Find Glitches Faster with PicoScope

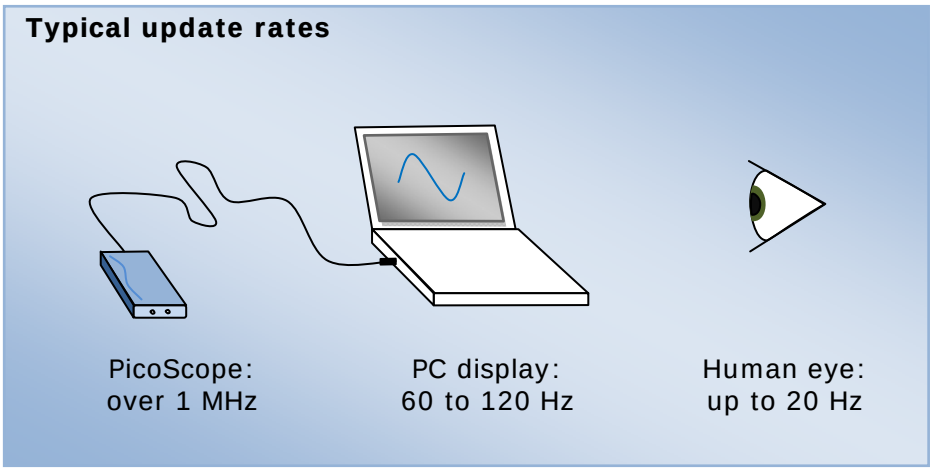
1. Introduction

In this article we explain how to find glitches more easily using PicoScope. Recent updates to PicoScope have increased the capture rate, which is an important parameter to consider when searching for intermittent events like glitches.

2. Capture rates of PicoScope oscilloscopes

There are currently 18 PicoScope real-time oscilloscopes, all with different speeds and other varying characteristics. What unites them and most other digital oscilloscopes is that the display hardware, which draws waveforms on the screen, is much slower than the data acquisition hardware, which converts the analog signals to digital data.

The update rate required for an oscilloscope display is usually assumed to be about 20 times per second, which is the minimum frame rate that the human eye finds comfortable when watching video. The upper limit is determined by the refresh rate of the screen, which is normally between 60 and 120 times per second.



The data acquisition hardware can often capture millions of waveforms per second. Clearly, the digital oscilloscope needs to reduce the amount of captured data before displaying it, but how can it do this without throwing away important information? The answer is: it's up to you, the user. If you understand how to use the scope to filter out unwanted data, you will be able to search more efficiently for intermittent events like glitches.

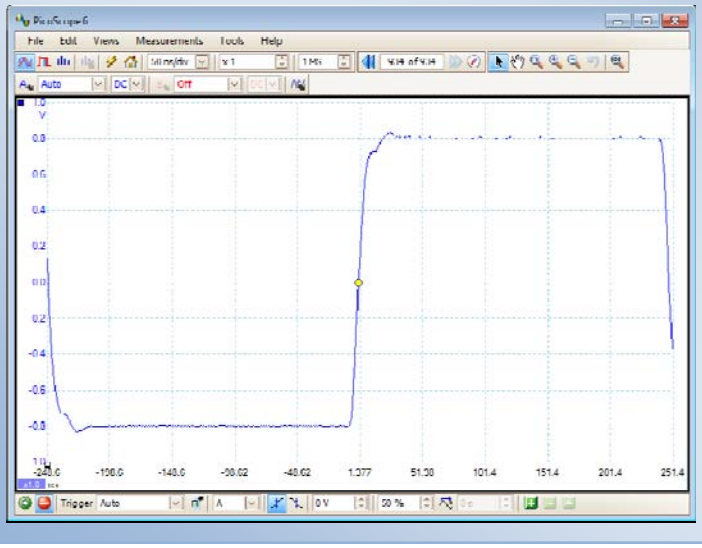
3. Real-time display mode

One way to match the capture speed to the display speed is simply to slow down the capture hardware. The scope's trigger is armed to capture one waveform, the waveform is displayed, and then the trigger is re-armed ready to capture another waveform. PicoScope operates in this way when you select Auto, Repeat or Single triggering mode. The drawback to this approach is that the scope spends only a small fraction of its time acquiring the signal, the rest being wasted waiting for the display. If an important event occurs in the signal during the waiting period, it will be missed.

With some deep-memory scopes, capturing large amounts of data can cause even slower updates. For example, with a 1 ms/div timebase you should be able to capture over 100 waveforms per second, but consider a deep memory scope running at 5 GS/s. It would acquire 50 million samples in this time, and displaying this amount of data could take many seconds per update. However, deep-memory PicoScope devices such as the 5000 and 6000 Series scopes avoid this problem by optimizing the data transfer, and can maintain a rate of 20 updates per second even under these conditions.

Real-time display mode example

Input signal:	2 MHz square wave
Timebase:	50 ns/div, 500 ns sweep time
Capture rate:	20 Hz
Display update rate:	20 Hz
Proportion of events displayed:	20 Hz / 2 MHz = **0.001%**



The above example shows that, even with PicoScope's optimized 20 Hz display update rate, only a tiny proportion of the input signal ever appears on the display. This makes it unlikely that you will spot an infrequent glitch. In some cases you can set up an advanced trigger to isolate the glitch, but this is possible only if you know what kind of signal you are looking for.

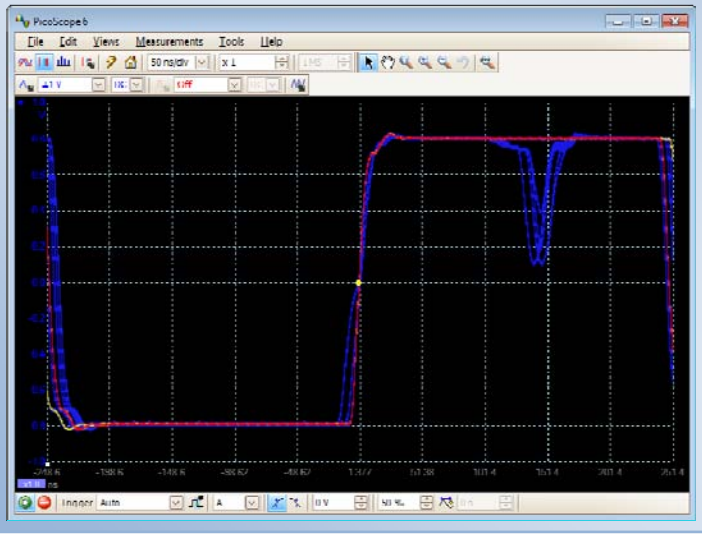
Deep memory scopes
An alternative method, if you have a deep-memory scope, is to capture a very long waveform containing many cycles. For example, a PicoScope 6000 Series scope could capture several seconds of a 1 MHz square wave. This gives you a good chance of capturing the glitch, but you must then search visually through the whole buffer to find the problem.

4. Persistence display mode

Persistence display mode overlays a large number of waveforms on the display, using various methods of color-coding or shading to distinguish frequent and infrequent data. You can arrange for old data to fade away after a specified time or to remain on the display until you erase it. Persistence mode has two advantages: it allows the scope to capture waveforms faster than it can update the display, and it makes it easier for you to spot a transient event.

Persistence display mode example

Input signal:	2 MHz square wave
Timebase:	50 ns/div, 500 ns sweep time
Capture rate:	≫ 20 Hz
Display update rate:	20 Hz
Proportion of events displayed:	variable



The hottest colors (such as red) indicate the densest areas of data where the majority of the waveforms are located. The coolest colors (such as blue) indicate transient events such as glitches and jitter.

The capture rate depends on the scope settings but is usually many thousands of waveforms per second. The display rate is typically 10 to 20 updates per second in this mode, although this is of little importance because each waveform persists for a long time on the display. In fact, with the persistence time set to infinity, you can leave PicoScope running overnight and still be sure of catching glitches.

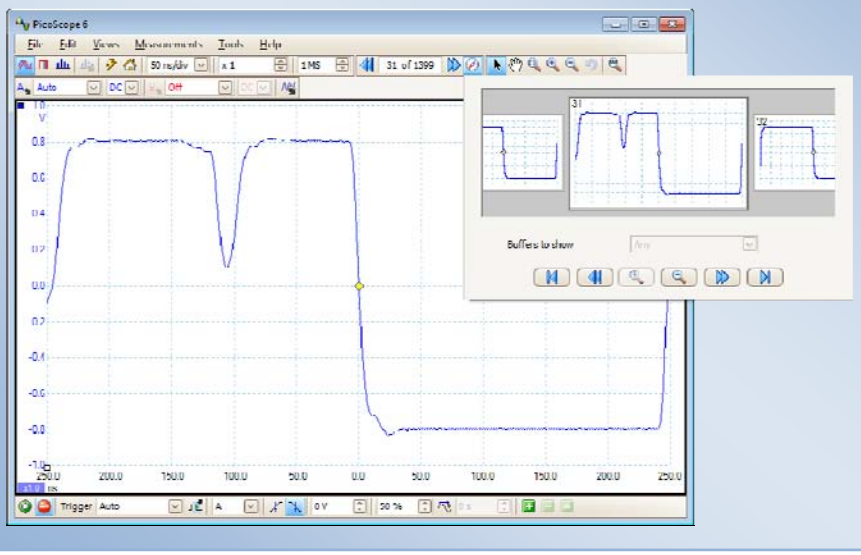
Persistence mode has allowed us to see a glitch that was practically invisible in real-time mode, and gives us a qualitative display of its rate of recurrence, but it cannot tell us exactly how often it occurs. Also, since the display is a composite of many waveforms, it is difficult to measure a single instance accurately.

5. Rapid trigger mode

In rapid trigger mode, the scope's trigger re-arms itself repeatedly with no need for intervention by the computer. This reduces the re-arm time to the order of a few microseconds or less, depending on the scope and the timebase selection. The scope rapidly fills the waveform buffer with a sequence of waveforms, resulting in a very high capture rate. PicoScope can achieve over 1 million captures per second in this mode, depending on the timebase selected.

Rapid trigger mode example

Input signal:	2 MHz square wave
Timebase:	50 ns/div, 500 ns sweep time
Capture rate:	1 MHz
Display update rate:	user-driven
Proportion of events captured:	1 MHz / 2 MHz = **50 %**



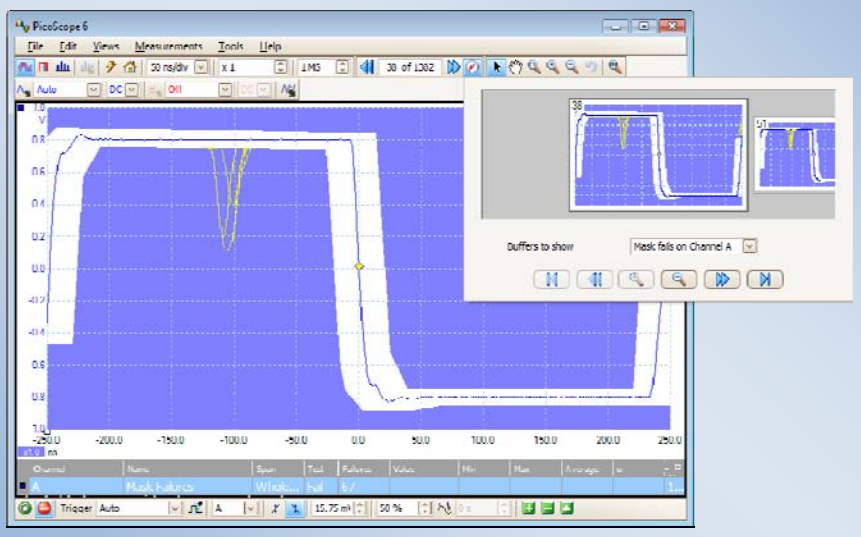
In the above example, the scope captured a closely spaced sequence of waveforms. By scrolling through the Buffer Overview, we found an example of the glitch. Selecting this waveform copied it to the main display, where a range of measurement tools such as vertical and horizontal rulers are available. We could also estimate the frequency with which the glitch occurs by counting the number of instances in the buffer.

6. Mask limit testing

Visually inspecting the Buffer Overview is easy with 20 waveforms but not practical when you have 10,000. This is where Mask Limit Testing comes to the rescue. First you set up a mask, and then you filter the Buffer Overview to show only those waveforms that failed the mask test. You don't need to know how to draw and edit masks, because PicoScope can do it all for you.

Mask limit testing example

Input signal:	2 MHz square wave
Timebase:	50 ns/div, 500 ns sweep time
Capture rate:	1 MHz
Display update rate:	user-driven
Proportion of events captured:	1 MHz / 2 MHz = **50%**



In the example above, we created a mask (the blue areas) around a good waveform using the **Tools > Masks** command and then the **Generate** button. This created a mask with the specified clearances around the waveform. We then set the Buffer Overview filter to "Mask fail on Channel A". This hid all of the 10,000 waveforms except the 100 or so that contained the glitch.

Mask limit testing works in any trigger mode. In this example we used it in combination with rapid trigger mode to capture closely-spaced waveforms at a rate of about 1 million per second.

7. Conclusion

We recommend that you familiarize yourself with PicoScope's Persistence Display mode, Rapid Trigger mode and Mask Limit Testing. They can all be used to improve your chances of finding an infrequent glitch, particularly when you don't know its shape or frequency of occurrence.

8. Further information

Download oscilloscope data sheets, User's Guides, Programmer's Guides and Software Development Kits from:

www.picotech.com

