the input signal, it can be even slightly faster) and a screen update rate of ~2Hz, which is still quick enough to give a live-view impression. With other settings, screen refresh can be much slower – and of course it depends on the number of captured segments, which can be limited in the SEQUENCE menu.

The screenshot below shows the trigger output signal – and it looks odd, because instead of a burst of trigger pulses we just see a static level change during acquisition. This is because Siglent changed the trigger pulse width to some 5.67µs, mainly because of their new application note "Triggering multiple instruments with an Oscilloscope" I guess. Most likely the bench multimeters wouldn't work with narrow pulses…

The 2nd screenshot below shows the new situation at 10ns/div, where the trigger rate is only 123kWfms/s and the individual pulses become visible again. This means, we cannot measure trigger rates >170kWfms/s with a frequency counter or another DSO anymore, but it is still possible to use the timestamps in the history for that purpose.



SDS1104X-E_TRIG_SEQ_50ns



SDS1104X-E_TRIG_Pulse

As stated before, screen update can be very slow in many cases. For instance, the maximum of 29140 segments available at 100ns/div takes some 10 seconds to refresh the screen. Yet we don't miss any events that have been captured, as will be demonstrated in the next screenshot, showing a 5MHz sine wave, frequency modulated with a max. deviation of 500kHz by a 400Hz sine:

SDS1104X-E_SEQ_100ns_5M_FM500k400_Display

The screen shows all 29140 acquisitions on top of each other at the same time, so we can see the entire captured data, even with intensity grading, and might enter history to examine every single record.

Let's do some blind time calculations for sequence mode. As calculated earlier in this document, the regular mode has a trigger rate of some 19.1kWfms/s and 97.32% blind time for the settings used in this scenario.

In sequence mode we get some 363.4k waveforms per second at 100ns/div. With 14 horizontal divisions, a single waveform equals 1.4µs and 363.4k waveforms are equivalent to a total acquisition time of 363400 x 1.4µs = 508760µs = 508.76ms. For each second, we get 508.76ms worth of actual sample data. This is equivalent to 50.88% of the total time, resulting in 49.12% blind time. This is vastly better than the 97.32% we got with regular acquisition.

Even though this might suggest that sequence mode is a great glitch hunting tool, its main purpose is capturing a high number of infrequent events. In the example above, we could have captured up to 29140 events, each with a maximum duration of 1.4µs, that might occur only once a second. If we tried to capture that as one single acquisition, we'd need some 30 Tpts (terapoints!) of acquisition memory, whereas in sequence mode it is just about 40.8Mpts (29140 segments of 1400 points each). This is by the way just another example where this scope utilizes much more memory than advertised; at 5µs/div it can be up to 55Mpts and since this is just for one channel group, we could say the SDS1104X-E (which has two groups) can use up to 110Mpts of total memory in situations like this.

The table below shows the record length, max. segments, the corresponding waveform update rate, blind time, trigger re-arm time, screen refresh time, sustained waveform update rate and total memory consumption in sequence mode for timebase settings up to 10µs/div.

Mem. Lim. [Pts]	14000000	Max. Seg.	80000	SR [Sa/s]	1,00E+9	20MHz					
Timebase [s]	Memory [Pts]	Seg.	t_1 [µs]	t_max [µs]	Diff [µs]	Rate [Wf/s]	Blind Time [%]	Re-arm Time [s]	Refresh [s]	Avg. Rate [Wf/s]	Total Mem. [Pts]
1,00E-9	14,00E+0	80000	0	5711669	5,71E+6	14,01E+3	99,98%	71,38E-6	31,600	2,5E+3	1,1E+6
2,00E-9	28,00E+0	80000	0	2895831	2,90E+6	27,63E+3	99,92%	36,17E-6	26,600	3,0E+3	2,2E+6
5,00E-9	70,00E+0	80000	0	647354	647,35E+3	123,58E+3	99,13%	8,02E-6	4,160	19,2E+3	5,6E+6
10,00E-9	140,00E+0	80000	0	651994	651,99E+3	122,70E+3	98,28%	8,01E-6	22,640	3,5E+3	11,2E+6
20,00E-9	280,00E+0	70894	0	336743	336,74E+3	210,53E+3	94,11%	4,47E-6	19,720	3,6E+3	19,9E+6
50,00E-9	700,00E+0	45526	0	93327	93,33E+3	487,81E+3	65,85%	1,35E-6	0,486	93,7E+3	31,9E+6
100,00E-9	1,40E+3	29140	1	80189	80,19E+3	363,40E+3	49,12%	1,35E-6	9,640	3,0E+3	40,8E+6
200,00E-9	2,80E+3	16943	1	71147	71,15E+3	238,14E+3	33,32%	1,40E-6	8,940	1,9E+3	47,4E+6
500,00E-9	7,00E+3	7574	3	63112	63,11E+3	120,01E+3	15,99%	1,33E-6	7,340	1,0E+3	53,0E+6
1,00E-6	14,00E+3	3912	7	60229	60,22E+3	64,96E+3	9,06%	1,39E-6	6,320	619,0E+0	54,8E+6
2,00E-6	28,00E+3	1962	14	58463	58,45E+3	33,57E+3	6,01%	1,79E-6	4,380	447,9E+0	54,9E+6
5,00E-6	70,00E+3	785	36	57503	57,47E+3	13,66E+3	4,38%	3,21E-6	1,900	413,2E+0	55,0E+6
10,00E-6	140,00E+3	392	72	57029	56,96E+3	6,88E+3	3,65%	5,30E-6	1,050	373,3E+0	54,9E+6

Sequence Mode Table 1

The trigger re-arm times particularly in the range 50ns/div to 2µs/div are absolutely impressive.

Note that the blind time becomes pretty much negligible at slower timebases like 1µs/div and above, even though the waveform update rates might not look all that impressive anymore. Yet they are about as good as it gets as they start approaching the theoretical maximum for the respective timebase.

Another interesting observation is the screen refresh rate with regard to the number of segments that is set up for sequence mode capturing. It has been done exemplarily for the 50ns/div timebase, where screen refresh is fast at the outset. The following table shows the test results. It can be seen, that screen refresh rates >20Hz are possible with trigger rates close to 500kWfms/s during acquisition. The sustained waveform update rate though, i.e. including the processing and display time, is still always lower as in regular mode.

Timebase [s]	Memory [Pts]	Seg.	t_1 [µs]	t_max [µs]	Diff [µs]	Rate [Wf/s]	Blind Time [%]	Re-arm Time [s]	Refresh [s]	Avg. Rate [Wf/s]	Total Mem. [Pts]
50,00E-9	700,00E+0	45526	0	92871	92,87E+3	490,21E+3	65,69%	1,34E-6	0,486	93,7E+3	31,9E+6
50,00E-9	700,00E+0	40000	0	81598	81,60E+3	490,21E+3	65,69%	1,34E-6	0,442	90,5E+3	28,0E+6
50,00E-9	700,00E+0	20000	0	40798	40,80E+3	490,22E+3	65,68%	1,34E-6	0,242	82,7E+3	14,0E+6
50,00E-9	700,00E+0	10000	0	20398	20,40E+3	490,24E+3	65,68%	1,34E-6	0,141	70,9E+3	7,0E+6
50,00E-9	700,00E+0	5000	0	10198	10,20E+3	490,29E+3	65,68%	1,34E-6	0,108	46,5E+3	3,5E+6
50,00E-9	700,00E+0	2000	0	4078	4,08E+3	490,44E+3	65,67%	1,34E-6	0,068	29,3E+3	1,4E+6
50,00E-9	700,00E+0	1000	0	2038	2,04E+3	490,68E+3	65,65%	1,34E-6	0,065	15,4E+3	700,0E+3
50,00E-9	700,00E+0	500	0	1018	1,02E+3	491,16E+3	65,62%	1,34E-6	0,065	7,7E+3	350,0E+3
50,00E-9	700,00E+0	200	0	406	406,00E+0	492,61E+3	65,52%	1,33E-6	0,047	4,3E+3	140,0E+3
50,00E-9	700,00E+0	100	0	202	202,00E+0	495,05E+3	65,35%	1,32E-6	0,044	2,3E+3	70,0E+3
50,00E-9	700,00E+0	50	0	100	100,00E+0	500,00E+3	65,00%	1,30E-6	0,044	1,1E+3	35,0E+3
50,00E-9	700,00E+0	20	0	39	39,00E+0	512,82E+3	64,10%	1,25E-6	0,044	451,5E+0	14,0E+3
50,00E-9	700,00E+0	10	0	19	19,00E+0	526,32E+3	63,16%	1,20E-6	0,044	226,2E+0	7,0E+3

Sequence Mode Table 2

This leads to the intriguing question, how the sustained waveform update rate and total blind time in sequence mode compares to regular mode in general. Is there a setting where sequence mode can beat the regular mode just at the expense of a slow screen refresh rate? The following table provides the answer.

Siglent SDS1104X-E Sustained Trigger Rate and Blind Time				
Timebase	Standard		Sequence	
[s]	Wfm/s	BT [%]	Wfm/s	BT [%]
1,00E-9	6,09E+3	99,991%	2,53E+3	99,996%
2,00E-9	9,84E+3	99,972%	3,01E+3	99,992%
5,00E-9	34,22E+3	99,760%	19,23E+3	99,865%
10,00E-9	12,89E+3	99,820%	3,53E+3	99,951%
20,00E-9	13,43E+3	99,624%	3,60E+3	99,899%
50,00E-9	107,69E+3	92,461%	93,67E+3	93,443%
100,00E-9	19,12E+3	97,324%	3,02E+3	99,577%
200,00E-9	13,36E+3	96,259%	1,90E+3	99,469%
500,00E-9	8,88E+3	93,788%	1,03E+3	99,278%
1,00E-6	7,29E+3	89,797%	618,99E+0	99,133%
2,00E-6	5,08E+3	85,770%	447,95E+0	98,746%
5,00E-6	2,28E+3	84,040%	413,16E+0	97,108%
10,00E-6	1,29E+3	81,954%	373,33E+0	94,773%

Sustained Trigger Rate Comparison

And the answer is … no. Except for the sweet spot at 50ns/div, where it comes pretty close, the sustained waveform update rate as well as the total blind time of sequence mode cannot compete with regular mode by quite a margin. Consequentially, sequence mode is not the ultimate glitch hunting tool for rare and completely random events, but is excellent for collecting infrequent events over a long time – which might include glitches. Consider a serial message, 100µs long, sent only once every second, that gets corrupted occasionally. Then we can trigger on the start of that message, use a 10µs/div timebase to capture a total of 140µs and will be able to capture up to 392 messages. When the error occurs we just stop the acquisition and inspect the history – and we need not even hurry, as we get plenty time and just need to stop the acquisition within some 6 minutes so the glitch event will not be overwritten. In this scenario, the scope will use close to 55Mpts of memory per channel group.