

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

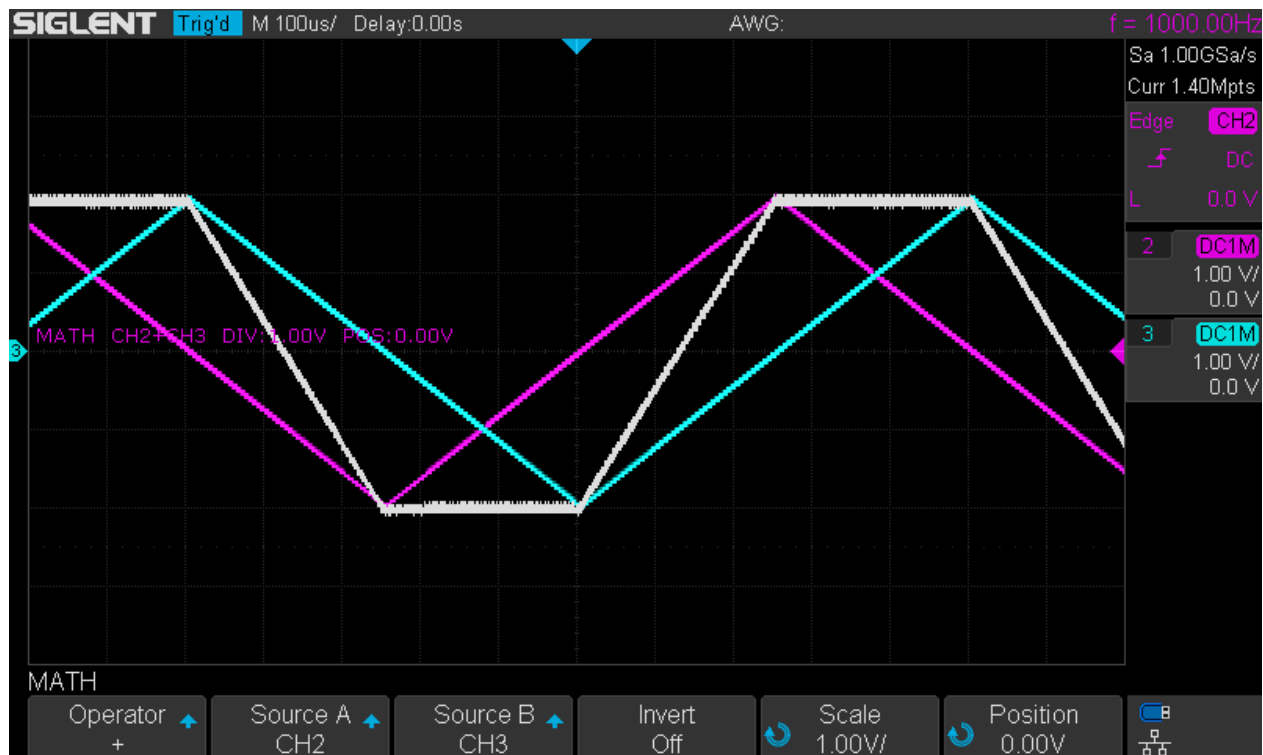
This is something even old analog scopes could do. In fact, it was only *Add* and a subtraction was accomplished by inverting the 2nd channel. There was also no dedicated math channel but the original traces were simply replaced by the sum of the two input channels.

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.

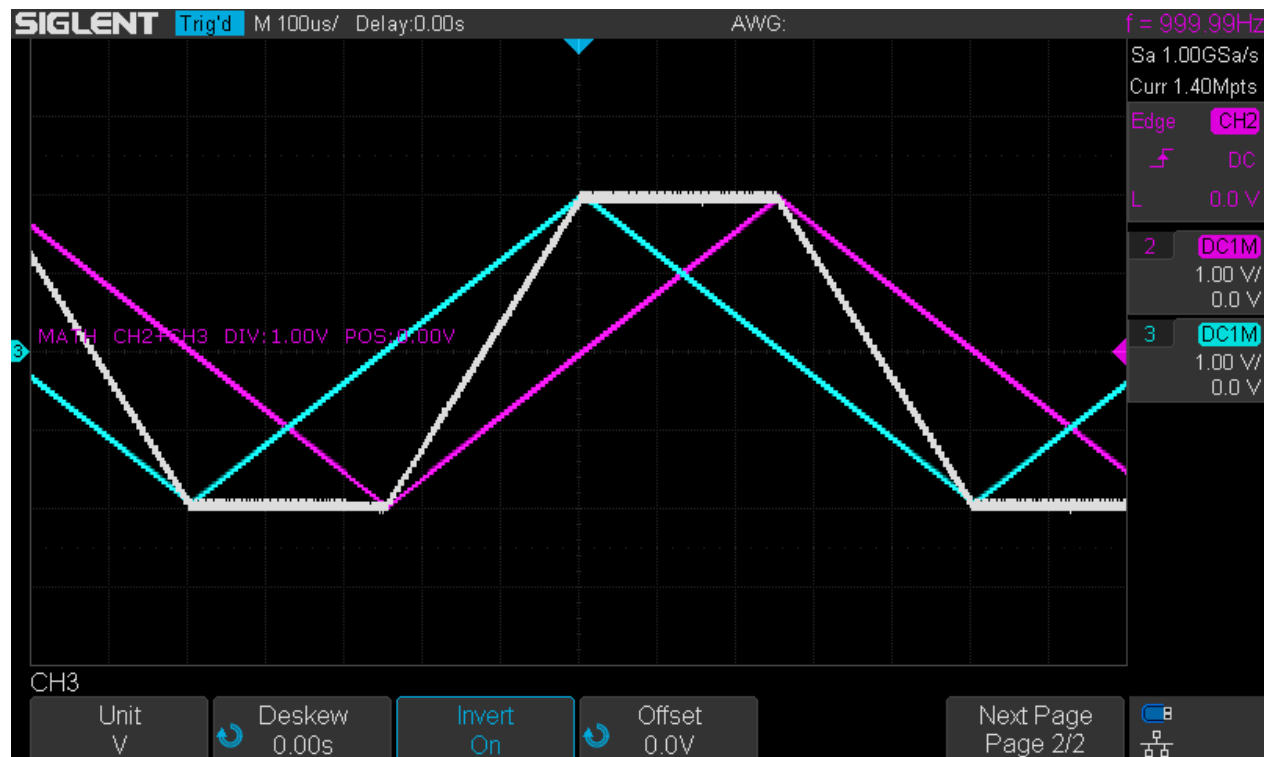
First is the sum of channels 2 + 3.

Second is the sum of channels 2 + inverted 3, to mimic the subtraction on an analog scope.

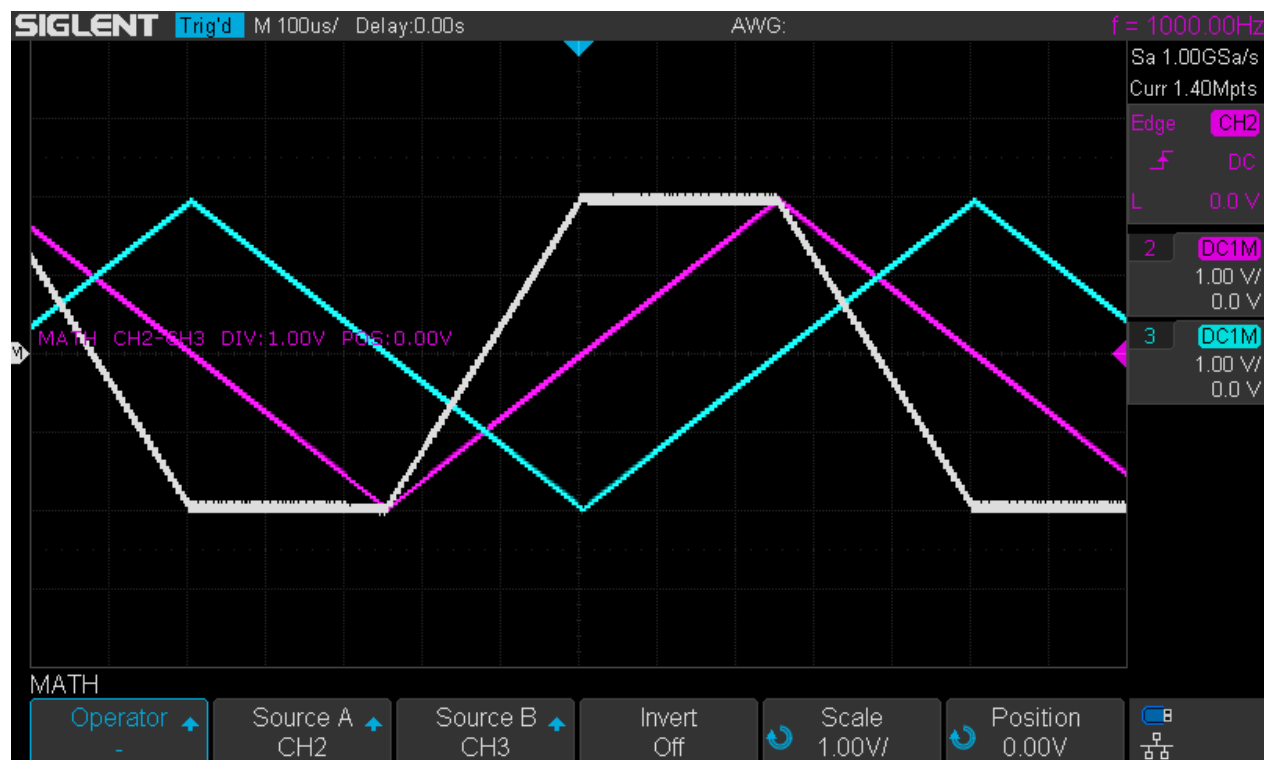
Third is the difference of channels 2 – 3, which should give the exact same result – and it certainly does.



SDS1104X-E_Add



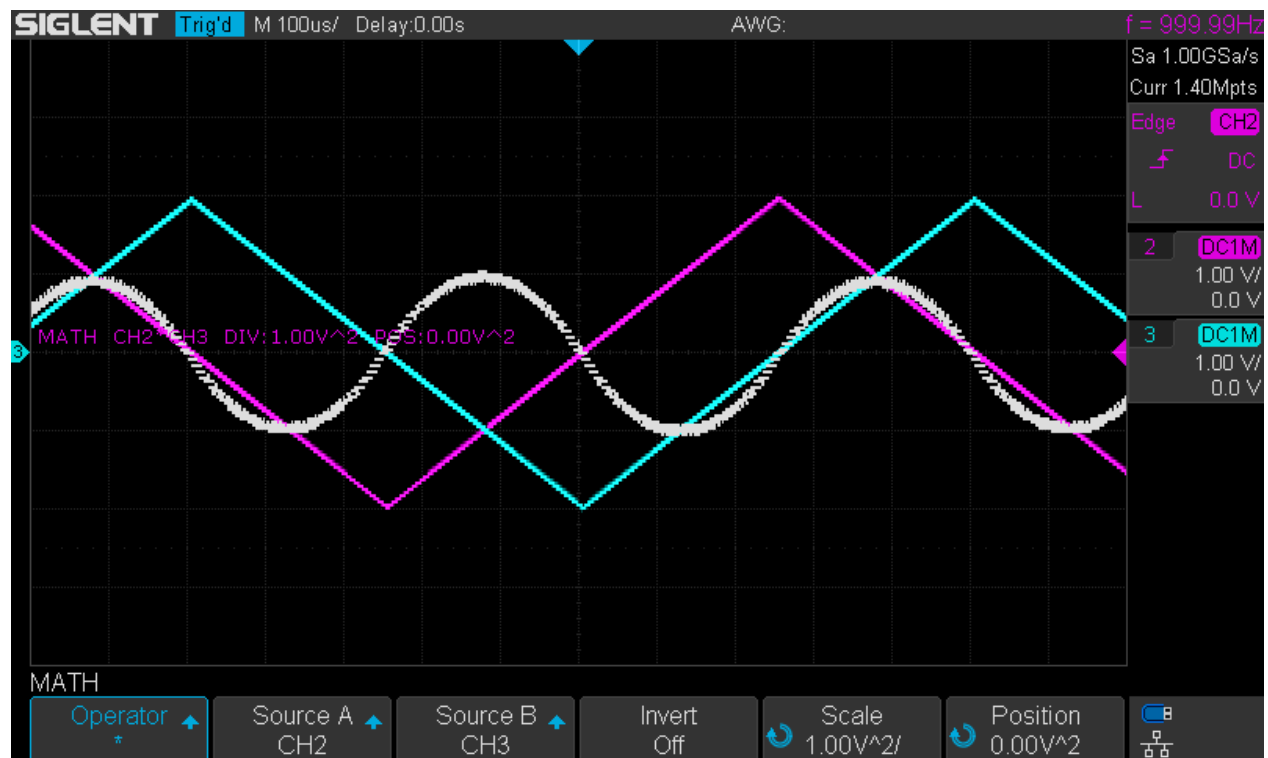
SDS1104X-E_Add_InvB



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.



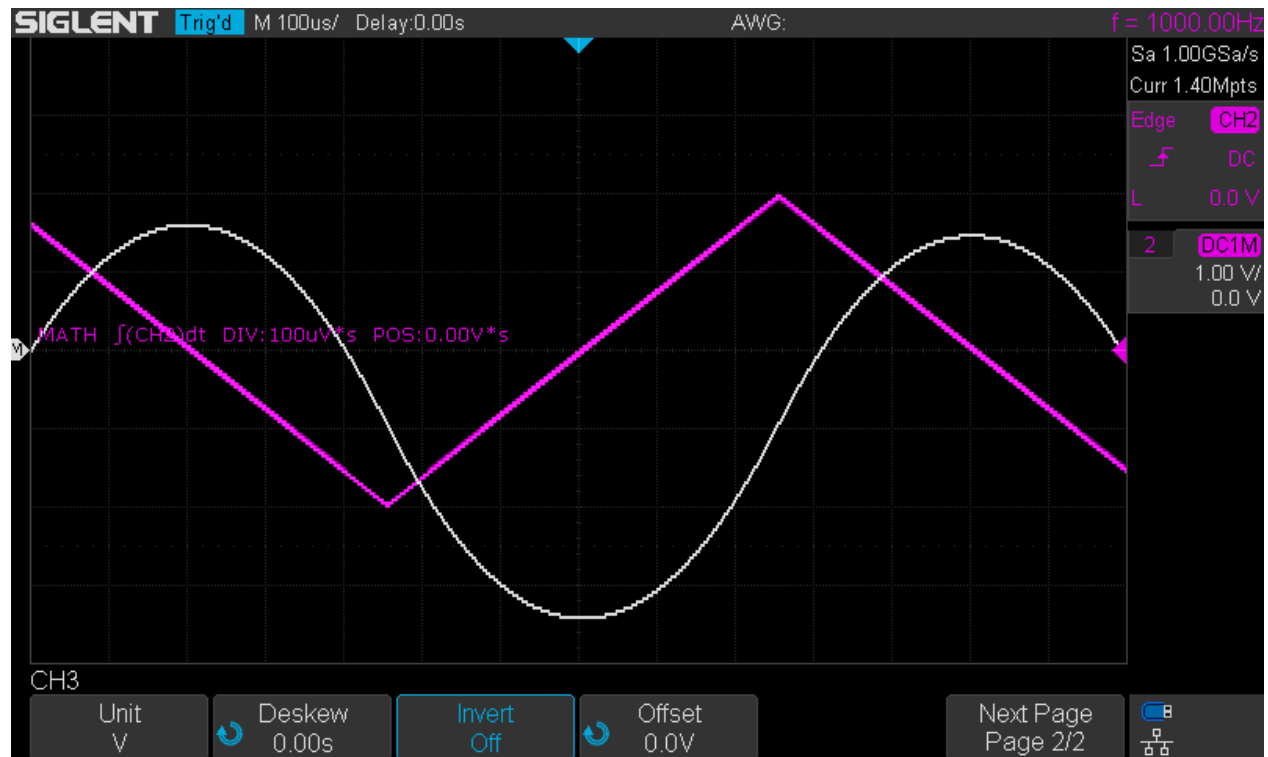
SDS1104X-E_Mul



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.

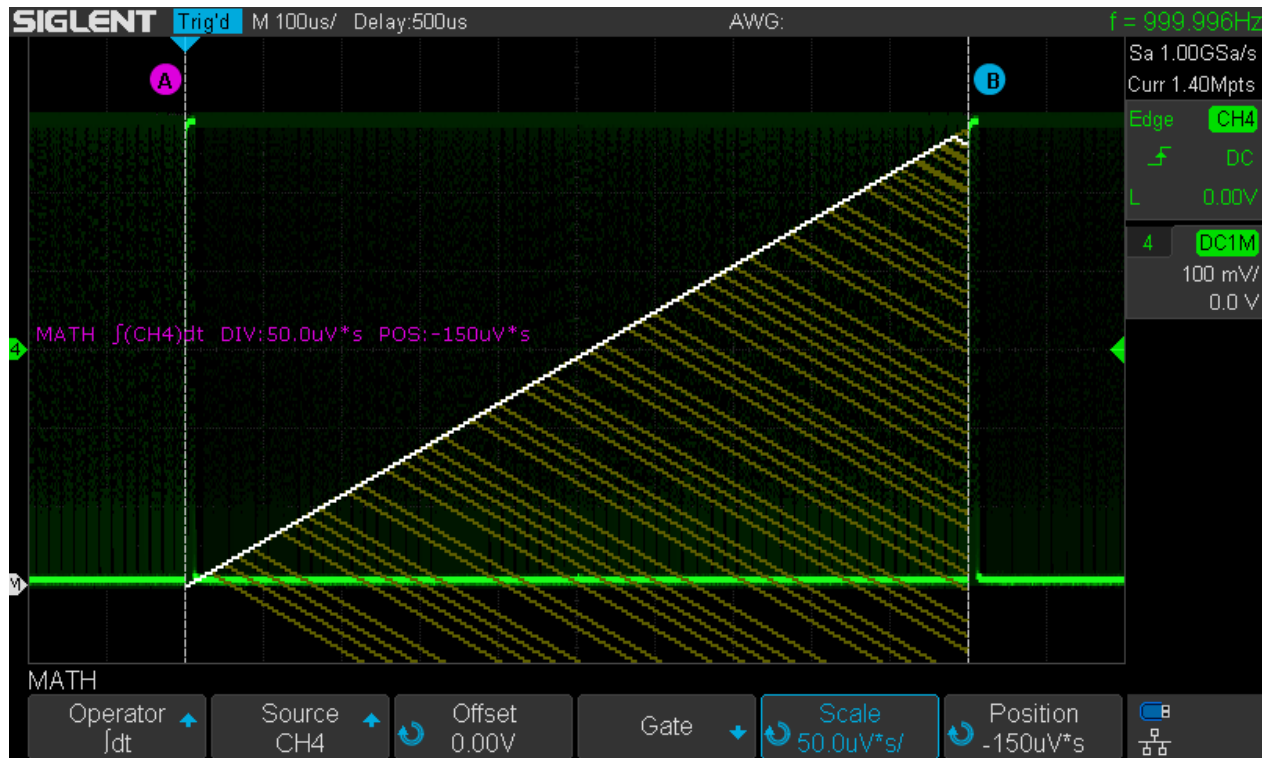
Integral



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

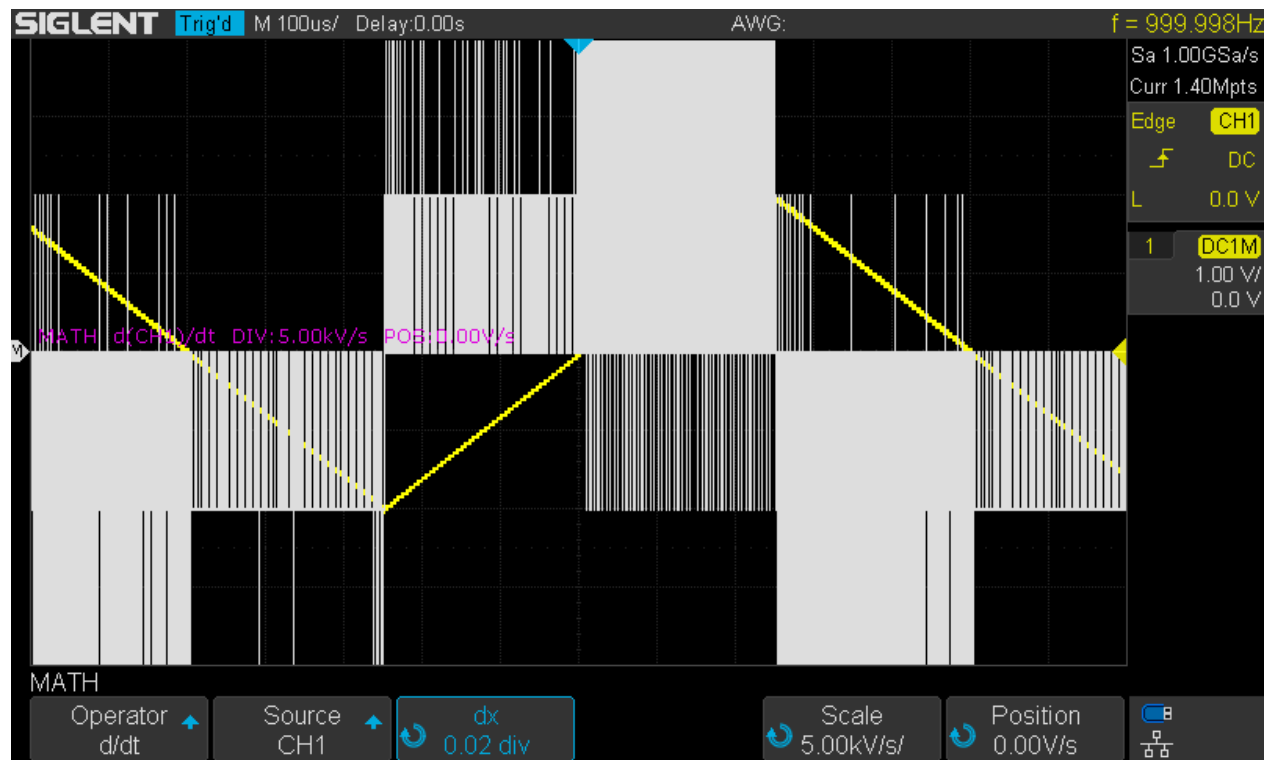
This is a particularly cumbersome math function that just cannot work well on an 8 bit system. It has a parameter dx , which determines the interval used for the difference computation. For an accurate and detailed result, we would want dx to be as small as possible and the lowest value that can be set 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.

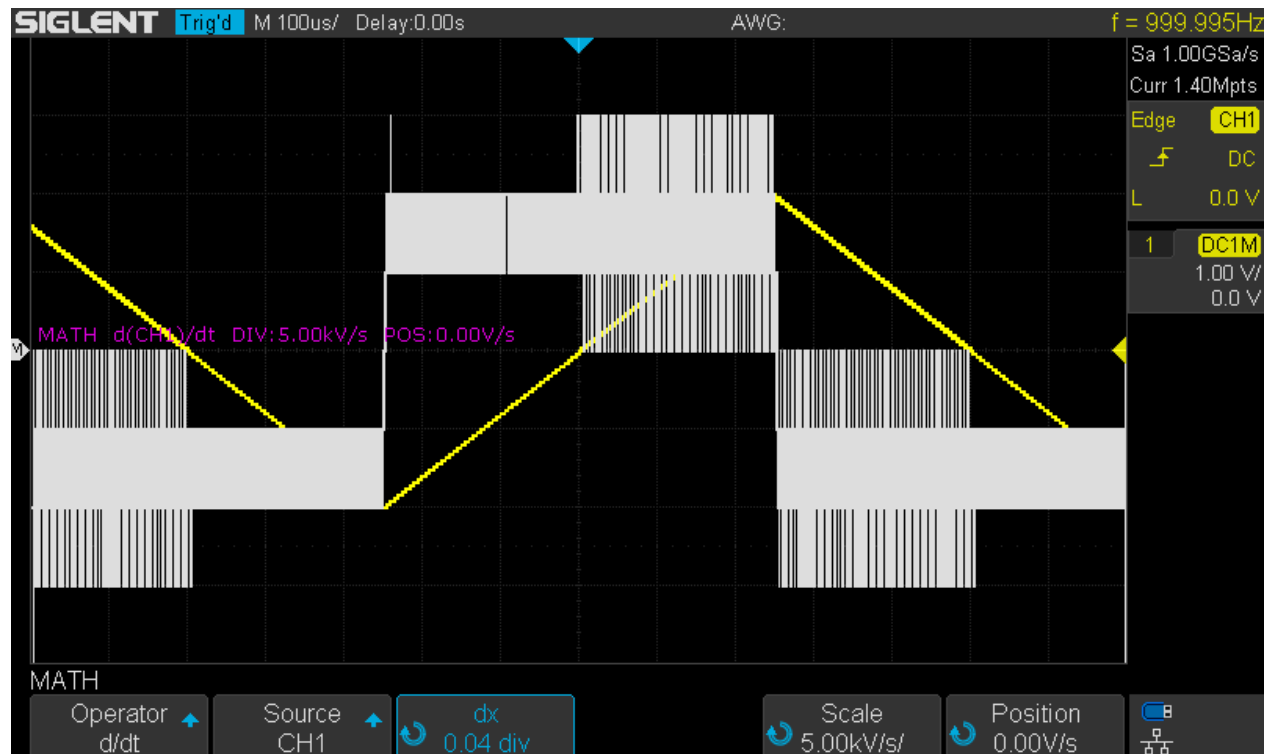
The screen is 800x480 pixels, with 700x400 dedicated to the trace area. Since there are 14 horizontal divisions, one division is $700/14 = 50$ pixels wide. The lowest dx parameter would thus be $0.02 \times 50 = 1$, which means the difference between two adjacent samples is calculated. This will work 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 500 μ s or 5 divisions at 100 μ s/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.

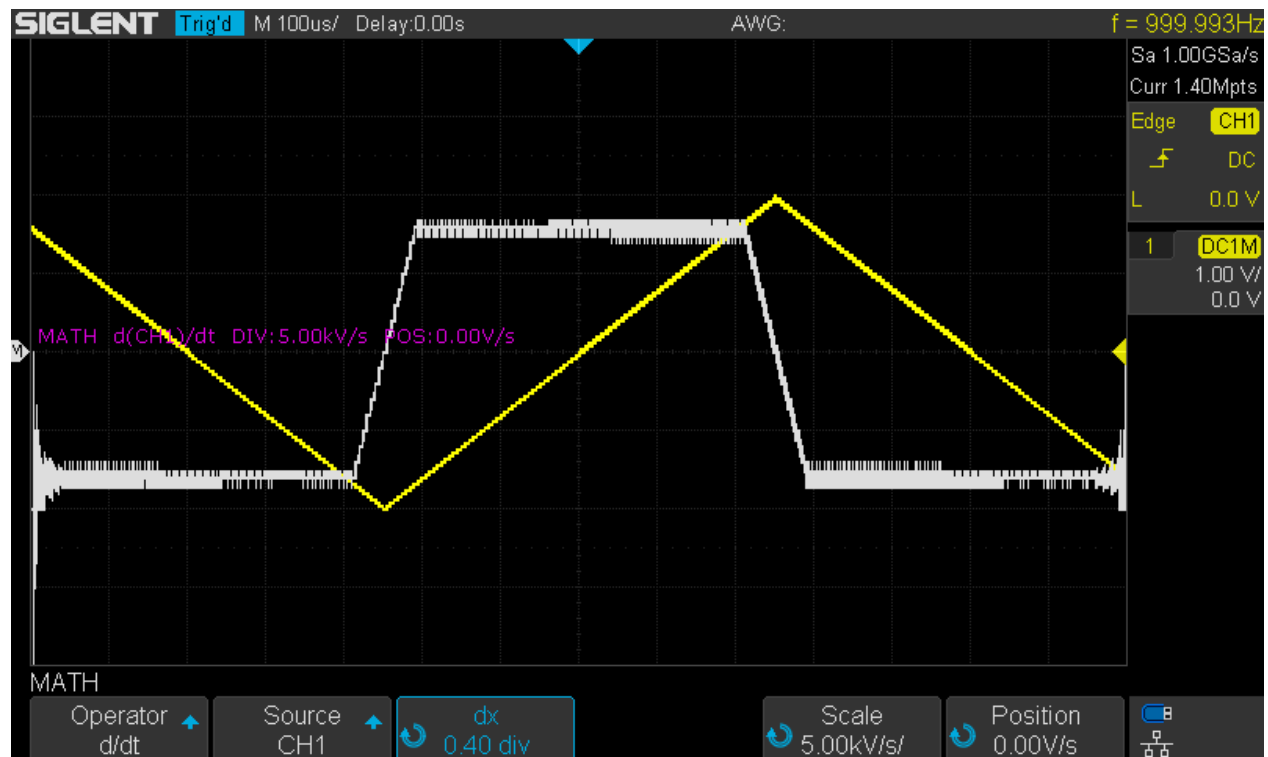


SDS1104X-E_Dif_Ramp_Avg16_2%



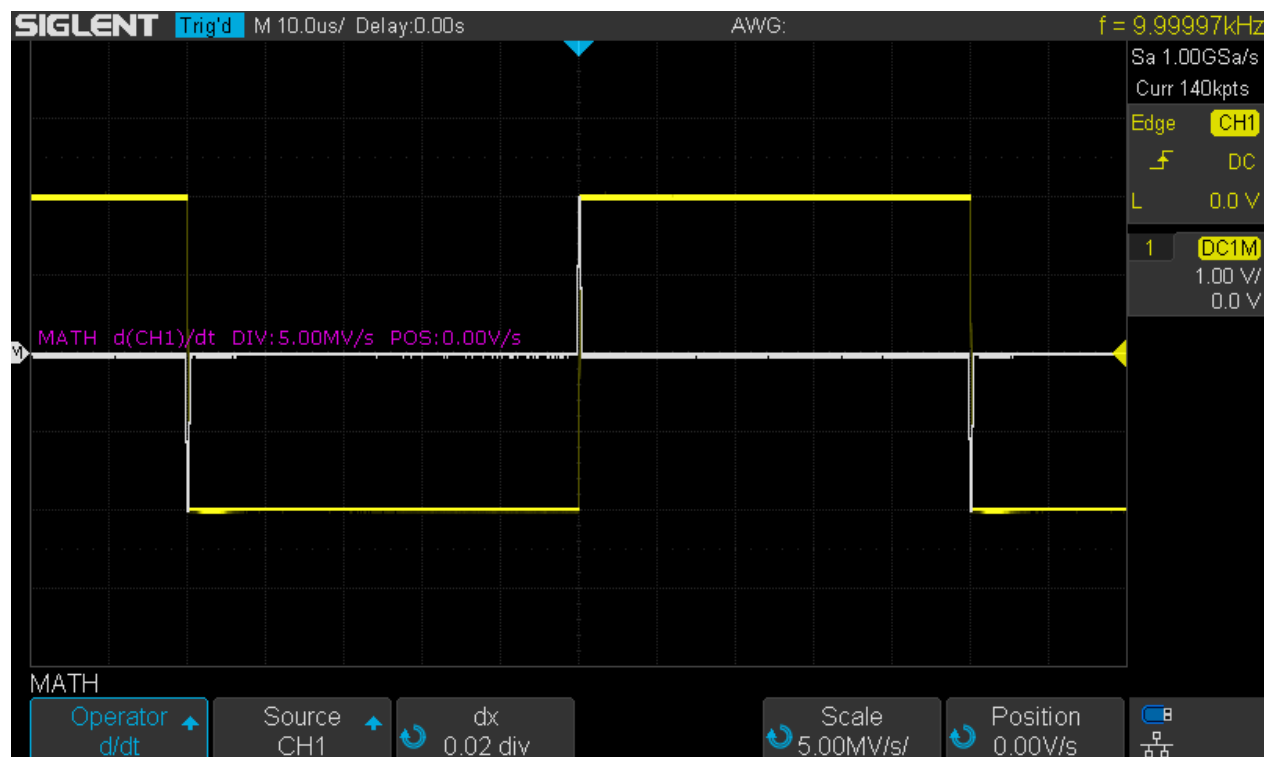
SDS1104X-E_Dif_Ramp_Avg16_4%

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%

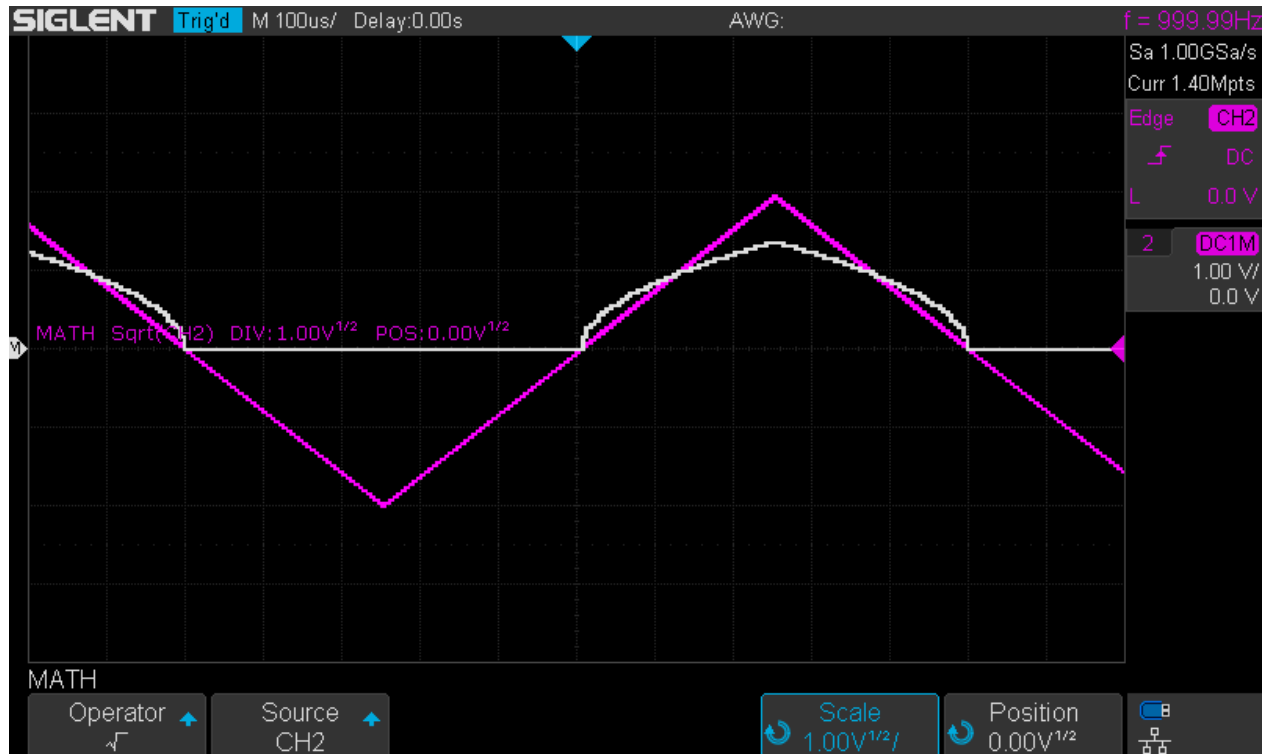
A square wave with fast transitions on the other hand is a perfect candidate for the differentiate function, and particularly so with the lowest possible dx parameter of 0.02.



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

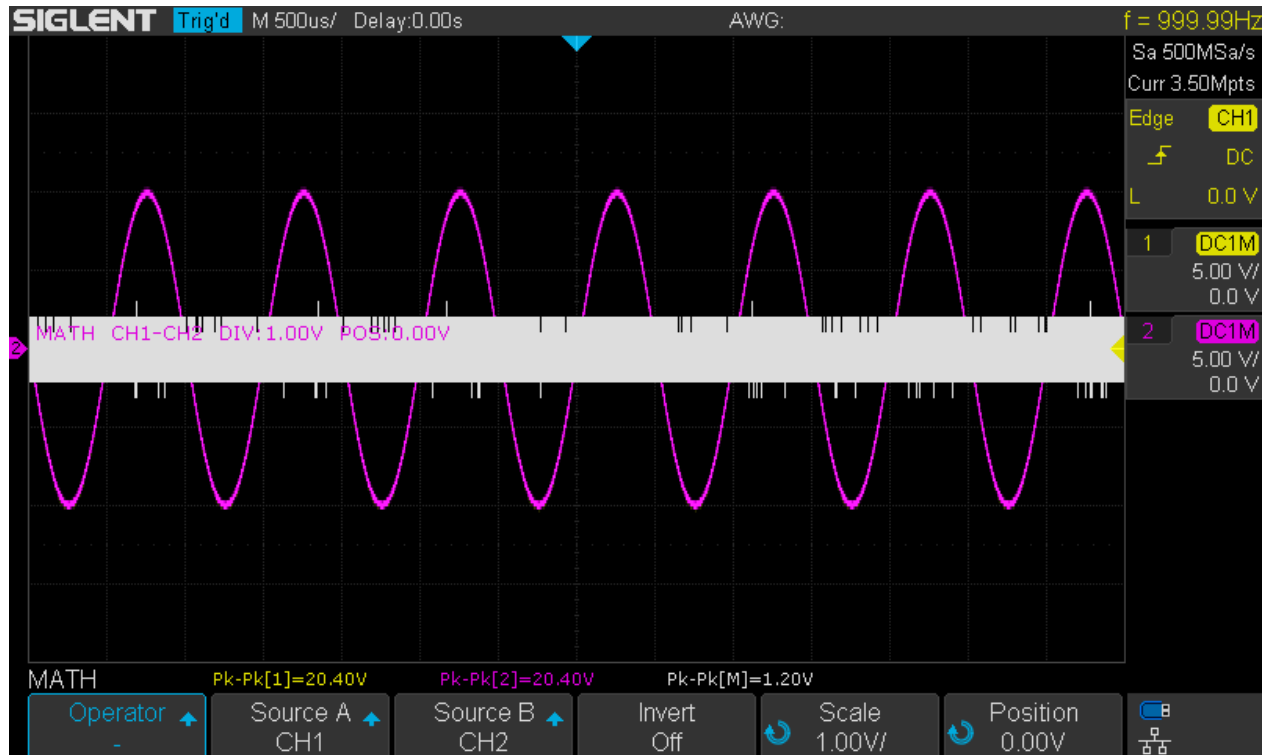
With analog scopes, we were able to combine two regular (single ended, ground referenced) channels into one differential channel. This was done by adding both channels with the 2nd channel inverted, whose gain had to be fine tuned in order to give a maximum of common mode rejection. Of course, this solution 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 $1.2\text{V}/20.4\text{V} = 0.0588 \sim -24.6\text{dB}$, 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.

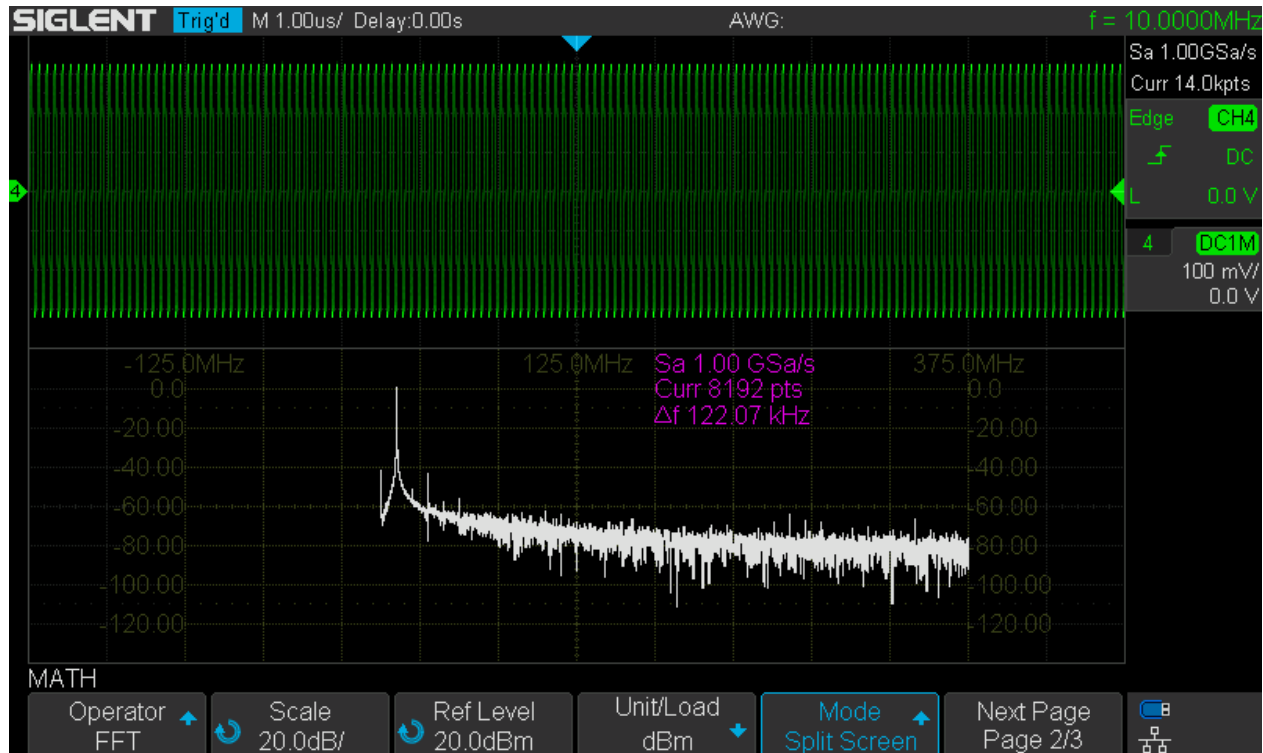
Generally speaking, we should not get too obsessed with features and gadgets, but concentrate more on the quality of measurements. All the bells and whistles won't serve us anyway, unless we get low distortion and noise, accurate levels, a reasonable high spurious-free dynamic range and adequate frequency 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

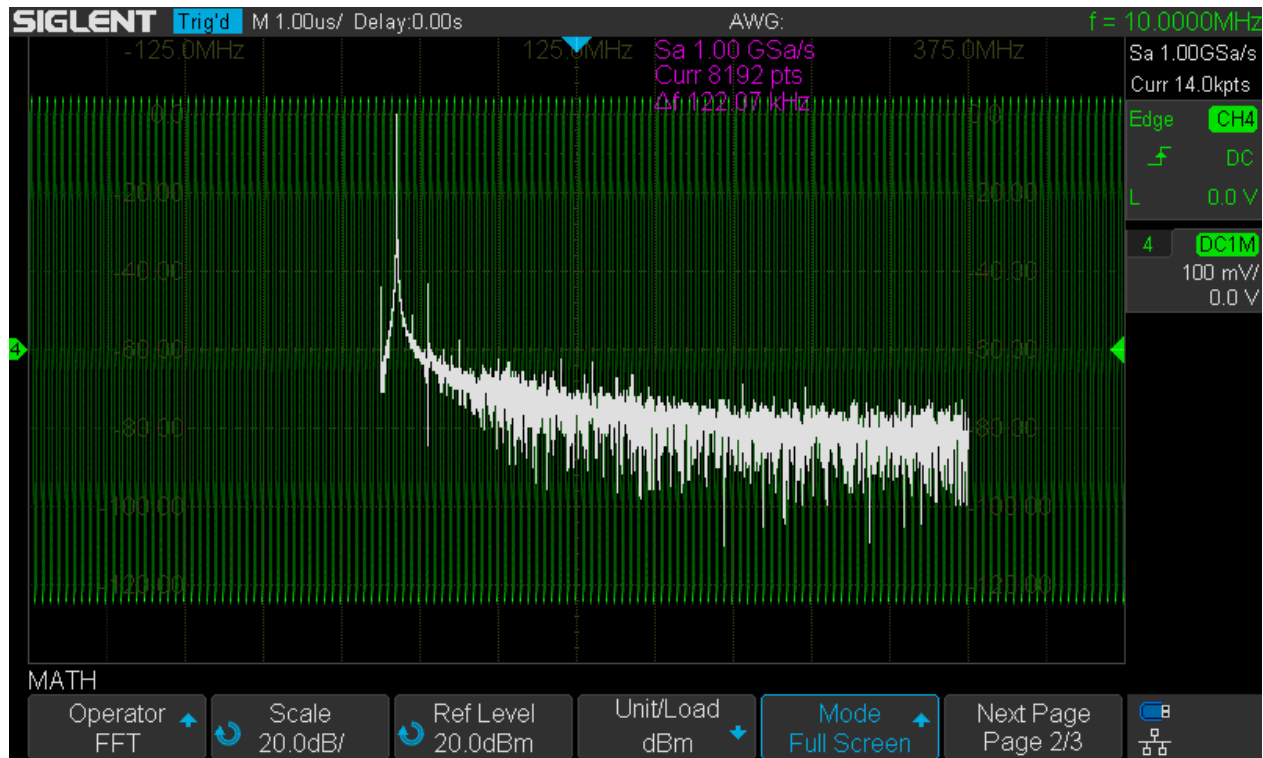
Screen Modes

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

- Split Screen
- Full Screen
- Exclusive

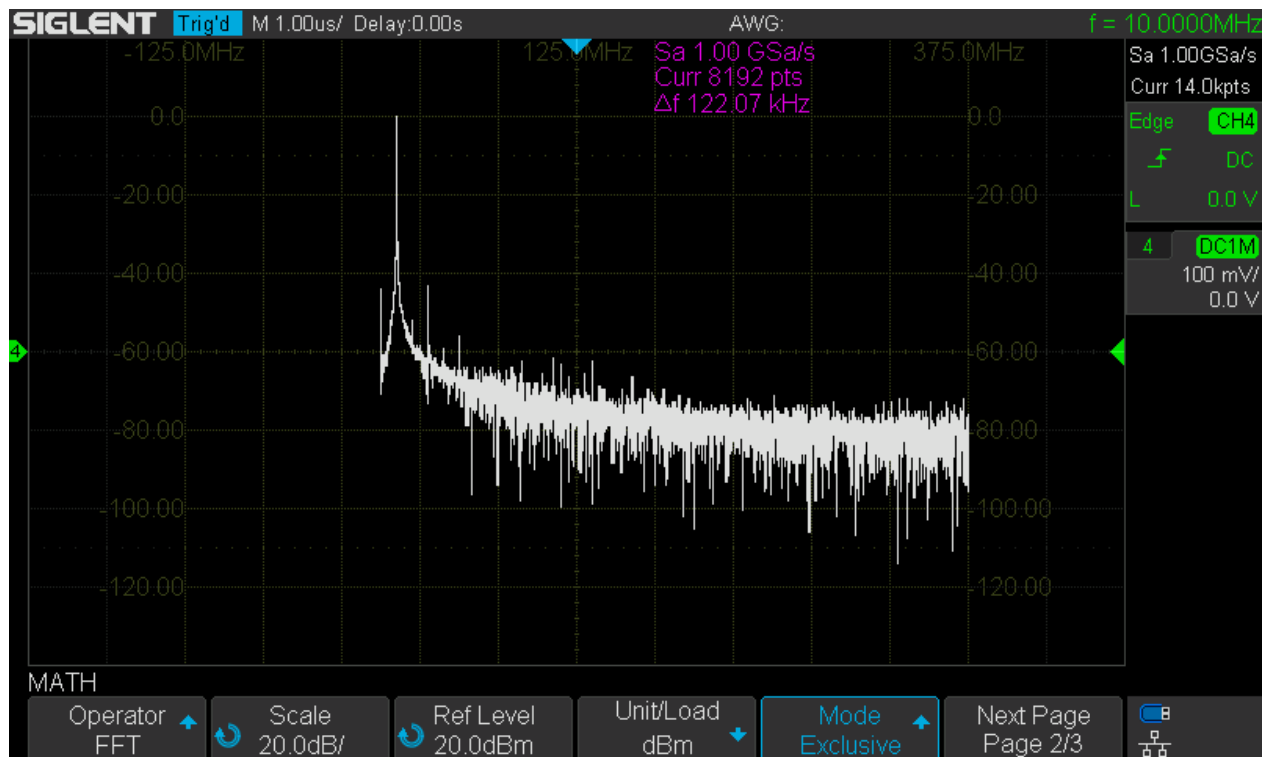
Split Screen as has been shown in the previous screenshot is most useful for setting up the measurement, as it allows the observation of the input signal in the time- and frequency-domain in parallel. This is important, because as a general rule we want the signal to nearly fill the screen height in 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

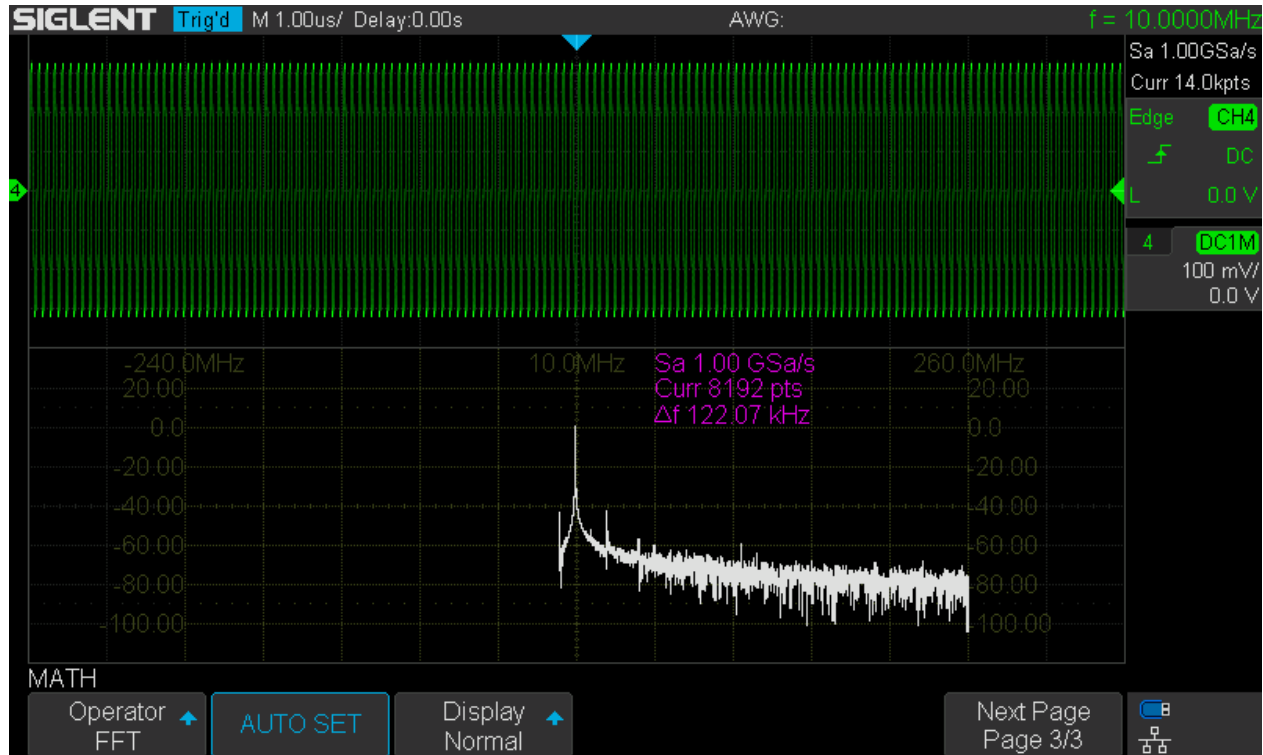
Exclusive shows the FFT exclusively and makes the instrument look more like a dedicated spectrum analyzer. Use this mode once the measurement is properly set up and the input signal amplitude is more or less stationary.



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.

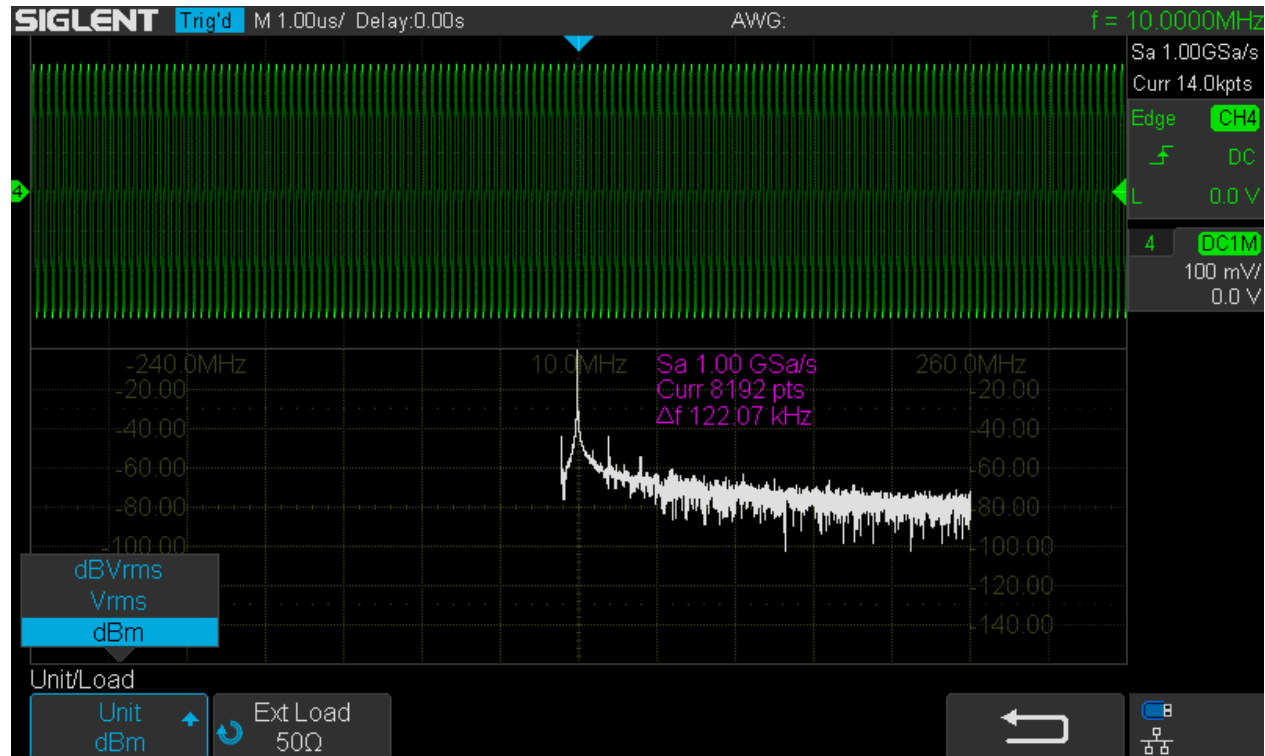
For general laboratory use, we would universally have an external 50Ω pass-through terminator, hence a 50Ω load impedance. The same is true for most RF applications, even though there are exceptions especially for the antenna inputs of domestic broadcast receivers. 75Ω is the standard for video signals, 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.

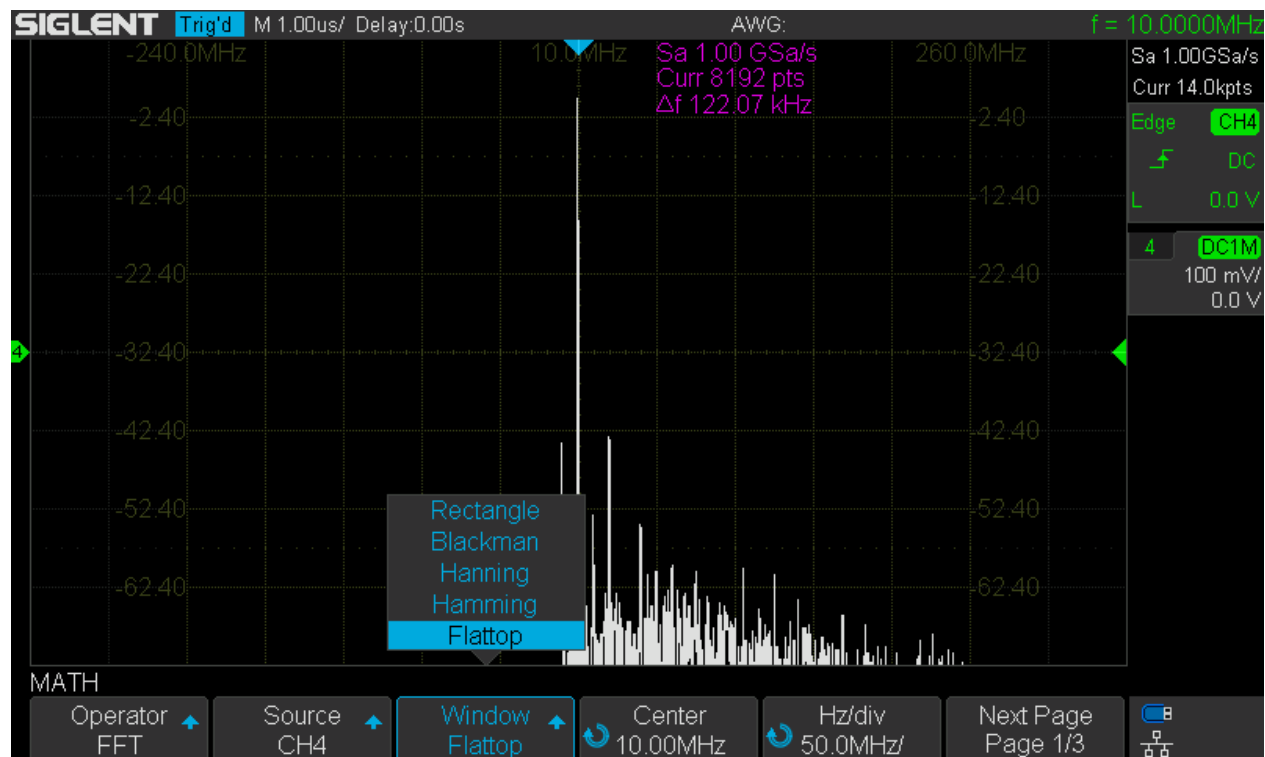
The screenshot below shows how the FFT is set up for dBm units and 50Ω external load. At the same time, an external 50Ω pass-through terminator has been fitted for channel 4.



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.

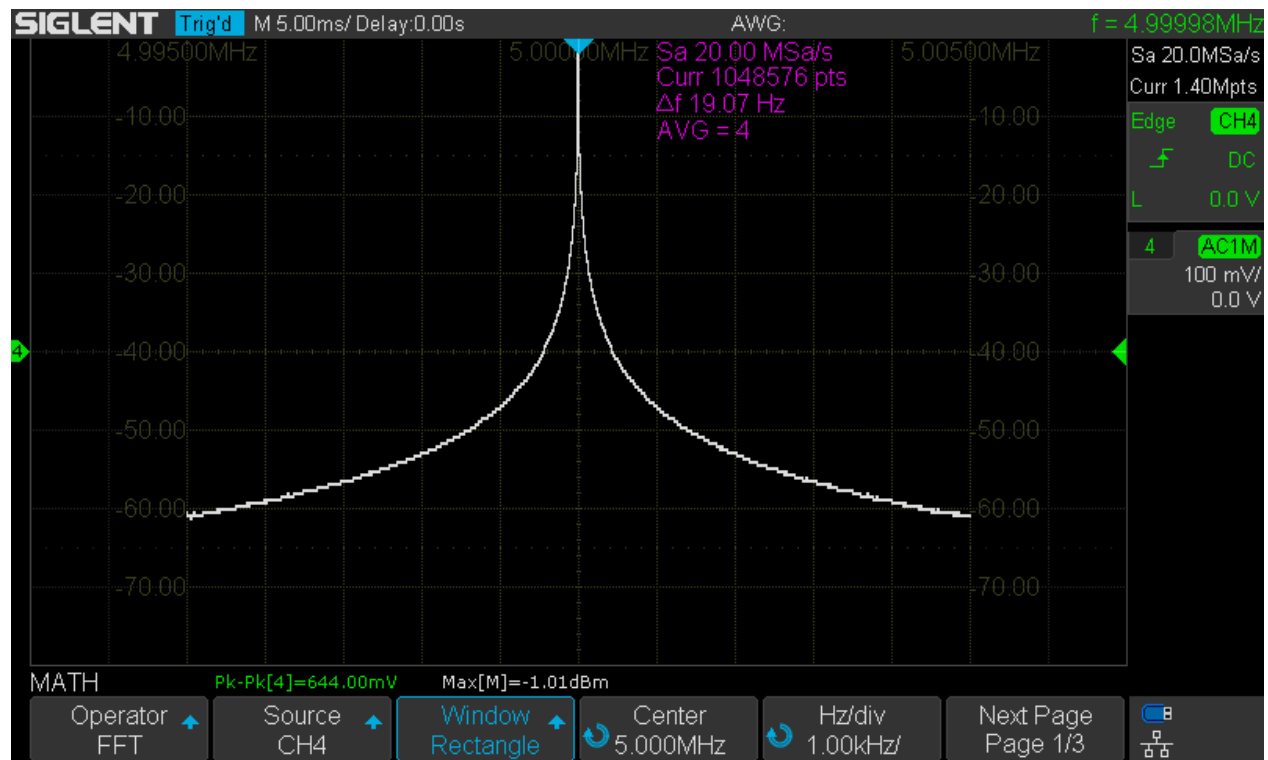
Window	-3dB BW (Bins)	Max. Side Lobe [dB]	Side Lobe roll-off [dB/Oct.]	Remark
Rectangle	0.89	-13.2	6	Min. 3dB bandwidth, used for short transients
Blackman	1.68	-58	18	High side lobe suppression, used for audio
Hanning	1.20 ~ 1.86	-23 ~ -47	12 ~ 30	Used for audio and vibration measurement
Hamming	1.30	-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):

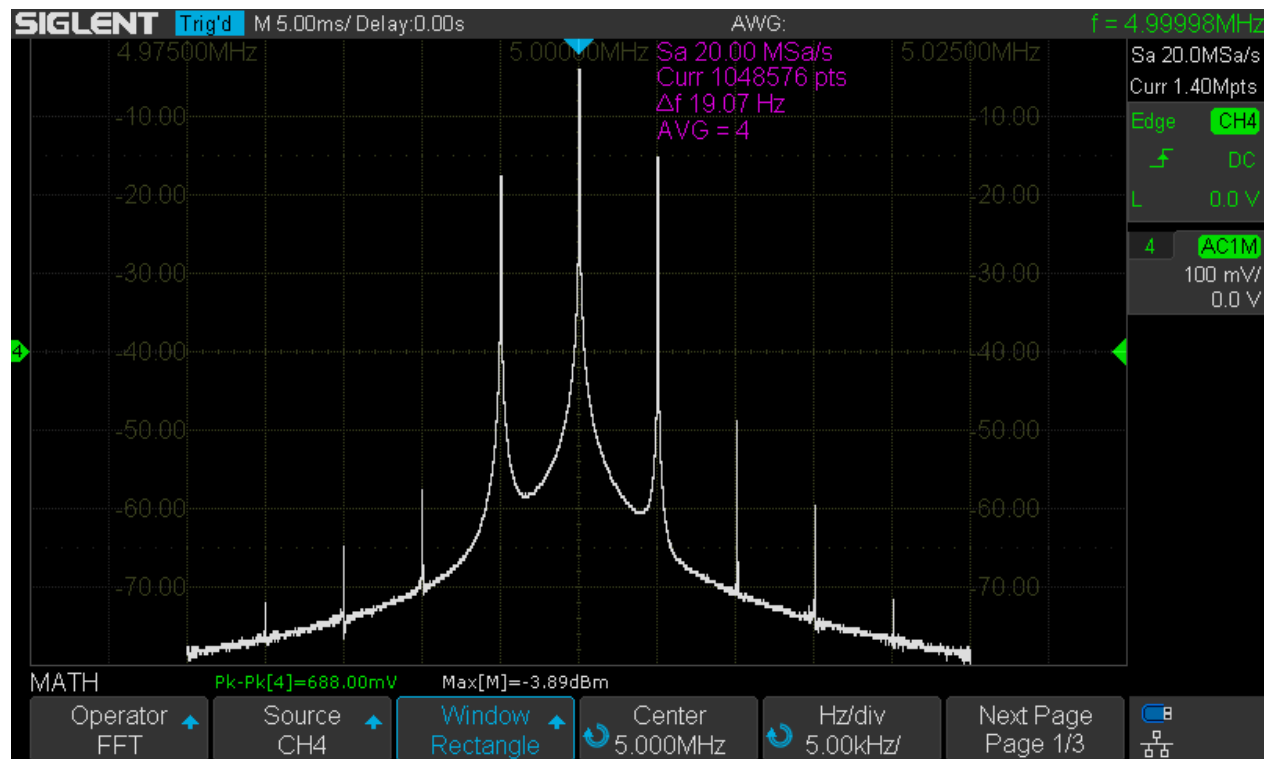
- Use Flat-Top whenever best amplitude accuracy is important. $RBW \sim 3 \times \Delta f$
- Use Blackman otherwise, especially when a high dynamic range is desired. $RBW \sim 1.7 \times \Delta f$

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

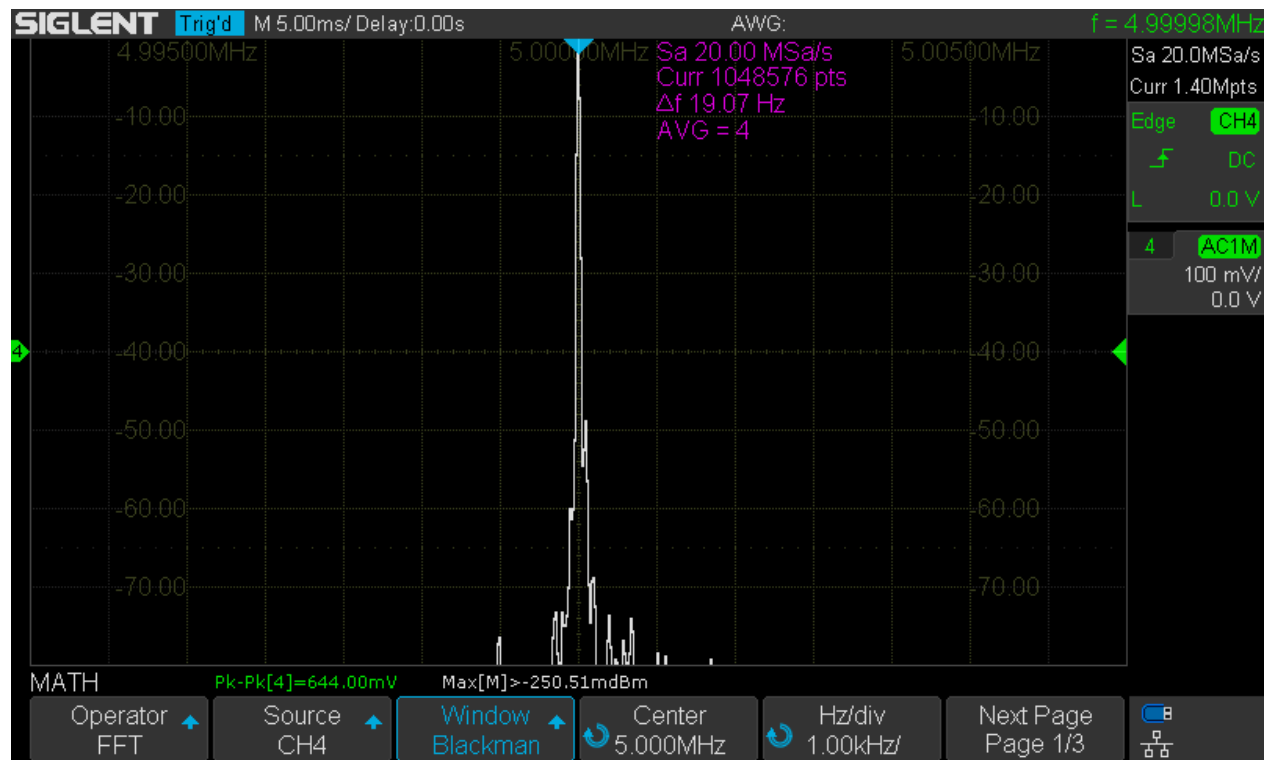
1. Sine, 5MHz, 0dBm, displayed at 1kHz/div to show the selectivity and filter shape
2. Sine, 5MHz, -3dBm and 50% AM with 5kHz, displayed at 5kHz/div



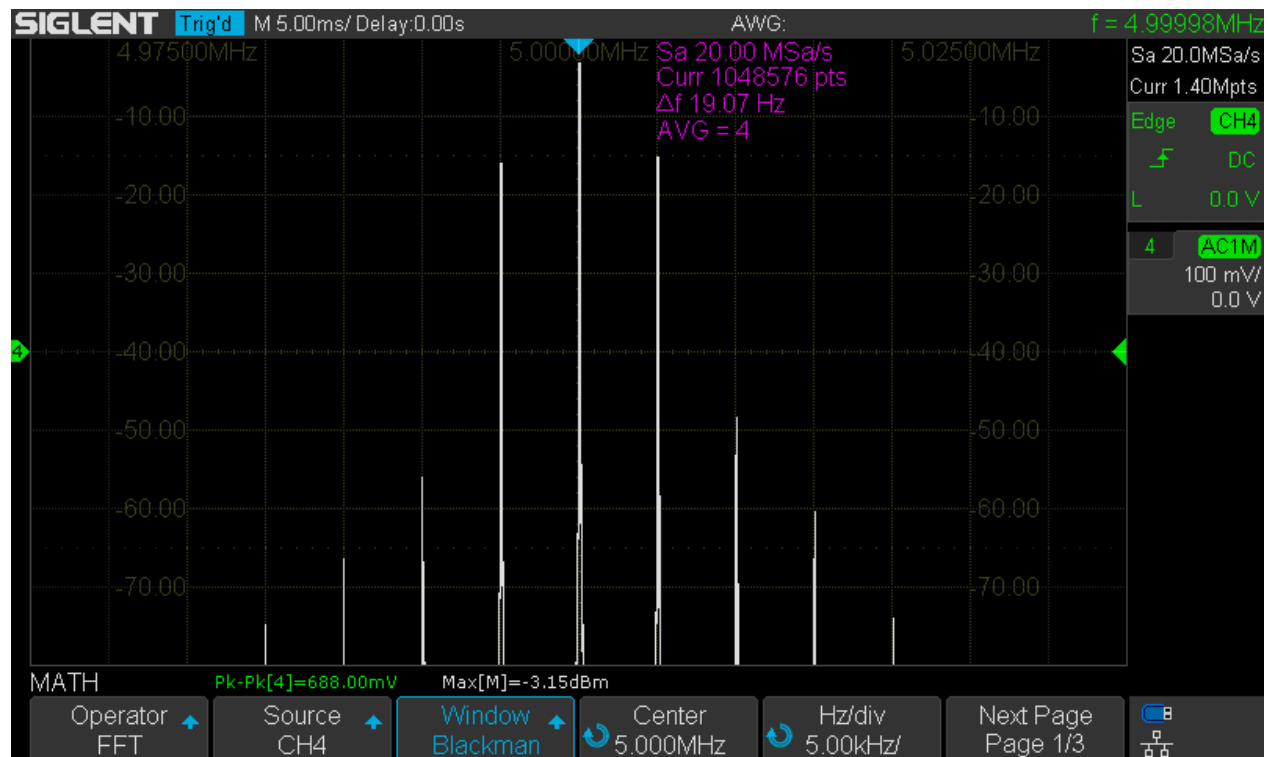
SDS1104X-E_FFT_Rectangle



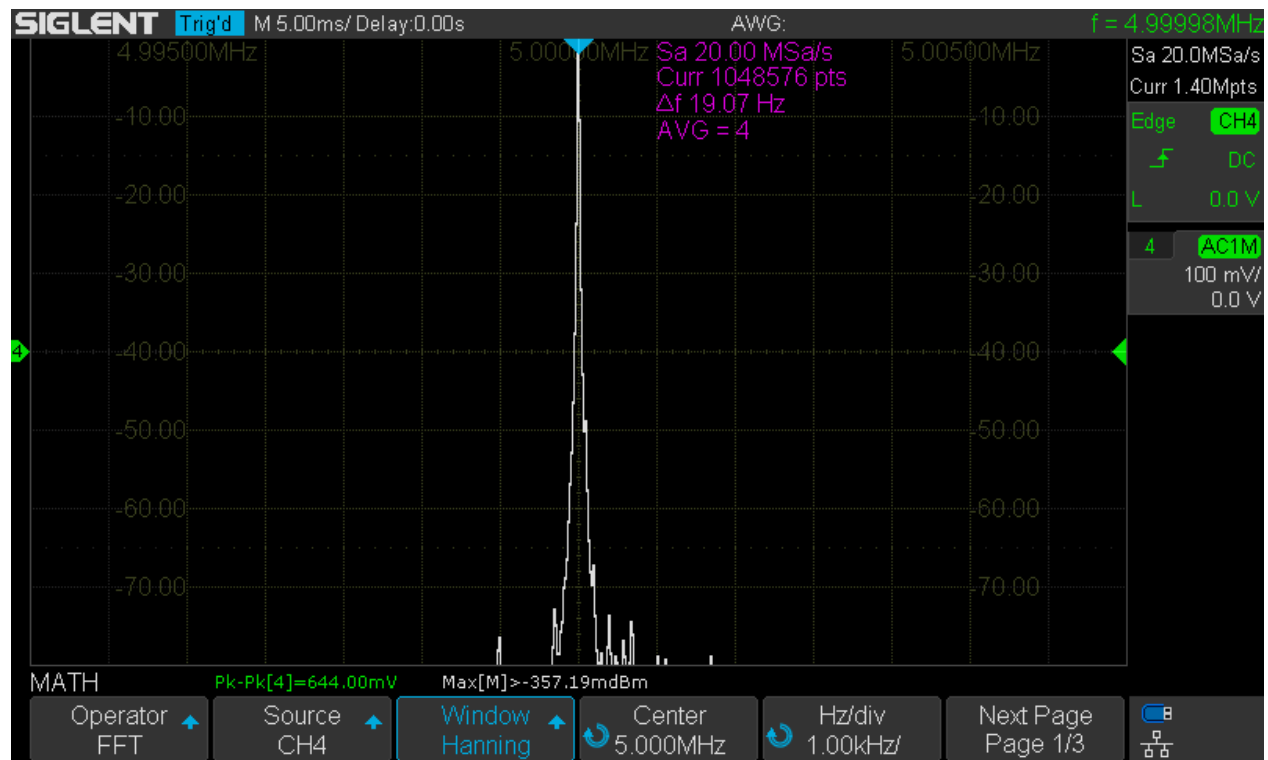
SDS1104X-E_FFT_Mod_Rectangle



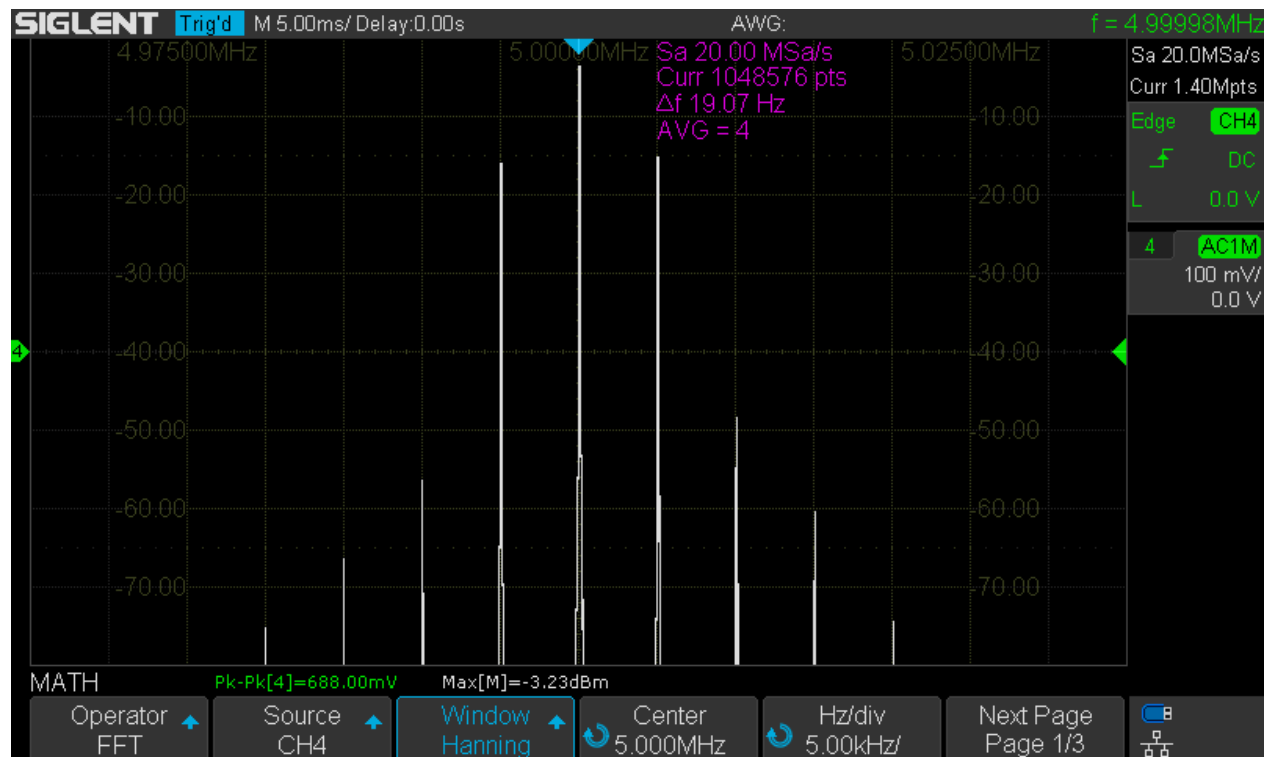
SDS1104X-E_FFT_Blackman



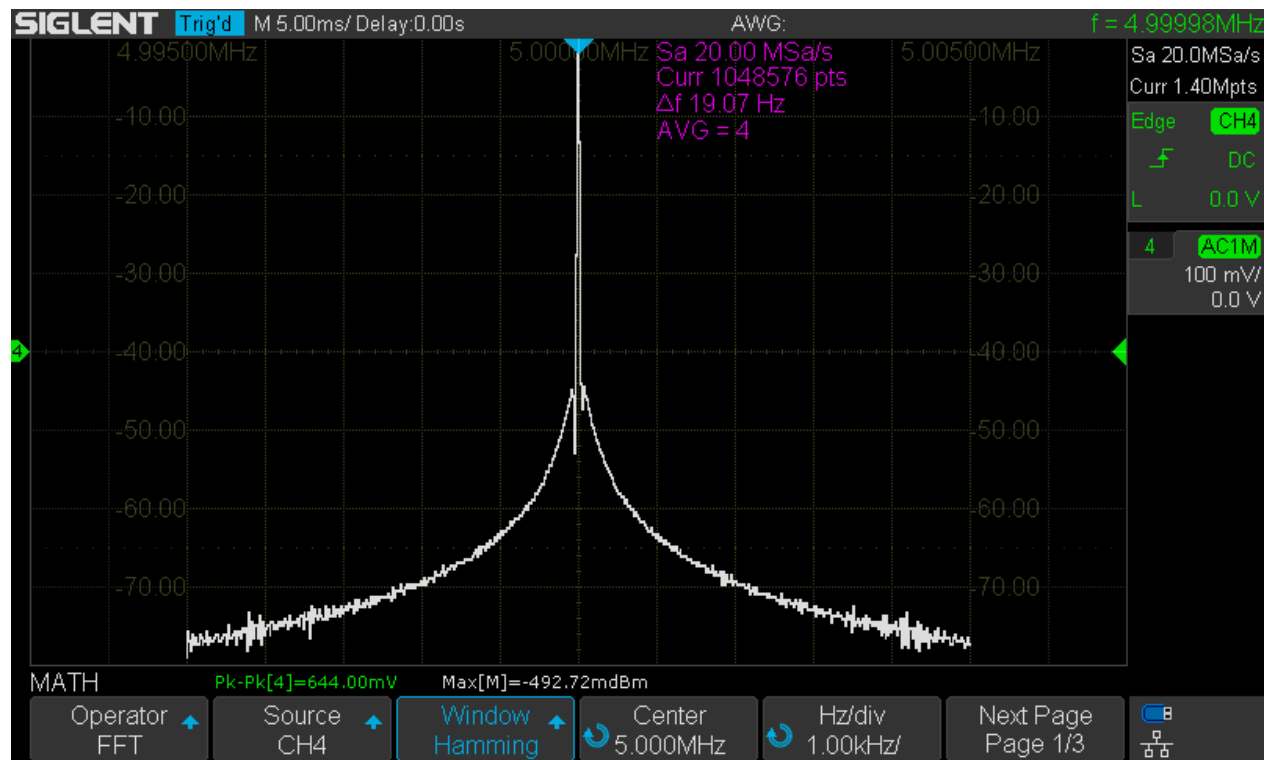
SDS1104X-E_FFT_Mod_Blackman



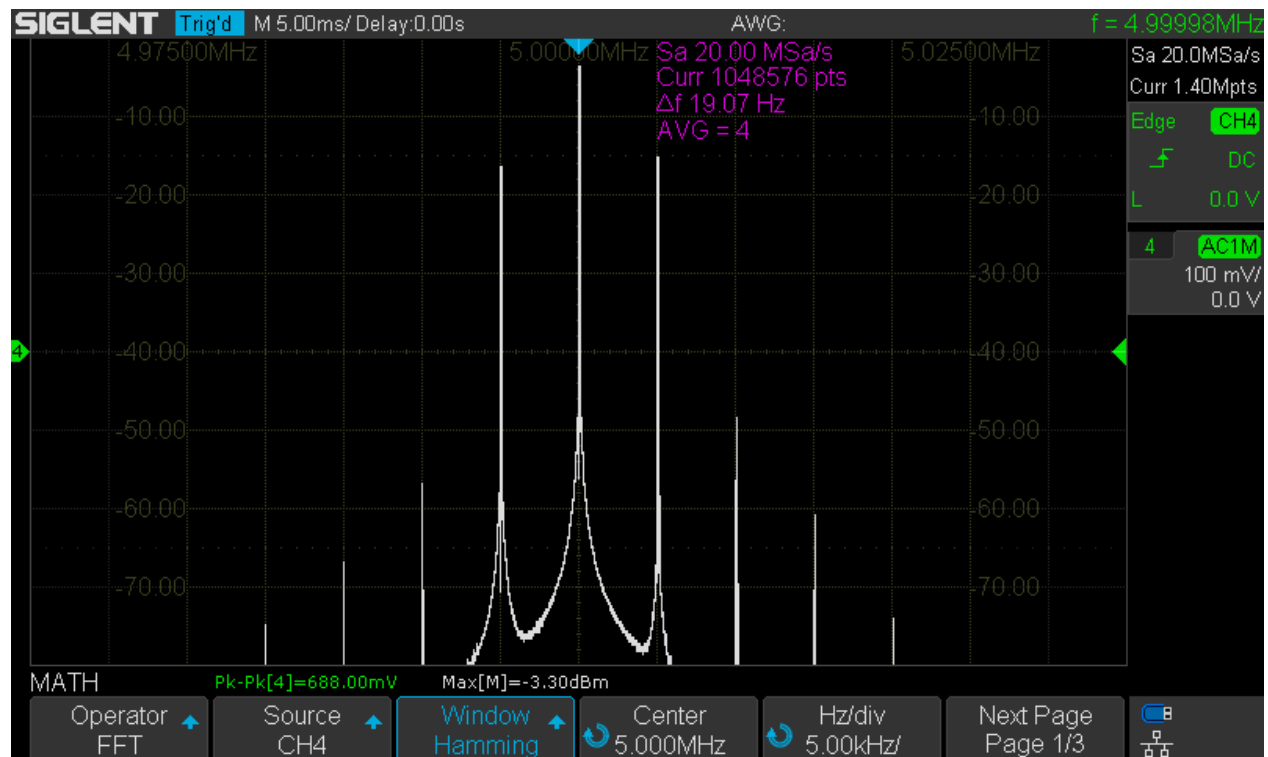
SDS1104X-E_FFT_Hanning



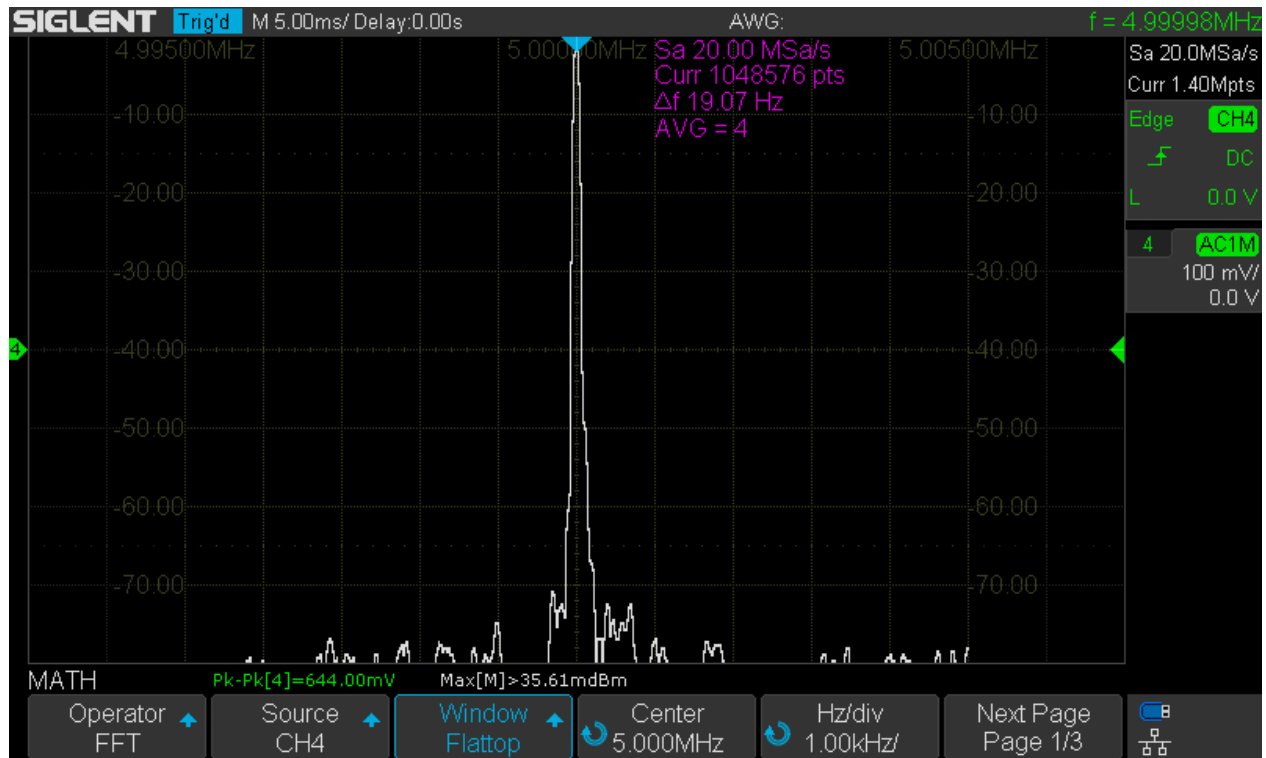
SDS1104X-E_FFT_Mod_Hanning



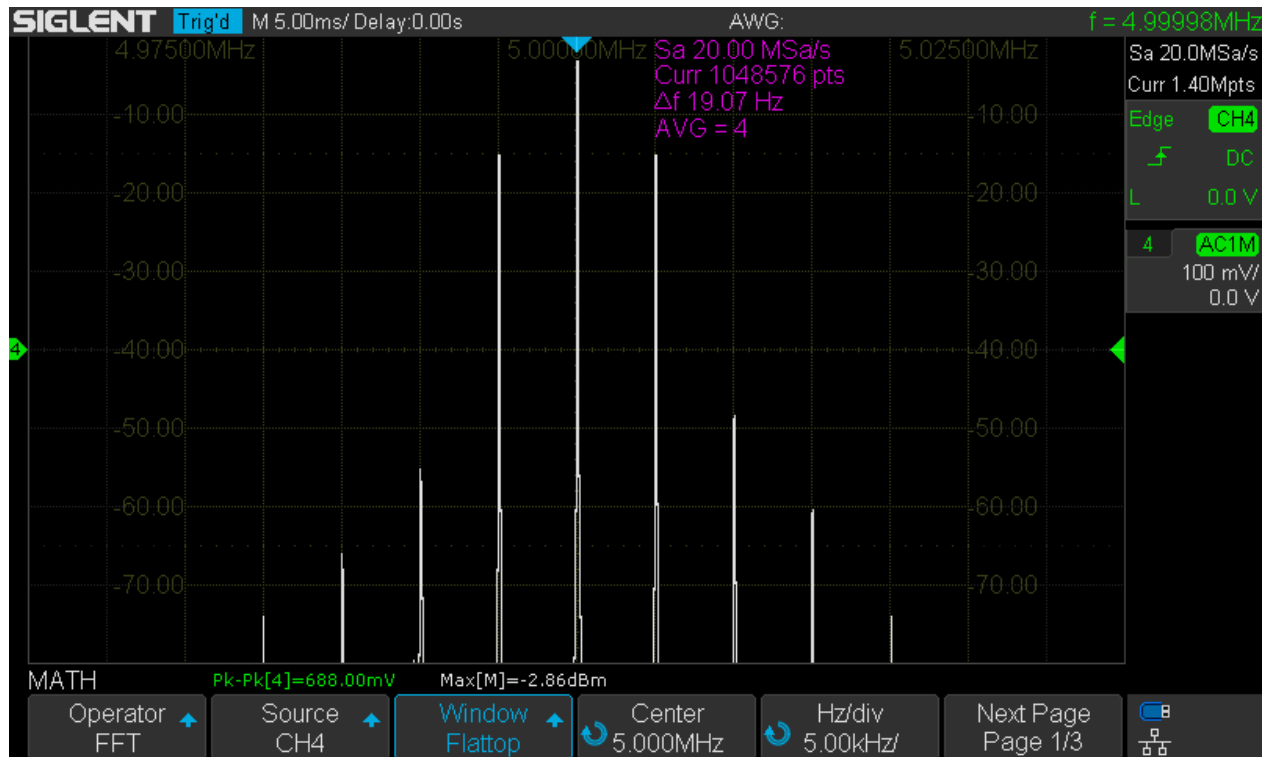
SDS1104X-E_FFT_Hamming



SDS1104X-E_FFT_Mod_Hamming



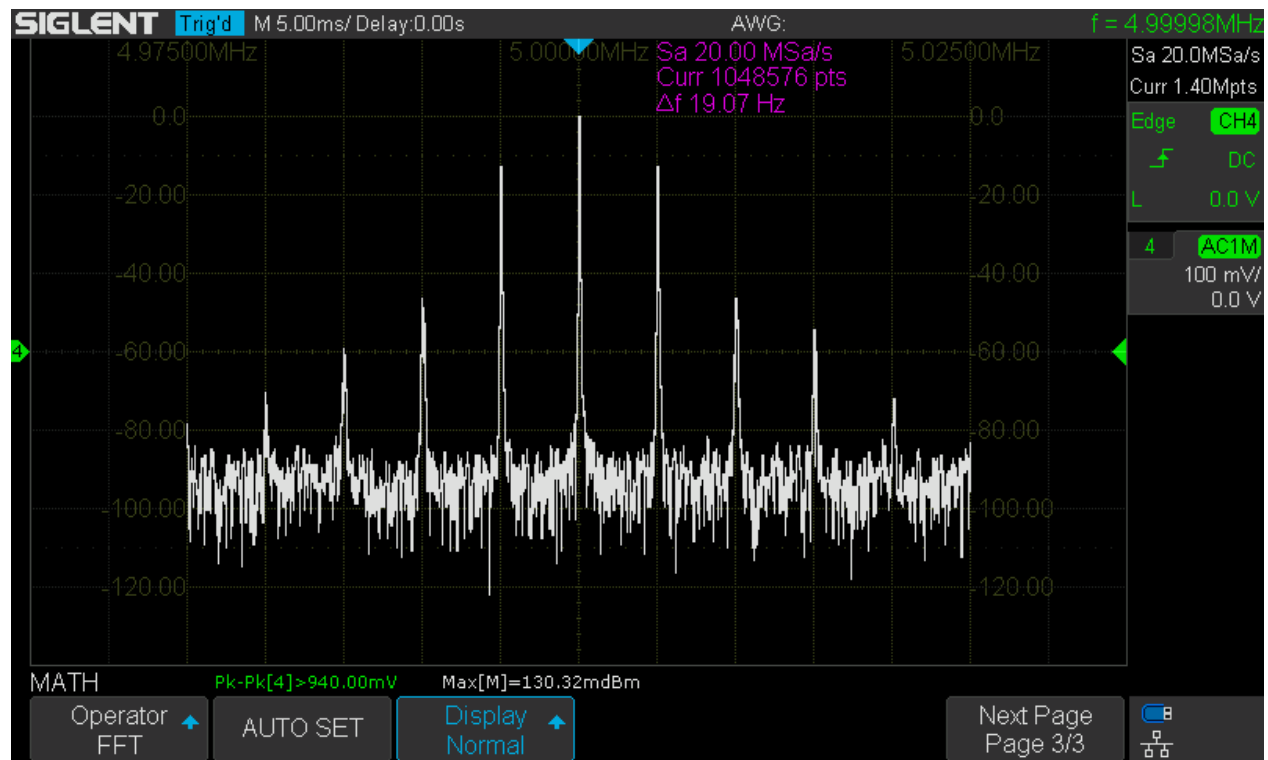
SDS1104X-E_FFT_Flattop



SDS1104X-E_FFT_Mod_Flattop

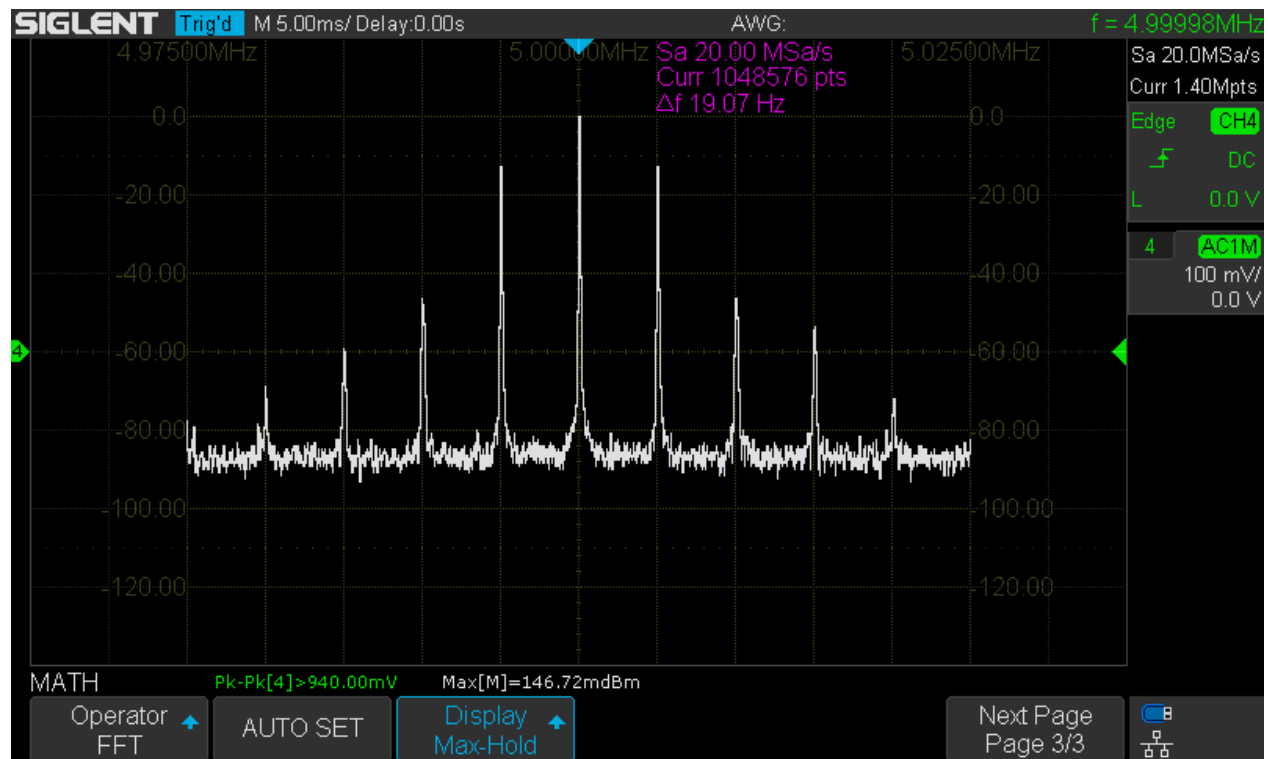
Display Modes

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



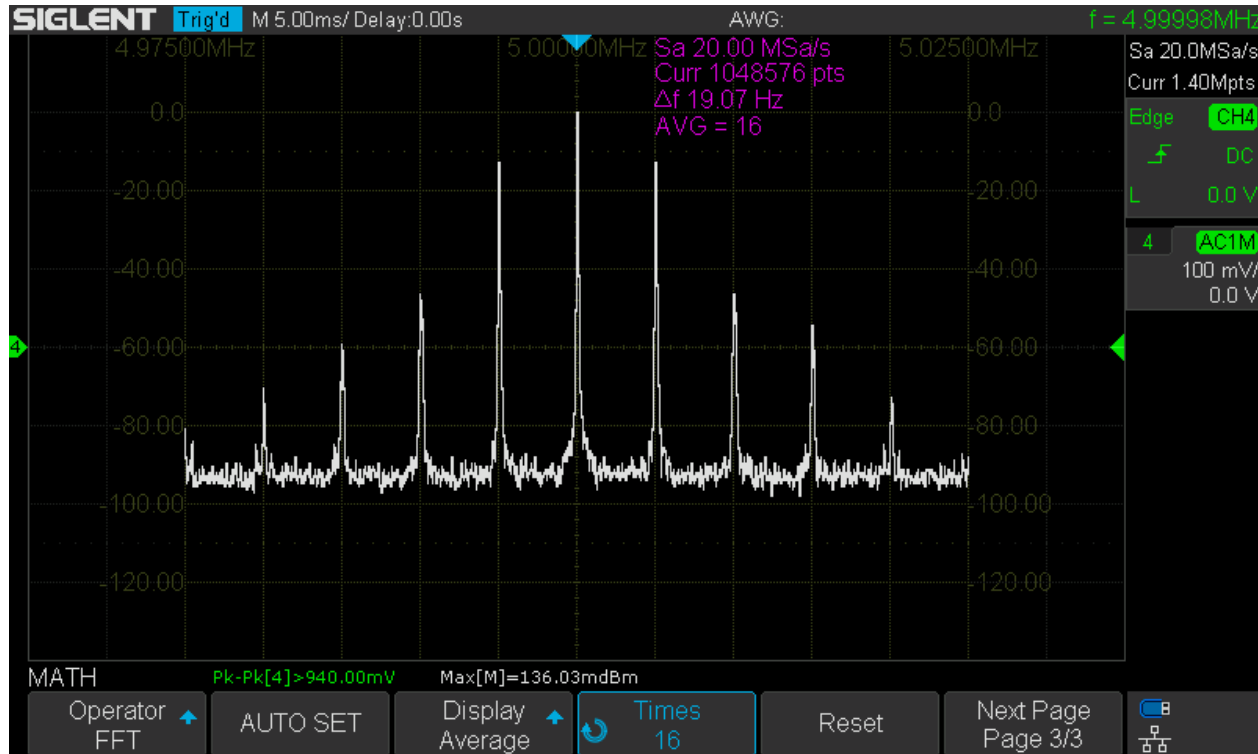
SDS1104X-E_FFT_Mode_Normal

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								
Timebase	14k		140k		1.4M		14M	
	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

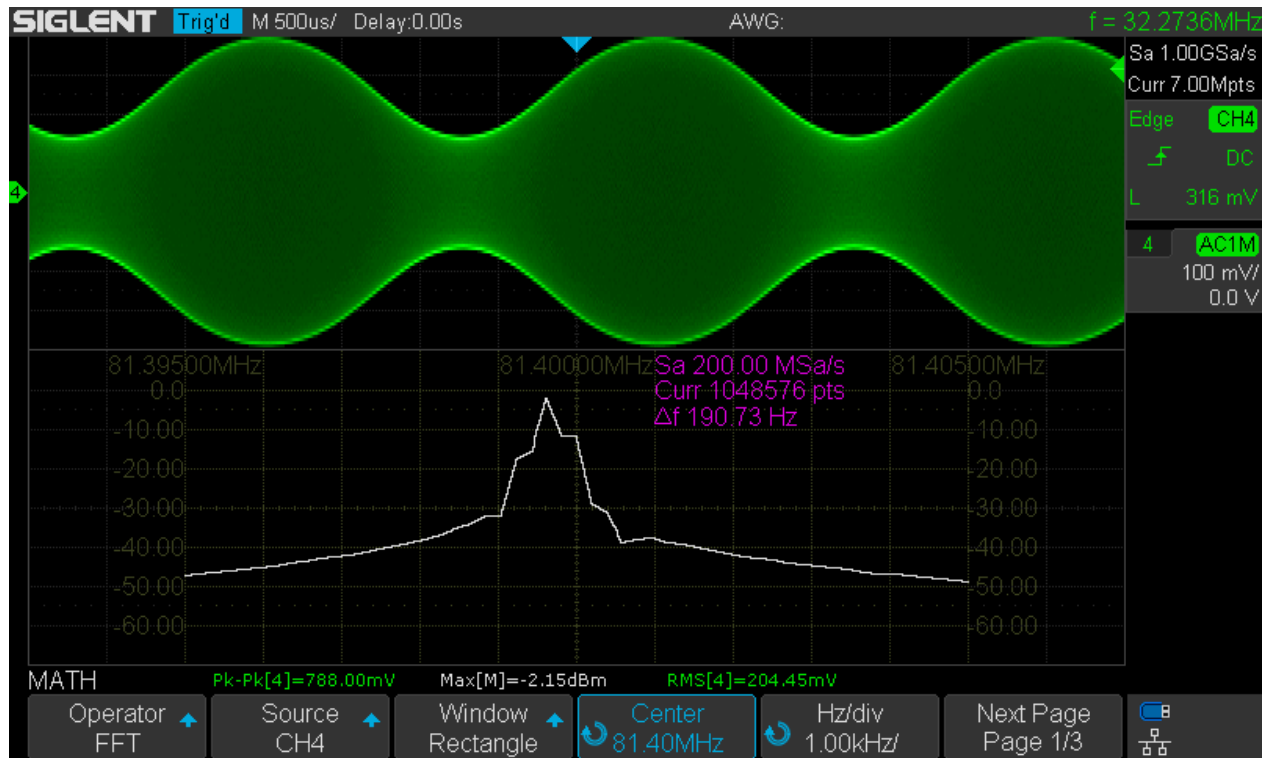
engaged and then some scope settings have been randomly altered just to get some halfway plausible but actually rather meaningless FFT graph, which was then either praised or criticized? Of course we can get away with some quick & dirty setup if we just want to get a quick overview, but for optimal speed, 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**.

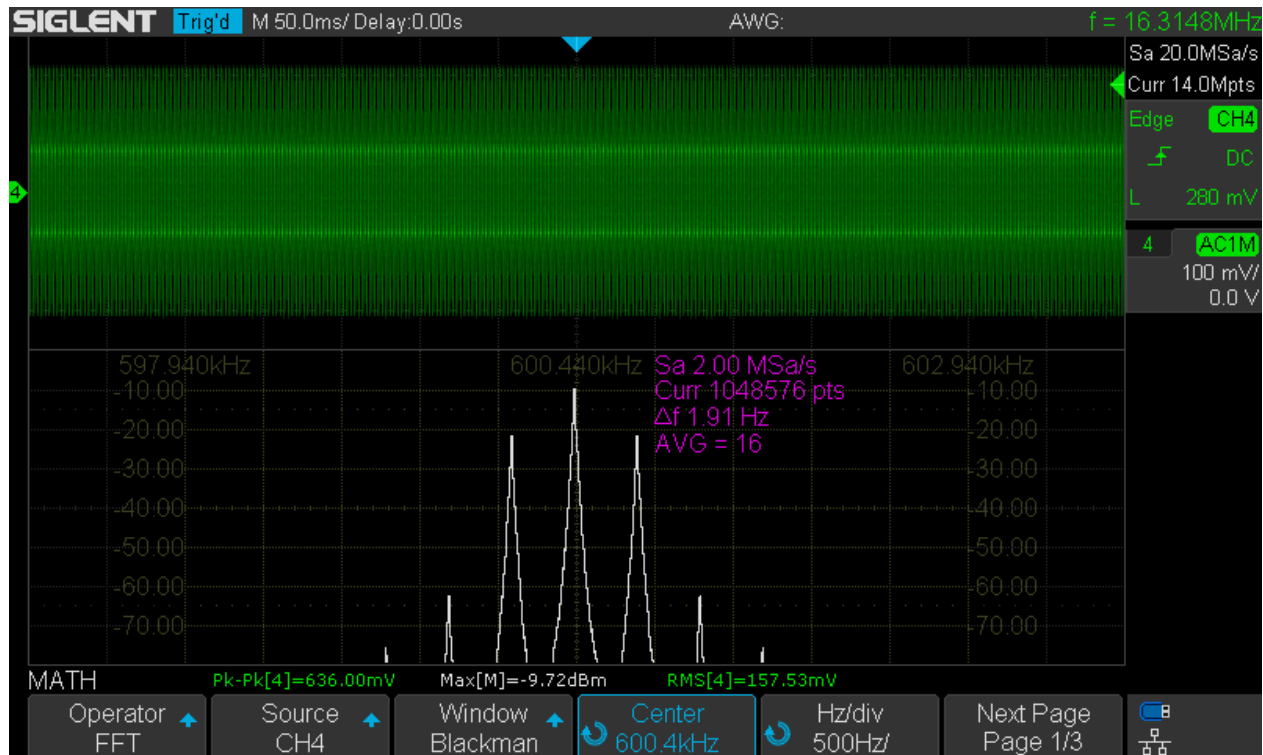
Looking at IF Signals

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 500 μ s/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 $\pm n \times f_i \pm m \times f_s$, where f_i is the

input frequency and f_s 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 $f_s/2$ and the output of the ADC has a brick-wall filter (then in the digital domain of course) that removes everything above $f_s/2$. 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 $-1 \times f_i + 41 \times f_s$, which is

$$41 \times 2\text{MHz} - 1 \times 81.4\text{MHz} = 82\text{MHz} - 81.4\text{MHz} = 600\text{kHz};$$

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.



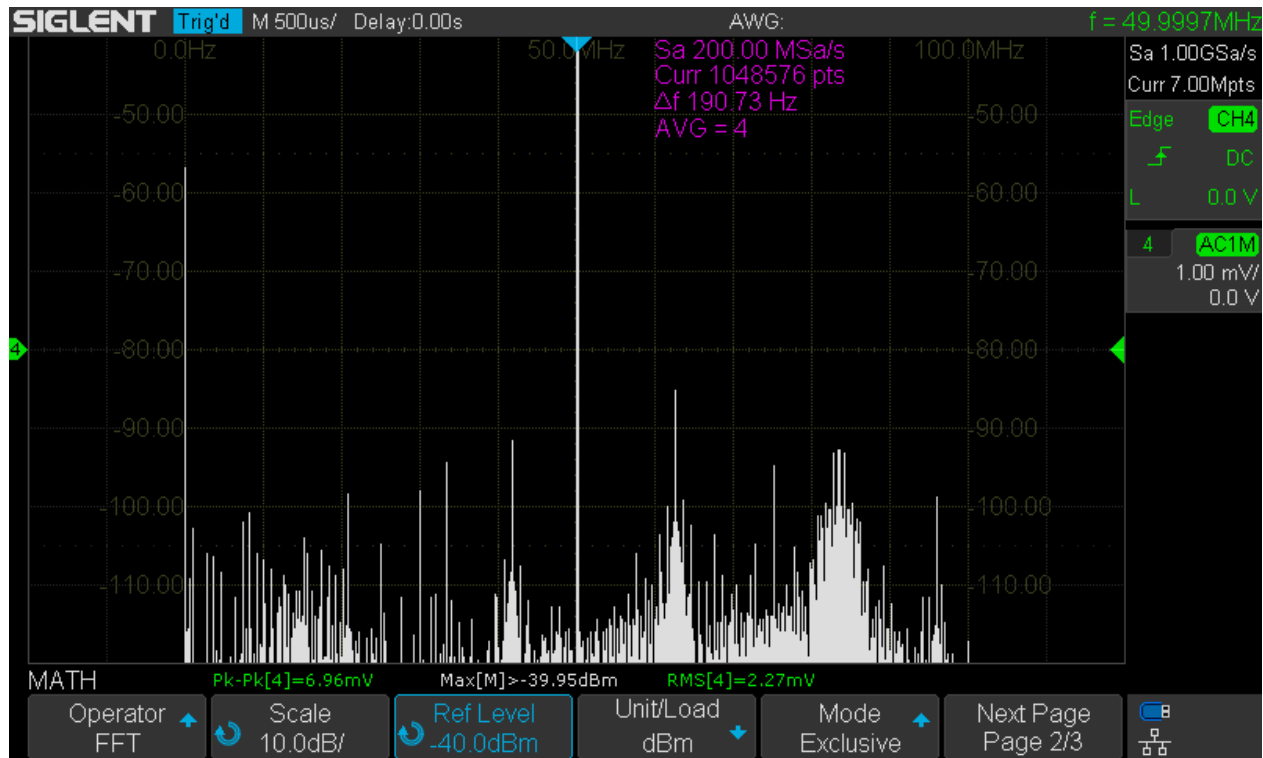
SDS1104X-E_FFT_100mV_Amp_-60dBm

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.

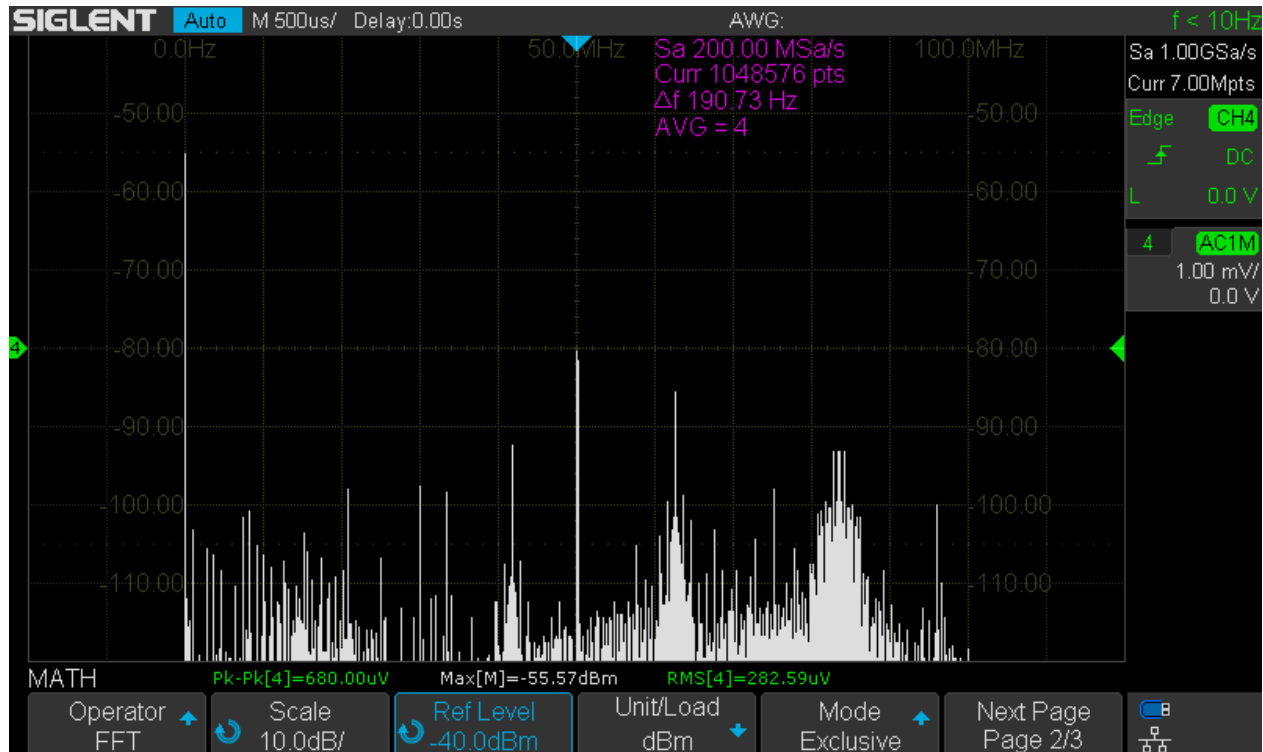
One might wonder how low we can go with the amplitude measurements. Up to now, we could demonstrate that it works fairly well down to -50dBm, which is just $707\mu\text{V}_{\text{rms}}$ or 2mV_{pp} . This is almost as sensitive as most AF or RF Millivolt meters with the added benefit of the selective 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 $7\mu\text{V}_{\text{rms}}$!

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 ($<22\mu\text{V}_{\text{rms}}$), 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 $10\mu\text{V}_{\text{rms}}$ whereas for known signals like in this test we would be able to go even lower and the proposed -90dBm should be realistic.

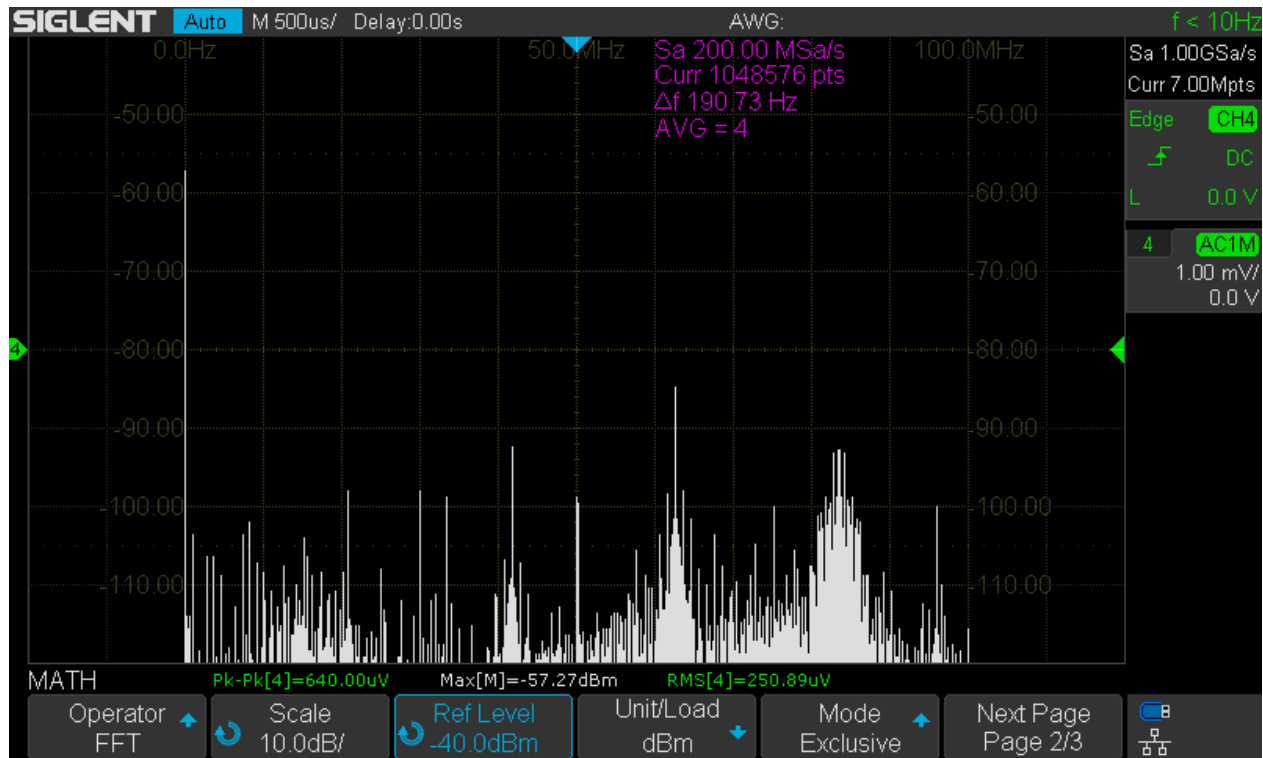


SDS1104X-E_FFT_1mV_Amp_-40dBm

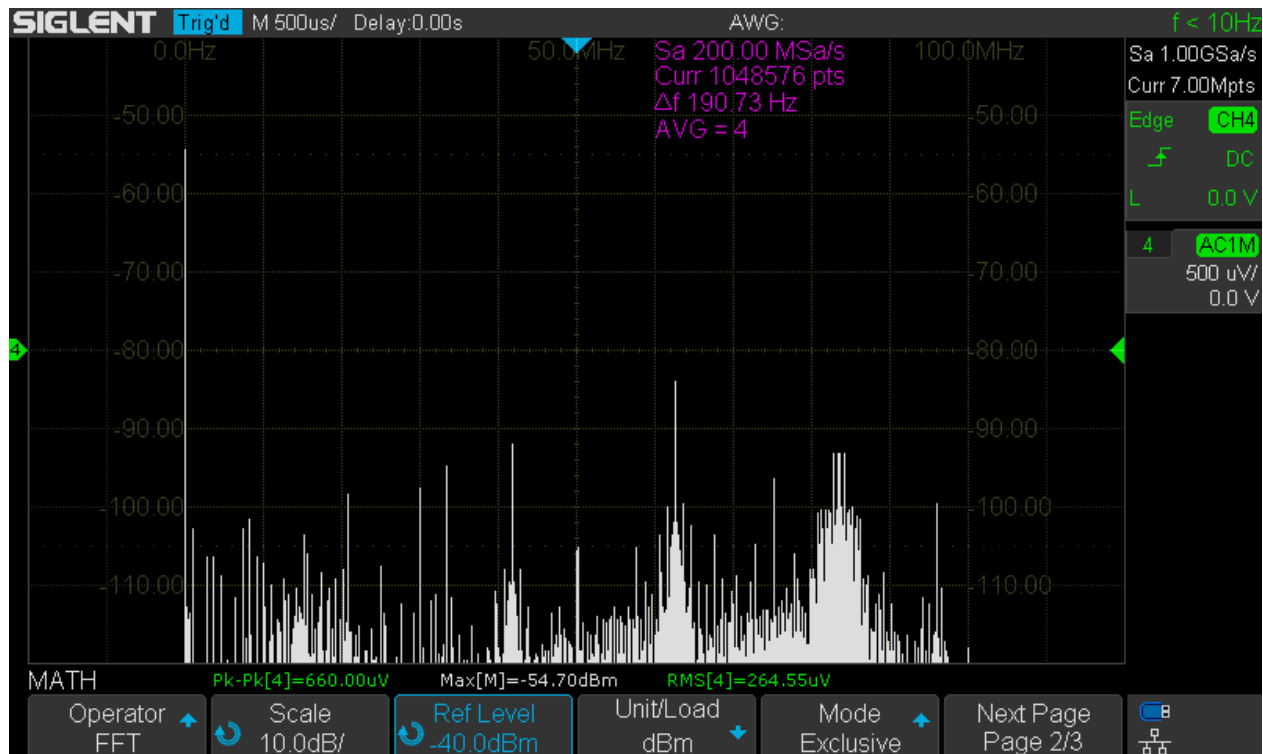


SDS1104X-E_FFT_1mV_Amp_-80dBm

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



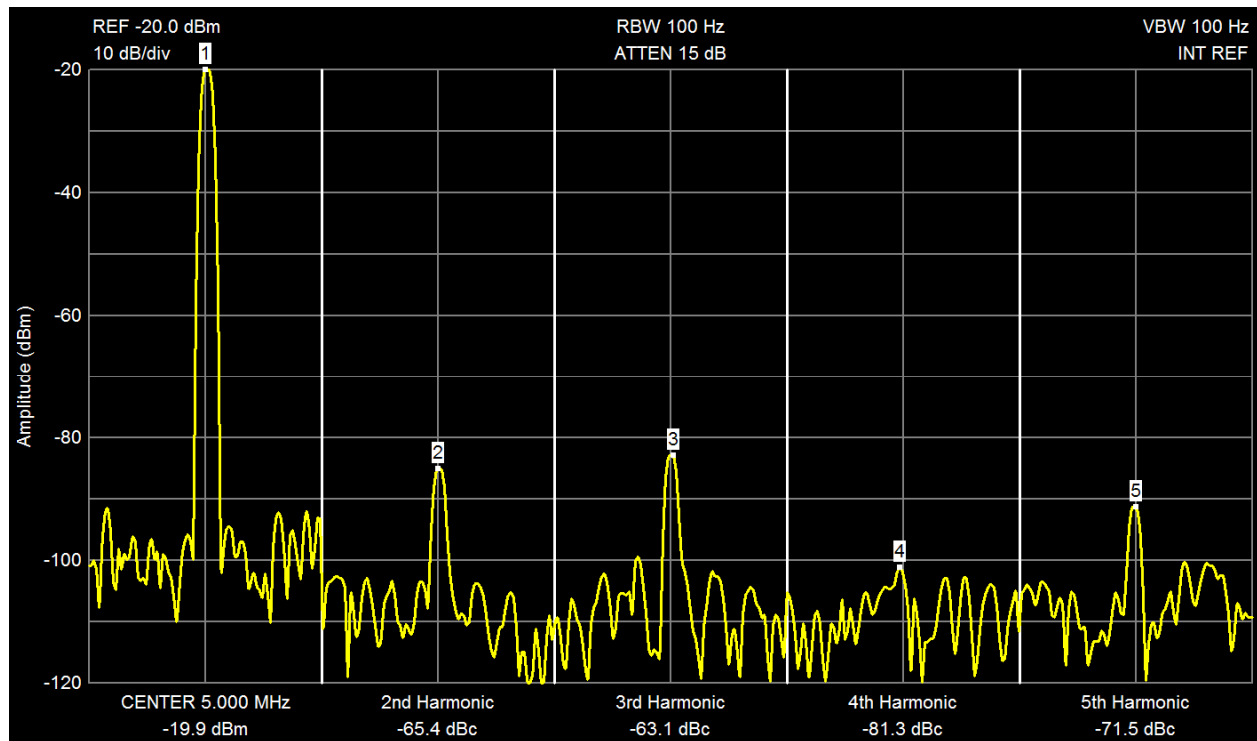
SDS1104X-E_FFT_500uV_Amp_-110dBm

As can be seen in the screenshots above, we can even measure the level of a known signal at -100dBm, that is $2,24\mu\text{Vrms}$ or $6,32\mu\text{Vpp}$. 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 $500\mu\text{V/div}$.

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

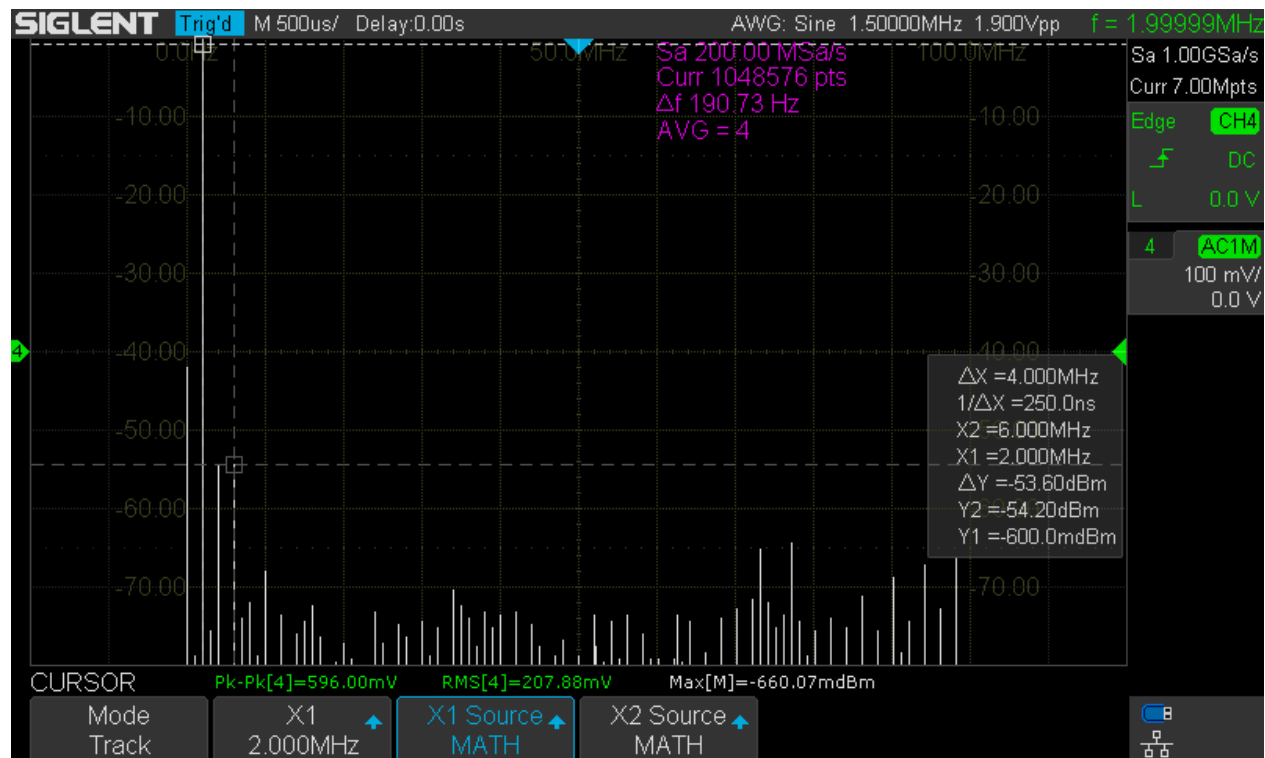
All the harmonics stay well below -60dBc at that frequency, which should be good enough for characterizing an 8-bit system.

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.

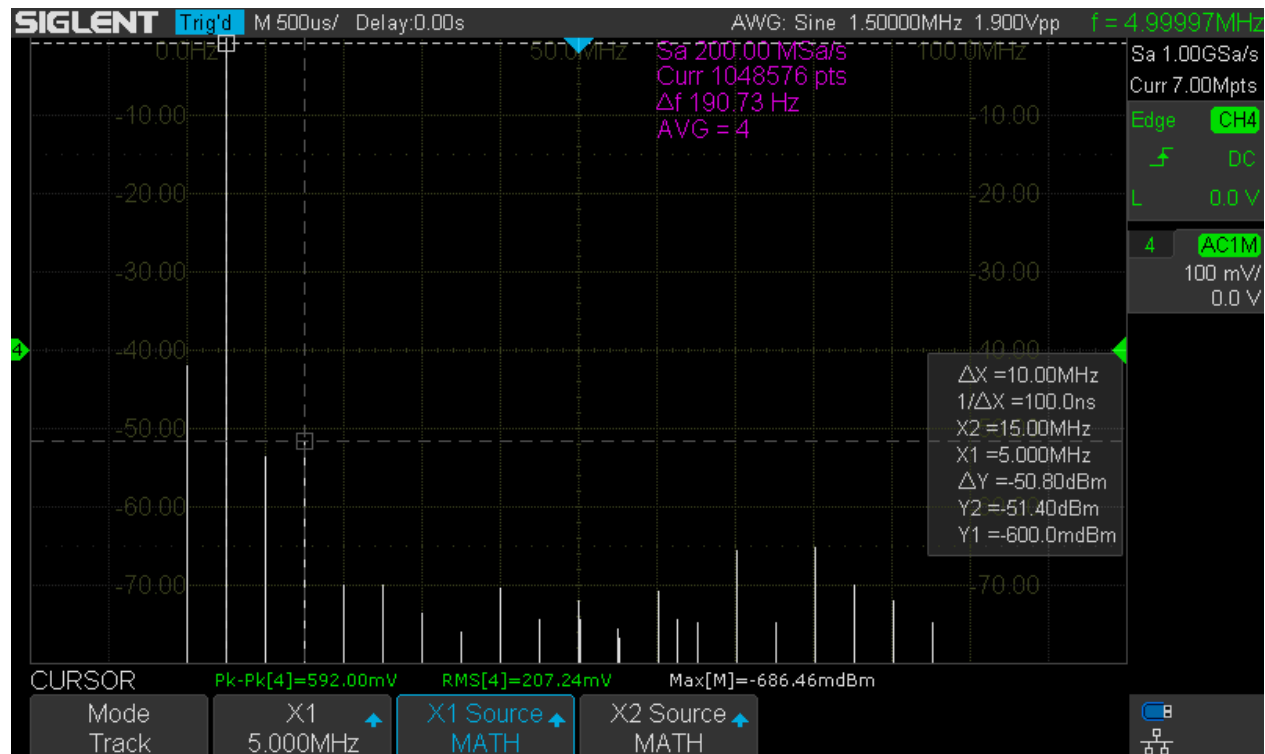
No automatic measurements for the harmonics are available, but tracking cursors on the math trace are a near perfect tool to both measure and highlight (in the screenshots) the strongest harmonic.

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!

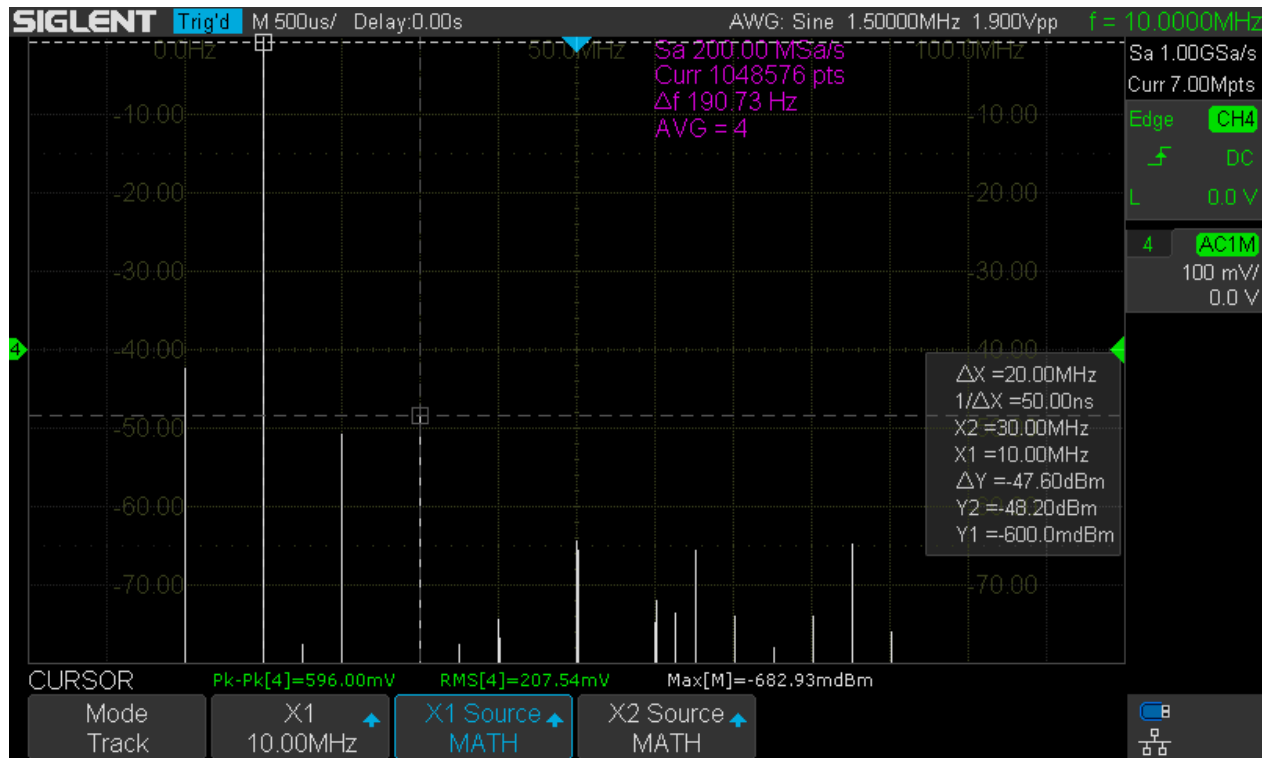
The conclusion can only be, while this scope certainly isn't up to the task of characterizing low distortion sine waves, high fidelity audio gear or high performance RF circuits, it still performs pretty good for the majority of undemanding tasks with less strict requirements.



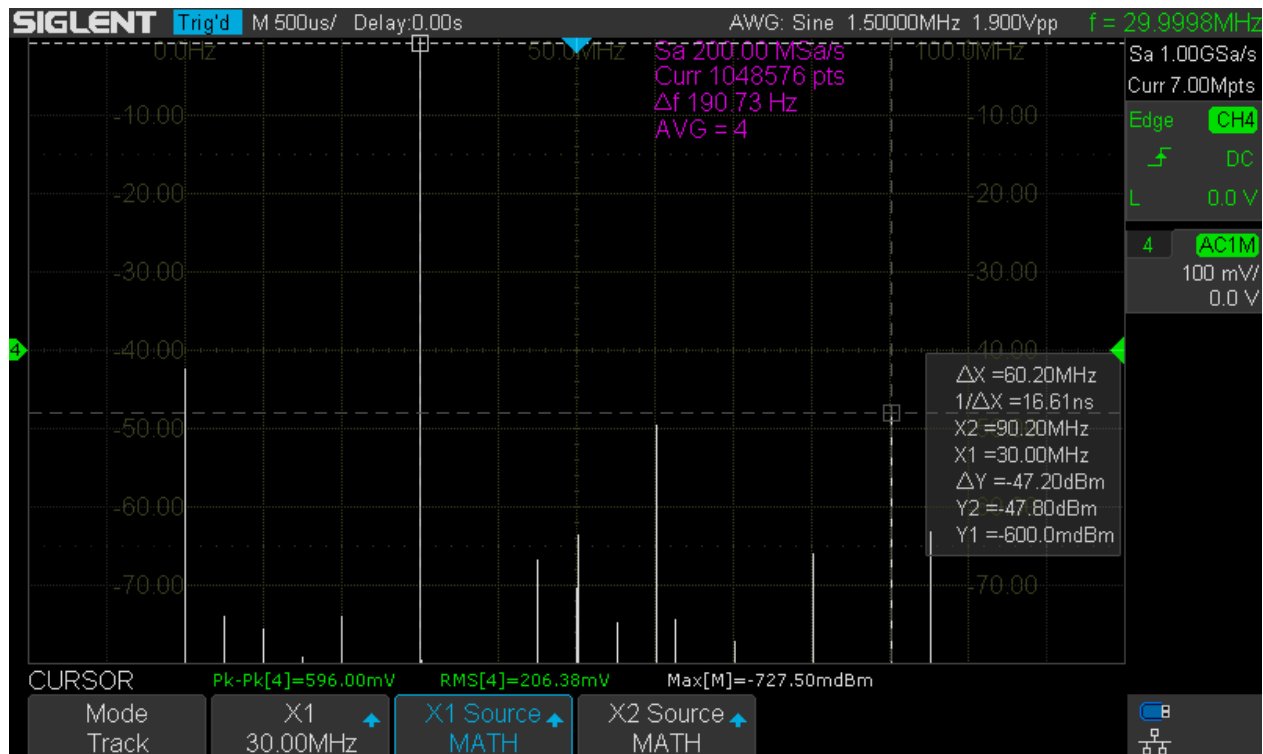
SDS1104X-E_FFT_THD_2MHz



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.

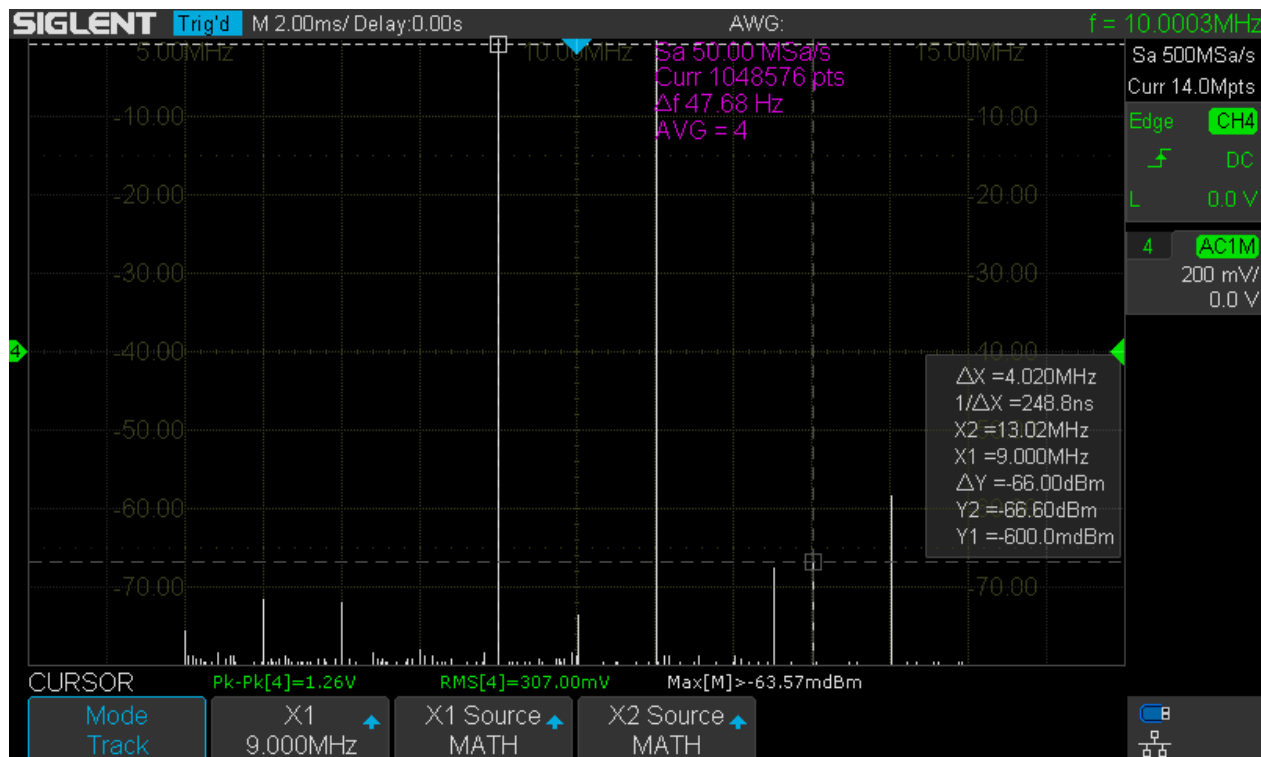
This test uses two input signals at 9 and 11MHz, both at a 0dBm level initially. The 9MHz signal is then attenuated step by step in order to check the amplitude accuracy of the measurements. At the same time, the 3rd order mixing products at 7 and 13MHz are measured, so the resulting IMD (intermodulation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

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

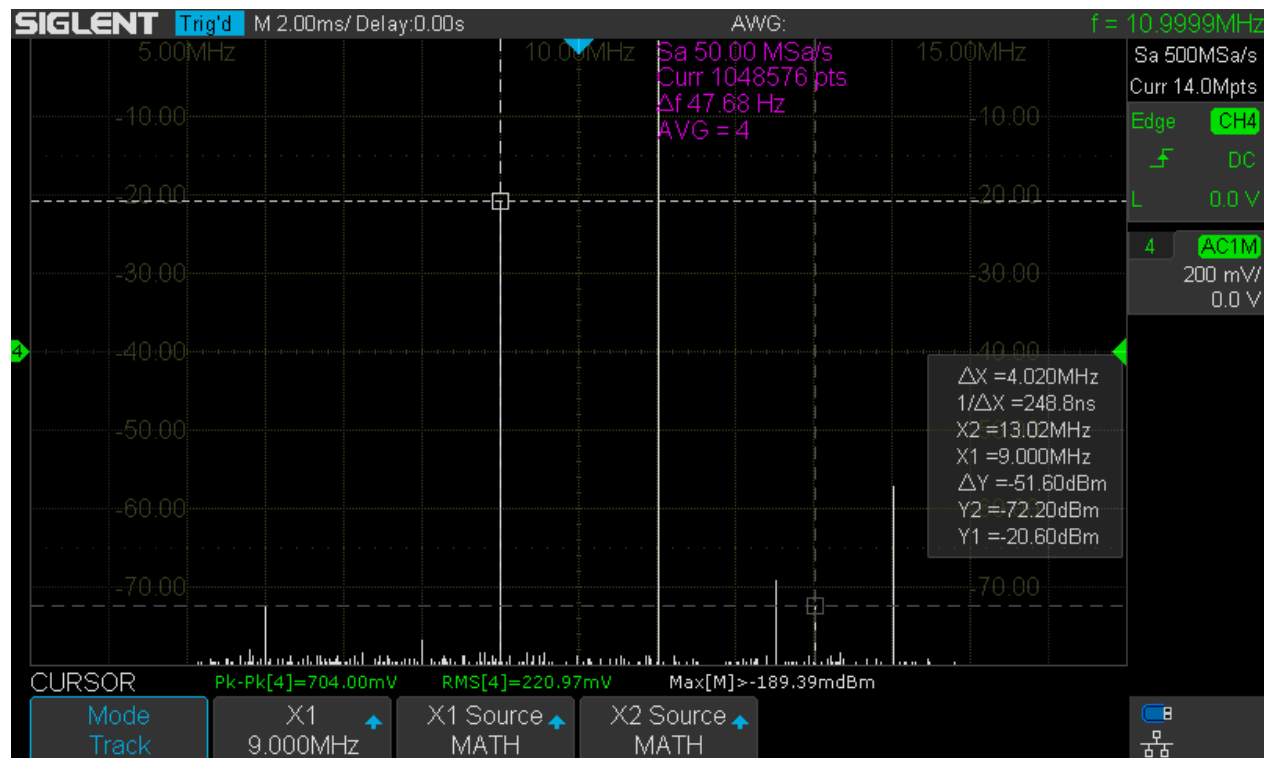
The 0dBm test would be the classical setup for determining the 3rd order intercept point. The unwanted product at 7MHz is weaker at -72dB, but the opposite intermodulation product at 13MHz isn't strong either at only -66dB. This means a 3rd order intermodulation distortion of -66dB at 0dBm input level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

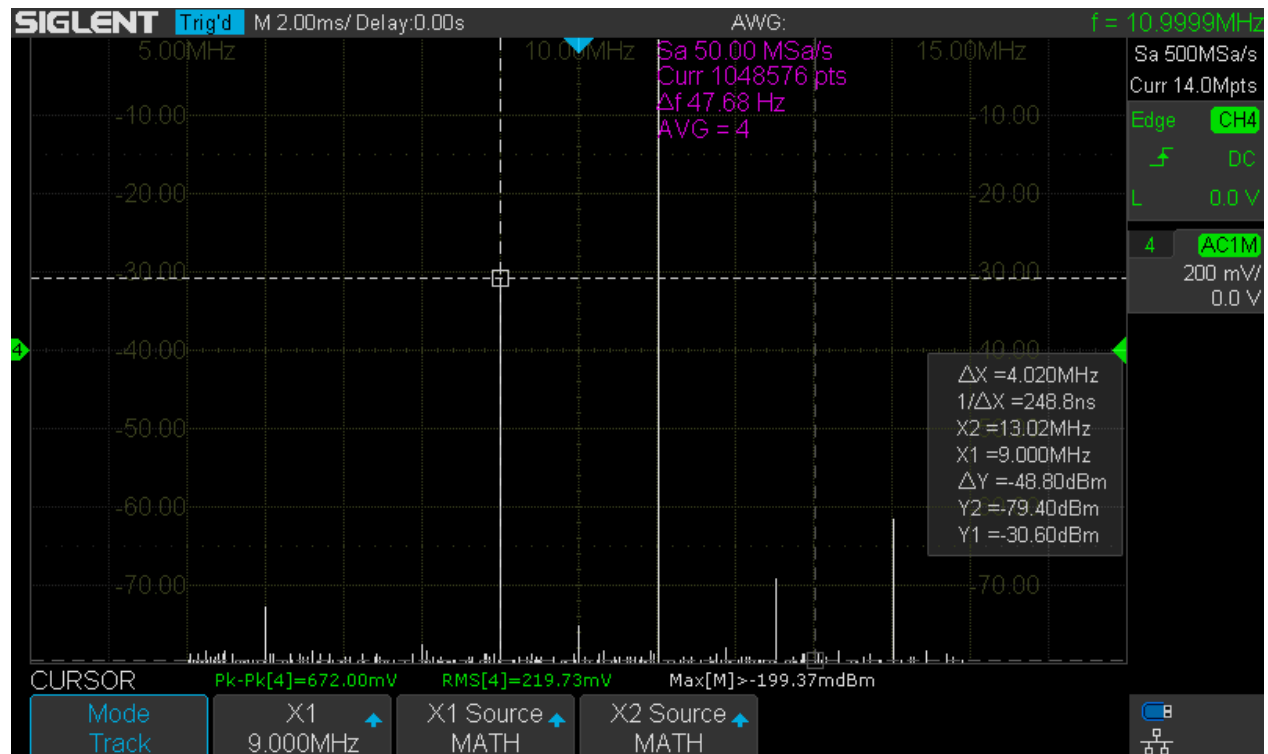
We can also measure levels below -50dBc quite accurately, simply because the 2nd signal stays fixed at 0dBm and serves as a dither for the ADC. As can be seen, measuring -70dBc isn't a problem at all. Just for fun, I've added a measurement at -76dBm level, which still works surprisingly well.



SDS1104X-E_FFT_IM3_0dBm



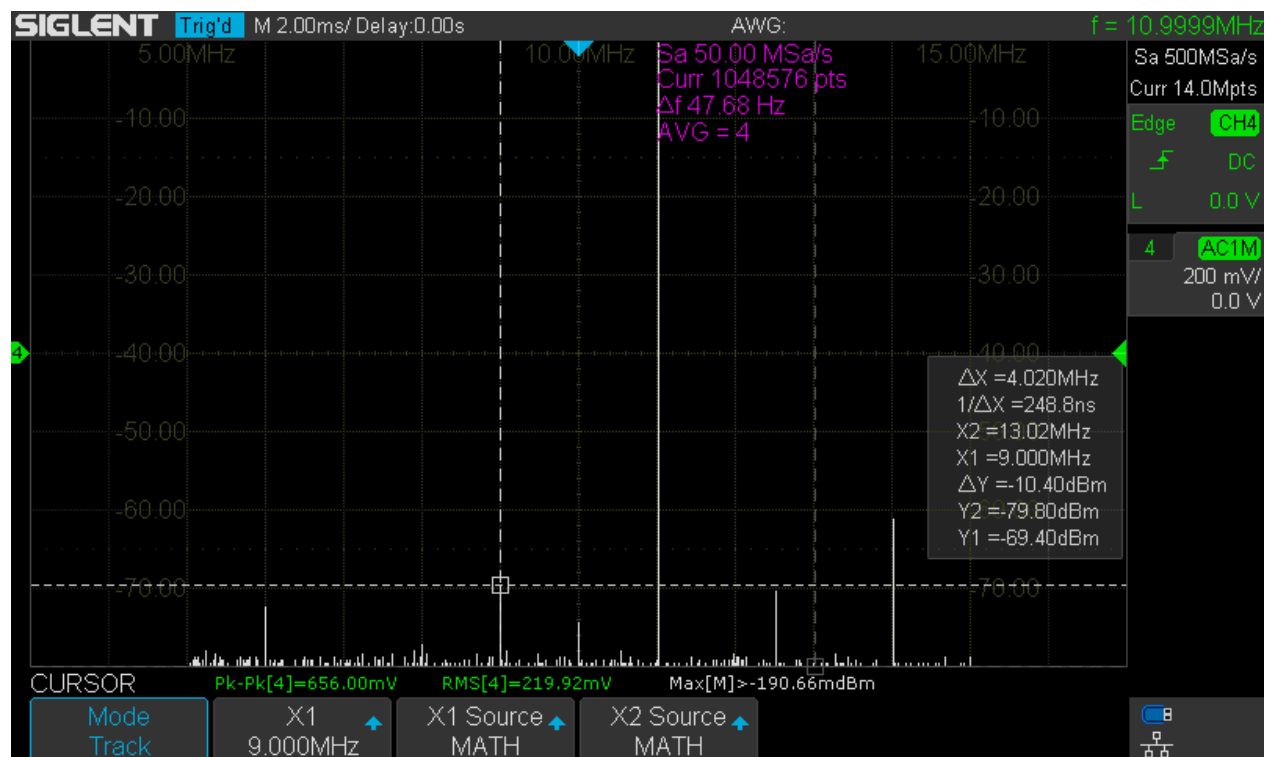
SDS1104X-E_FFT_IM3_-20dBm



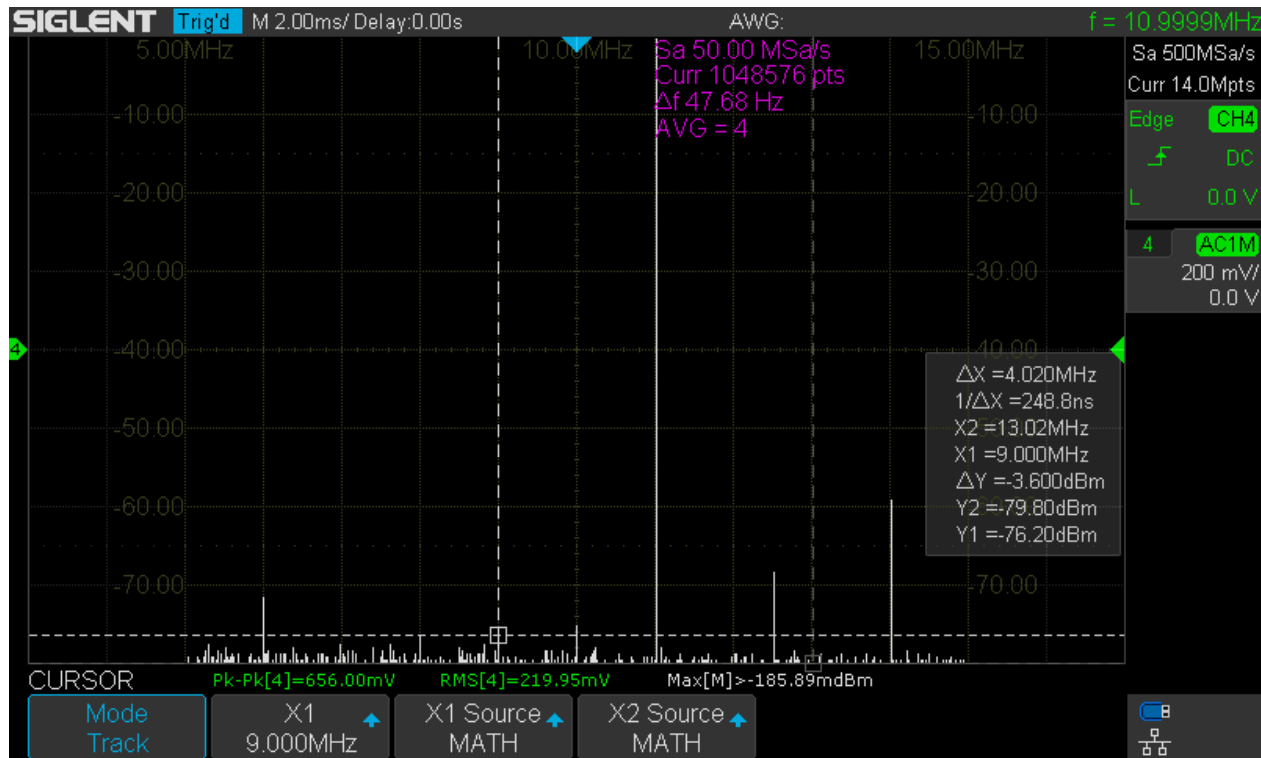
SDS1104X-E_FFT_IM3_-30dBm



SDS1104X-E_FFT_IM3_-60dBm



SDS1104X-E_FFT_IM3_-70dBm



SDS1104X-E_FFT_IM3_-76dBm

Application Examples

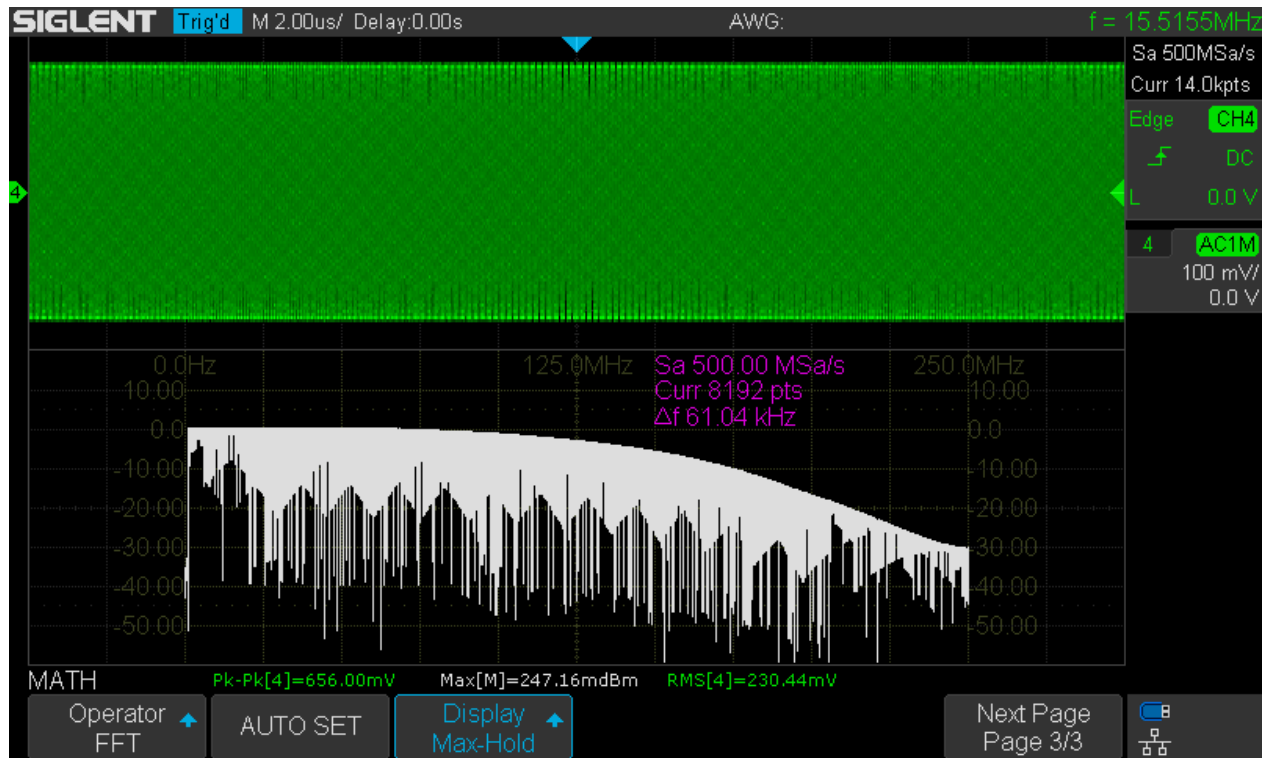
In this chapter we're looking at some more practical examples for using the SDS1104X-E FFT.

Wideband Measurement

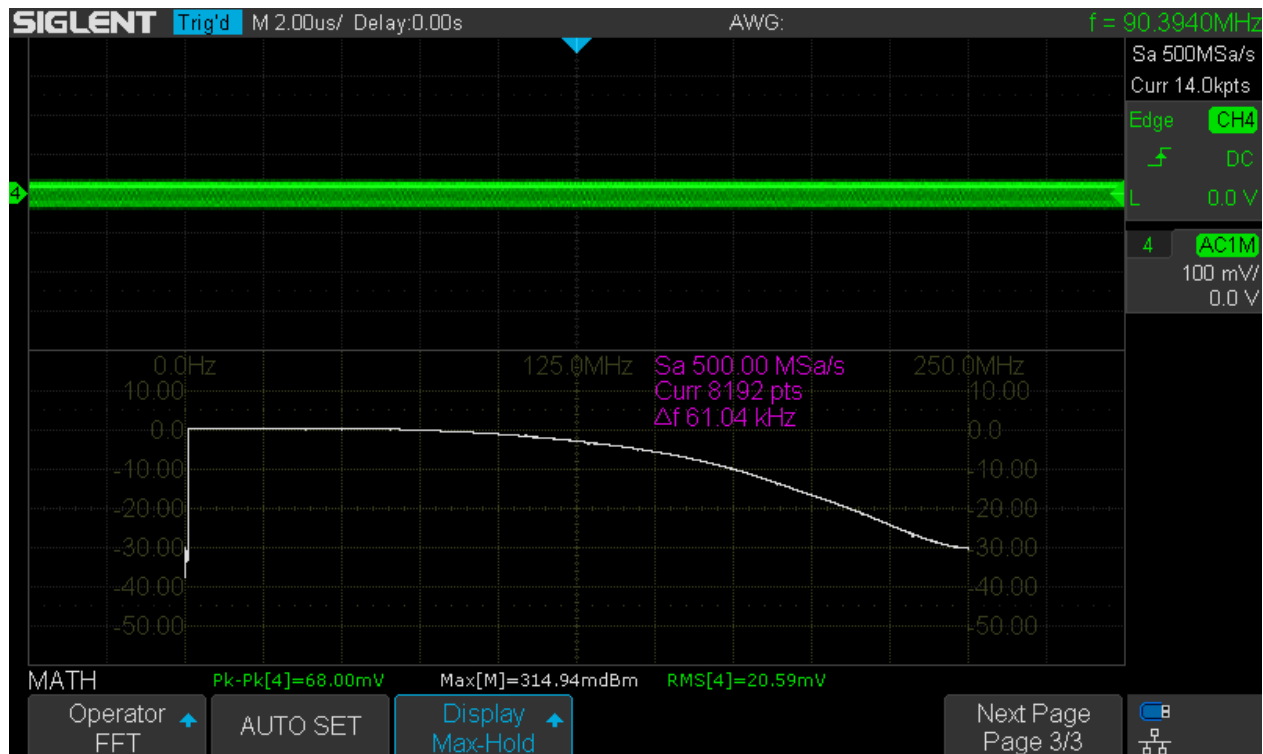
We can plot the frequency response of the SDS1104X-E in the range of 1MHz to 250MHz using its FFT and an external signal source that covers the entire frequency range. There are three possibilities:

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 $f \times n$, 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



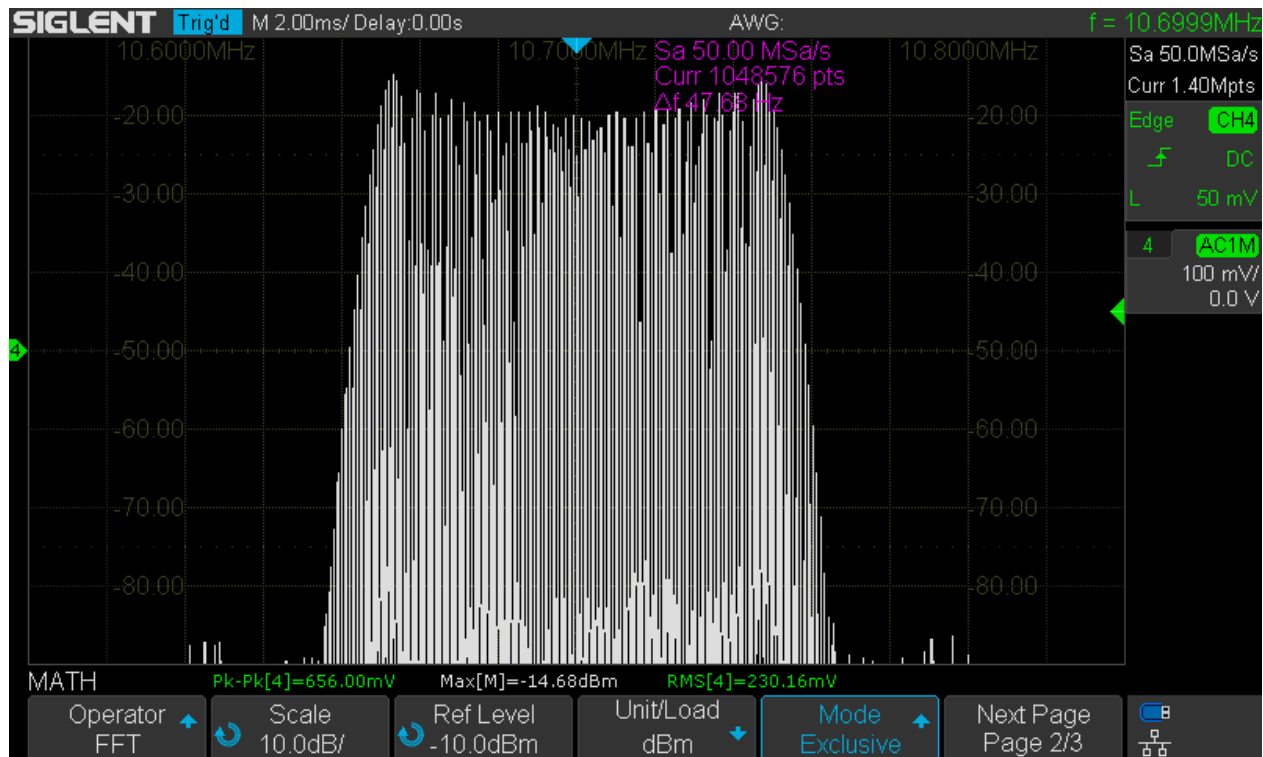
SDS1104X-E_FFT_Sweep_1M-250MHz_final

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

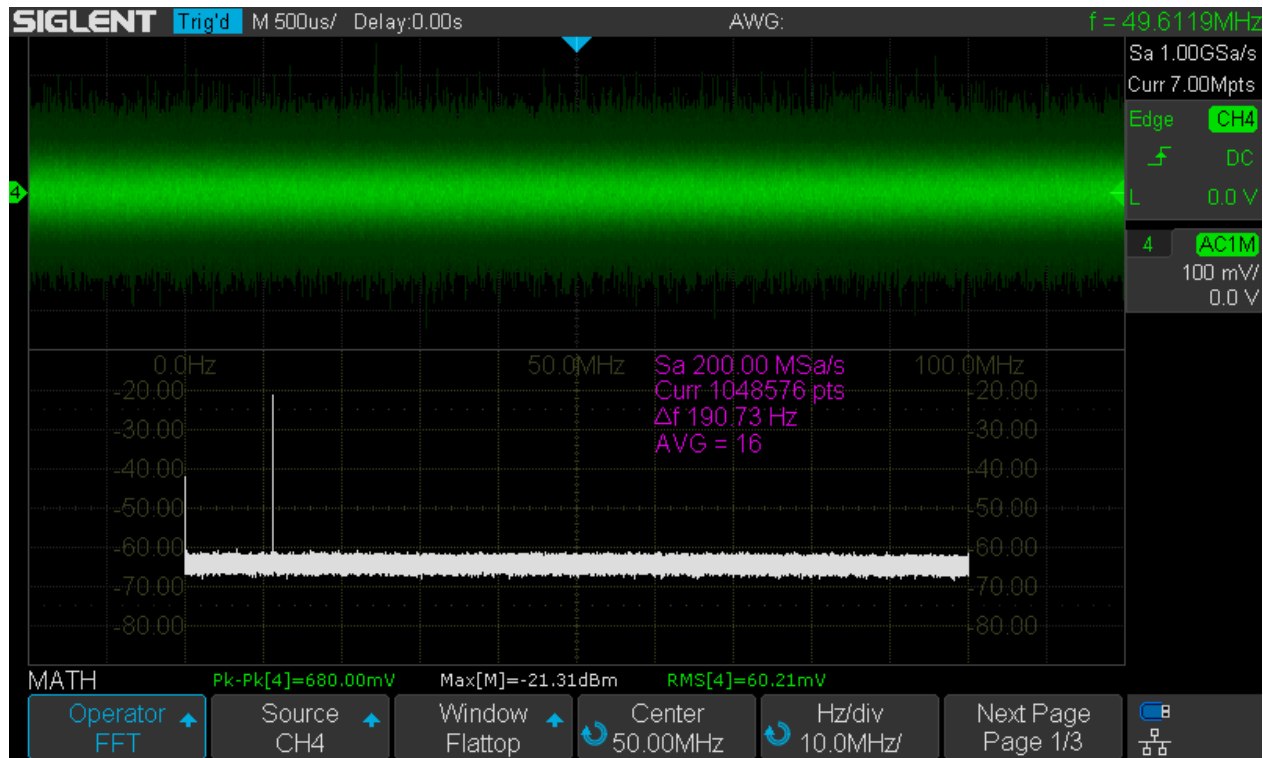
We can look at the IF signal of a domestic FM receiver at 10.7MHz, with a 1kHz frequency modulation and 50kHz max. deviation. For this we choose an analysis bandwidth of 25MHz – 12.5MHz would be even better, but we'd need to enable the 2nd channel in the group (Ch. 3 in this example) since this BW is 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

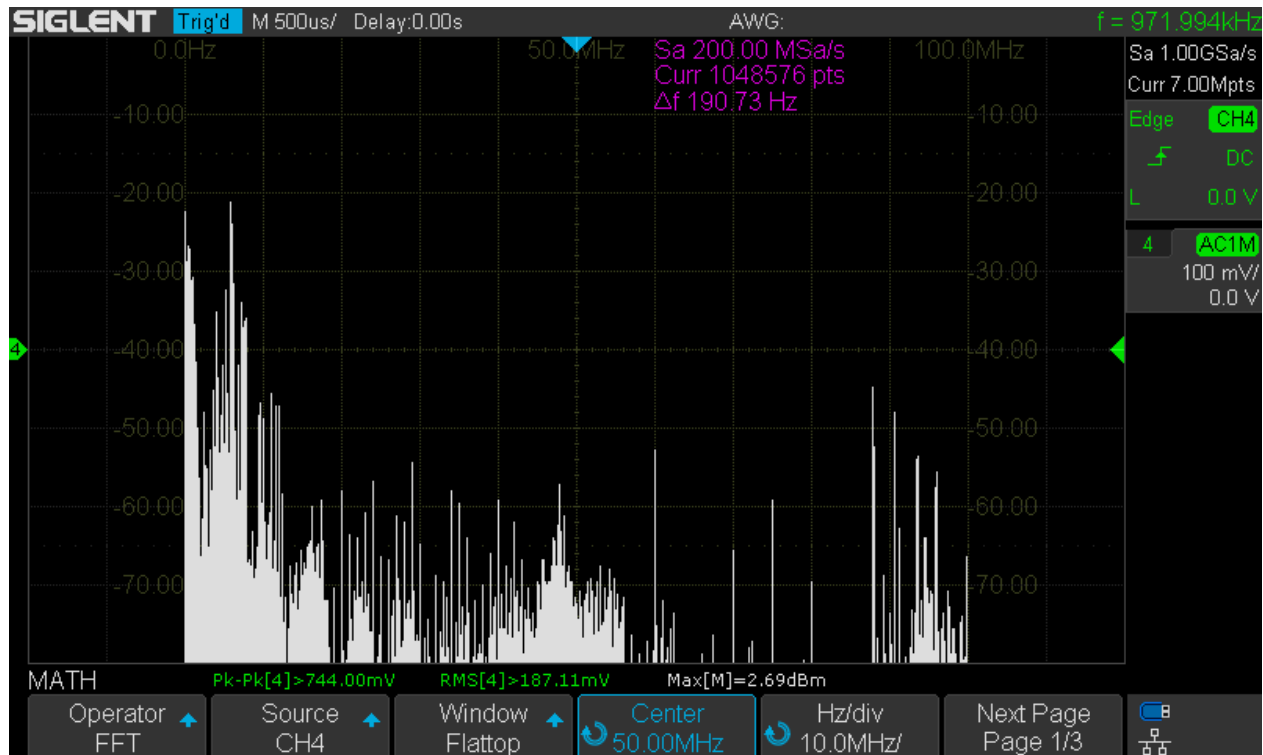
Signal buried in Noise

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 [s ⁻²]
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

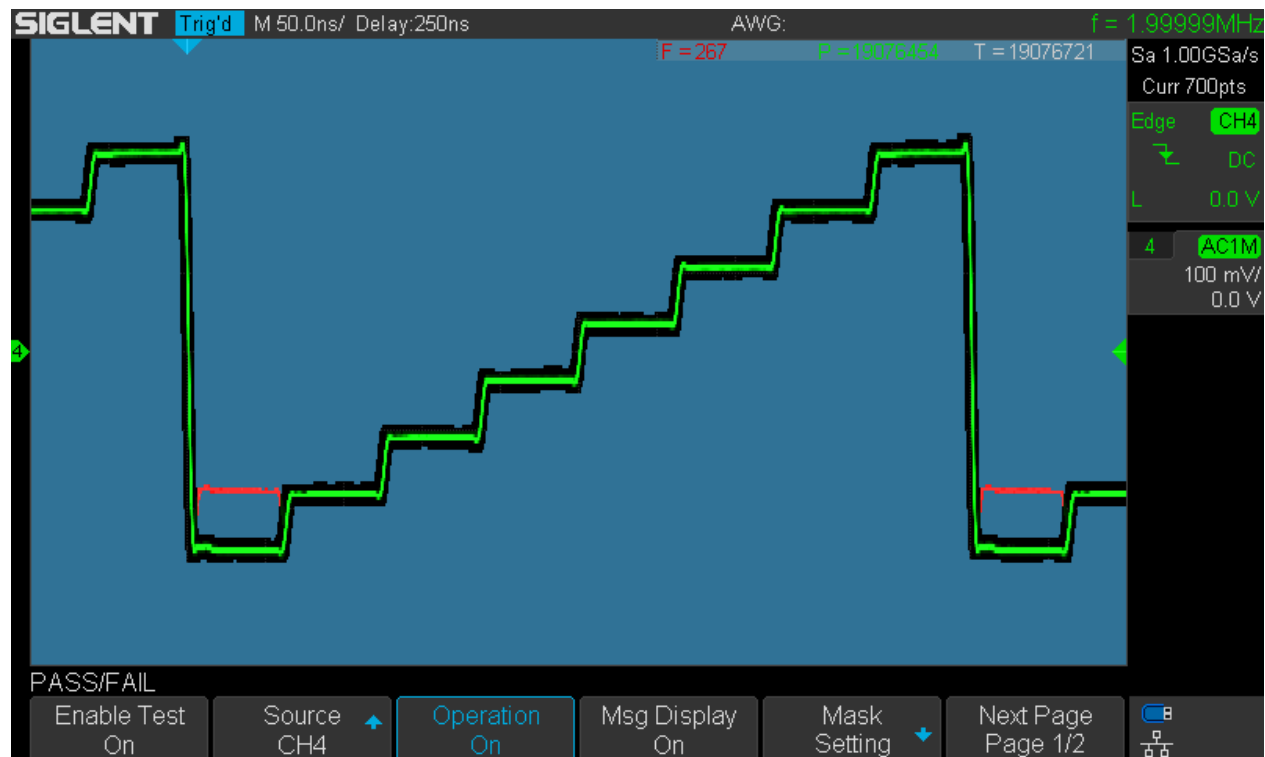
As an example, let's assume a step-up curve where the bottom step is occasionally missing. The signal repetition rate is 2MHz and the fault occurs at 20Hz, hence an original fault rate of 1:100000 or 0.001%.

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

$$\text{Fault_detection_rate} \times \text{timebase} \times \text{horizontal_divisions} = 107600 \times 50\text{e-}9 \times 14 = 0.07532 = 7.532\%;$$

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 $20 \times 177 = 3540$ faults where 267 got detected. $267 / 3540 = 0.0754 = 7.54\%$; Average detection time was $177\text{s} / 267 = 0.663\text{s}$ and for 1Hz fault rate this would be $0.663\text{s} \times 20 = 13.26\text{s}$. 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.



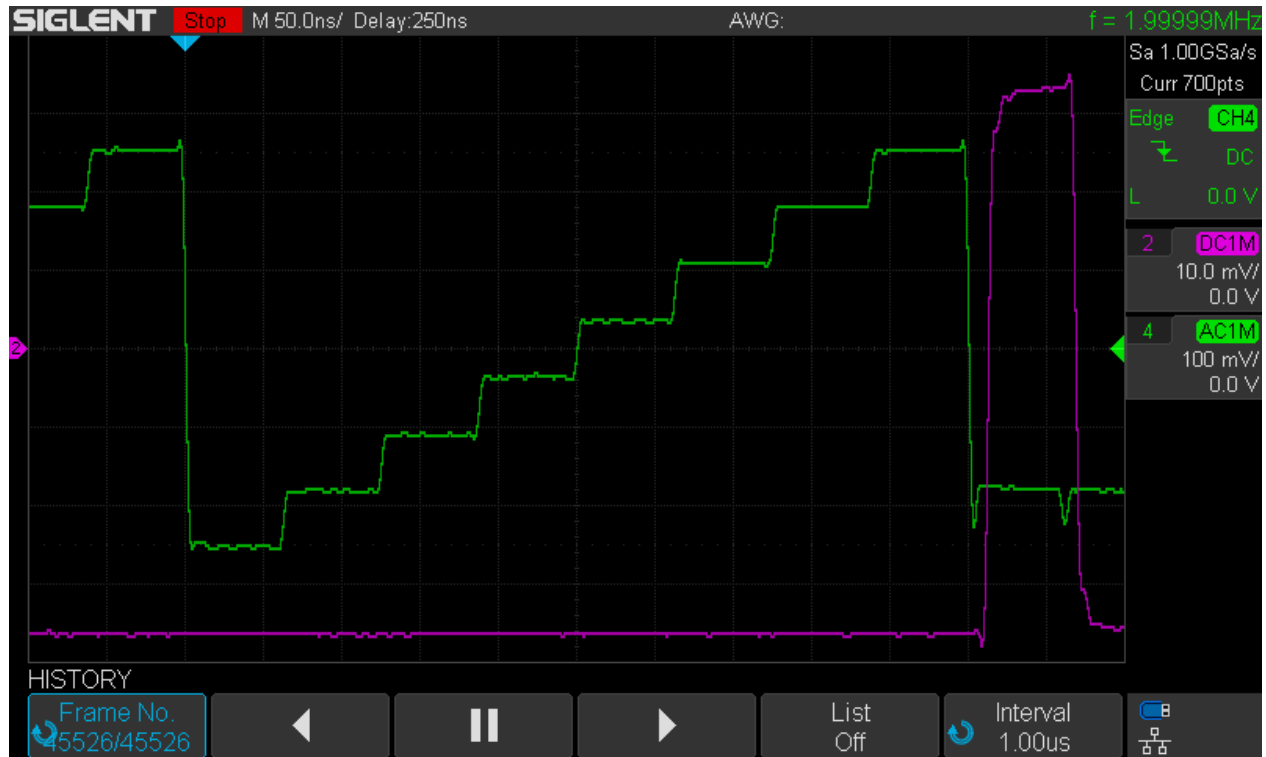
SDS1104X-E_Masktest_50ns_2MHz_20Hz_177s

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.

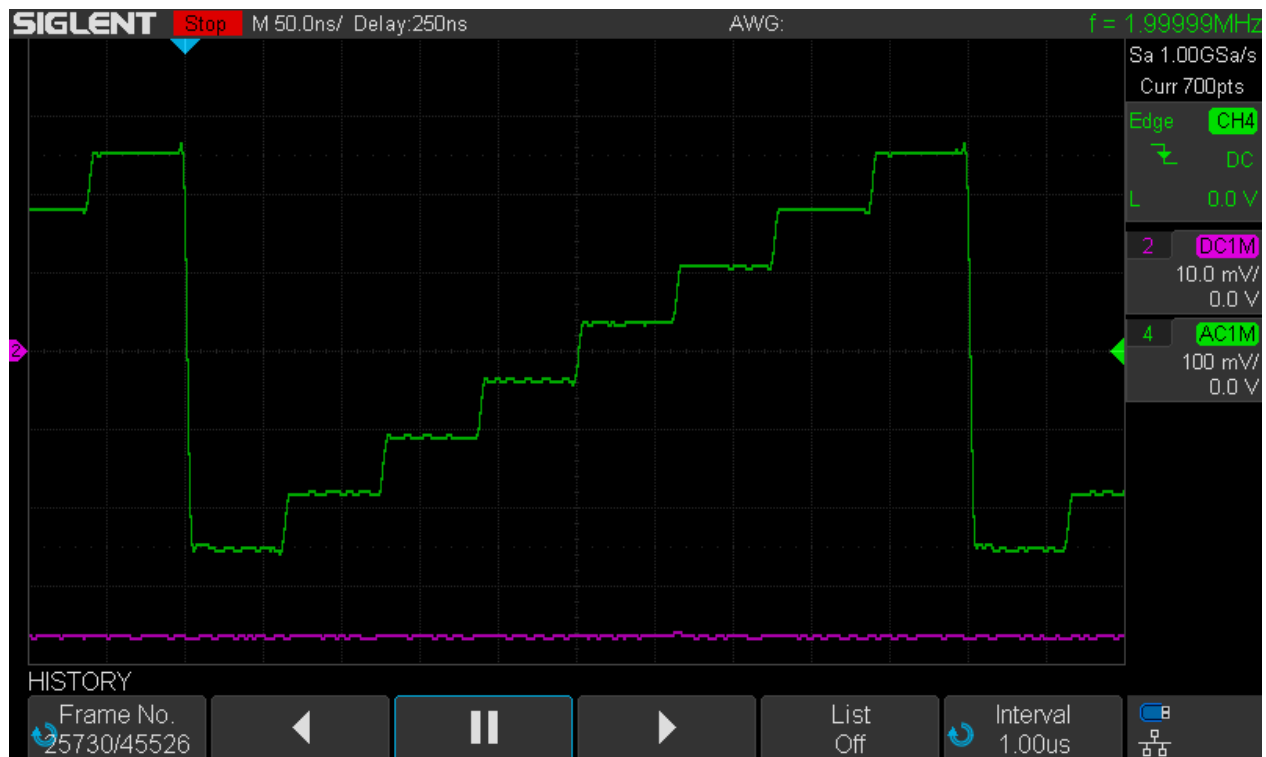


SDS1104X-E_Masktest_50ns_2MHz_20Hz_Stop

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.

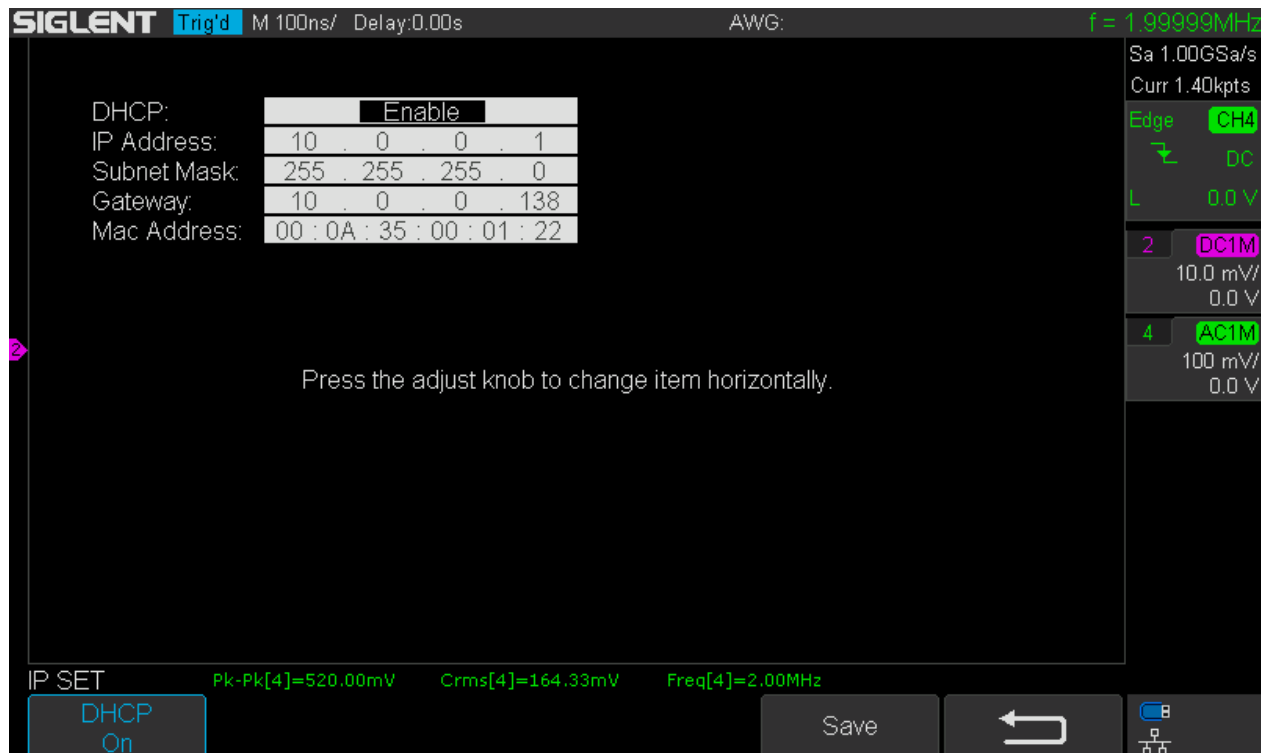
You might ask what the advantages of mask testing are compared to just using (infinite) display persistence alone. Well, there is a couple...

- 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 $267 / 19076721 \sim 14e-6$; this is not exactly the true fault rate which would be $20\text{Hz}/2\text{MHz} = 10e-6$, 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 $<30\text{Hz}$ 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

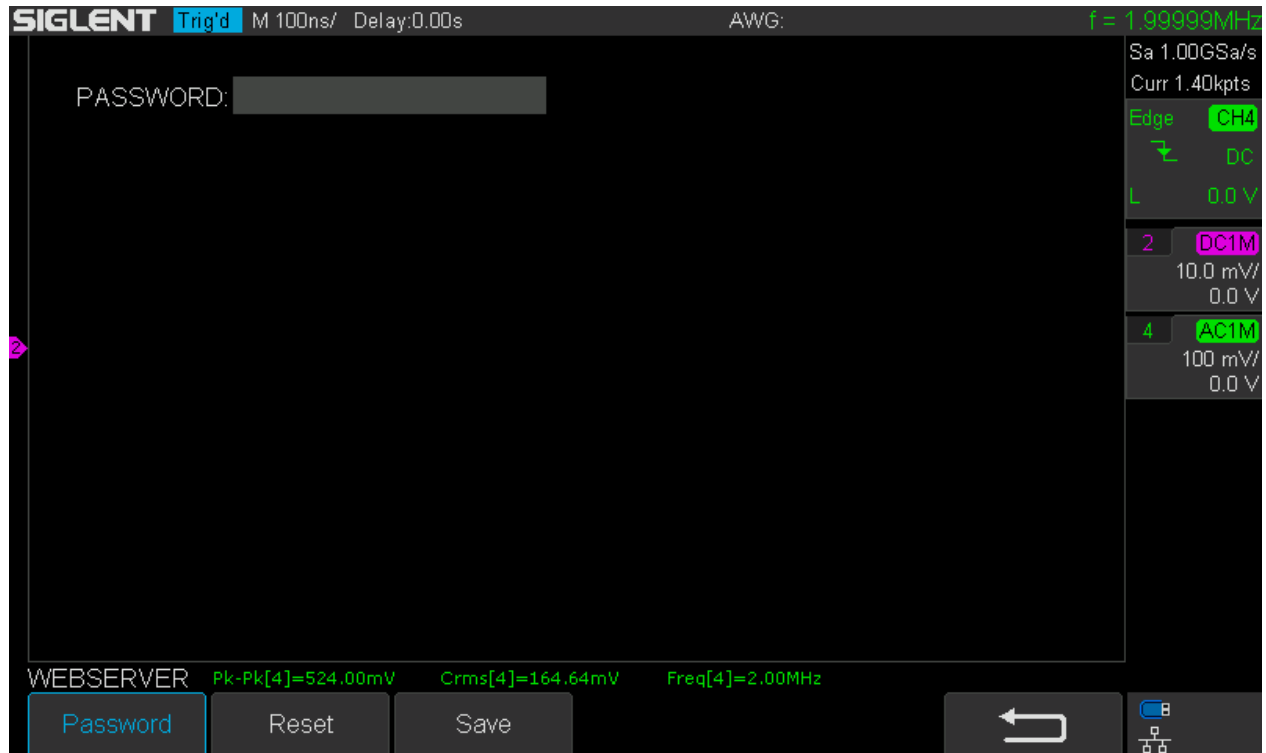
For the first time, a Siglent DSO has a built-in web server. It is nothing exciting, and does not show a live view of the DSO screen, so it's by no means a substitute for an USB scope. It just allows remote setup of the most common functions and pulling individual screenshots. A nice first shot nevertheless...



SDS1104X-E_IP_setting

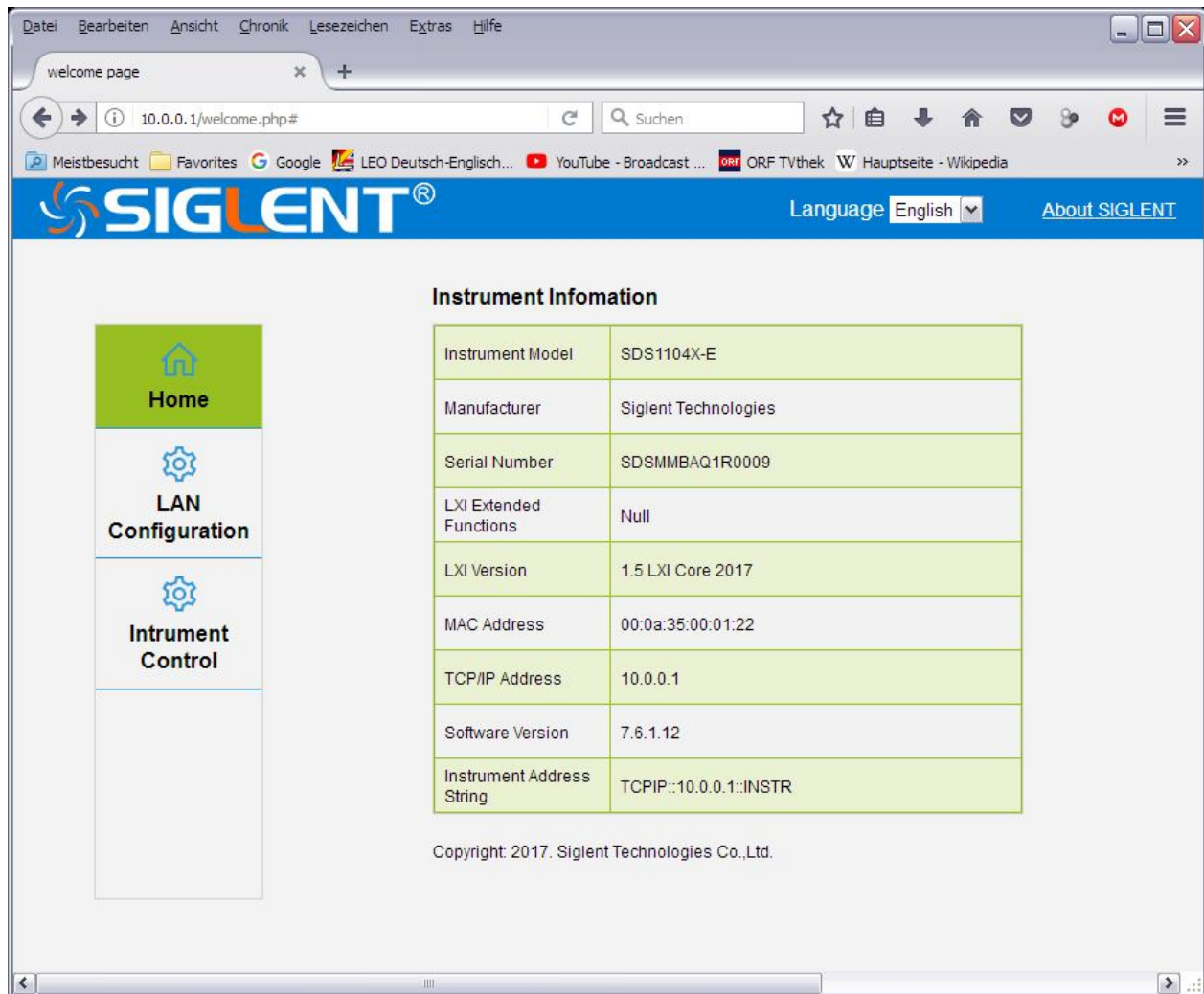
On page 2 of the *Utility* menu, we can set up the IP configuration. This is totally easy as long as a network router with integrated DHCP server is available. All that's needed is enabling DHCP and waiting a moment until the scope has received its individual IP address, see screenshot above. In a simpler 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.



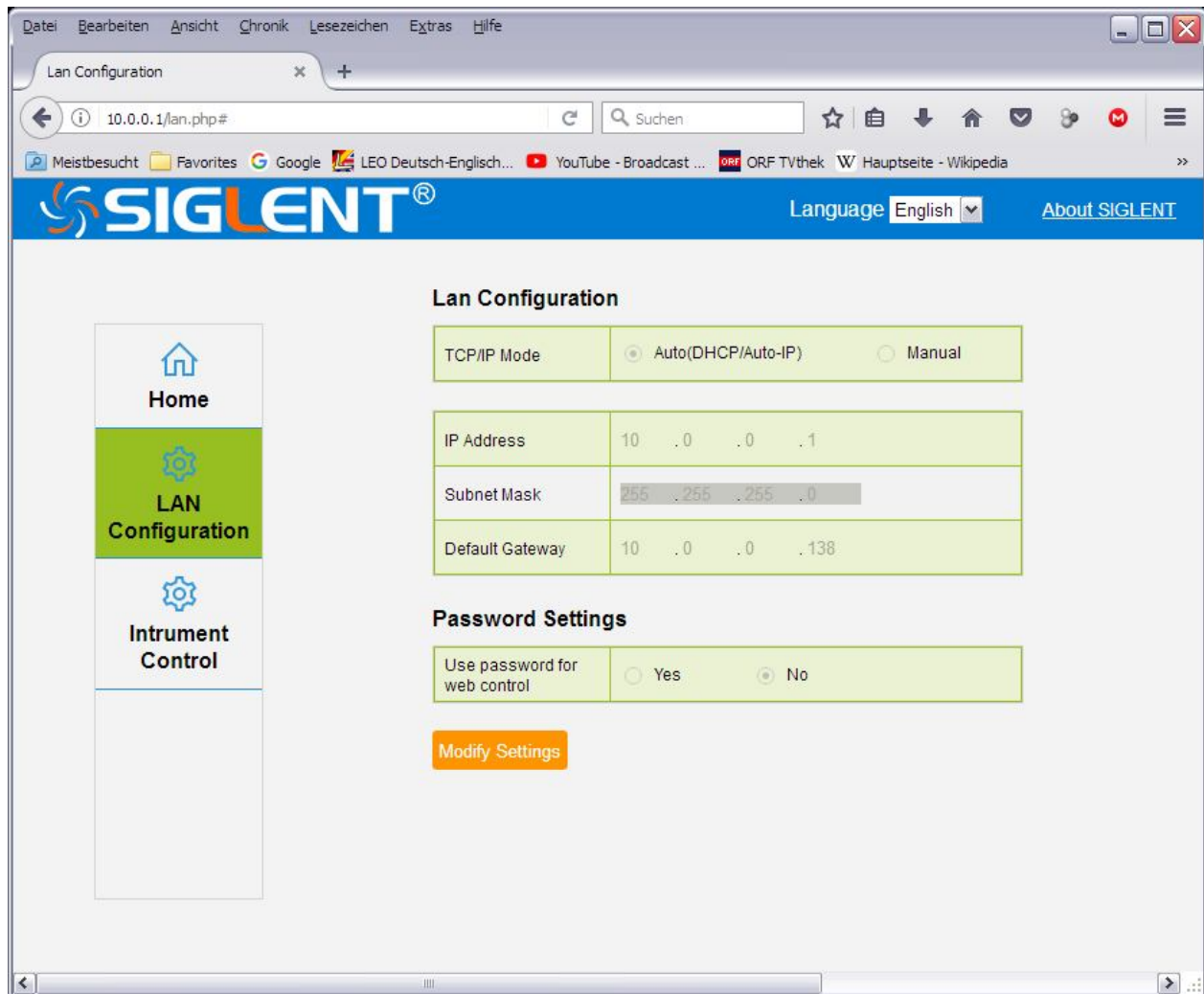
SDS1104X-E_WEB_Password

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

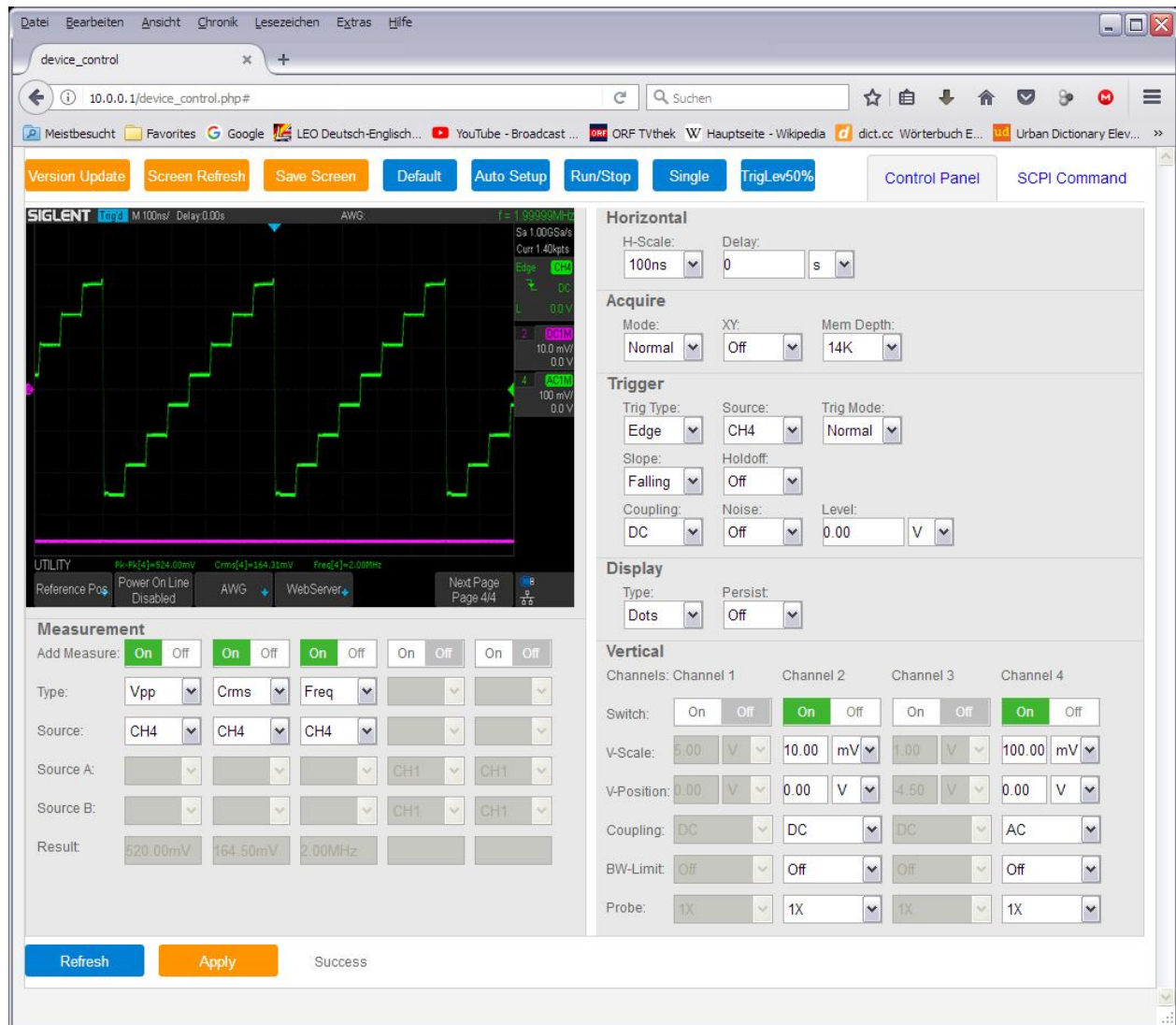
This contains some useful information about the instrument, like model, serial number and firmware version. This screen offers access to the LAN configuration as well as the Instrument Control. Let's have a look at LAN Configuration first.



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.



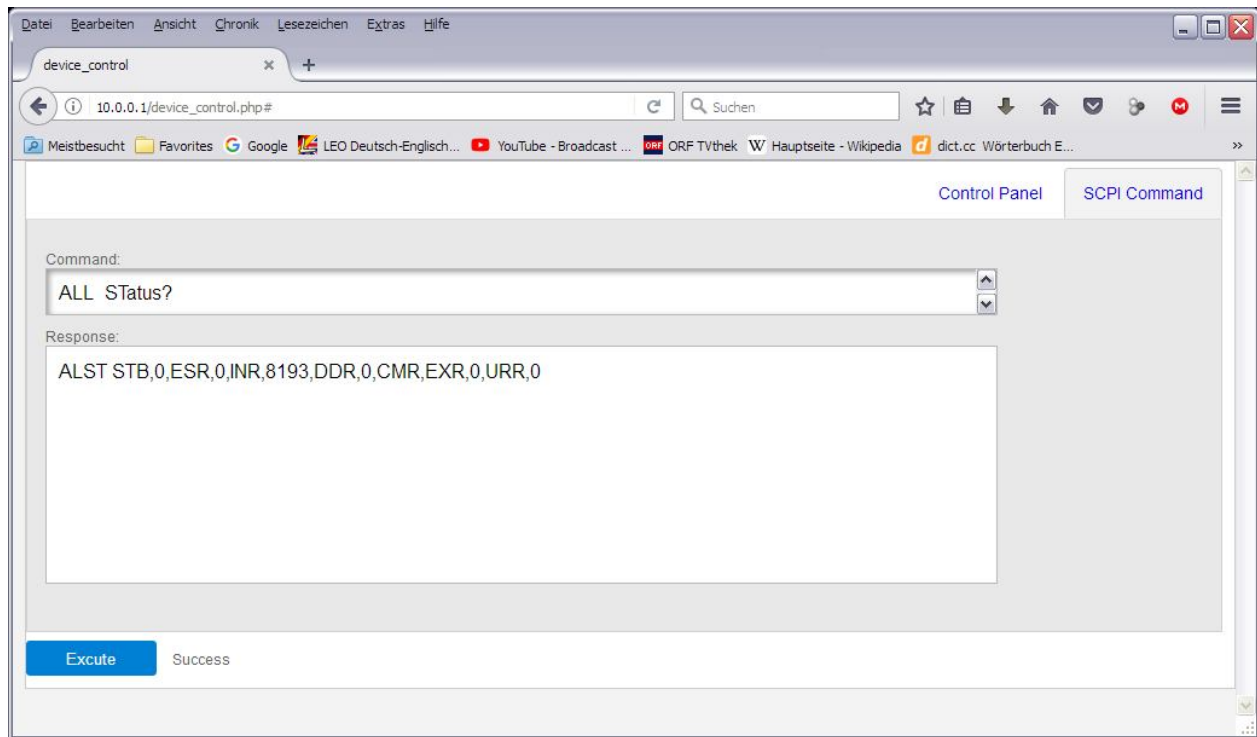
WEB_Control

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

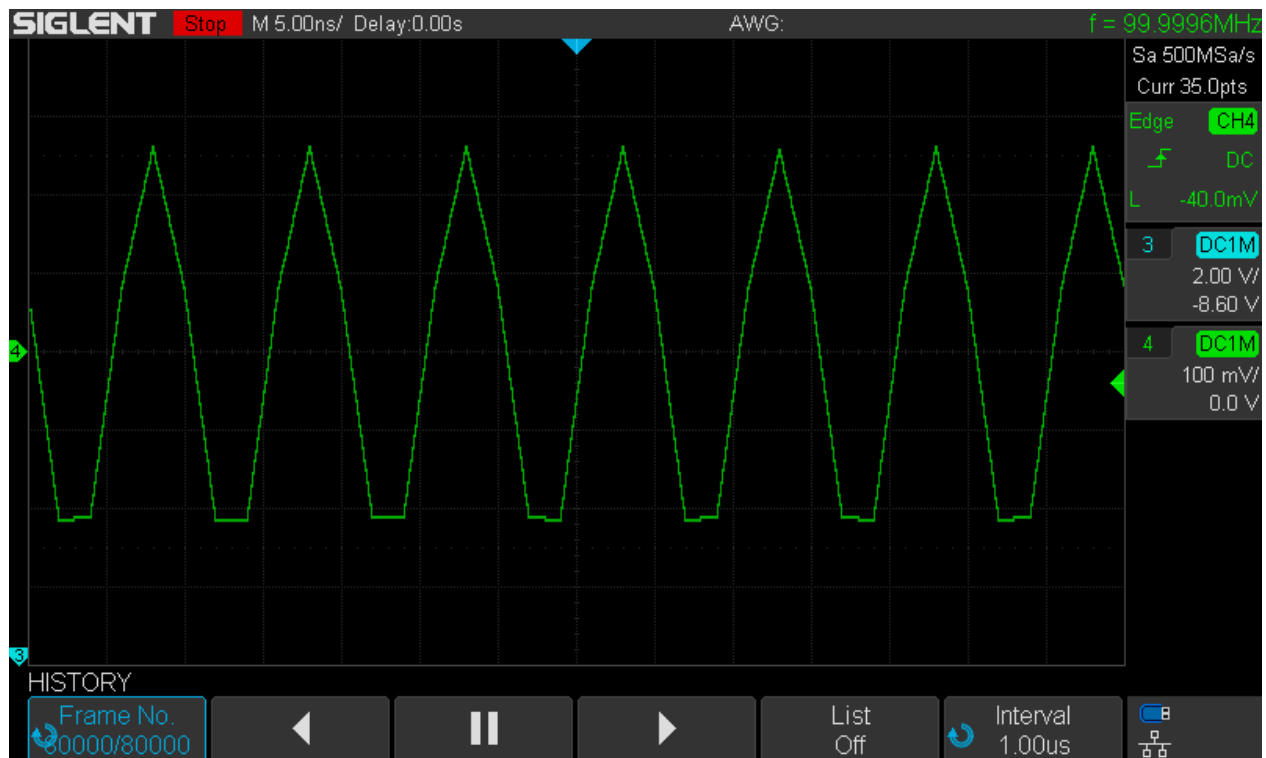
As has been already stated, this is not a special mode, but works silently in the background all the time. Consequently, history is available whenever we need it.

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



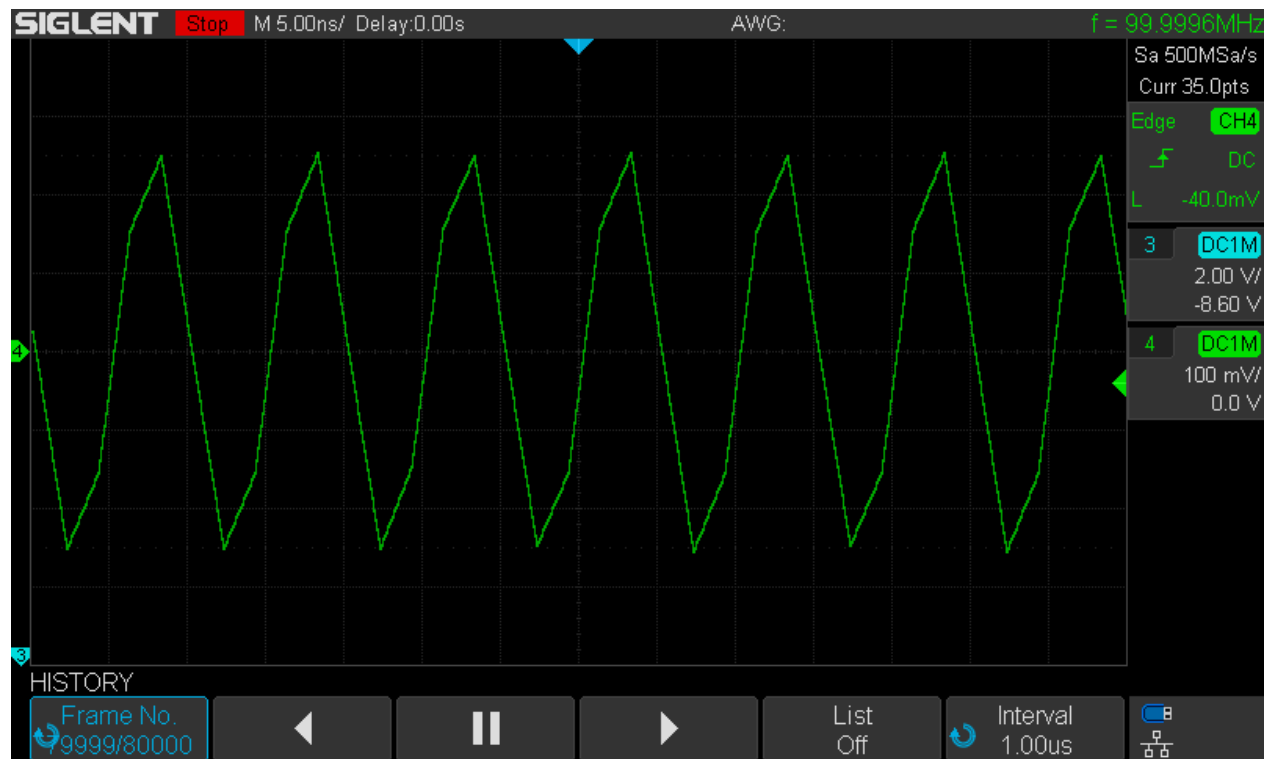
Hist_100MHz_vectors_x_run

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

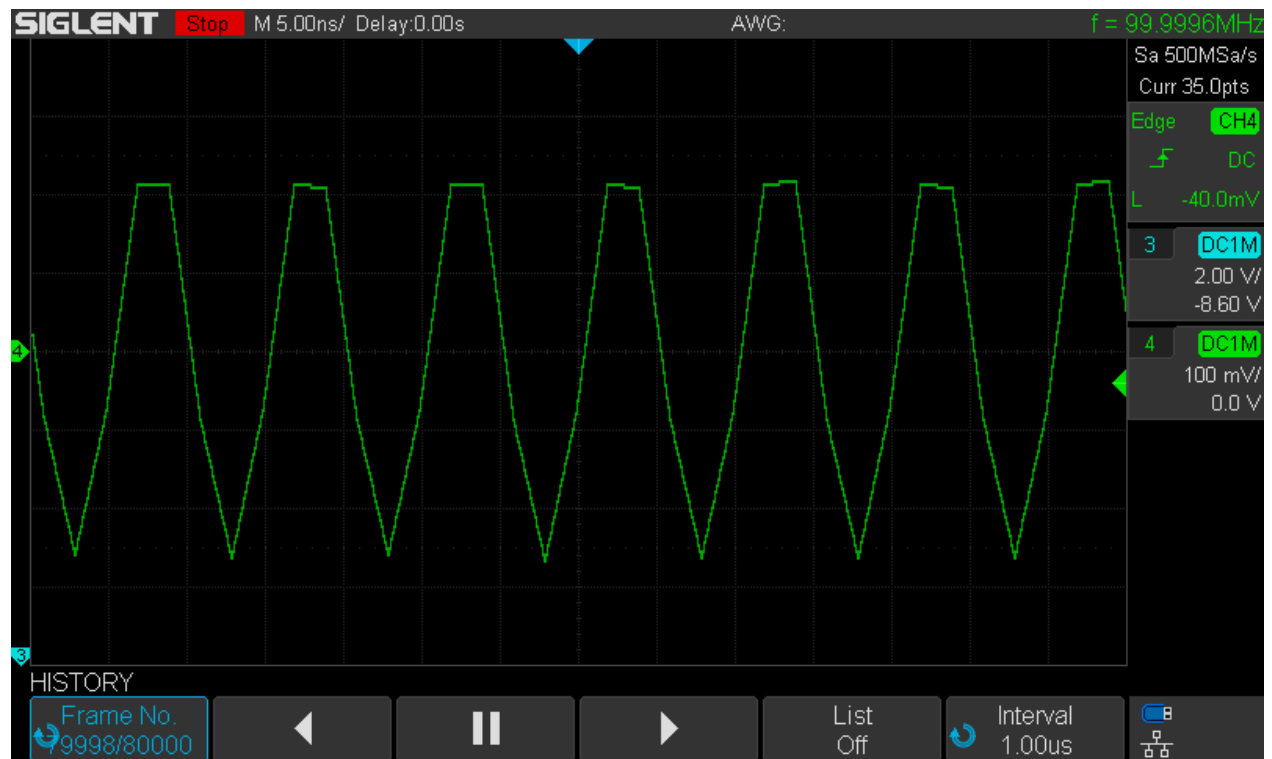


Hist_100MHz_vectors_x_80000

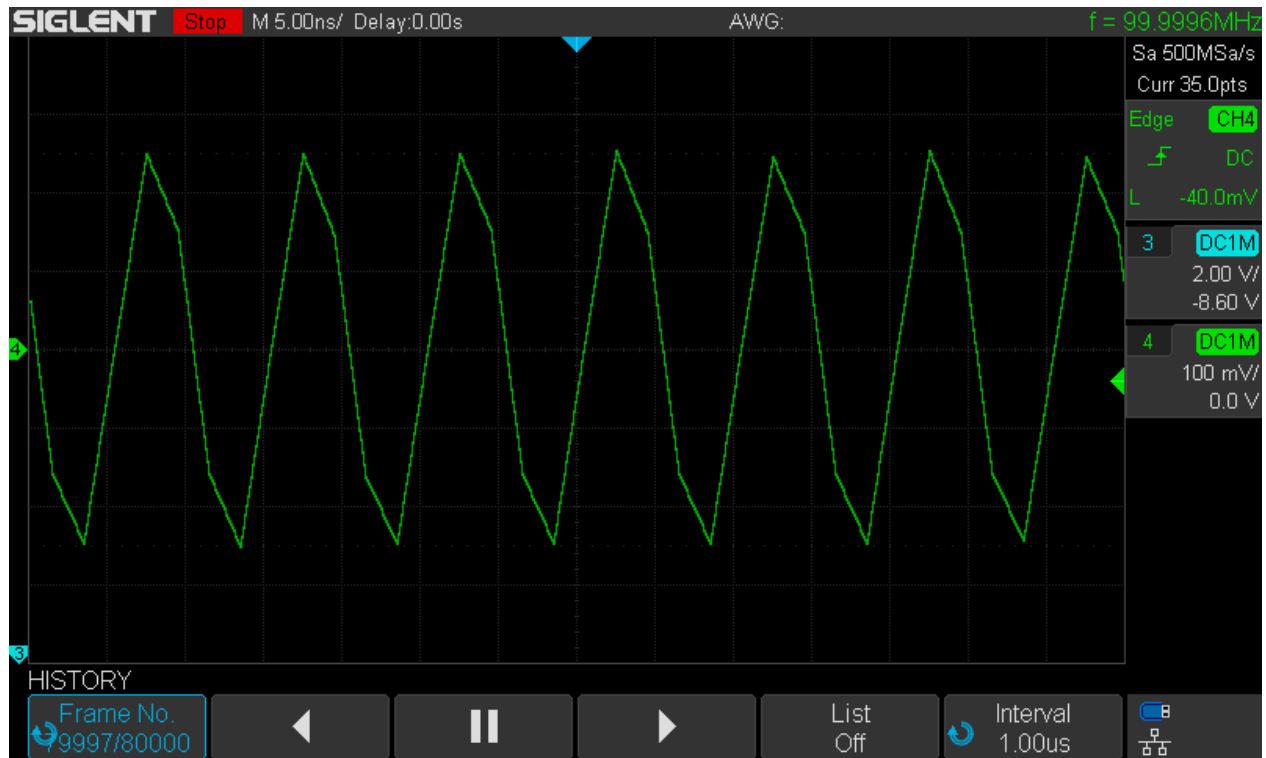
Now we can see that 80000 records (frames) have been stored in the history and we can analyze every single one if we want to. Initially, we're just seeing the last one, but can scroll through the frames in order to see the different variations of the misshaped signal caused by the simple x-interpolation.



Hist_100MHz_vectors_x_79999

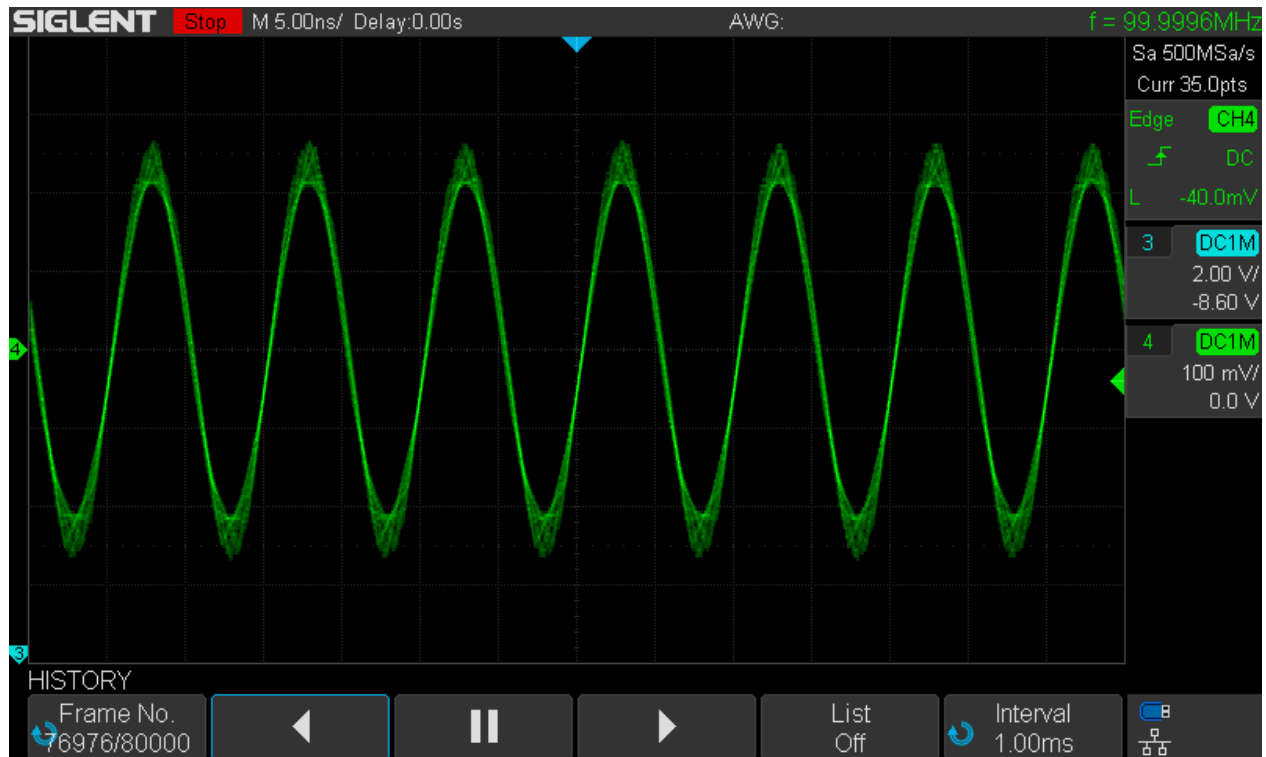


Hist_100MHz_vectors_x_79998



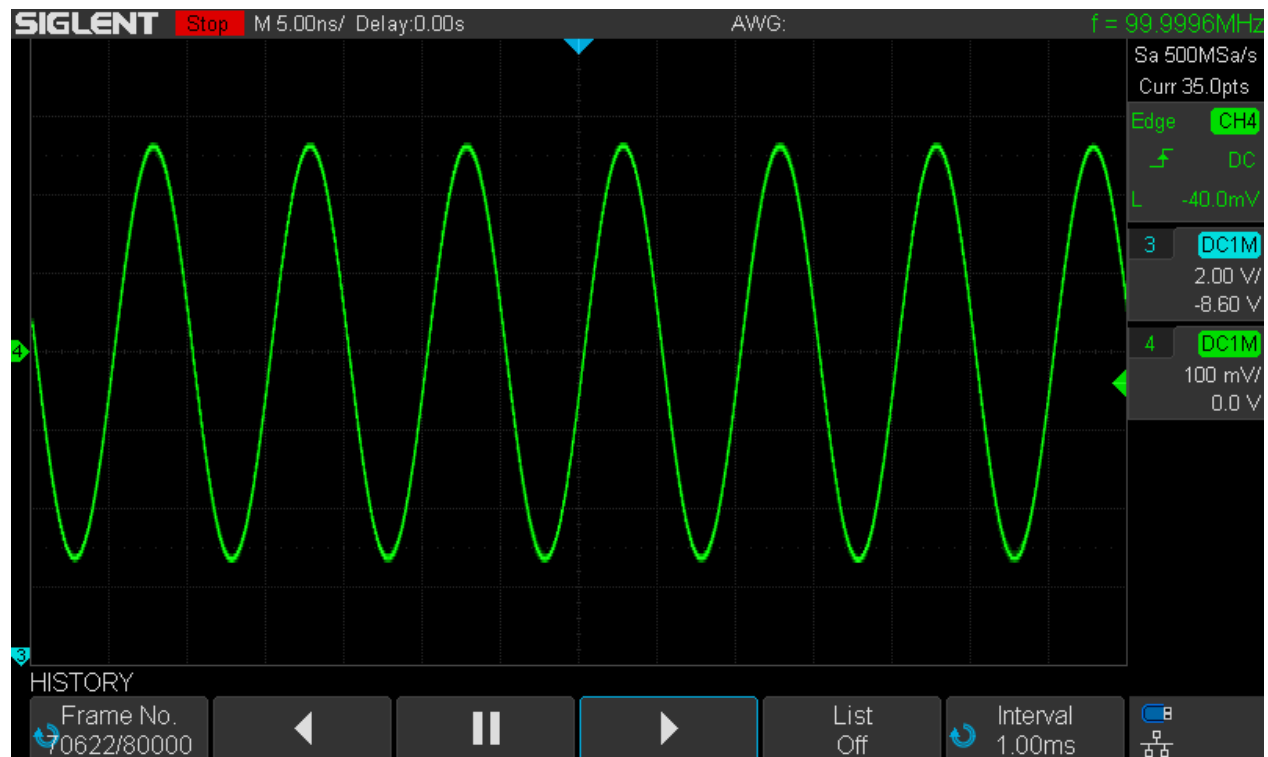
Hist_100MHz_vectors_x_79997

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:

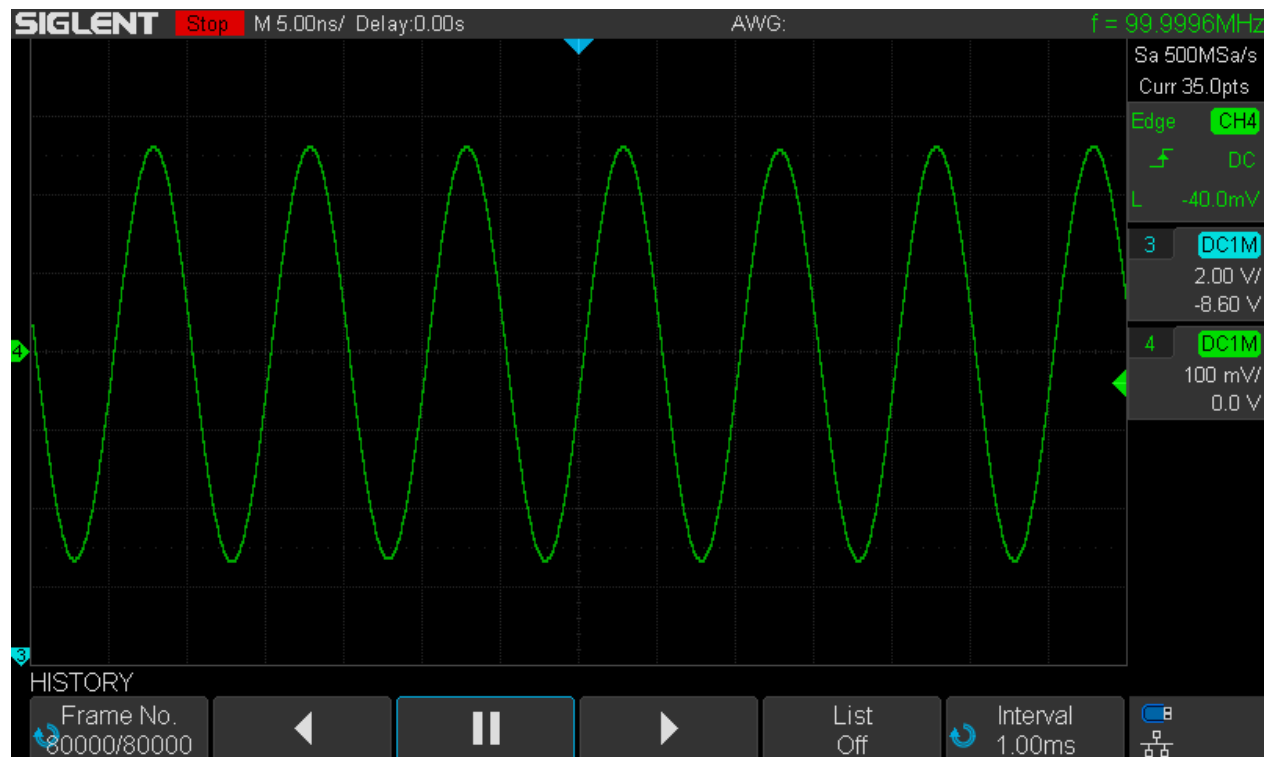


Hist_100MHz_vectors_x_Playback

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):



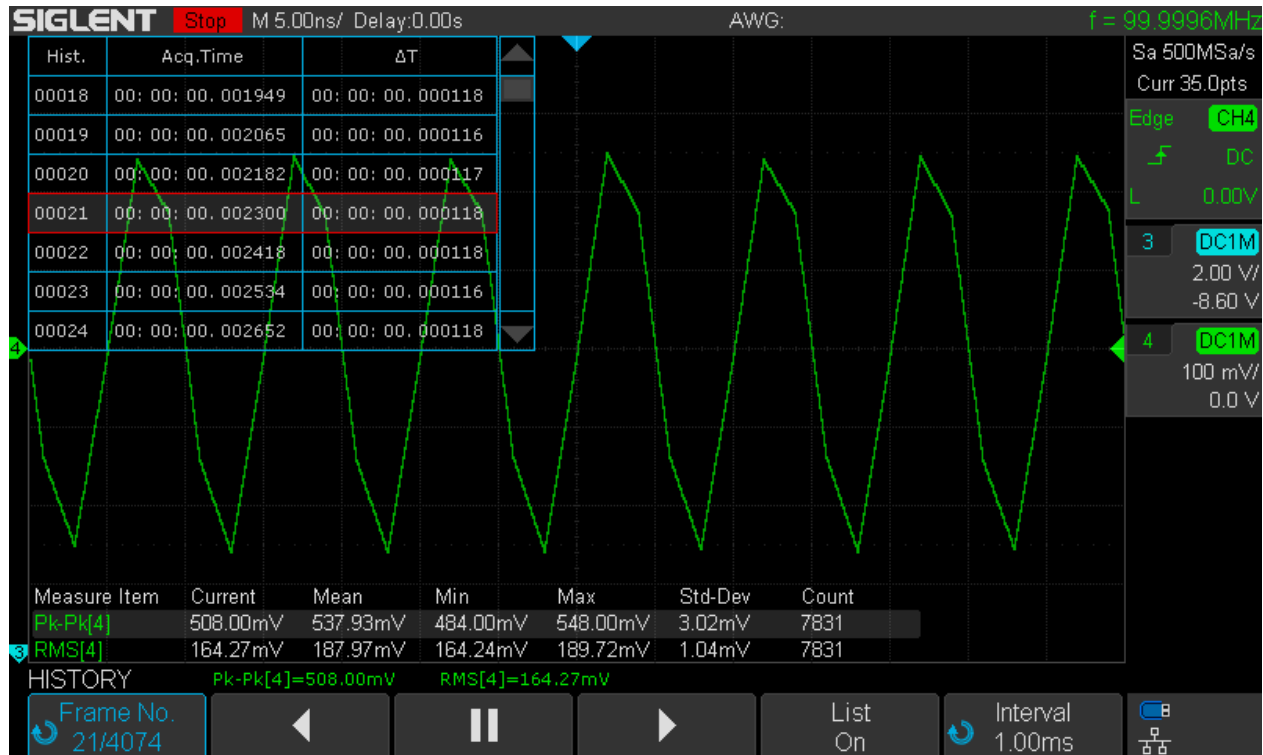
Hist_100MHz_vectors_sinx_Playback



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.



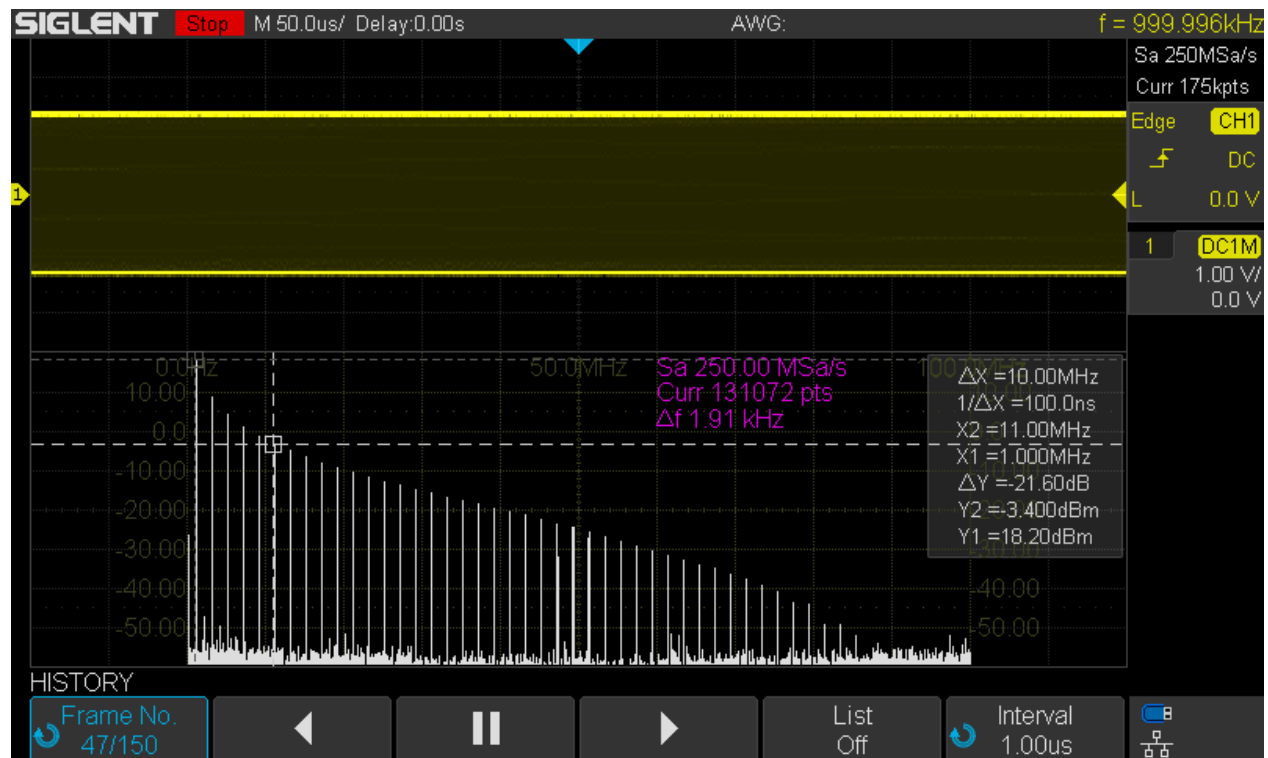
Hist_100MHz_vectors_x_list

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:

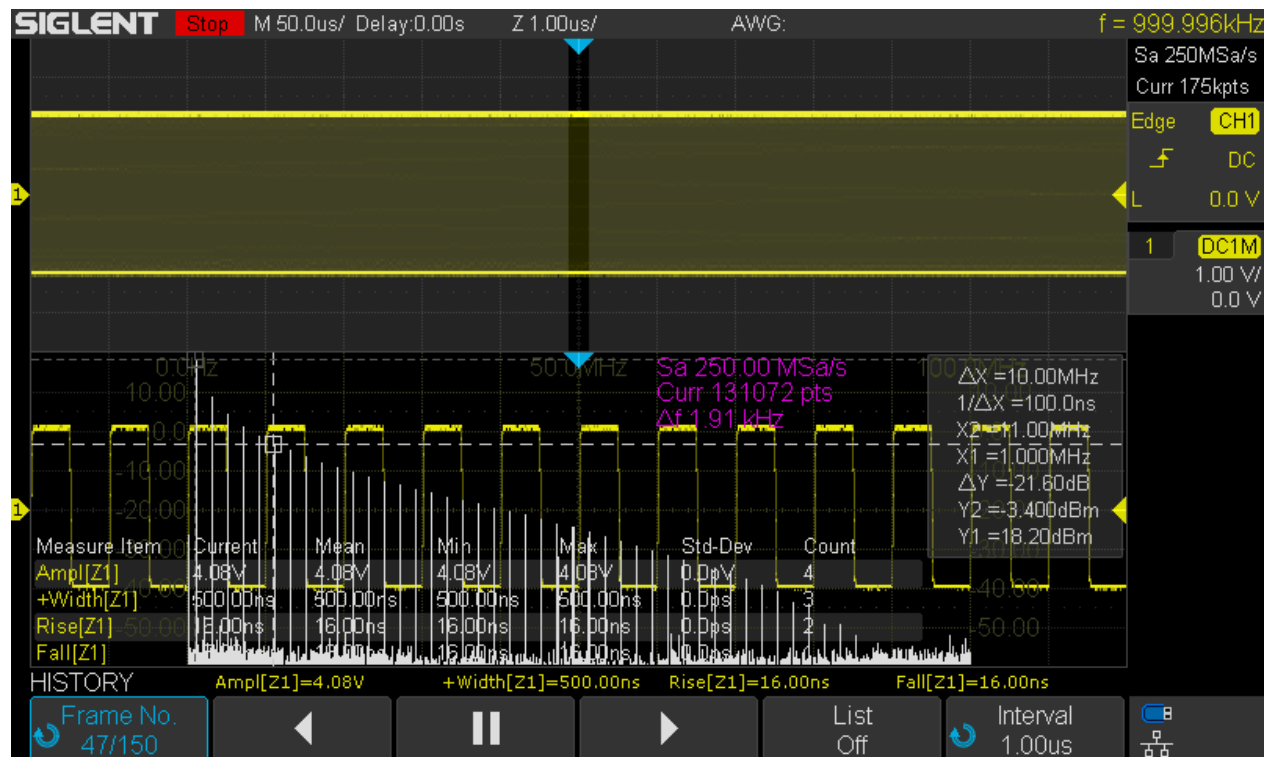
1. Push the universal select knob to display a dialog where we can jump to a certain record instantly by entering the desired frame number.
2. Use the playback function at an appropriate frame rate to emulate a fast forward or rewind like on a tape recorder.
3. Bring up the *Navigation* menu by pressing 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

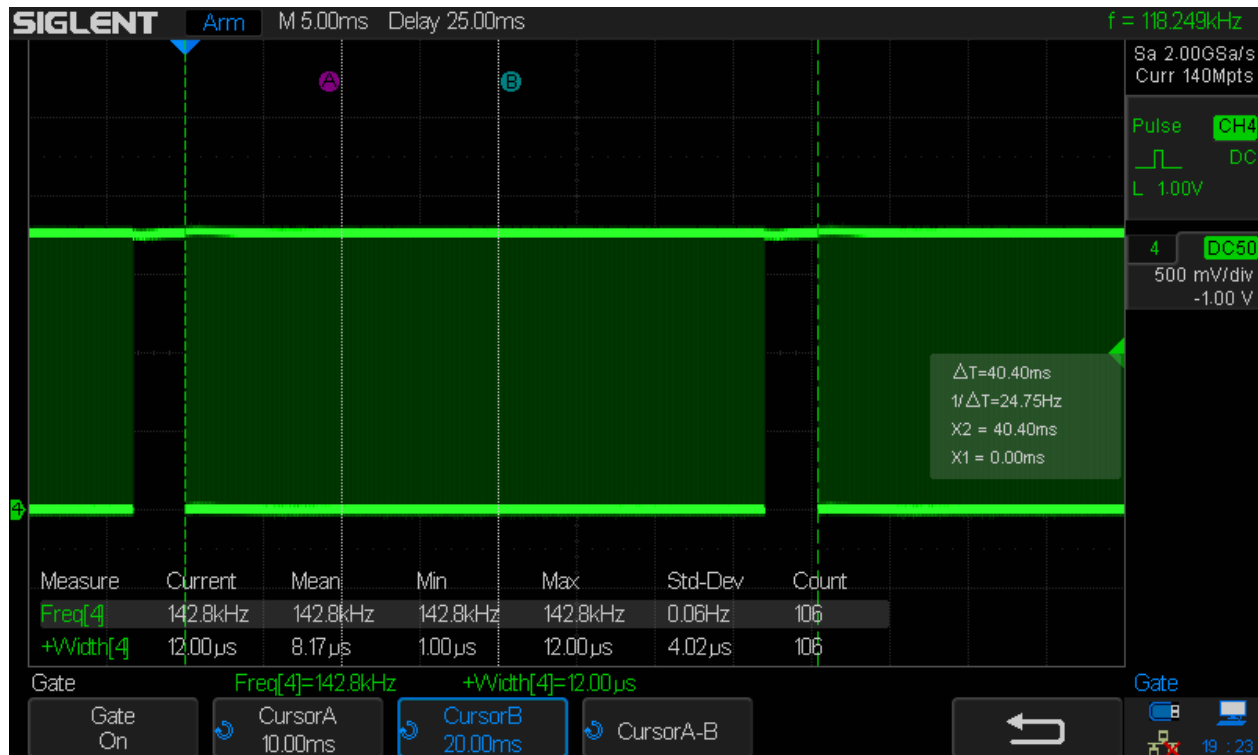


Hist_Square_10MHz_Zoom_FFT_Cursors_Meas

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):



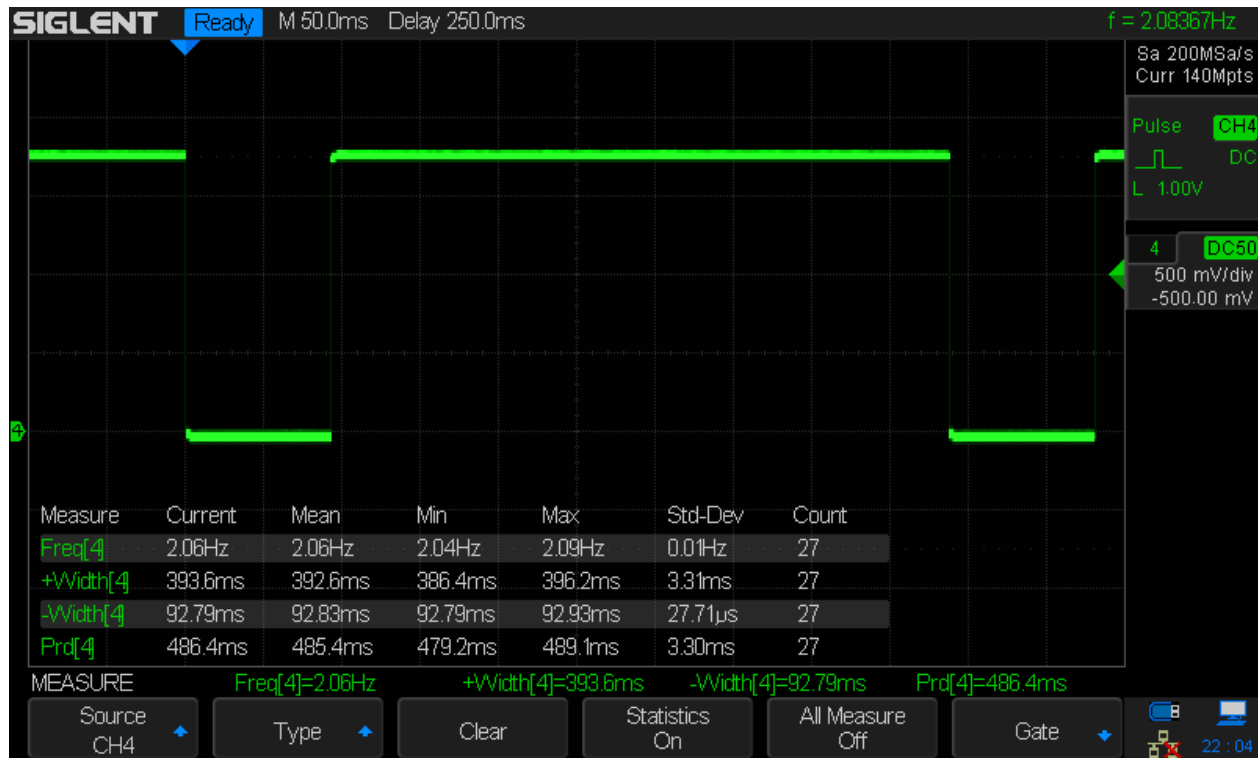
SDS1104X-E_TRIG_Std_50ns

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.

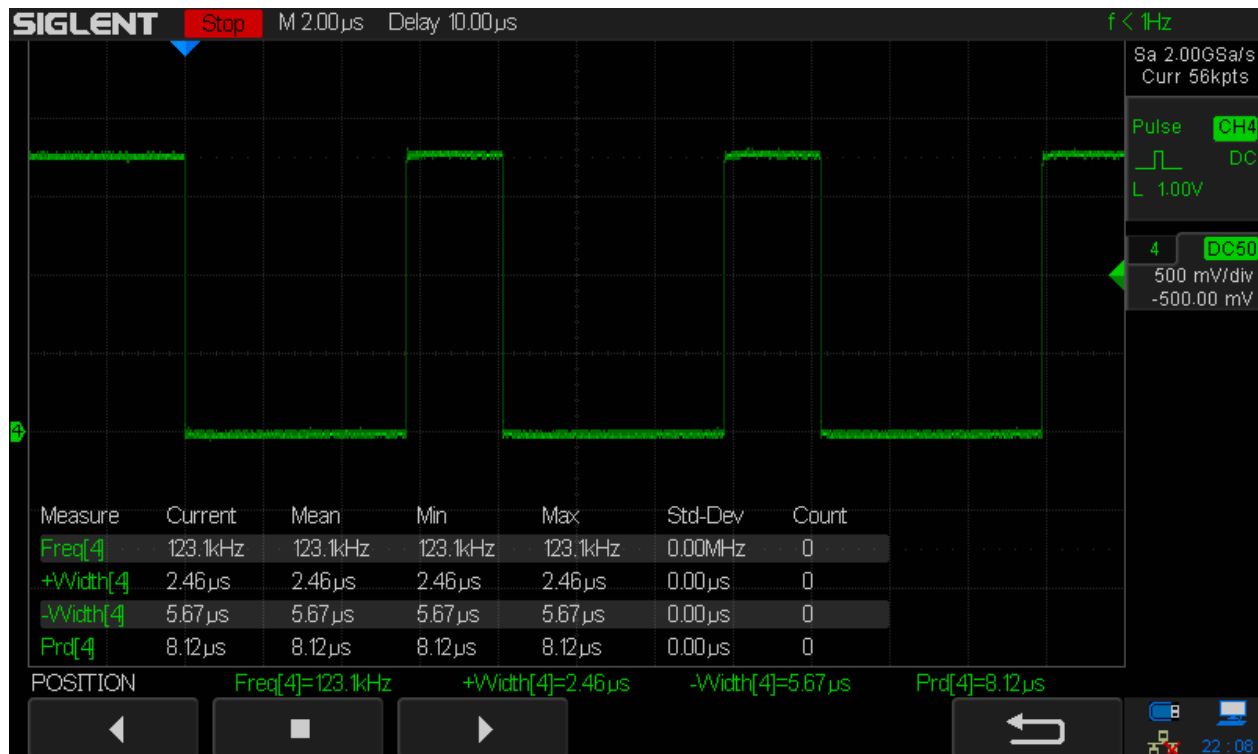
Now let's have a look at sequence mode. The sweet spot setting is with a timebase of 50ns/div and dots display, where the waveform update rate is measured as nearly 488k (depending on the input signal, it can be even slightly faster) and a screen update rate of ~2Hz, which is still quick enough to give 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:



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 $1.4\mu\text{s}$ and 363.4k waveforms are equivalent to a total acquisition time of $363400 \times 1.4\mu\text{s} = 508760\mu\text{s} = 508.76\text{ms}$. 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 $1.4\mu\text{s}$, 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 $5\mu\text{s}/\text{div}$ 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 $10\mu\text{s}/\text{div}$.

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 [s]	Standard		Sequence	
	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.