Siglent SDS1104X-E Review

Segmented Memory

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

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

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

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

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

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

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

Record_Length_Limit_1ms

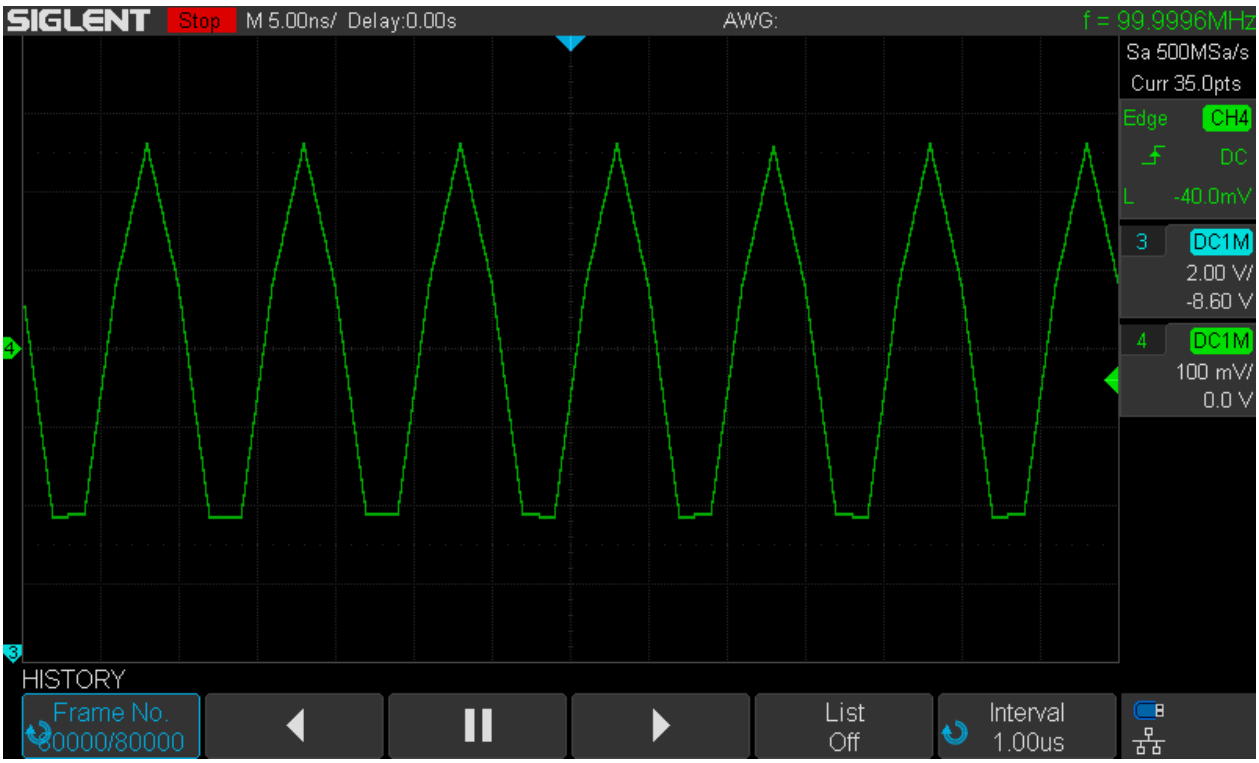
History

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

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

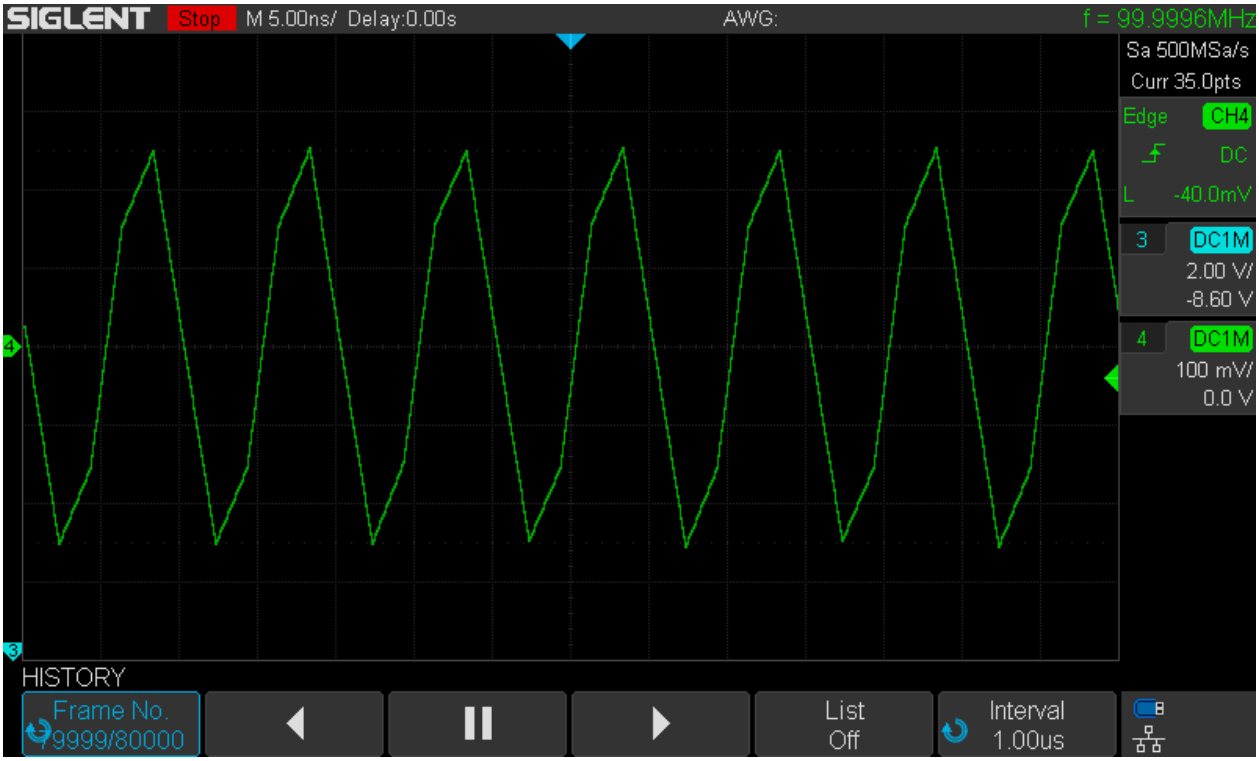
Hist_100MHz_vectors_x_run

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

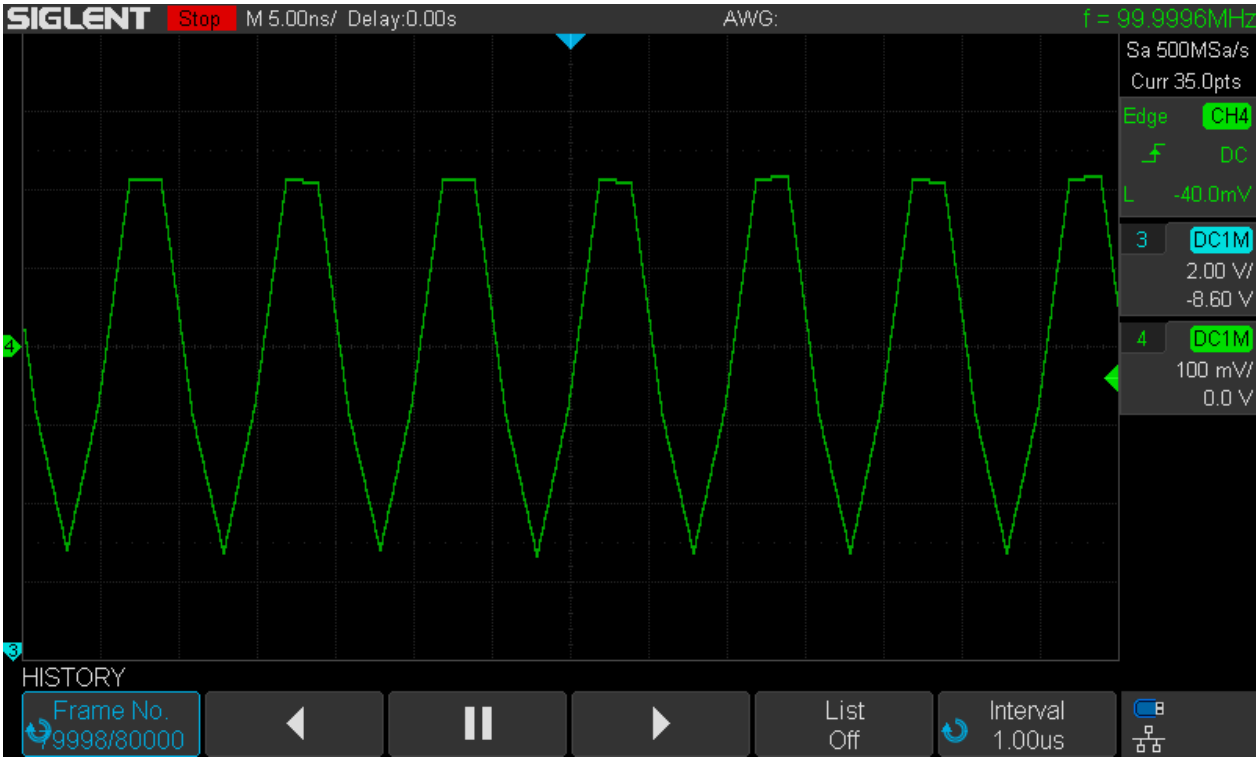


Hist_100MHz_vectors_x_80000

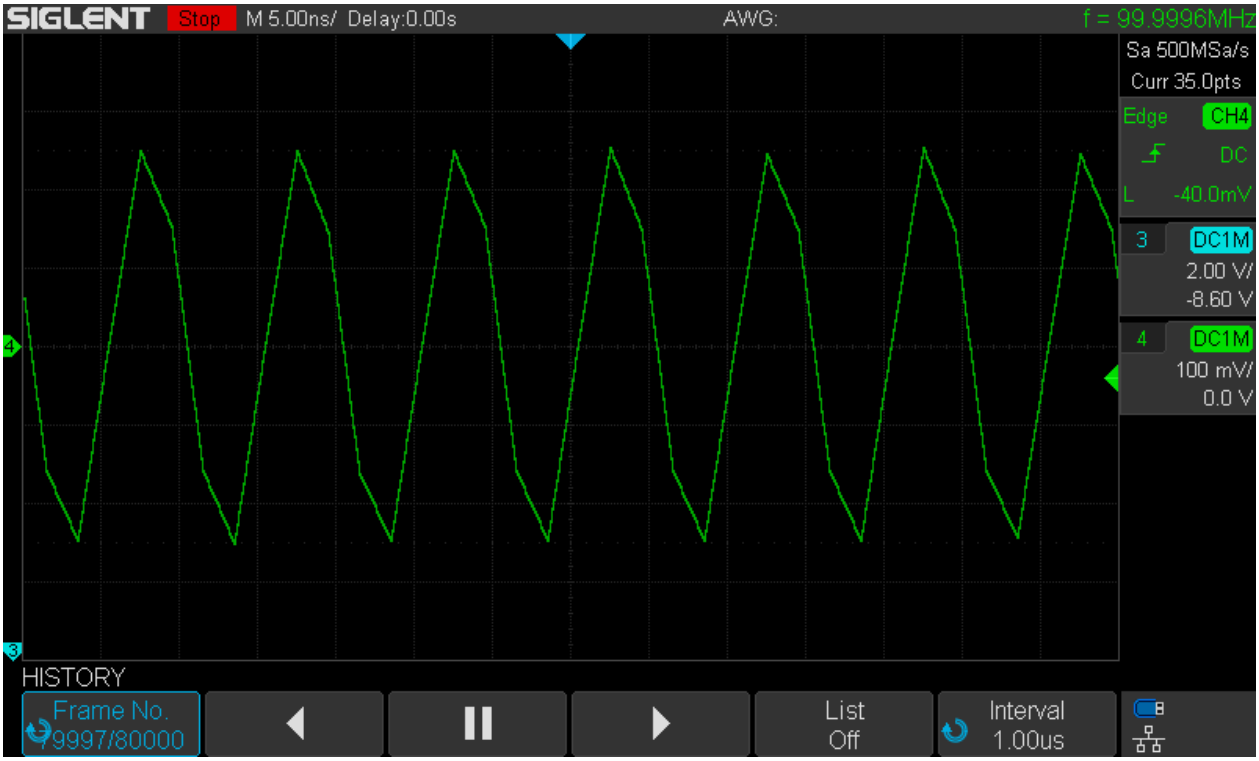
Now we can see that 80000 records (frames) have been stored in the history and we can analyze every single one if we want to. Initially, we're just seeing the last one, but can scroll through the frames in order to see the different variations of the misshaped signal caused by the simple x-interpolation.



Hist_100MHz_vectors_x_79999

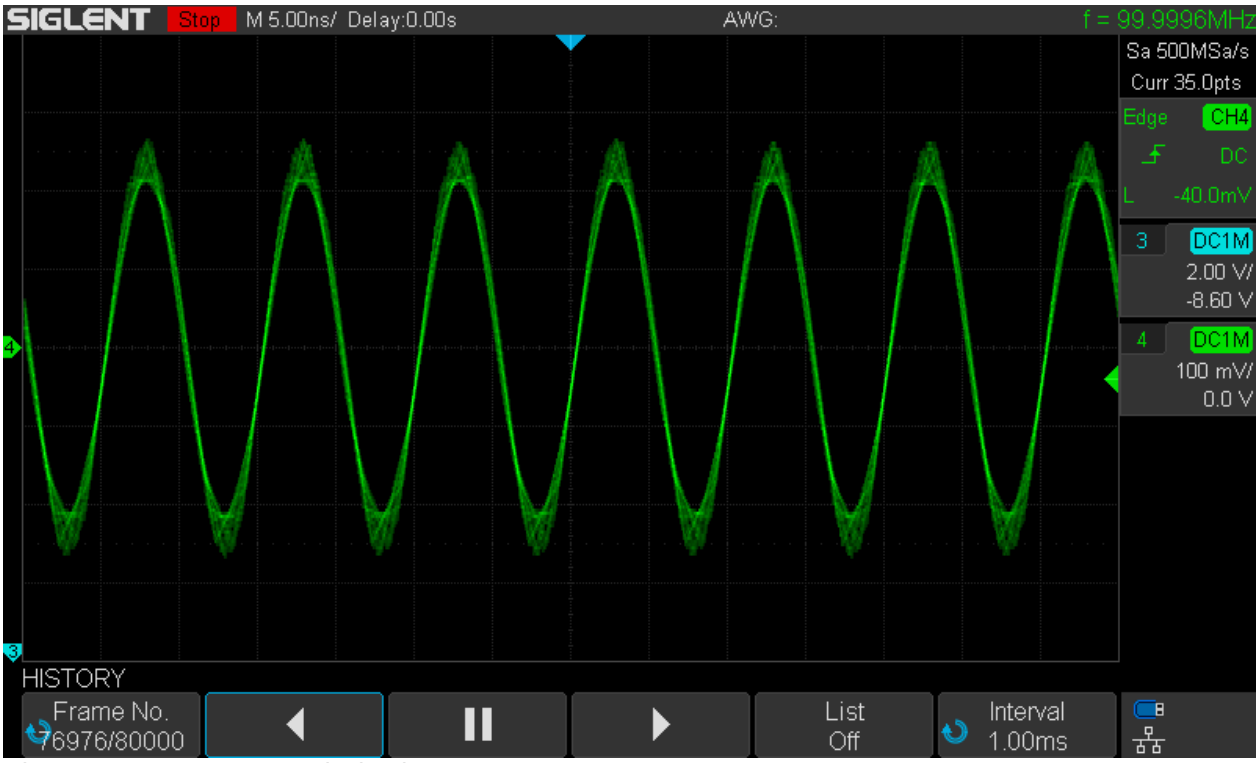


Hist_100MHz_vectors_x_79998



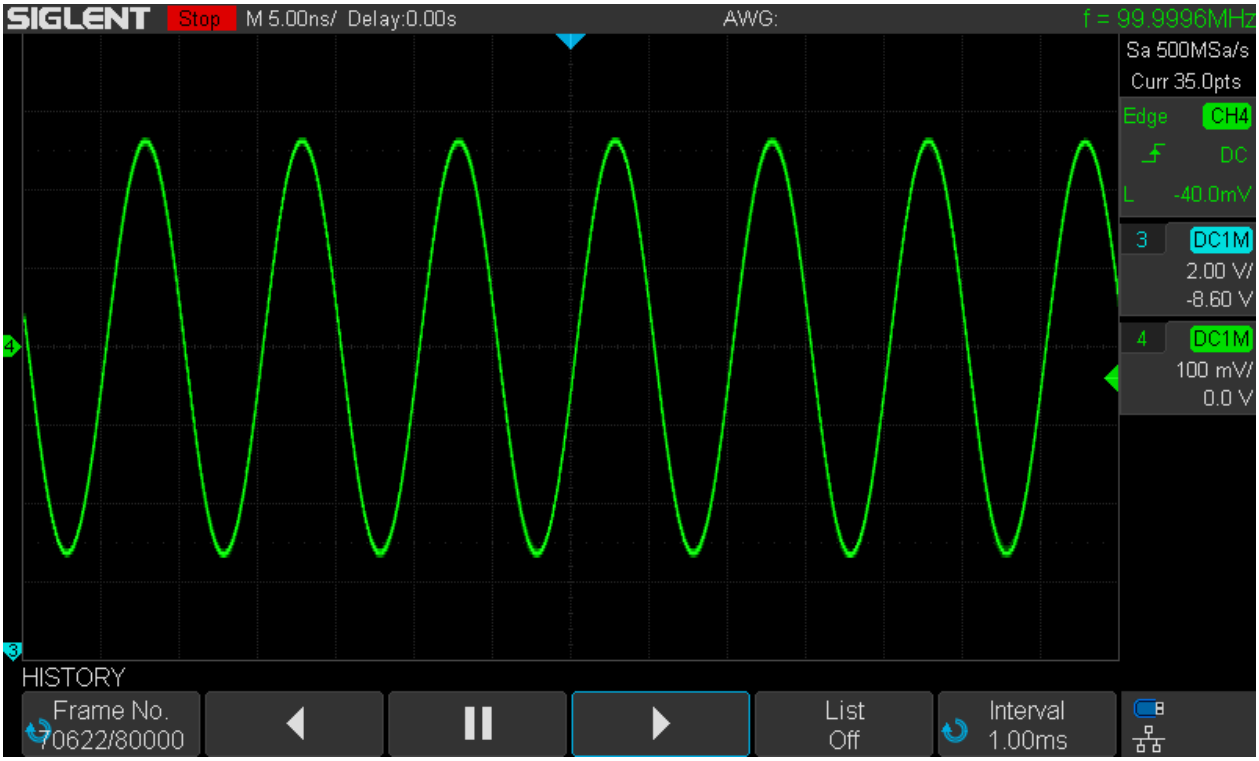
Hist_100MHz_vectors_x_79997

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

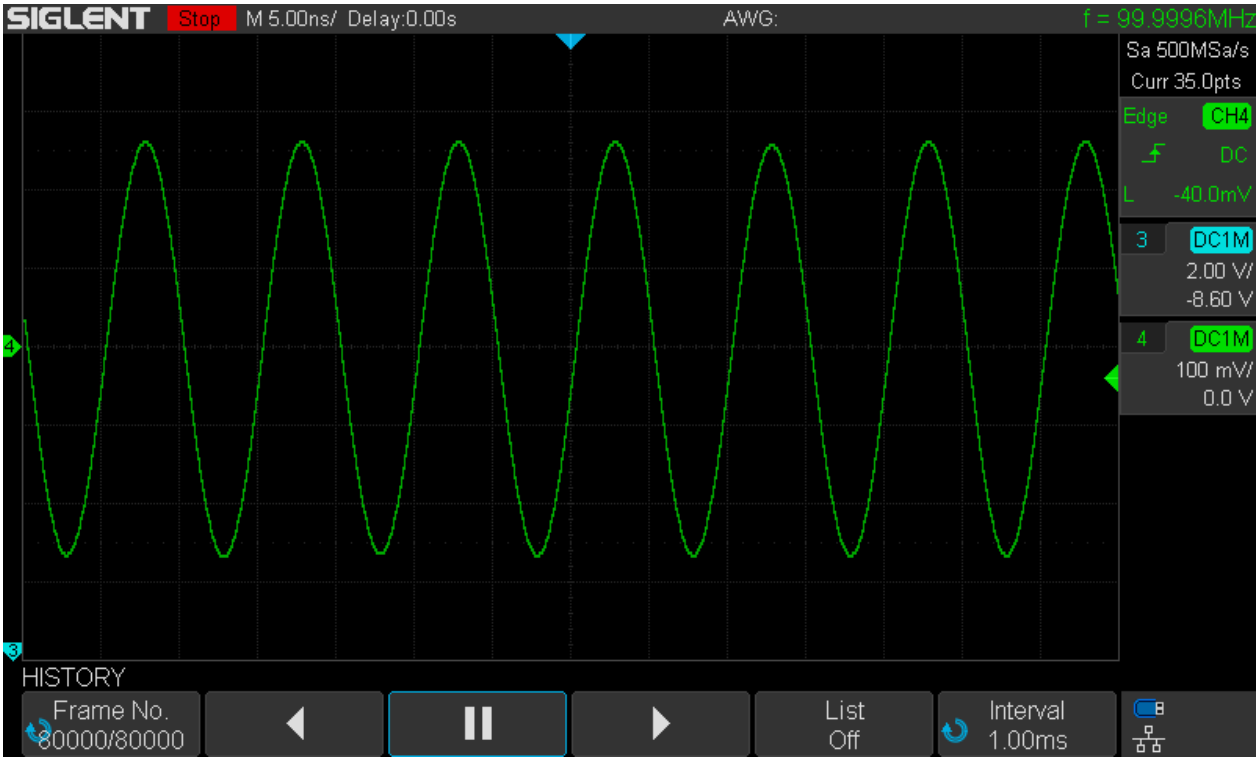


Hist_100MHz_vectors_x_Playback

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



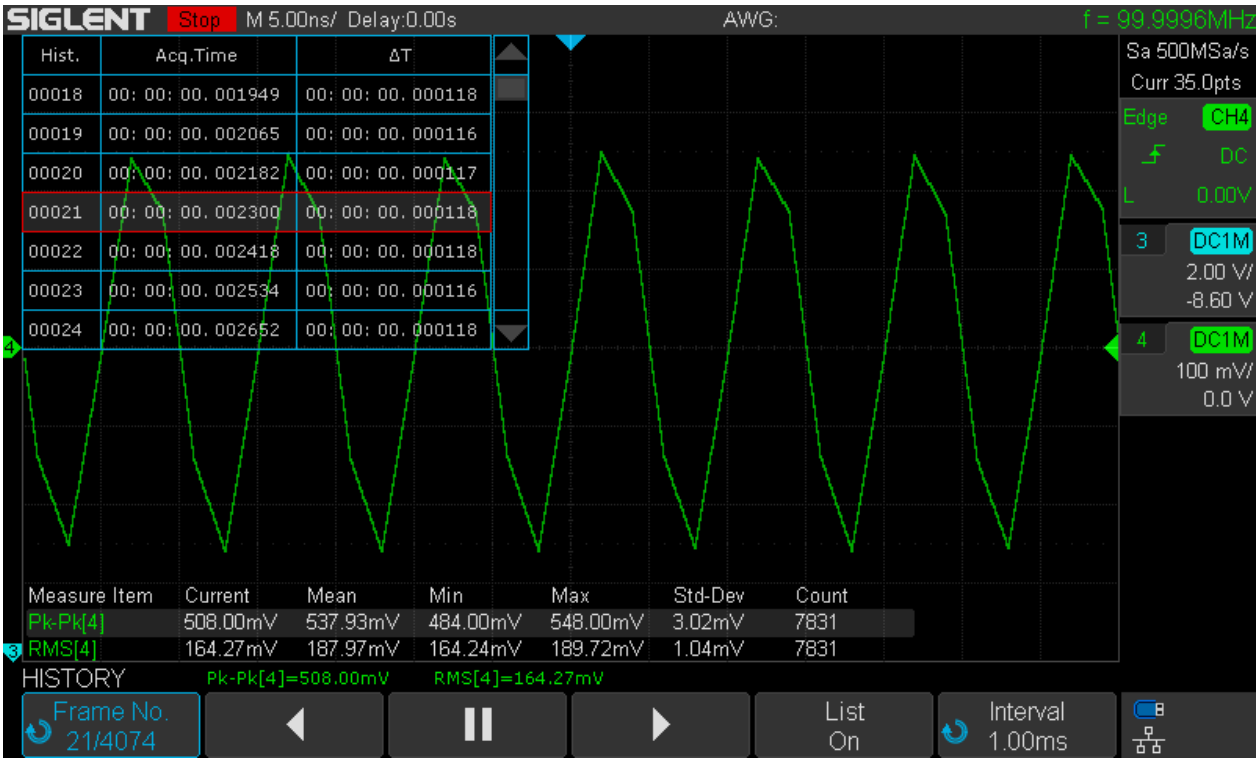
Hist_100MHz_vectors_sinx_Playback



Hist_100MHz_vectors_sinx_80000

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

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



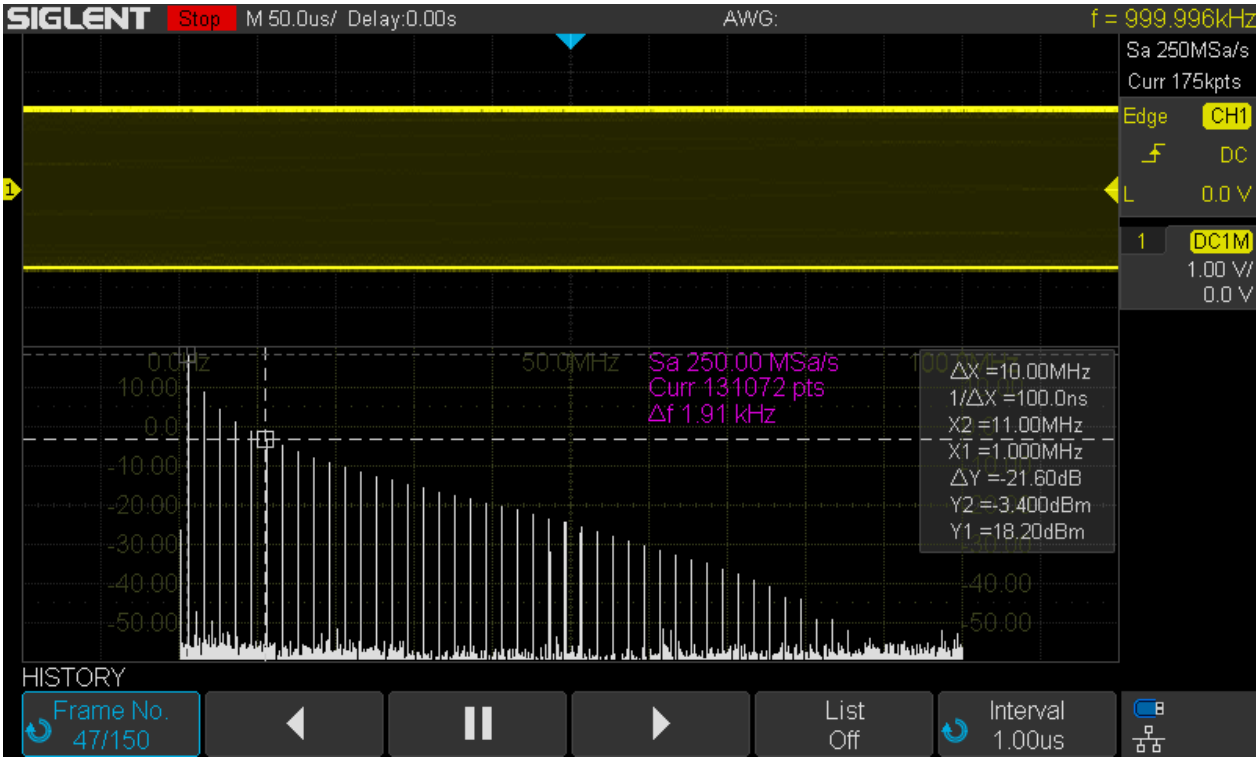
Hist_100MHz_vectors_x_list

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

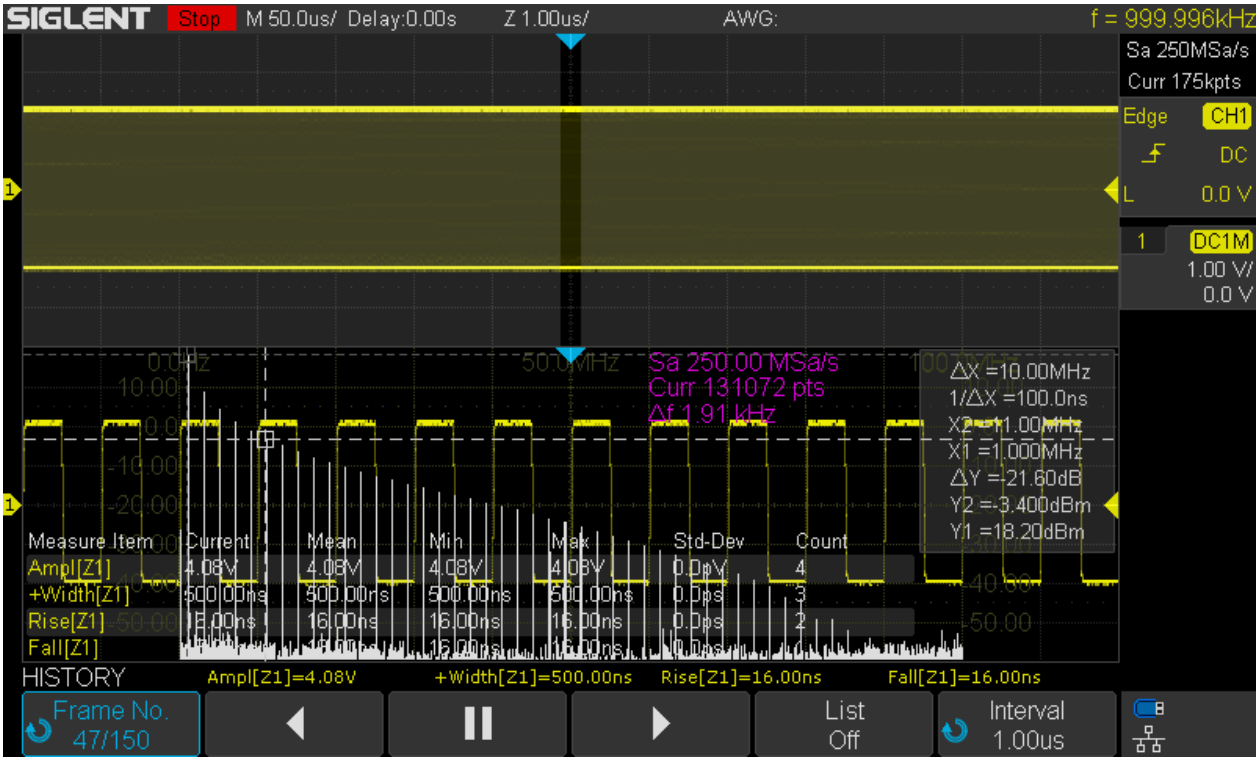
1. Push the universal select knob to display a dialog where we can jump to a certain record instantly by entering the desired frame number.
2. Use the playback function at an appropriate frame rate to emulate a fast forward or rewind like on a tape recorder.
3. Bring up the *Navigation* menu by pressing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

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

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



Hist_Square_10MHz_FFT_Cursors

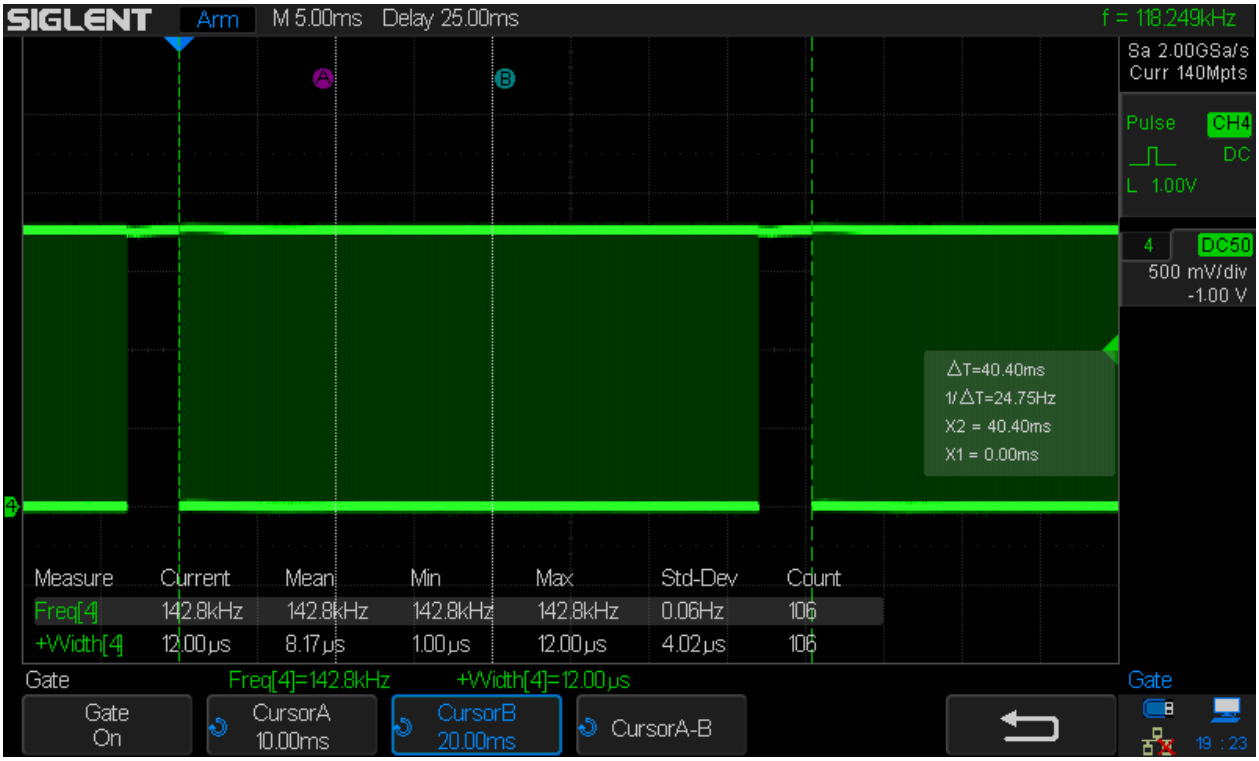


Hist_Square_10MHz_Zoom_FFT_Cursors_Meas

Sequence Mode

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

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



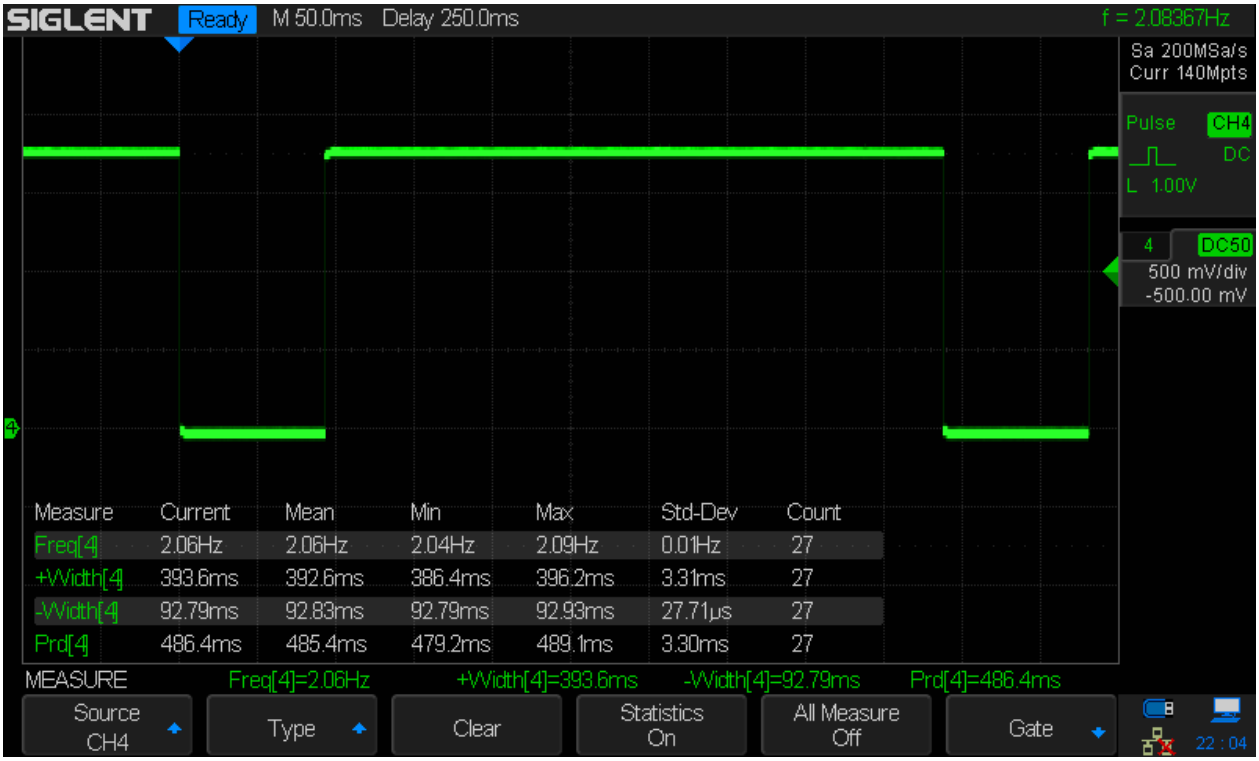
SDS1104X-E_TRIG_Std_50ns

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

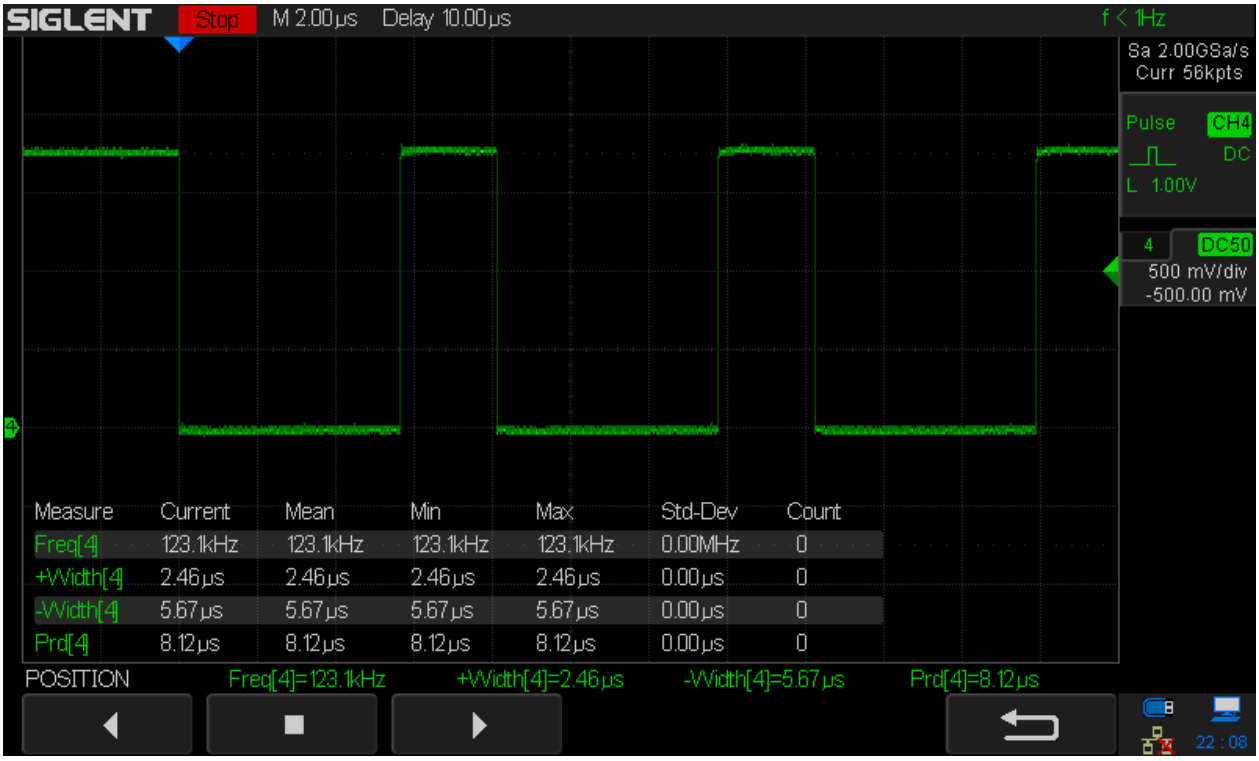
Now let's have a look at sequence mode. The sweet spot setting is with a timebase of 50ns/div and dots display, where the waveform update rate is measured as nearly 488k (depending on the input signal, it can be even slightly faster) and a screen update rate of ~2Hz, which is still quick enough to give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

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

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



SDS1104X-E_TRIG_SEQ_50ns



SDS1104X-E_TRIG_Pulse

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

SDS1104X-E_SEQ_100ns_5M_FM500k400_Display

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

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

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

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

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

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

Sequence Mode Table 1

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

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

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

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

Sequence Mode Table 2

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

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

Sustained Trigger Rate Comparison

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