Siglent SDS1104X-E Review

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

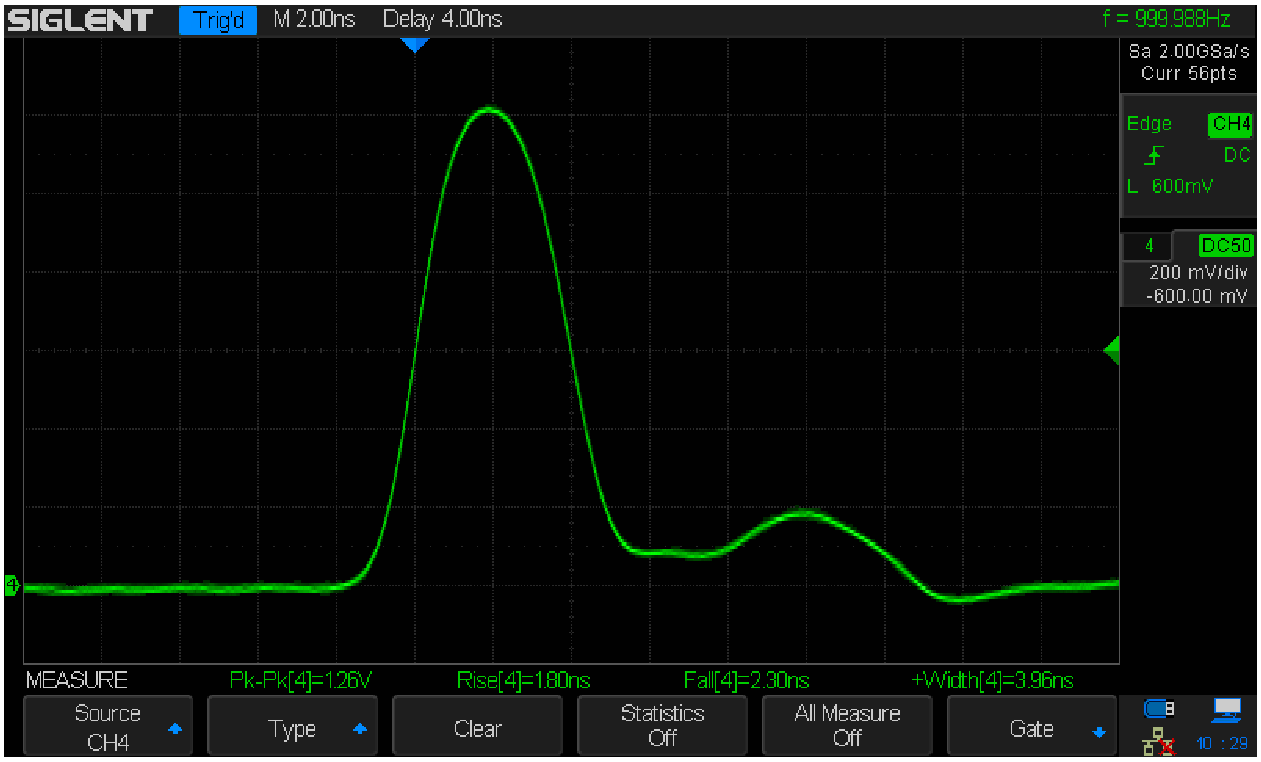
There are 4 dedicated basic acquisition modes, which determine the way captured data from the ADC is preprocessed (and decimated, if necessary). On top of that, roll mode also qualifies as acquisition mode, but could be looked at as an extension to normal and peak detect modes. It also has a dedicated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

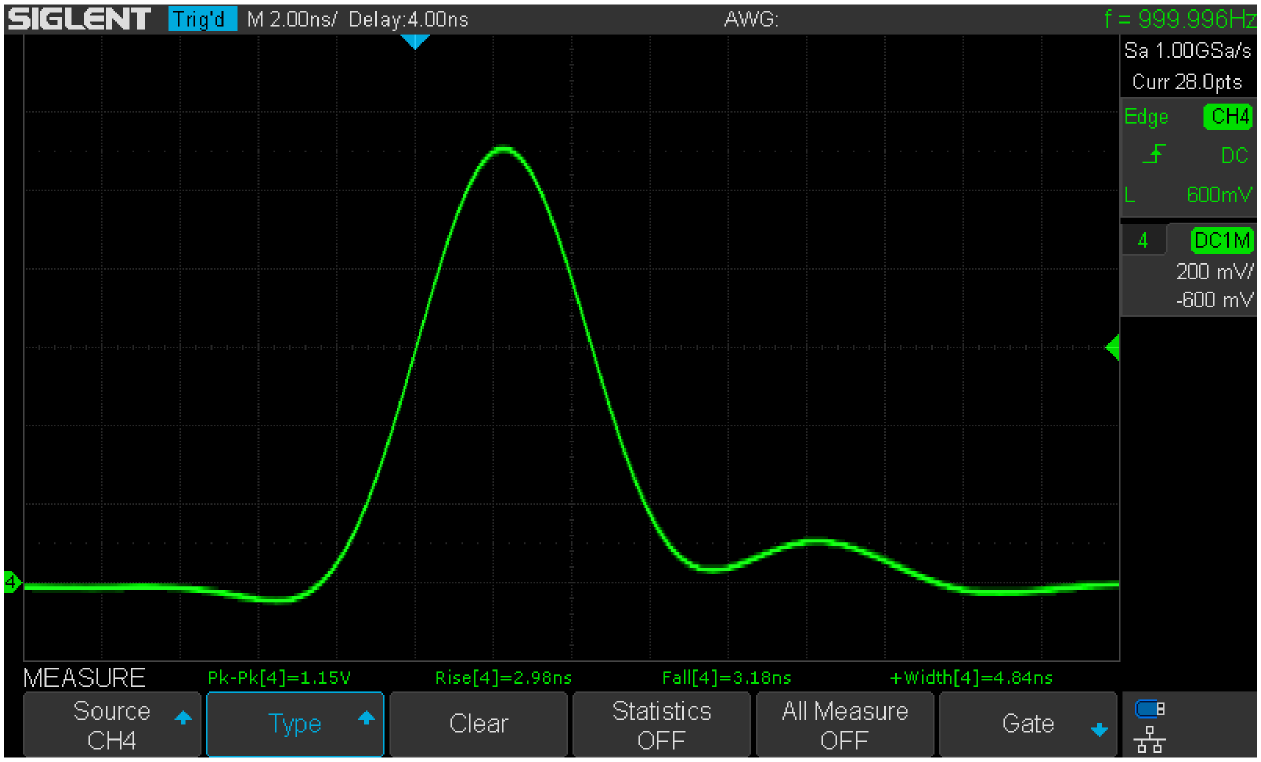


Pk-Pk[4]=1.26V Rise[4]=1.80ns Fall[4]=2.30ns +Width[4]=3.96ns

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:



Pk-Pk[4]=1.15V Rise[4]=2.98ns Fall[4]=3.18ns +Width[4]=4.84ns

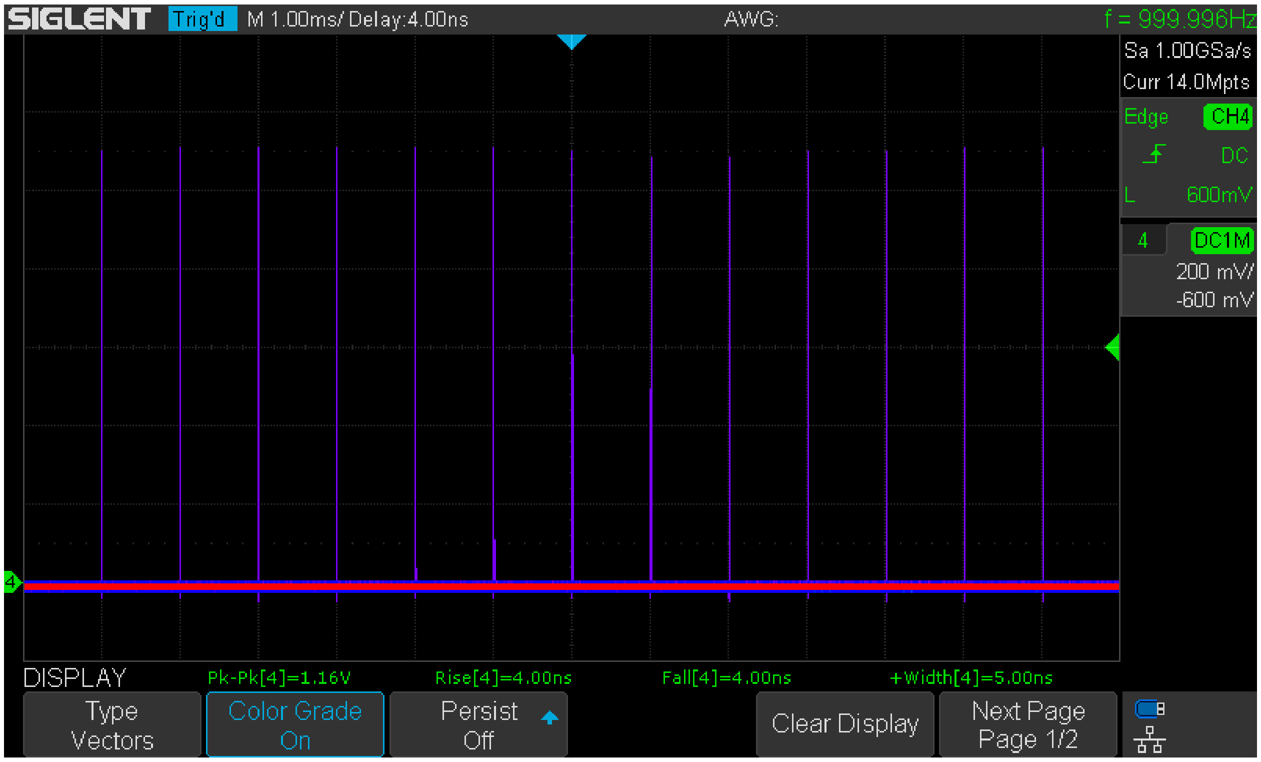
Pulse_1kHz_4ns_BW109MHz

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.

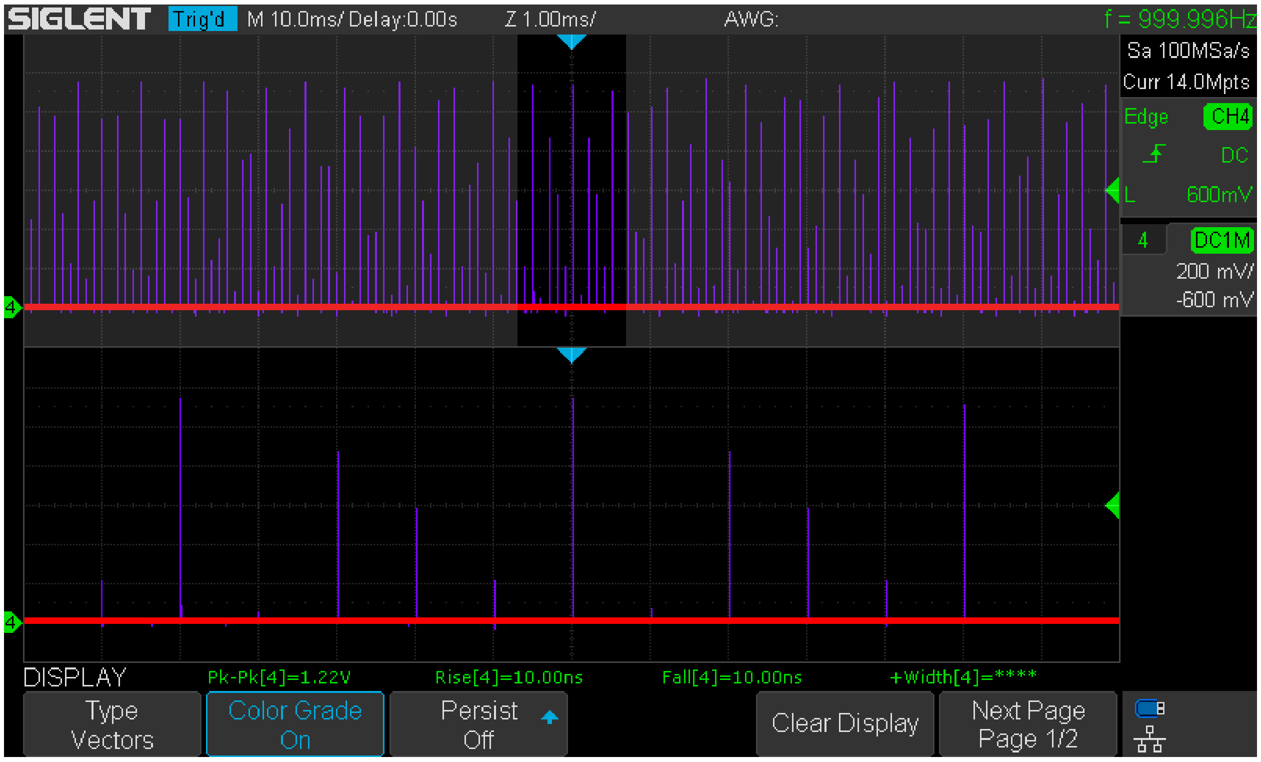
First we look at the pulse train at a timebase of 1ms/div. In this scenario, the full 14Mpts of acquisition memory are utilized and the full sample rate of 1GSa/s is still available. As expected, the pulses are captured without problems. Color graded display is used for all the pulse tests, as this provides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

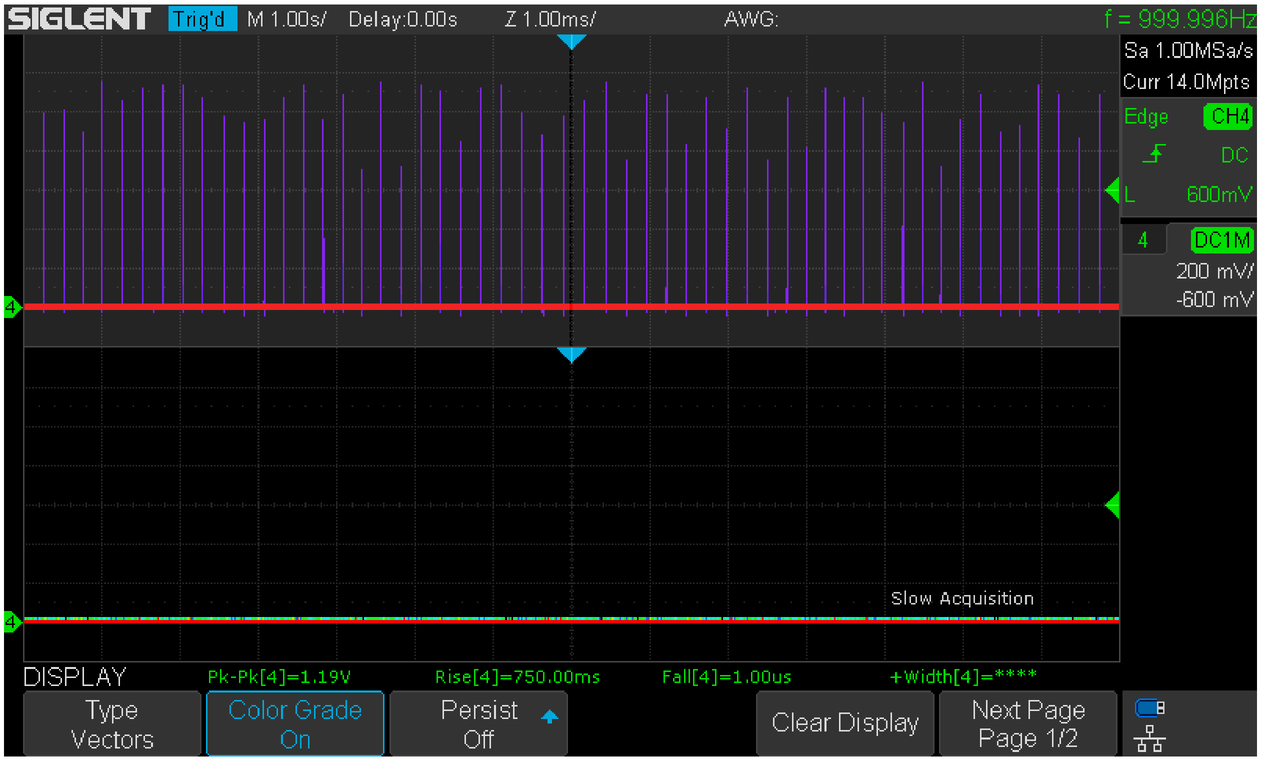
Lowering the timebase to 10ms/div reduces the sample rate to 100MSa/s. A zoom window of 1ms/div is used to provide a direct comparison with the first screenshot.



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

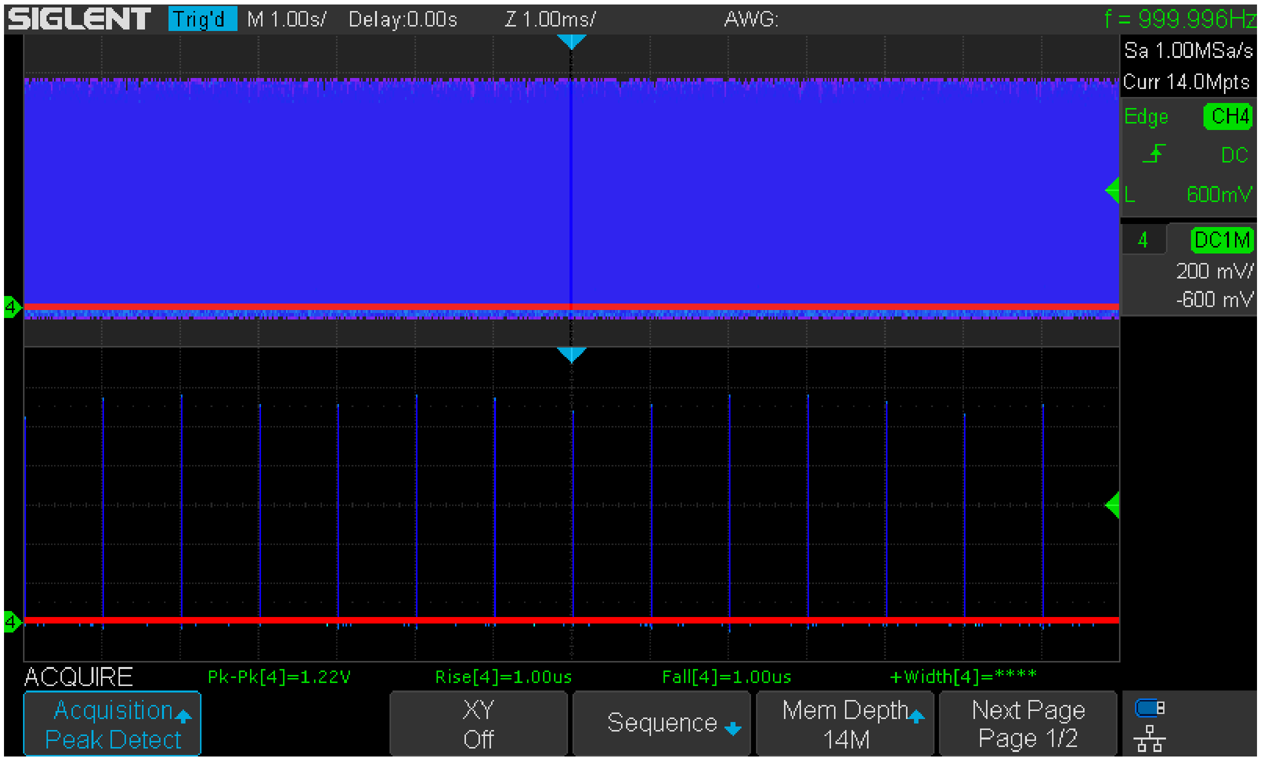
Sample rate drops at slow time bases, which raises the risk of losing important signal details, like spikes and glitches. Peak detect mode uses a different strategy for data decimation: instead of using every n^{th} sample from the ADC stream, one min/max pair out of *2n* samples is stored. This way 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.



Pk-Pk[4]=1.22V Rise[4]=1.00us Fall[4]=1.00us +Width[4]=****

Peak_Pulse_1kHz_4ns_1s

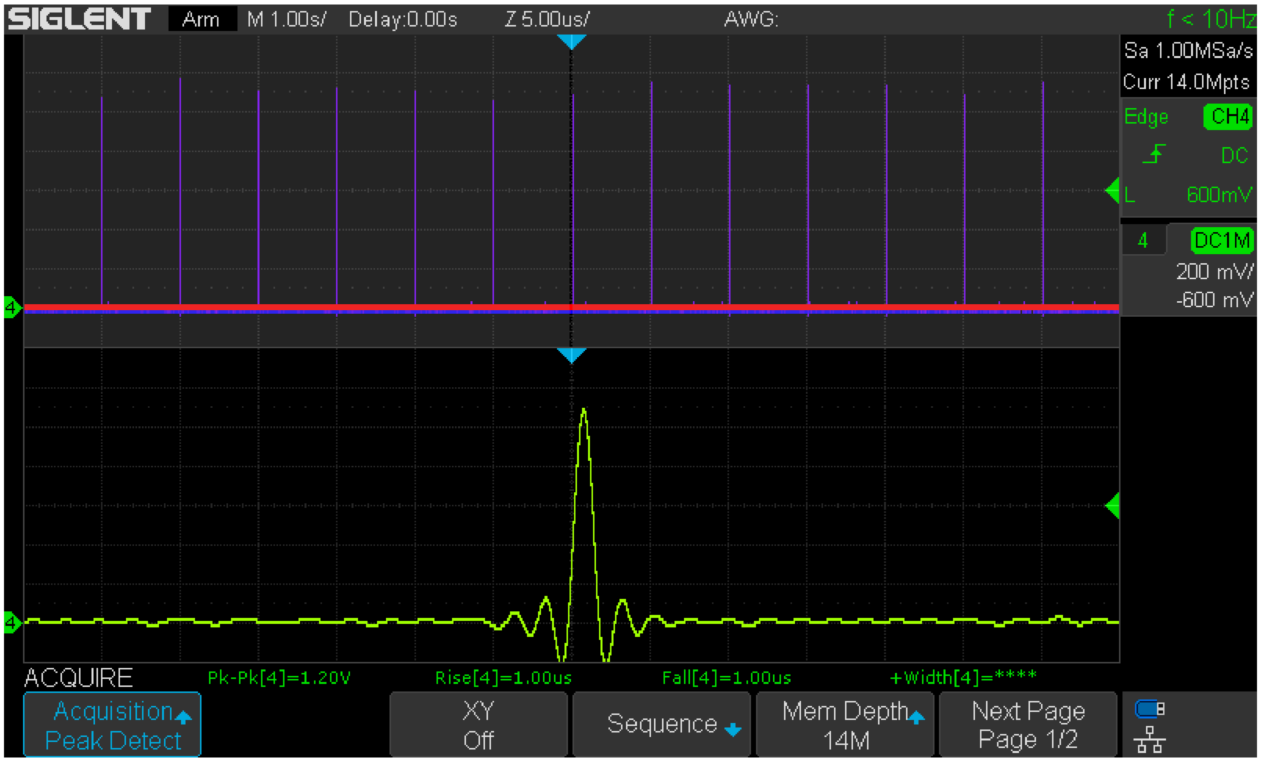
Now this is something completely different. Even at a sample speed of only 1MSa/s, the main window is completely filled with pulses and the zoom window shows that all the pulses are indeed captured. The amplitude measurement is fairly accurate, but transition times are far off and the system cannot determine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

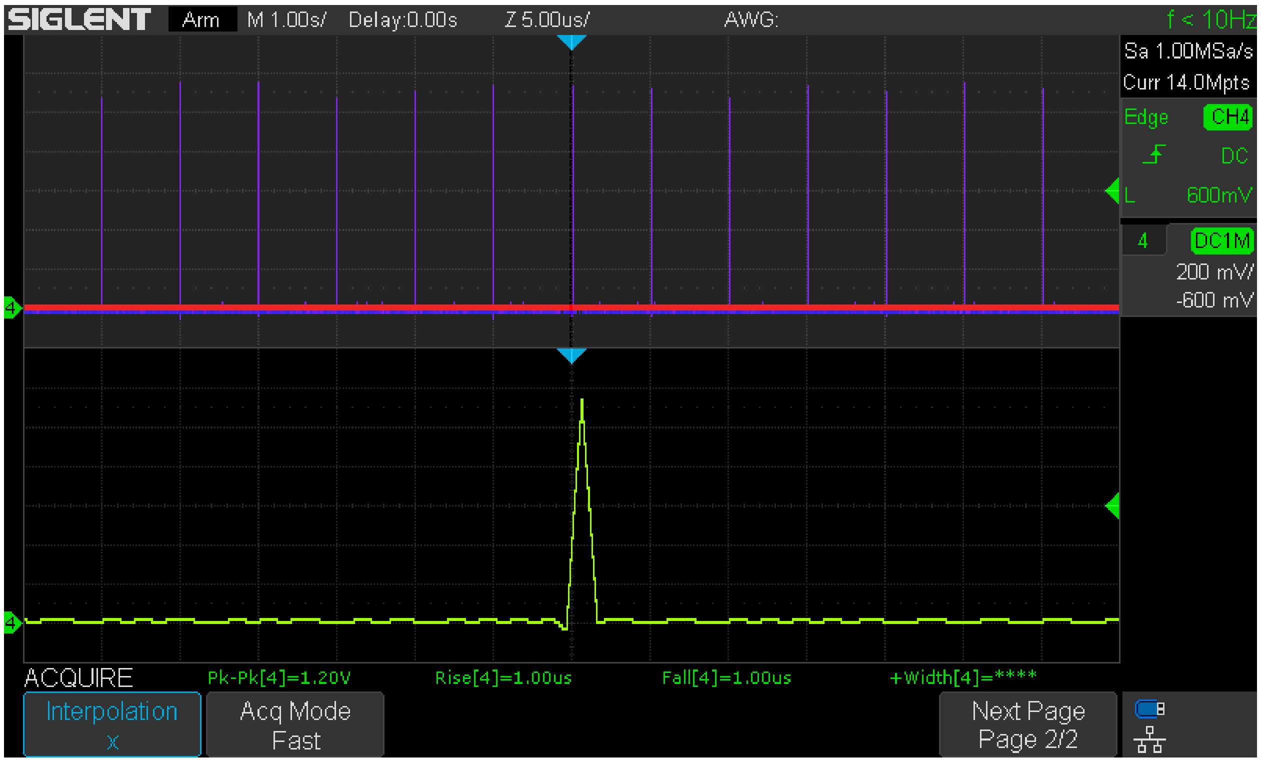
At 5µs/div, we can clearly see the Sinc pulse resulting from the sin(x)/x reconstruction of the severely undersampled and misaligned data produced by the peak detect decimation scheme. According to what we can see in the zoom window, the pulse would indeed have 1µs transition times, and the pulse width 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.

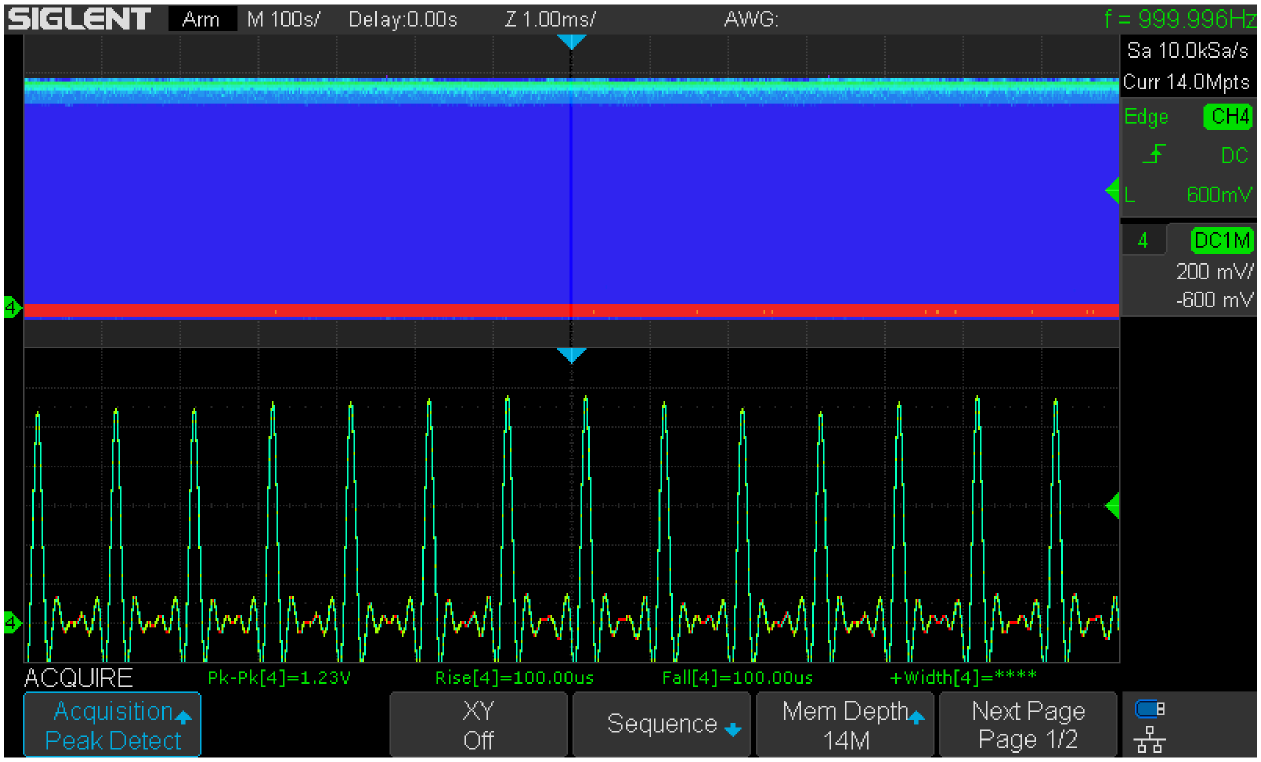


Peak_Pulse_1kHz_4ns_1s_Z5us



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.

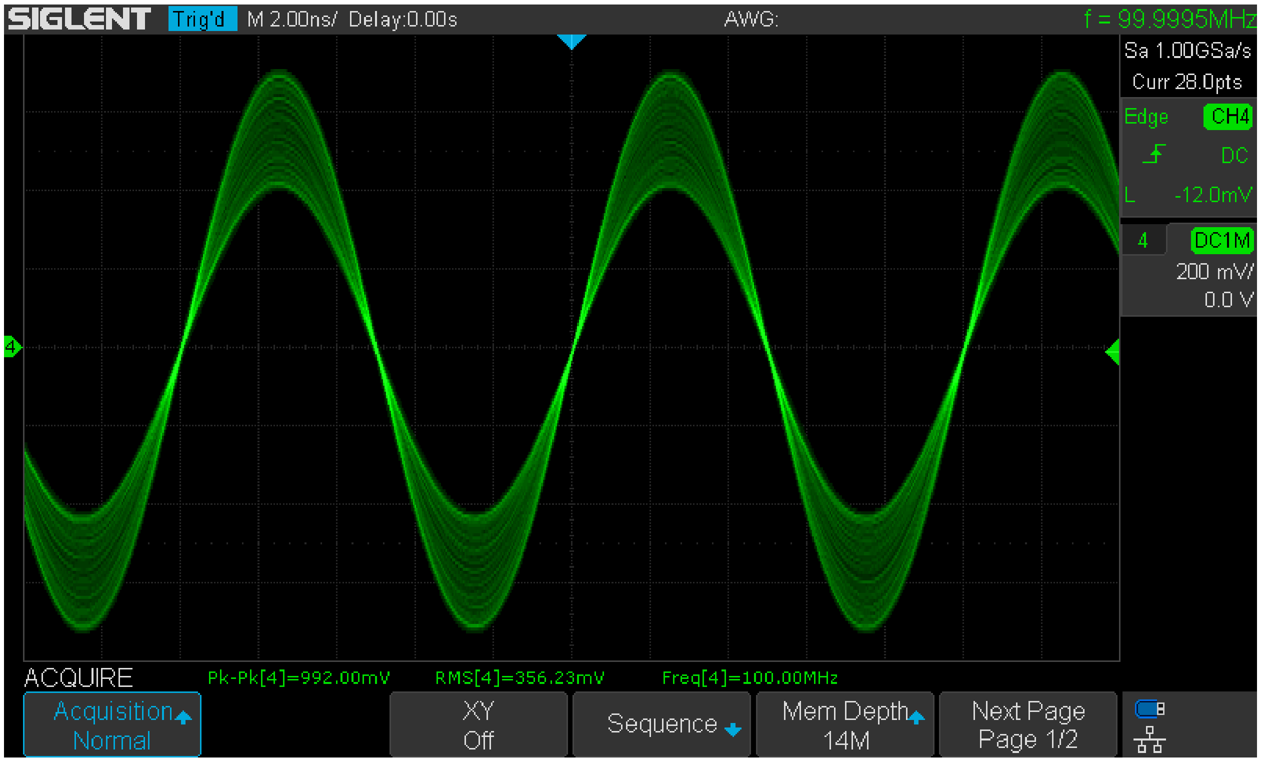
CAUTION: As it just fits the topic, a warning appears in place here. Never ever use peak detect together with FFT mode. Peak detect mode violates a fundamental requirement of digital signal acquisition, it is the evenly spaced samples with regard to the time axis. This produces some artificial signal 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).

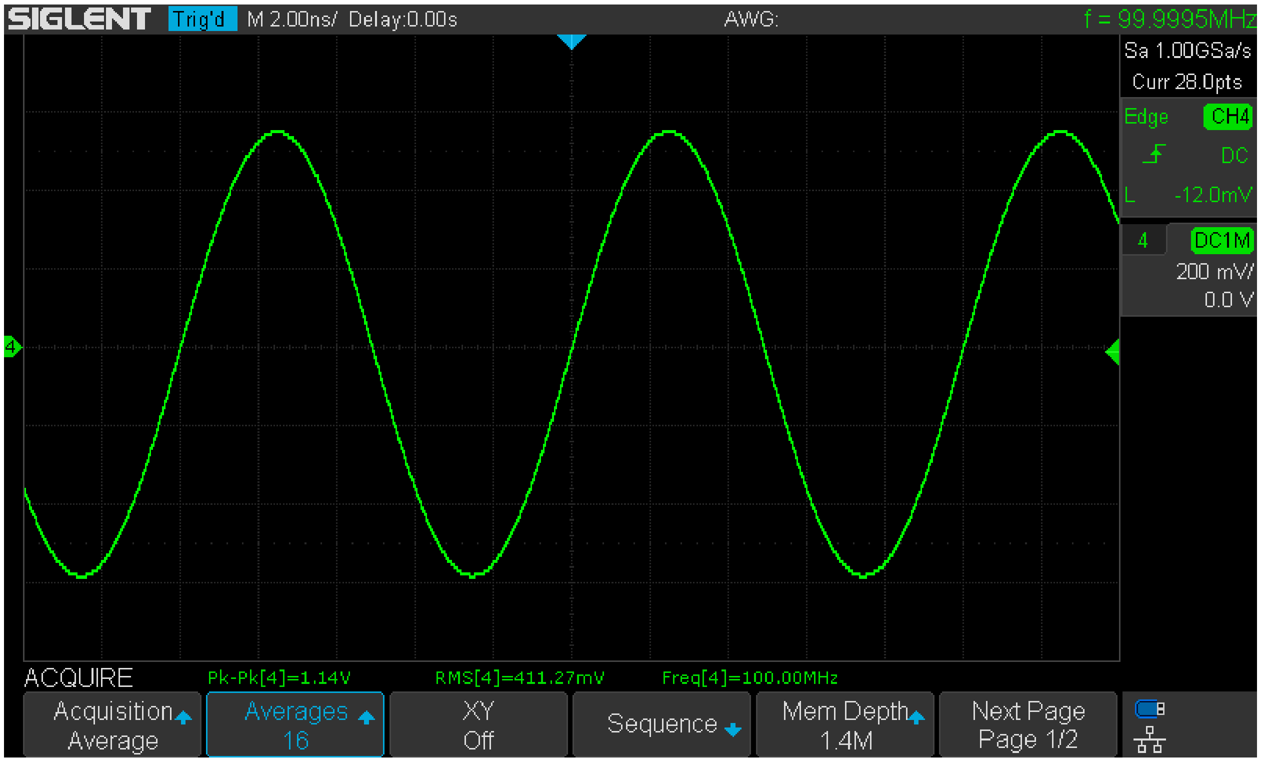
Let's have a look at a 100MHz carrier 25% amplitude modulated with a 400Hz sine in normal acquisition mode, shown in the screenshot below. The amplitude measurements are more or less meaningless, as they are changing all the time at high speed, according to the modulation.
The actual carrier level is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



Pk-Pk[4]=992.00mV RMS[4]=356.23mV Freq[4]=100.00MHz

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.



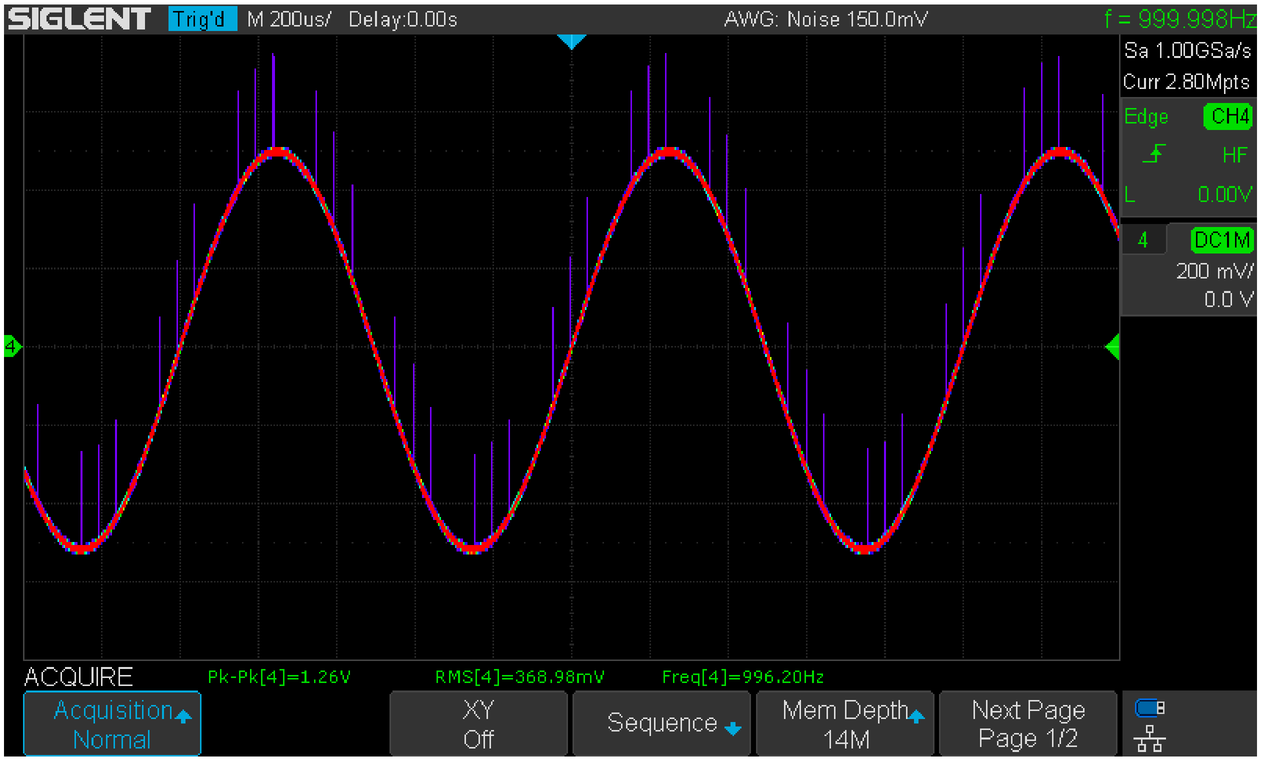
Pk-Pk[4]=1.14V RMS[4]=411.27mV Freq[4]=100.00MHz

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.



Pk-Pk[4]=1.26V RMS[4]=368.98mV Freq[4]=996.20Hz

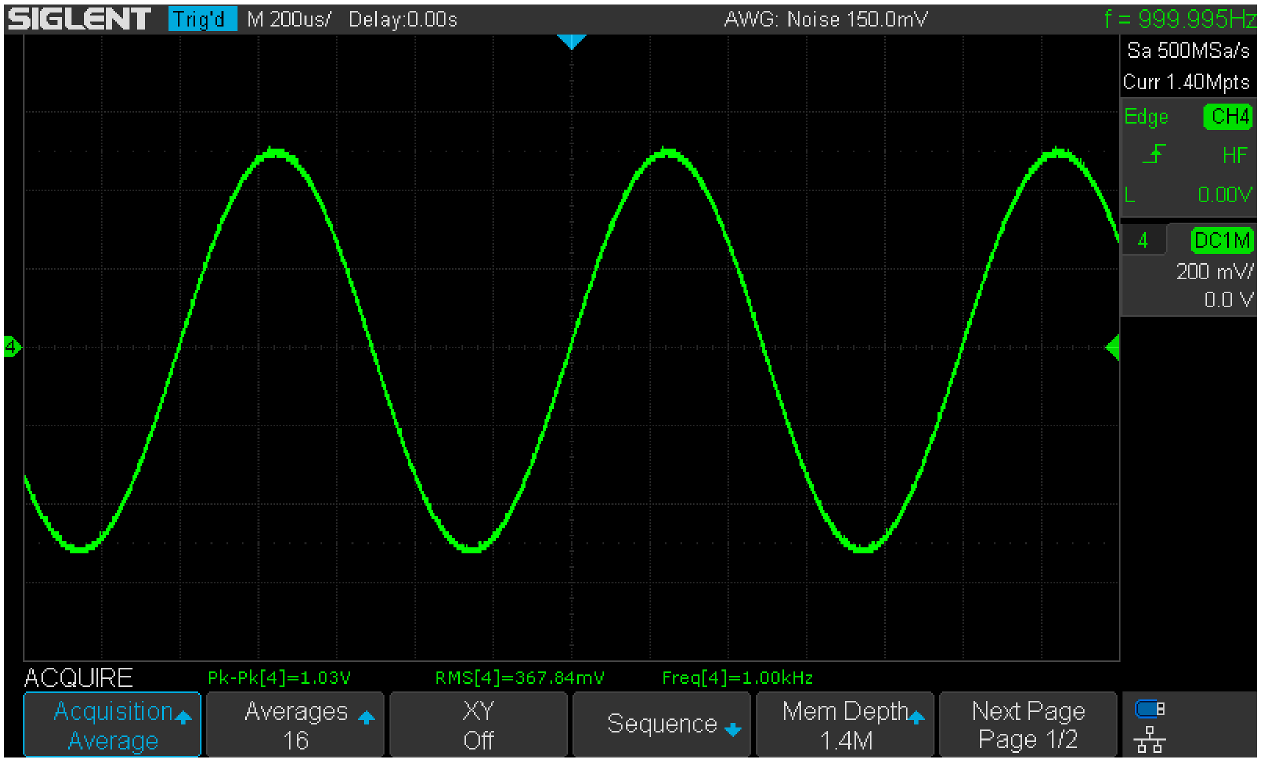
AVG_1kHz_Sp4981Hz_normal

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.

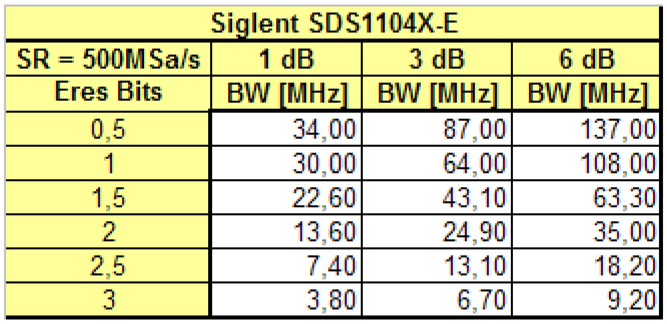
The table below shows the measured bandwidth of the SDS1104X-E at all available Eres bit settings for a sample rate of 1GSa/s in interleaved mode (only one channel per channel group enabled).

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.

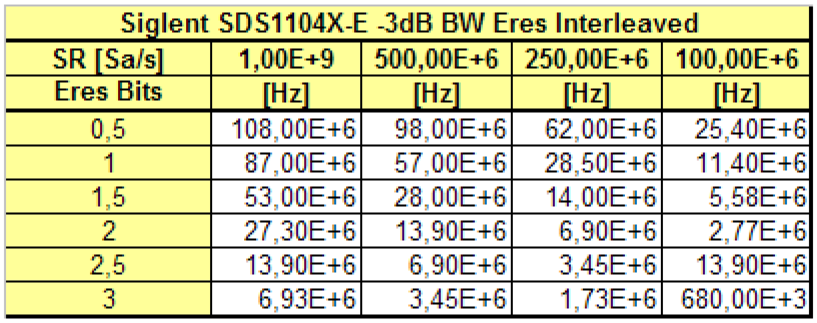
A 2nd series of measurements shows the Eres bandwidth for 500MSa/s in individual mode (both channels in a group enabled), see the table below.



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:

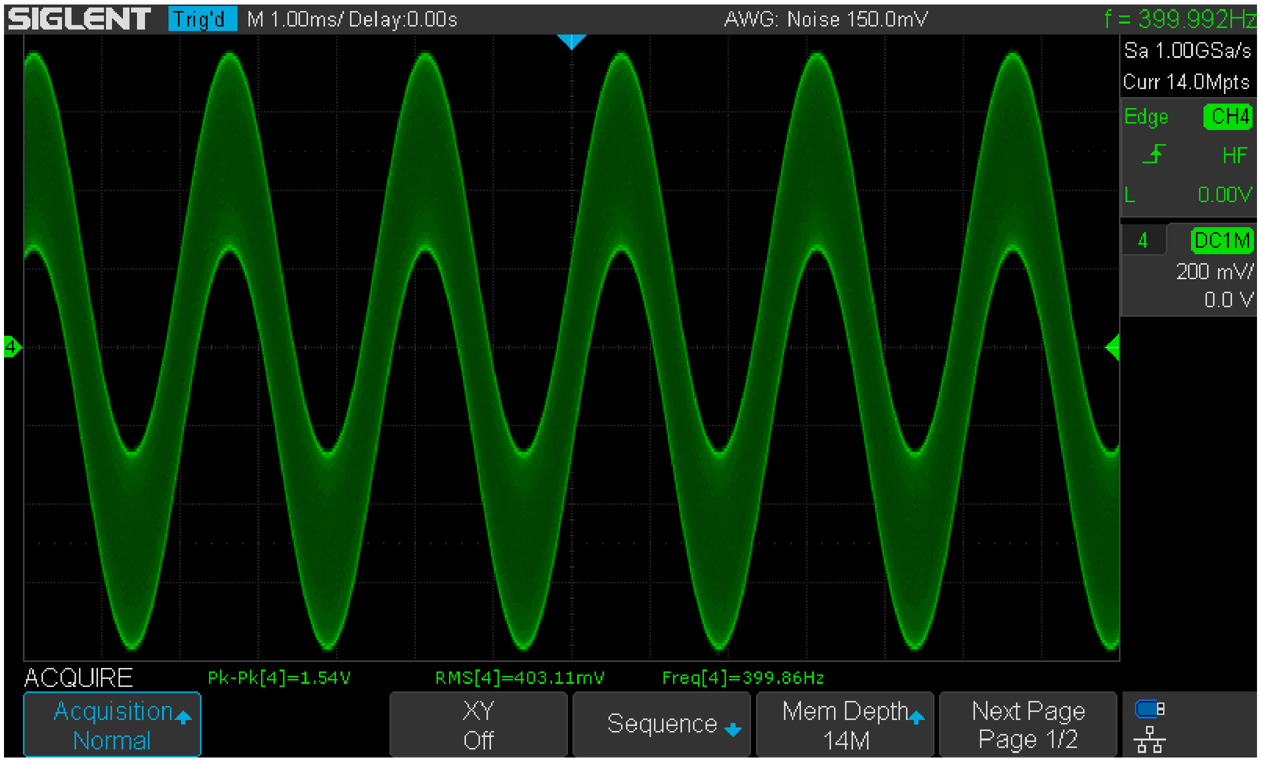
Resolution Enhancement [Bits]	-3dB Bandwidth (x Nyquist]
0.5	0.5
1	0.241
1.5	0.121
2	0.058
2.5	0.029
3	0.016

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).

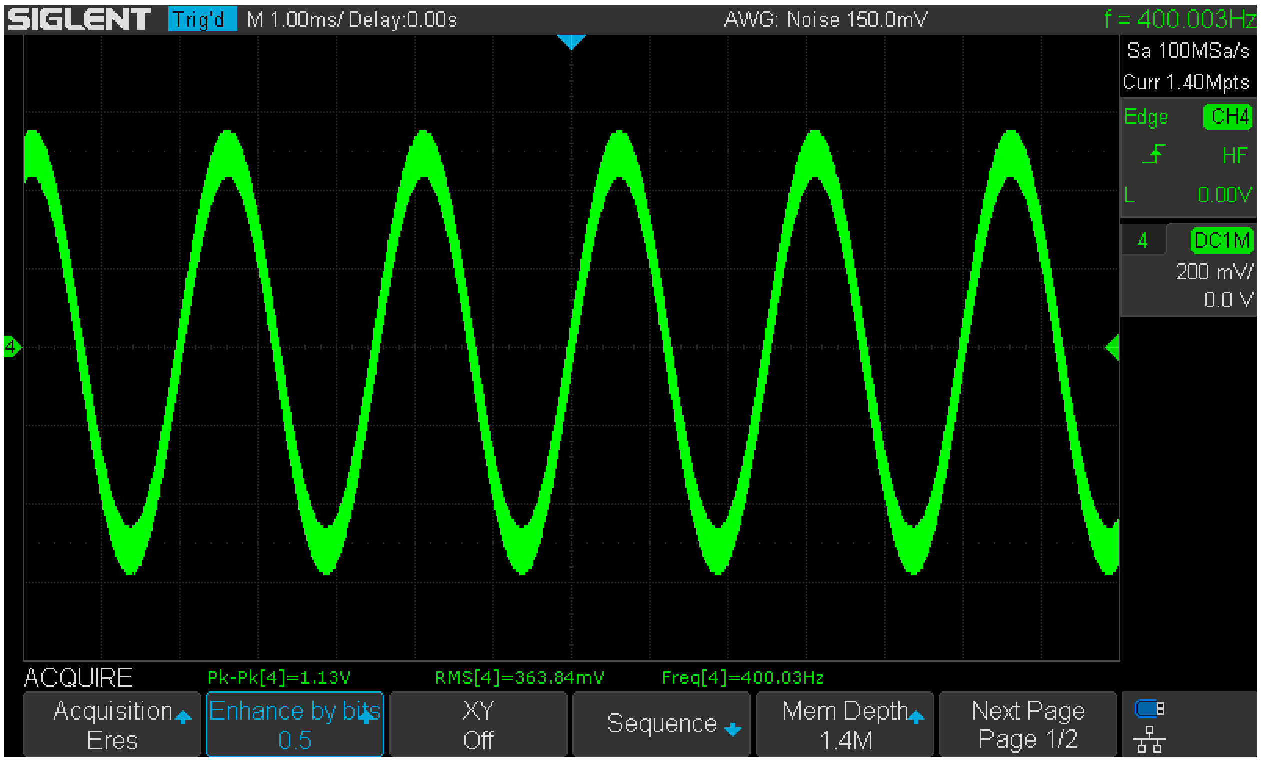
Beware that for the time being, Eres mode provides no true resolution enhancement. This is mainly because of the display interface which is still limited to 8 bits. Siglent have promised to change this at some point in the future, but this is treated as a low priority task, so it may take a while.

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.



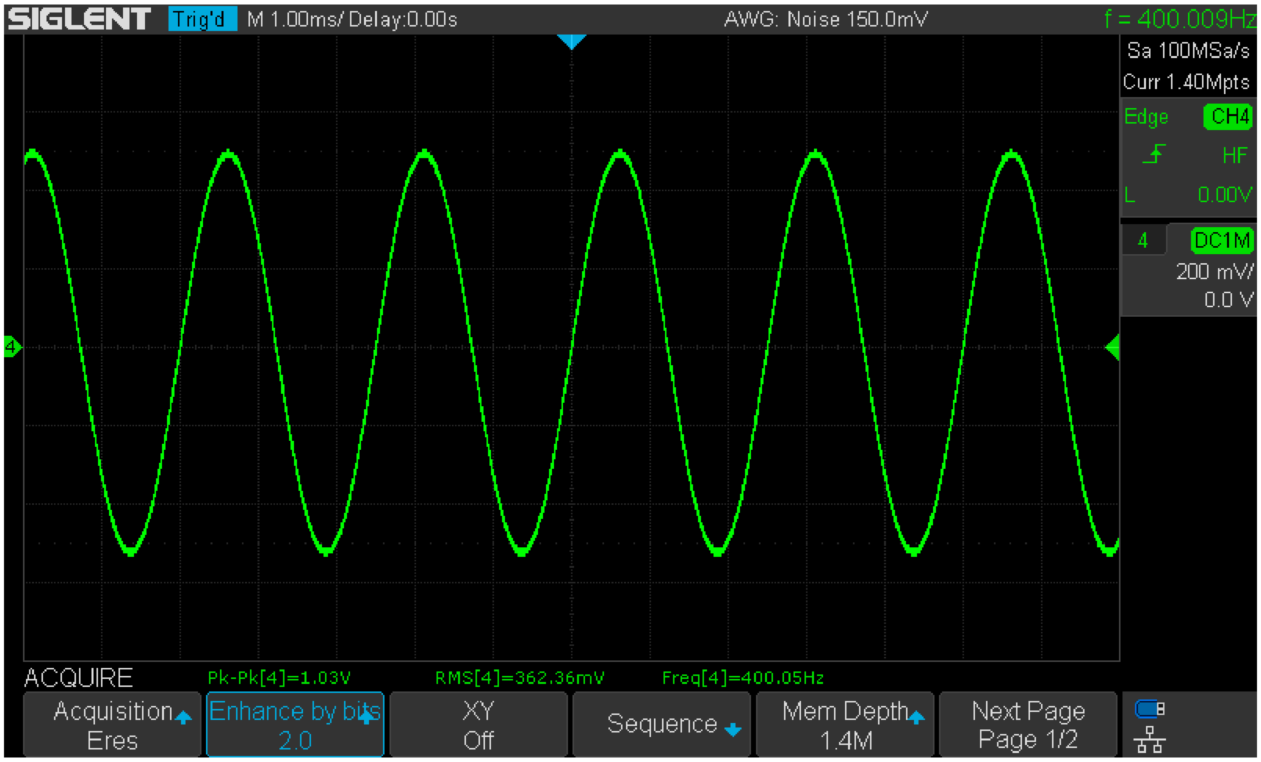
Eres_normal_400Hz_56MHz

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.

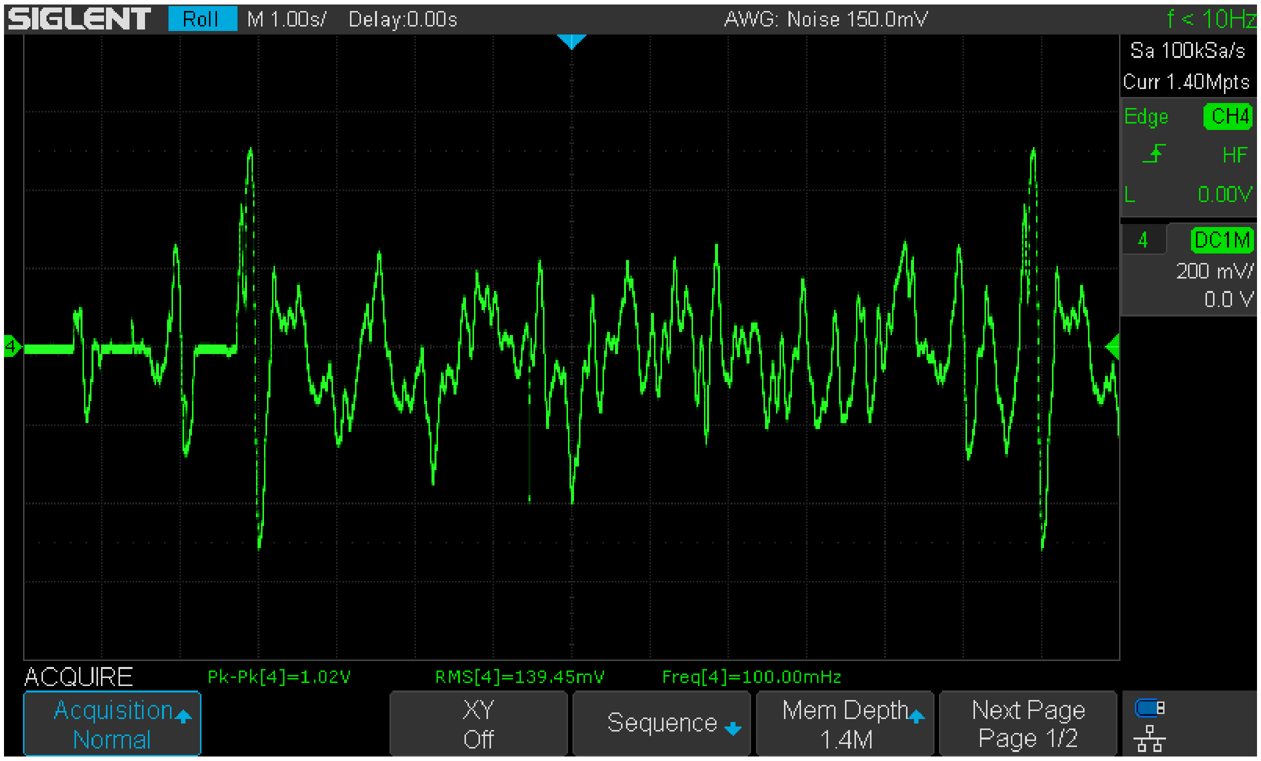
Consequently, roll mode is a non-triggered continuous acquisition in either normal or peak detect mode, similar to a strip chart recorder. The big advantage of this mode is the total lack of blind time, but there are several restrictions to keep in mind:

- 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.

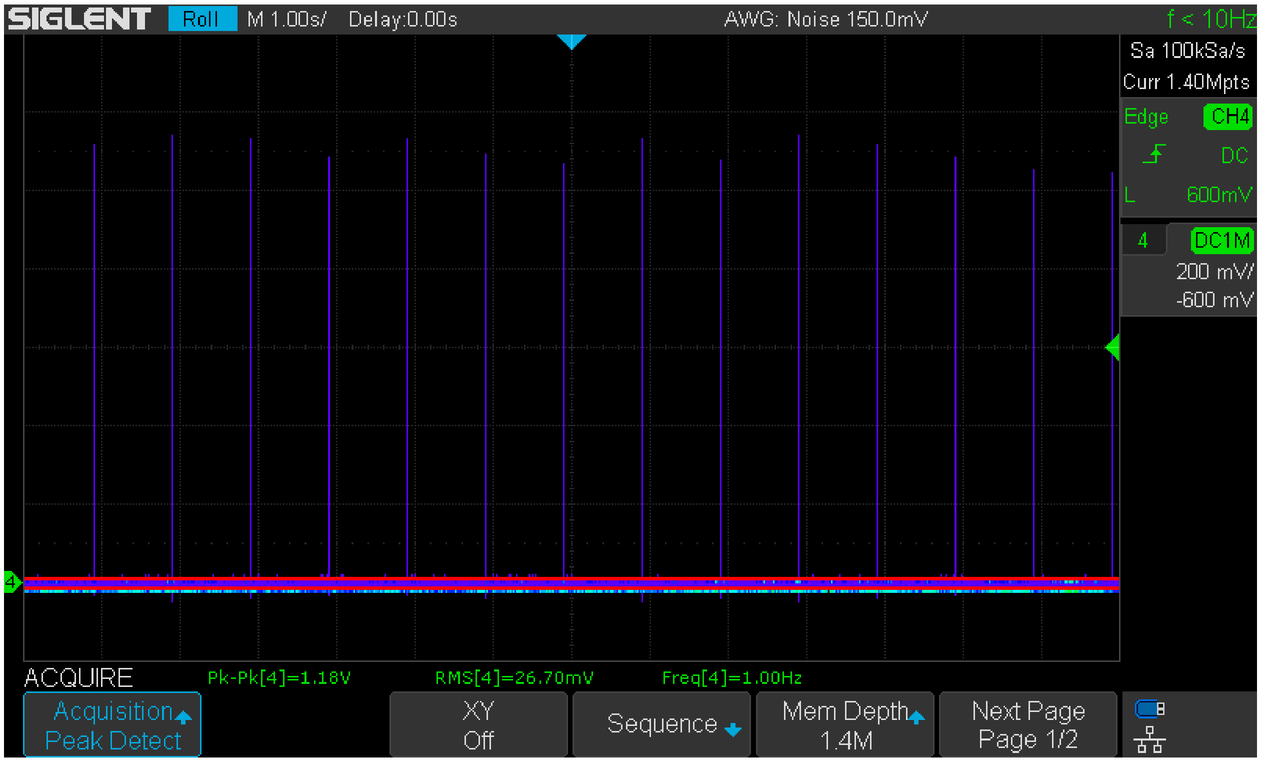
The screenshot below shows an example of normal roll mode at 1s/div.



Pk-Pk[4]=1.02V RMS[4]=139.45mV Freq[4]=100.00mHz

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.



Pk-Pk[4]=1.18V RMS[4]=26.70mV Freq[4]=1.00Hz

Roll_peak

X-Y

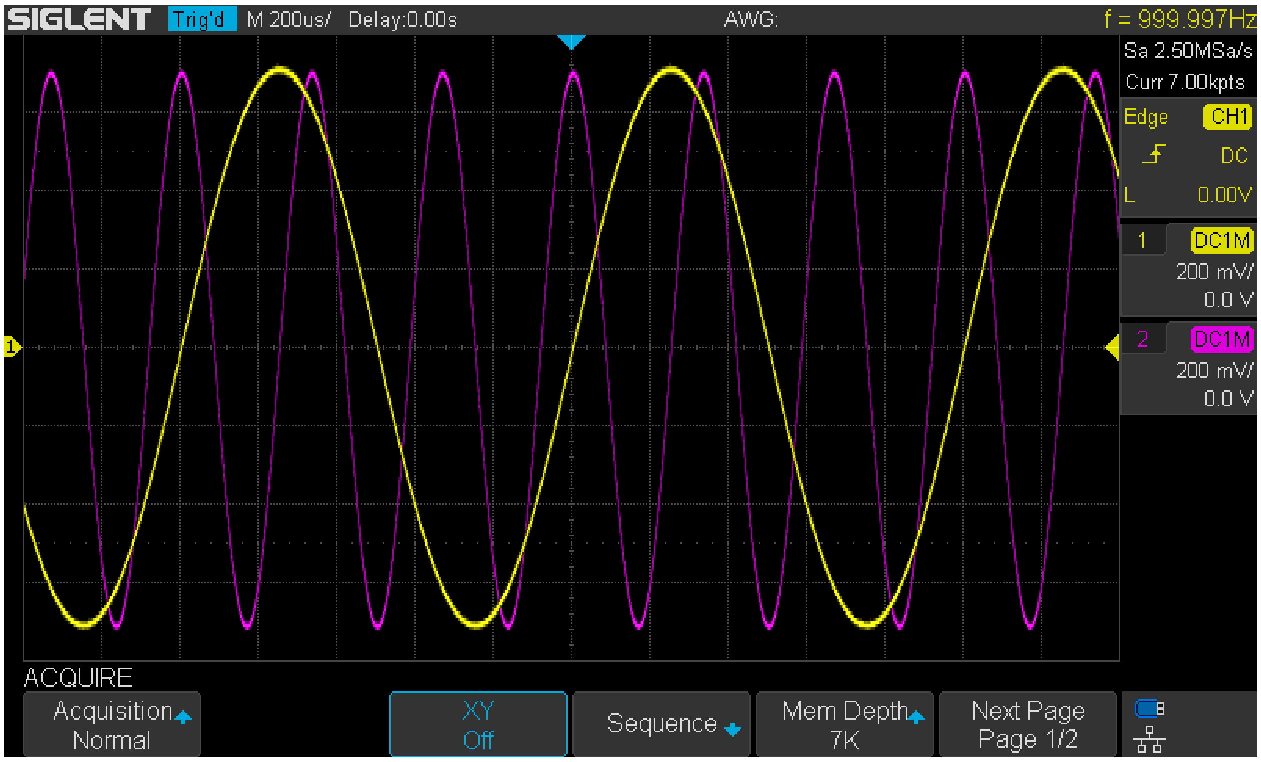
In contrast to analog scopes, X-Y mode on a DSO is quite different and in any case not straight forward. The timebase is still active and important, as the X-Y picture is based on regular Y-t acquisitions of Ch.1 and Ch.2 (and/or Ch.3 and Ch.4, as the SDS1104X-E can show two independent X-Y traces in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

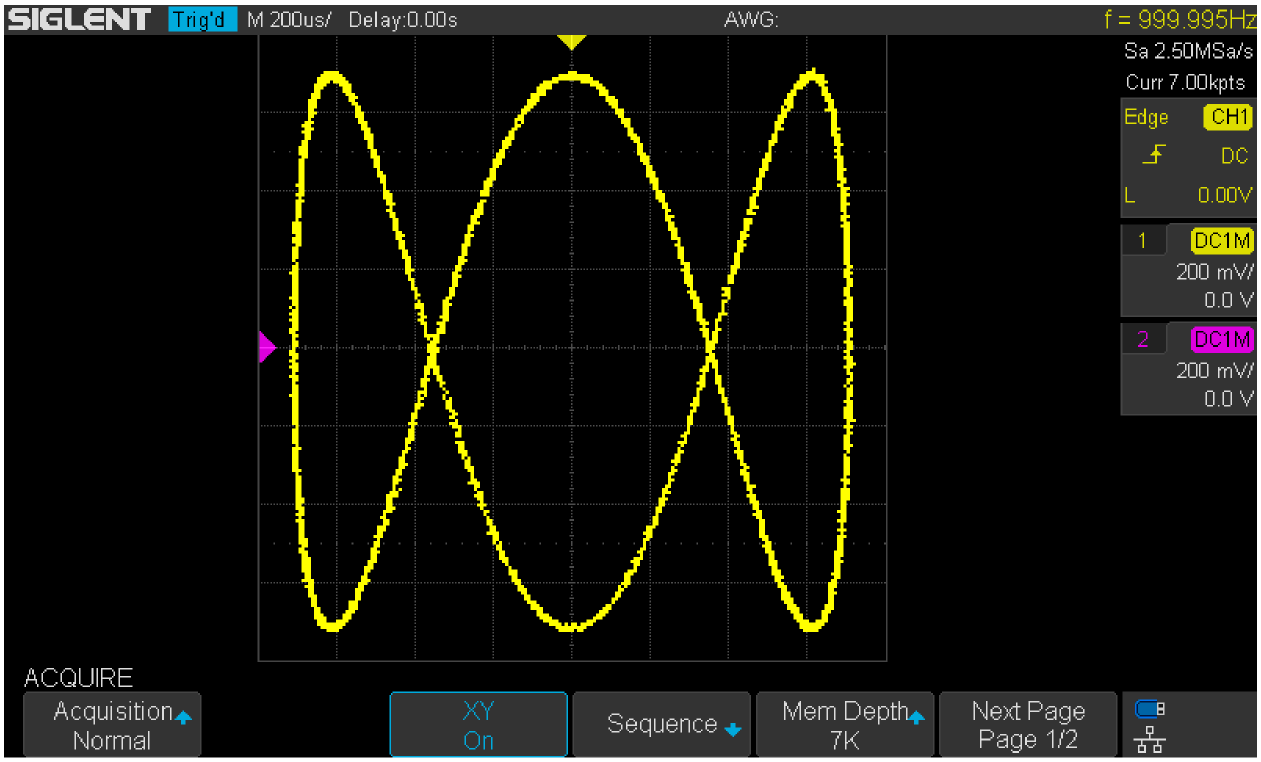
As a result, we have to carefully adjust our signals and memory depth so that the screen shows at least one full period of each signal involved and no aliasing occurs. At least we get a very reasonable update speed with short memory. I've yet to find a reliable way to estimate the exact update rate,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

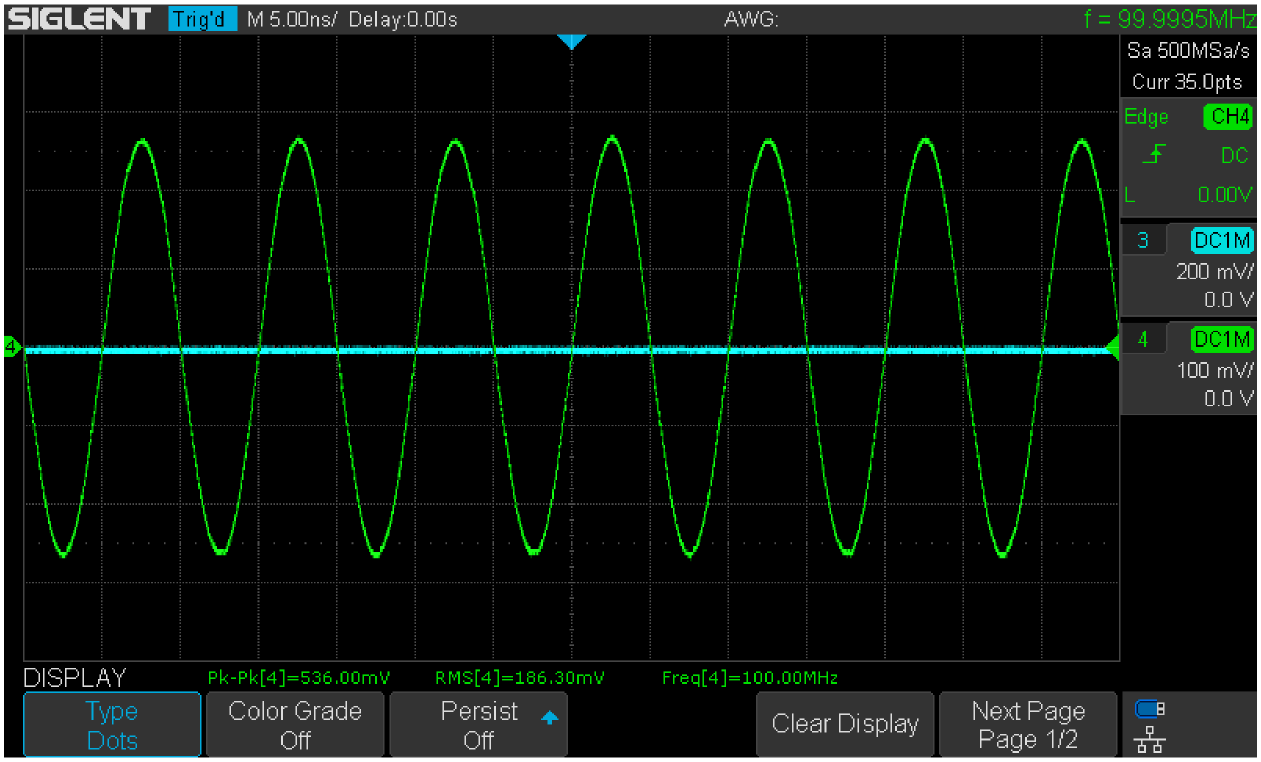
With Display Type Vectors and Interpolation x, a bunch of additional display points is generated as a linear interpolation from one sample to the next. In other words, the existing ADC samples are interconnected by straight lines. This is useful for “beautifying” signals like pulses and ramps, especially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

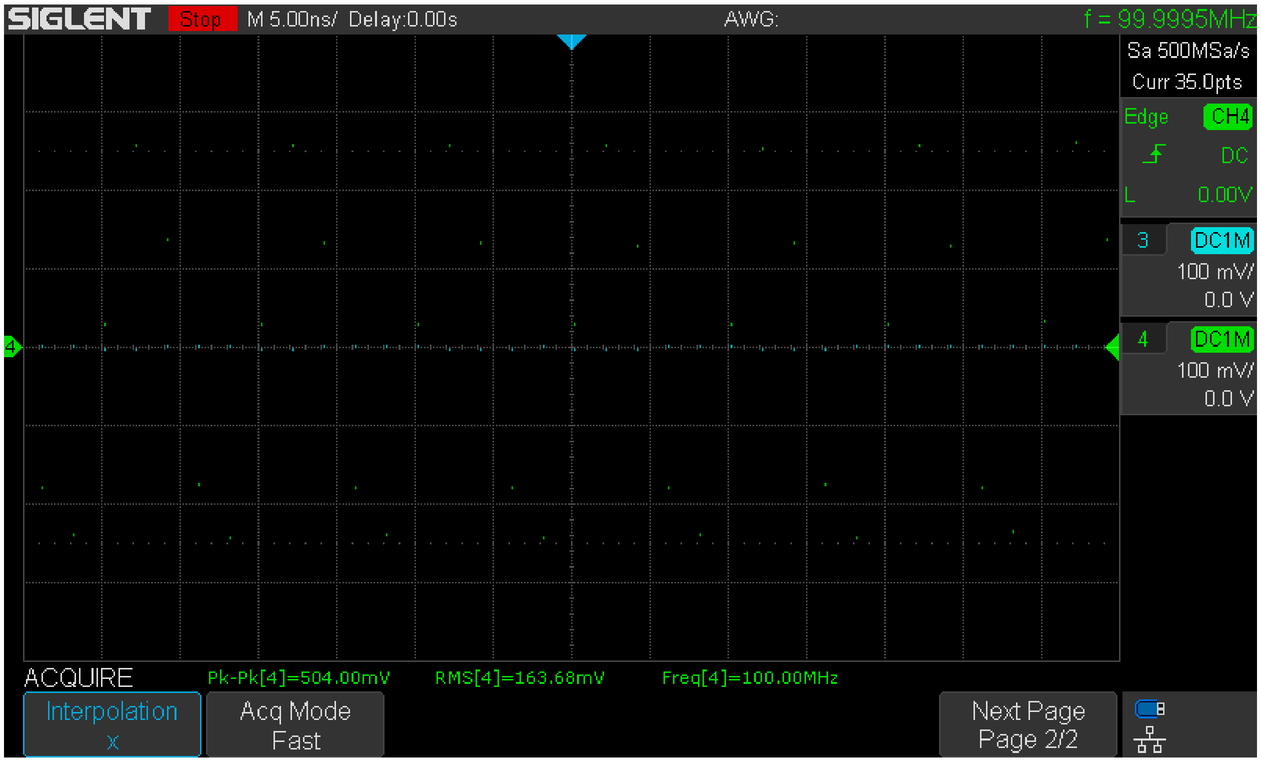
Examples

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

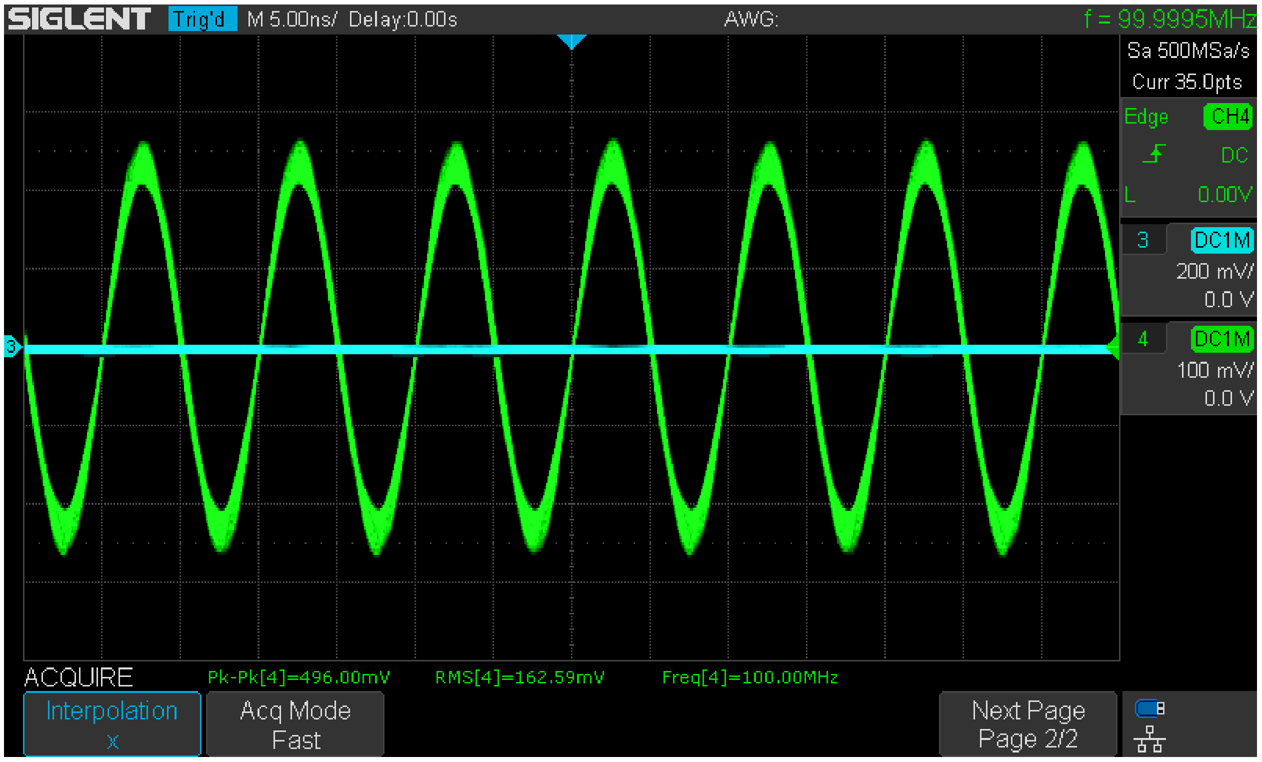
The next screenshot shows display mode vectors together with x-interpolation. This looks quite different to the dots display we've seen before. With vectors and linear x interpolation, we seem to get an amplitude-modulated waveform rather than a stationary signal.

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.

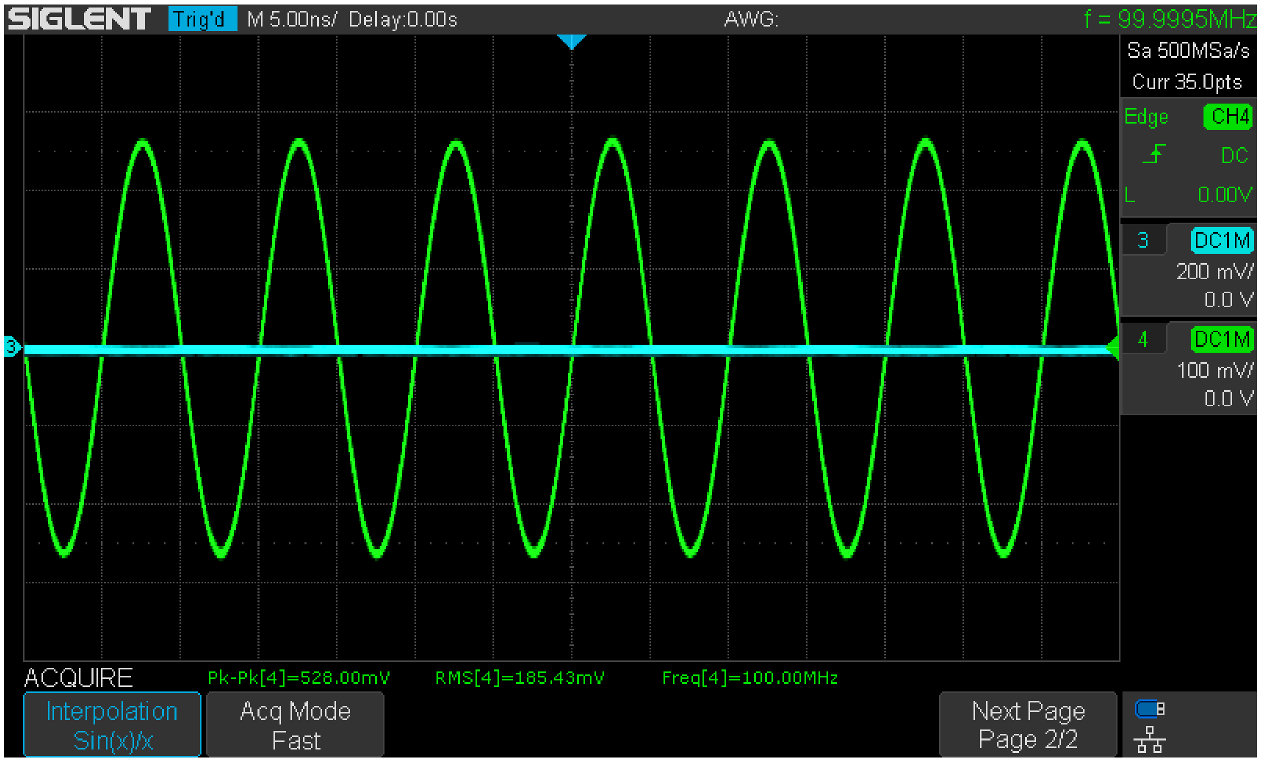
Dots display always gives a faithful representation of the input signal, as long as there are no signal components exceeding the Nyquist frequency (half the sample rate).

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.



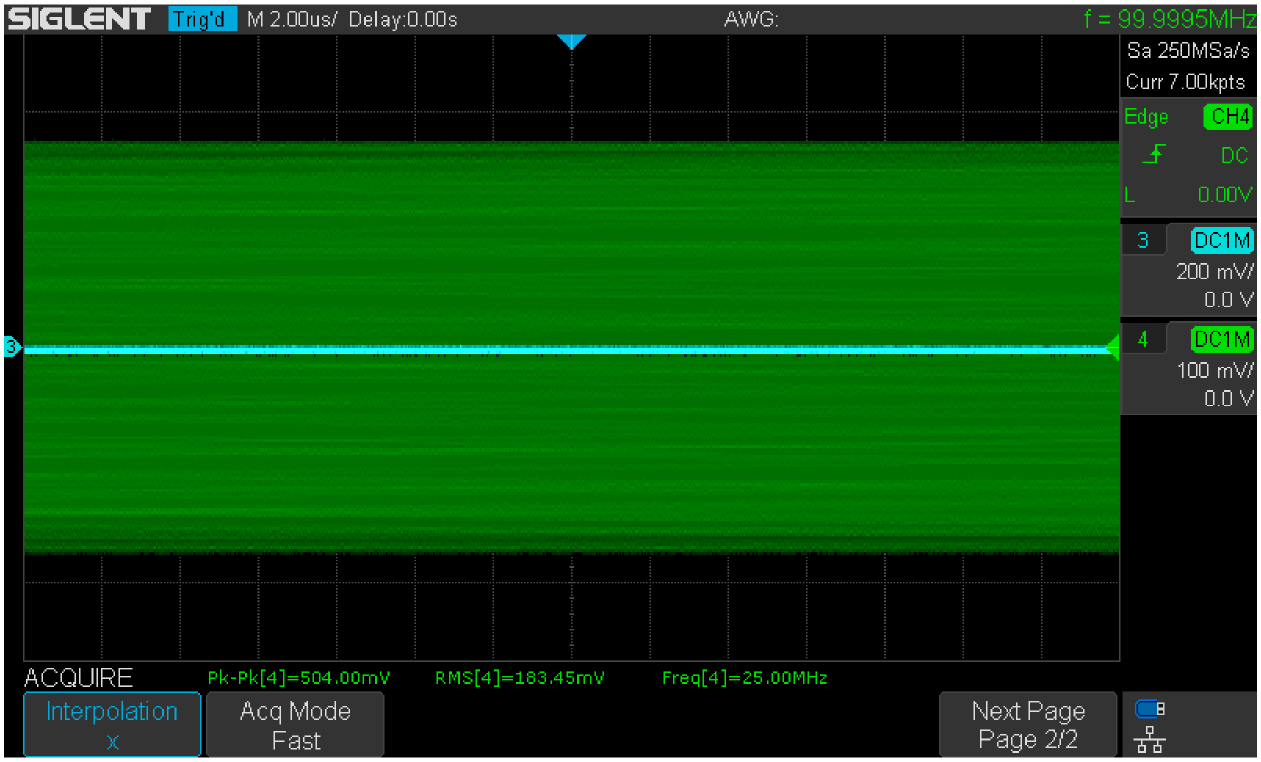
INT_100MHz_500MSa_5ns_x



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.



Pk-Pk[4]=504.00mV RMS[4]=183.45mV Freq[4]=25.00MHz

INT_100MHz_250MSa_2us_x

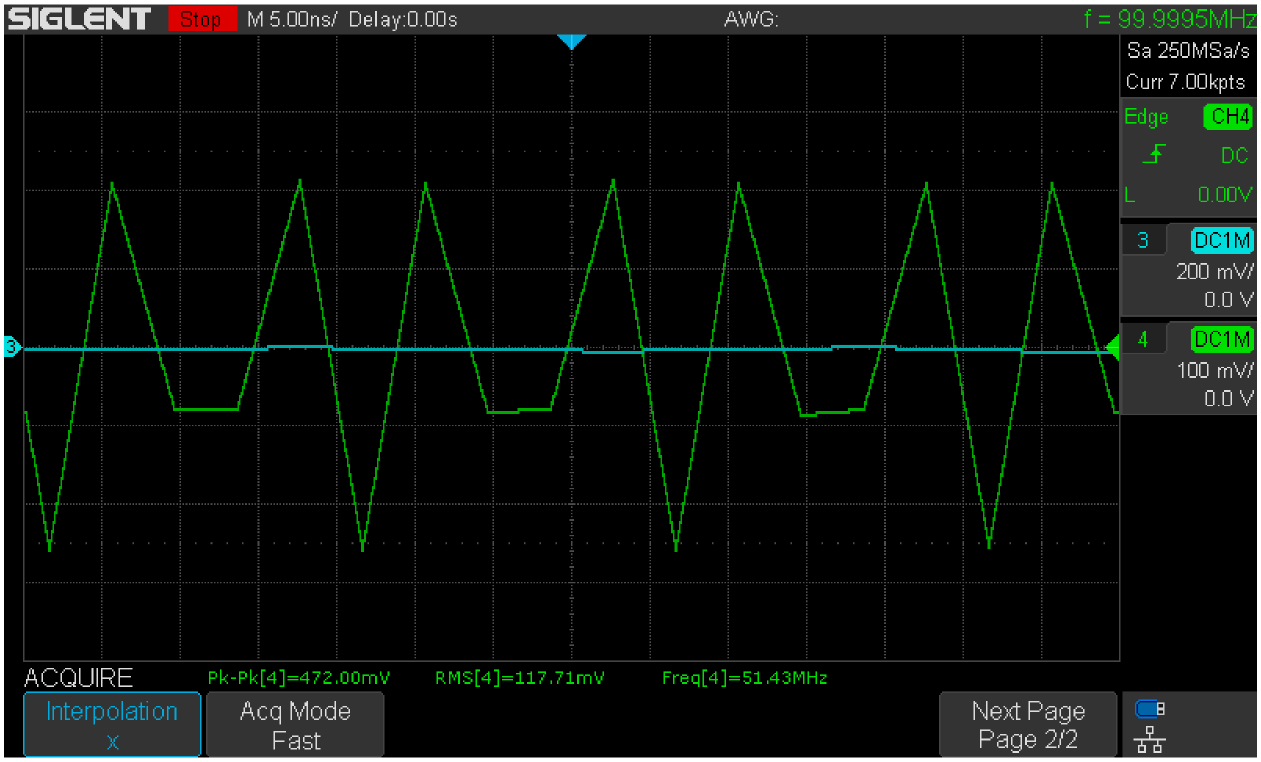
Of course, with a time-base that slow we cannot see anything meaningful. So we have to stop the acquisition and then zoom into the waveform by simply adjusting the horizontal time-base to 5ns/div.

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

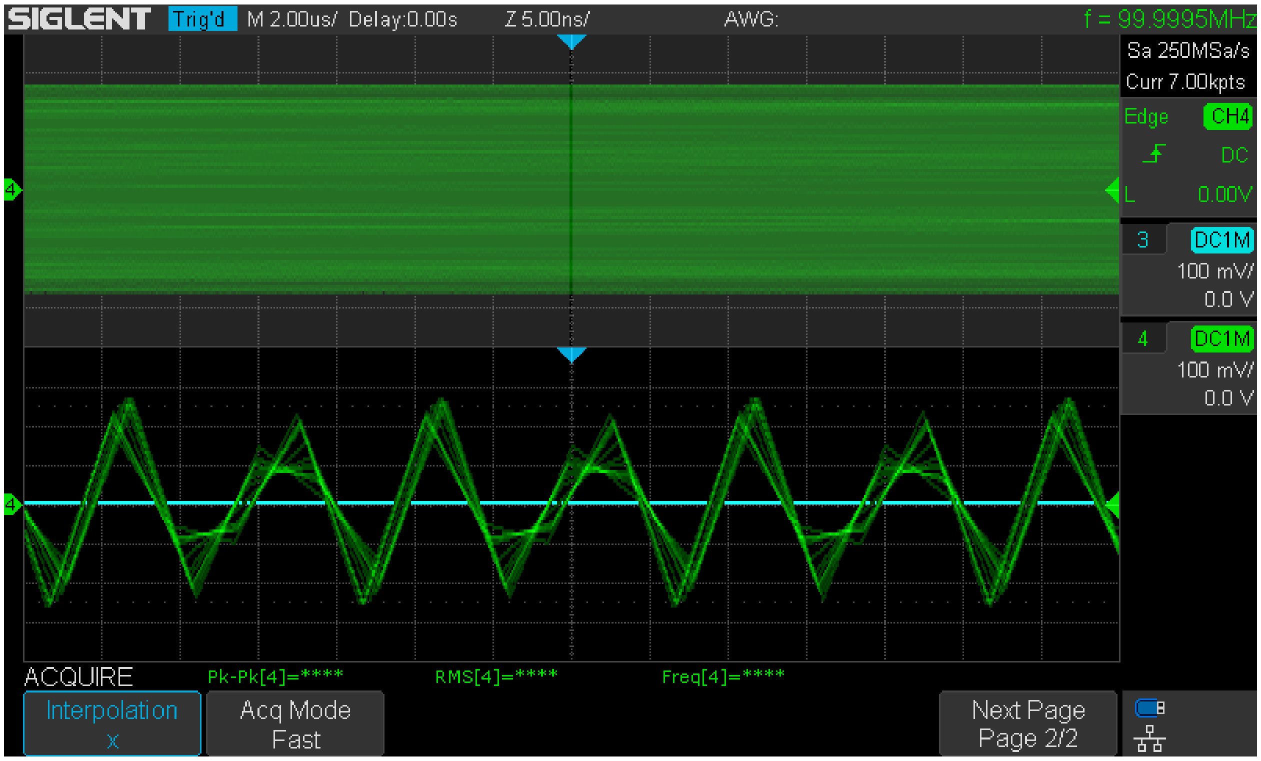
The first following screenshot shows vectors with linear x-interpolation. The similarity with the original signal is quite low – to say the least – and we already get an idea how this would look like in run mode.

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.

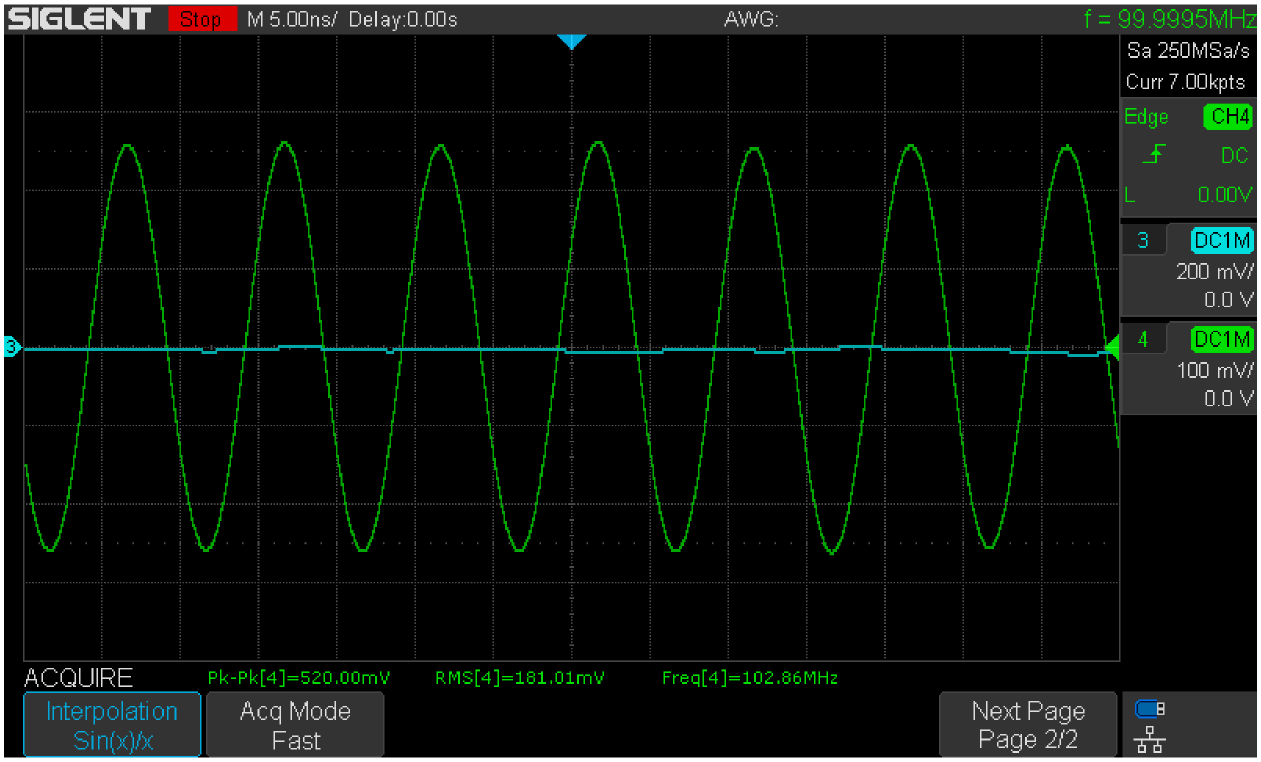


INT_100MHz_250MSa_5ns_x



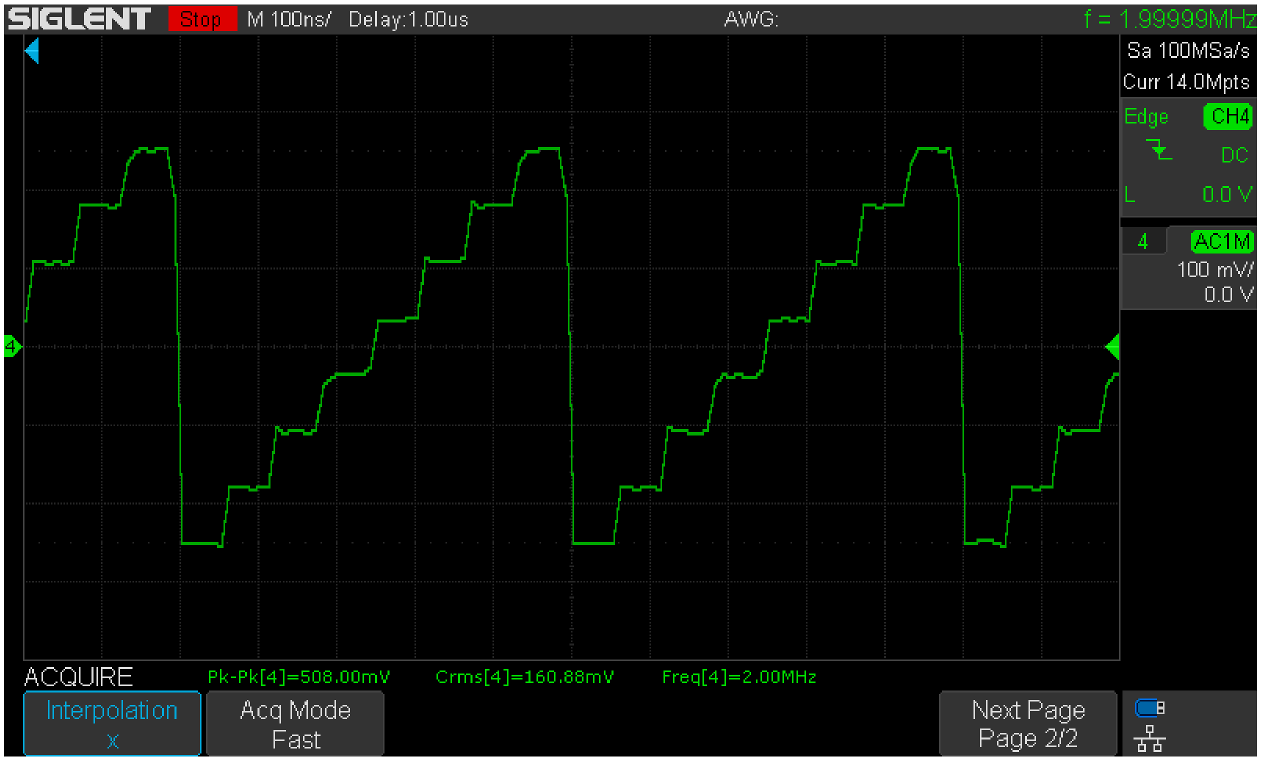
INT_100MHz_250MSa_2us_x_Z5ns

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.

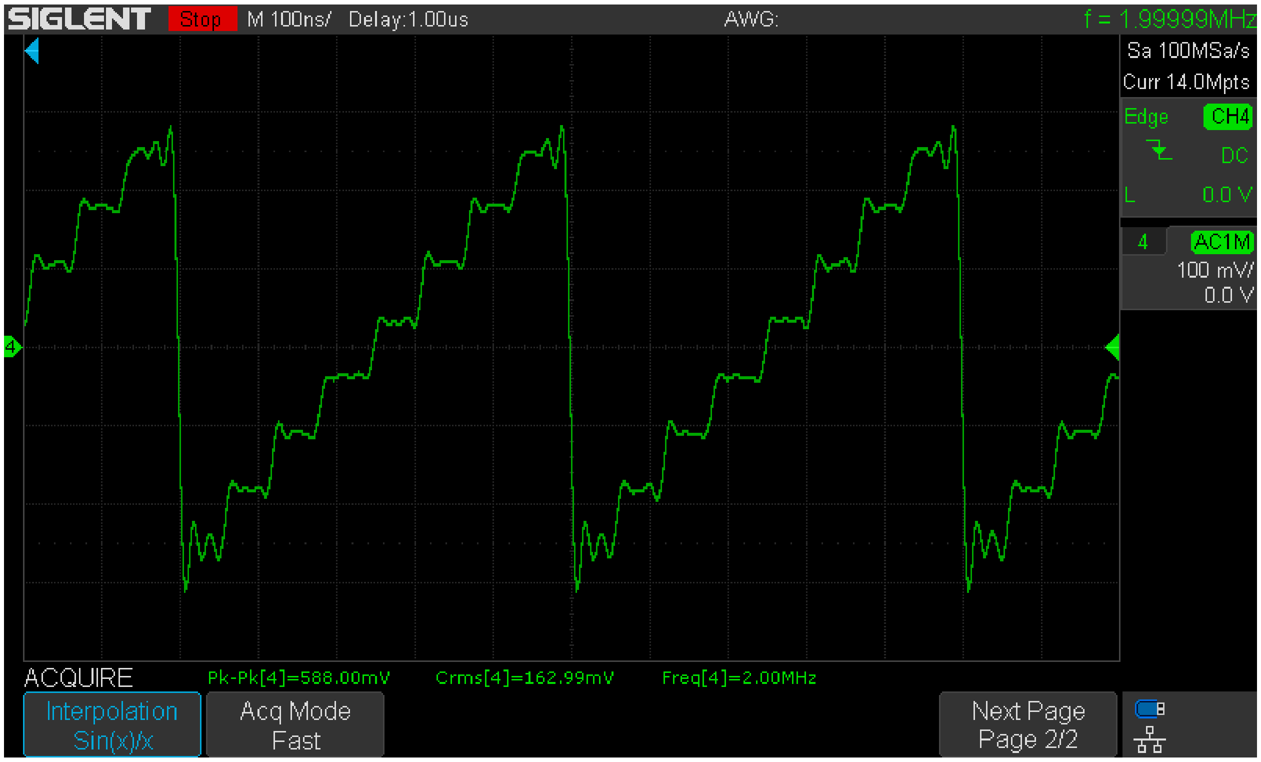


INT_100MHz_250MSa_5ns_sinc

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



INT_2MHz_stepup_100ns_x



Pk-Pk[4]=588.00mV Crms[4]=162.99mV Freq[4]=2.00MHz

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.