DAQ Syncing and Triggering

Cole Meisenhelder

January 16, 2019

1 Introduction

We determined that the ACME II DAQ system had a triggering issue that was the primary cause of our noise problem, but was able to be fixed with three changes. This writeup will lay out the DAQ system design, describe the problem, and describe the three changes that eliminated our noise problems.

2 The System

The primary DAQ system in ACME II was a NI PXIe-5171R 8 channel PXI Oscilloscope, which we typically refer to as the FPGA. This FPGA oscilloscope was housed in a PXIe-1075 18 slot chassis, which was connected to our DAQ computer by a PXIe-8375 MXI Express card paired with a PCIe 8375 for remote control. Each of the 8 channels was normally connected to one of the 8 experiment PMT signals, and was provided triggering information through the PFI0 input, which is accessible through a NI breakout box that we connect to with a NI SHH19-MH19-AUX cable. Note that this looks like an HDMI cable, and has the same ends, however it apparently has nonstandard wiring, which caused problems when Adam first tried to work with the system, so make sure that the Aux I/O cable is an NI cable. This triggering signal is produced by our DG645 and should have extremely low jitter of 10^{-8}, which we can confirm on a bench-top oscilloscope where we see no timing error. The bit file for this FPGA was developed by NI engineers in contact with Adam to allow us to use the PFI0 trigger, but is otherwise the same as the example file available online.

We also have a backup system consisting of 4 PXI-5922 PXI oscilloscope cards. This system is simpler in some ways, but each card only uses two input channels, and each card has an option for a clock input and trigger. This clock input must be a square wave unlike the FPGA which can take a square or sine wave input. Additionally, this system has been configured from the start to use the NI-SCOPE software driver.

3 Trigger Noise

After finishing the ACME II measurement we discovered that our excess noise was correlated with the beginning and end of a molecular beam trace. Further investigation revealed that there was a significant source of timing noise in the DAQ system. This was very evident when the polarization switching control pulses from the DG645 were sent directly into the PXIe-5171R module and recorded in LabView. Specifically we sent in two signals which were 10 ms long pulse trains of 200 kHz square waves repeating at 50 Hz that are roughly 180 degrees out of phase with each other. The trigger pulse was sent from the DG645 every 500 ms, as we did normally during the run of ACME II. We could see that the arrival time of these 50 Hz pulse trains varied over a range of about 120 ns. This 120 ns corresponds to shifting one full digitization point in either direction, and the traces can be seen to drift somewhat continuously over this range as shown in figure 1. This jitter also appears to be linear and periodic within this range.

4 Solving Trigger Noise

Once we had indentified that there was a timing issue, we checked that it was not an issue with the signals themselves by connecting them to a standard lab scope and saw that the problem disappeared. This confirmed that what we were seeing was being caused by the DAQ and was not a true signal error. We were eventually able to fix the problem by implementing three changes.

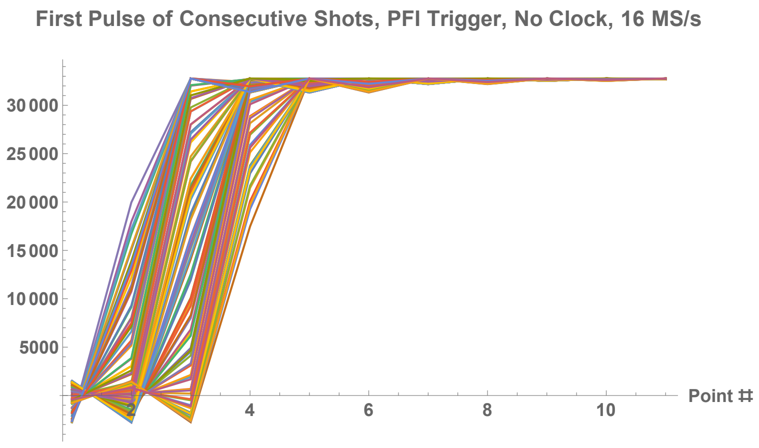

Figure 1: The leading edge of consecutive 10 ms pulse trains overlapped. No clock reference was used, and we sampled at 16 MS/s so each point corresponds to 62.5 ns.

4.1 Clock Syncing

Our first thought was that there must be some sort of clock syncing error, as we had not synced our SRS Rb clock to the DG645 and we had no reference clock synced to the FPGA itself. The 5171R has a timebase accuracy of only ±25 ppm when using the onboard clock, which is enough to explain the behavior we saw. We first synced the DG645 to the clock using the clock input on the DG645, but saw no change in the behavior. We then determined that we could sync the clock to the FPGA through two methods. In either method you can use either a sine wave or a square wave. The first is to use the AUX I/O inputs to connect the clock, which is through pin 6 of the breakout box (this can be found online in the getting started guide for the 5171). This method requires that the input source of the clock be set in LabView to "VAL_CLK_IN". The second is to use the onboard clock of the chassis, which has a BNC input for syncing with an external clock. This is probably preferable if you are syncing multiple cards to the clock, and requires you to use the LabView source name "VAL_PXI_CLK". For the FPGA we observed no difference between these methods, but with the PXI-5922 cards we saw one card show jitter when using this clock method while it was fine with the direct clock input. This needs to be investigated further before trusting though.

The PXI-5922 cards can be synced to a clock using either the same chassis clock sync method as the FPGA, or by using the front panel CLK IN connection. There are some concerns with using the chassis clock as it may not cleanly distribute the clock signal to all boards as previously mentioned. However, the front panel clock input required us to generate a square wave synced to our Rb clock as our clock source only outputs a sine wave. For this a SRS signal generator was sufficient. We also had to input a clock signal individually for all 4 boards. The LabView settings for the clock input with these cards are significantly easier to remember as LabView will create a dropdown menu of the available choices.

When synced with a reference clock there was a clear improvement in the behavior as the drift rate between consecutive pulses clearly decreased, but the drift over three points remained. This was clear as the overlapping pulses no longer were able to drift over the full two point range in a single 500 ms triggering window, and the traces overlap significantly more. This behavior can be seen in figures 2 and 3, where the gap in the overlap clearly shows the periodic nature of the drift. These two plots were generated with the FPGA synced to the DG645 clock output and the Rb clock respectively, and in both cases the DG645 was synced to the Rb clock.

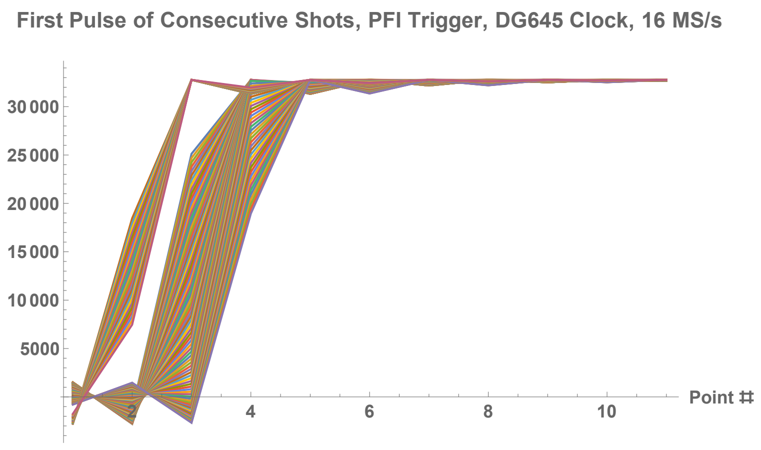

Figure 2: The leading edge of consecutive 10 ms pulse trains overlapped, sampling at 16 MS/s. The clock was synced to the clock output of the DG645.

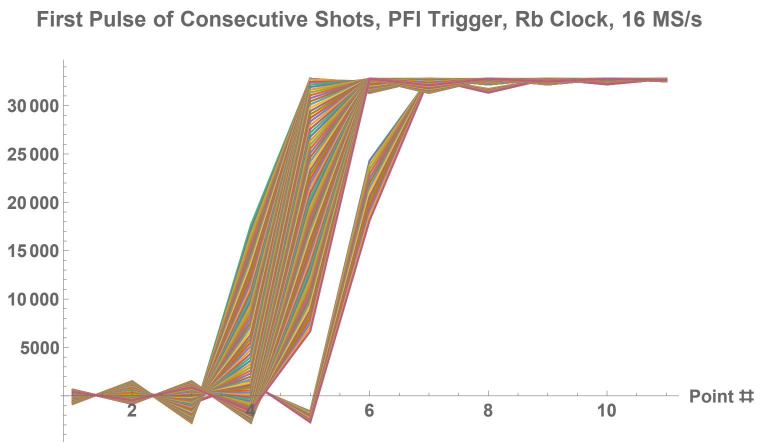

Figure 3: The leading edge of consecutive 10 ms pulse trains overlapped, sampling at 16 MS/s. The clock was synced to the Rb clock.

While this did not fix the issue, it did improve the jitter that we were seeing. To try to better understand the source, we looked at the overlap of the rising edges of the 200 kHz square wave within a single 10 ms pulse. We did this both with and without a clock and found that there was significant drift over a range between two digitization points that went away when synced to a clock. This drift without a clock corresponds to a drift rate of at least 6 ppm, which is only a lower bound, but is within the FPGA spec. These results can be seen in figures 4 and 5 where in the clock case all of the traces overlap so well as to almost look like a single trace. This is the behavior we want to see at this level, and indicated that the clock was doing its job, and most likely was not the problem anymore.

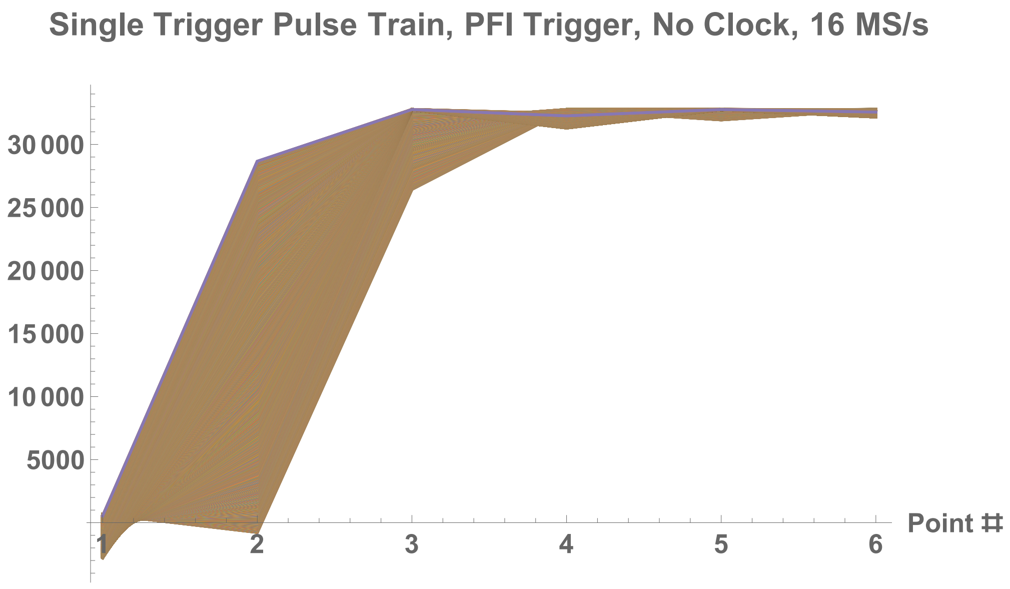

Figure 4: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 16 MS/s. No clock reference was used, and the signal drifts at a rate of at least 6 ppm.

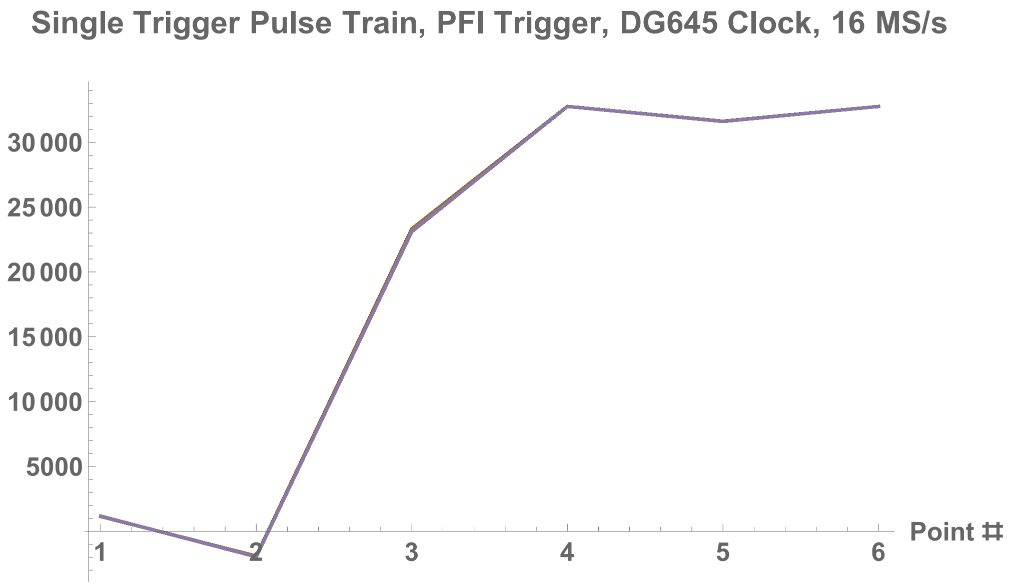

Figure 5: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 16 MS/s. The clock was synced to the clock output of the DG645.

4.2 Sample Rate Selection

The FPGA clock runs at 250 MHz, and so we were unsure of how the software actually handles decimation by non-integer values, like the 16 MS/s sample rate we had used in ACME II and in our tests up to this point. The manual indicates that the system has the option to decimate by n. The choice of 16 MS/s had been made for ACME II in spite of this in order to cleanly fit an even number of digitization points within the 200 kHz polarization switching cycle. In LabView we set a sample rate rather than the decimation, and the FPGA bit file directly takes this input and decides how to decimate. However, we were unable to determine directly from this file or from conversation with NI engineers how the program interprets a non-integer input for the decimation factor. The best understanding of how this must be occurring is that the FPGA somehow interpolates between the two nearest integer decimation rates to almost sample at the specified rate, however NI could not confirm how the system operates without integer decimation.

We experimented with changing the sample rate and found that by switching to integer decimation the drift rate decreased significantly once again. This is visible in figure 6 as the drift, while still periodic, covers significantly less of the three point region, leaving a larger gap than in figure 3.

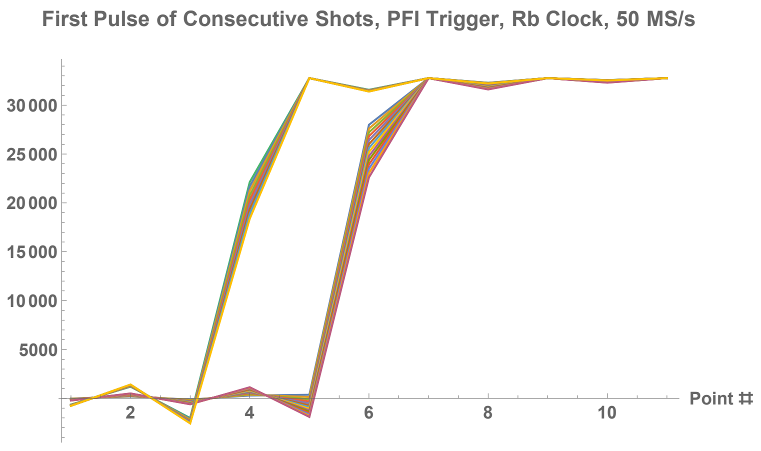

Figure 6: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 50 MS/s, and using a digital trigger. The FPGA was synced to the Rb clock. There is a clear decrease in the linear drift rate at this integer decimation sample rate.

In trying to track down the cause of the remaining drift and periodicity, we repeated these tests with the analog trigger settings instead of the PFI trigger. For this we used the same trigger signal sent into channel 1 of the FPGA. We tried this test because the FPGA has a trigger jitter spec that is 8 ns for a digital trigger and one Sample Clock timebase period for an analog trigger. This difference initially did not make sense to us until discussing with an NI engineer who told us that the the digital trigger uses a 125 MHz clock rather than the 250 MHz clock used for the analog trigger. The tests seemed to somewhat confirm this spec as when triggering off of the PFI input we saw drift over 3 points at all sample rates, and when using an analog trigger we saw drift over just two points at all sample rates. This can be seen in figures 6 and 7. However, this did not match the 8 ns spec for digital triggers because of how they scaled with the sample rate. NI also could not confirm whether this behavior was expected or not as even R and D seemed to lack an understanding of how they defined these things.

The PXI-5922 has a similar spec for the analog triggers (the only option) so we repeated these tests with this system, and found that using integer decimation sampling rates and syncing to a clock we no longer had any discernible jitter. The results of this test can be seen in figure 8 This confirmed that there must either be a problem with our FPGA code or the FPGA hardware itself that was out of spec.

4.3 Software Changes

The final change required to remove the noise was switching over from our old code. We previously had used the ACQ session palette tools to control and read from the FPGA, but NI engineers suggested we try using the newer NI-SCOPE driver as the ACQ tools are no longer supported. Additionally, the tests with the PXI-5922 suggested that this might be a potential solution as it was one of the few differences between the code we used for the different systems. Switching over fixed the remaining issues, but neither we or NI can actually explain why this worked. It seems to be a bug in the old code that will never be fixed with unsupported drivers. Figures 9 and 10 show this clean data for both digital and analog triggers.

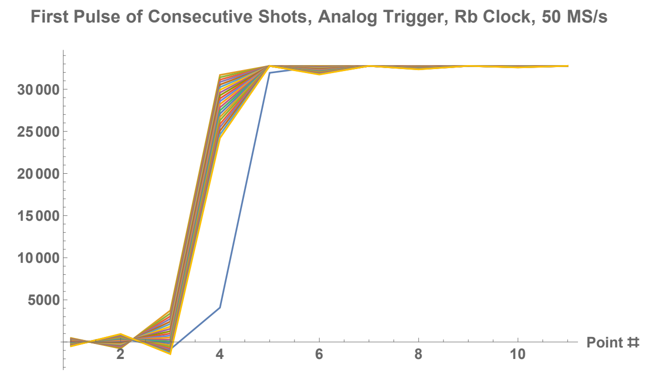

Figure 7: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 50 MS/s, and using an analog trigger. The FPGA was synced to the Rb clock. The drift appears periodic over two digitization points as opposed to 3 in figure 6.

5 Conclusion

After implementing these three changes we were able to remove the noise to the level that we expect to be able to measure. Going forward three things should always be implemented. First, always sync the scope to a clock source. Second, make sure that you are clearly choosing a sample rate that is an integer decimation of the sample clock. For ACME III this will most likely be 12.5 MS/s, as we don't want to increase our data taking rate for space reasons, and we will have to adjust our polarization switching rate to match the sample rate nicely. Third, never use the ACQ session palette tools to connect to the scopes as it is unsupported software with dangerous bugs we do not understand. The NI-SCOPE driver is currently supported and shows no problems. For what it's worth, it is also easier to work with.

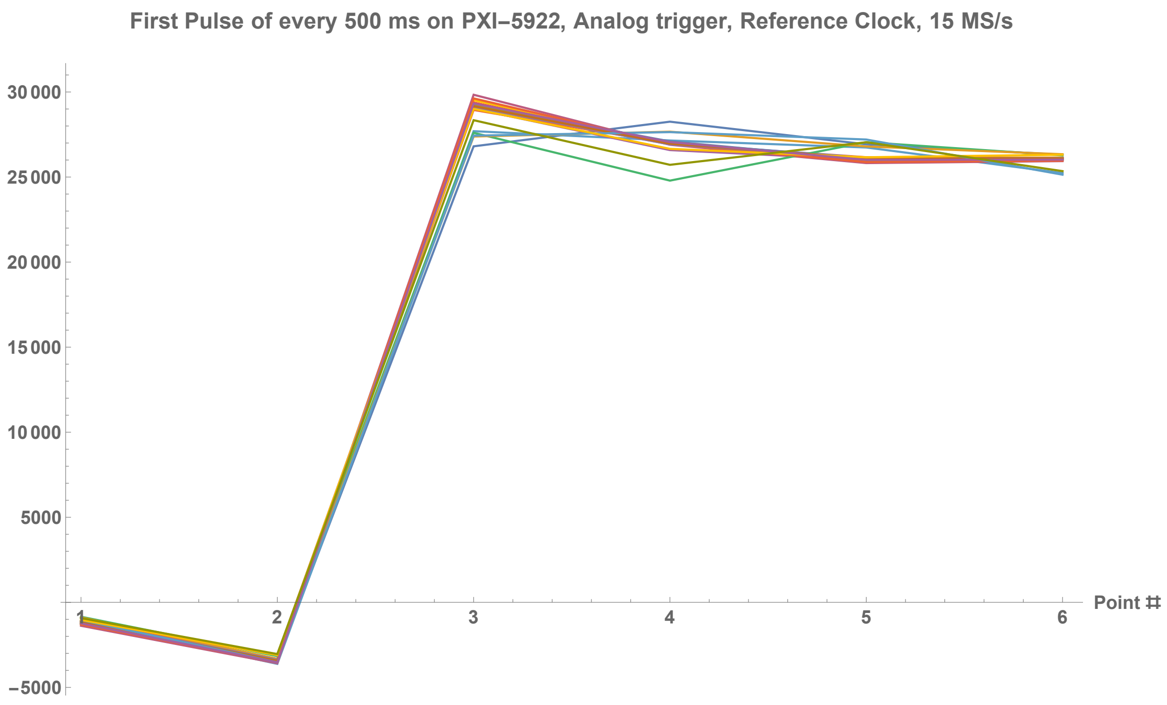

Figure 8: Data taken with the PXI-5922 DAQ cards. The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 15 MS/s, and using an analog trigger. Synced to the Rb clock by a 10 MHz square wave generator directly synced to the Rb clock.

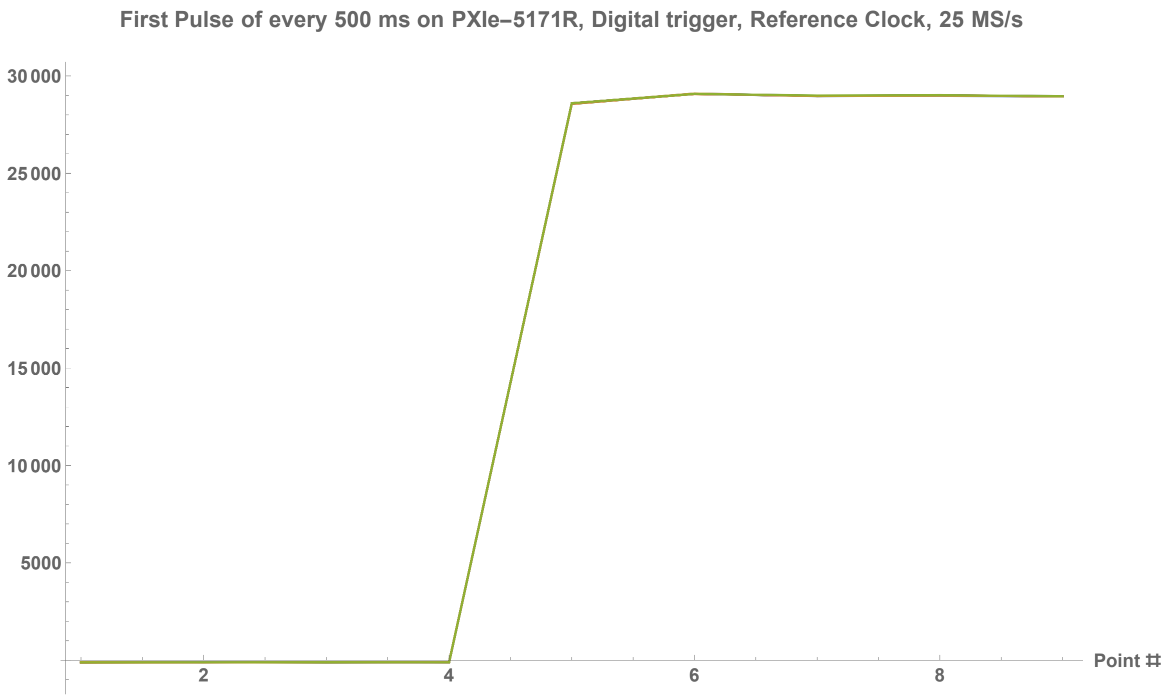

Figure 9: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 25 MS/s, and using a digital trigger with the NI-SCOPE drivers. No timing jitter is visible.

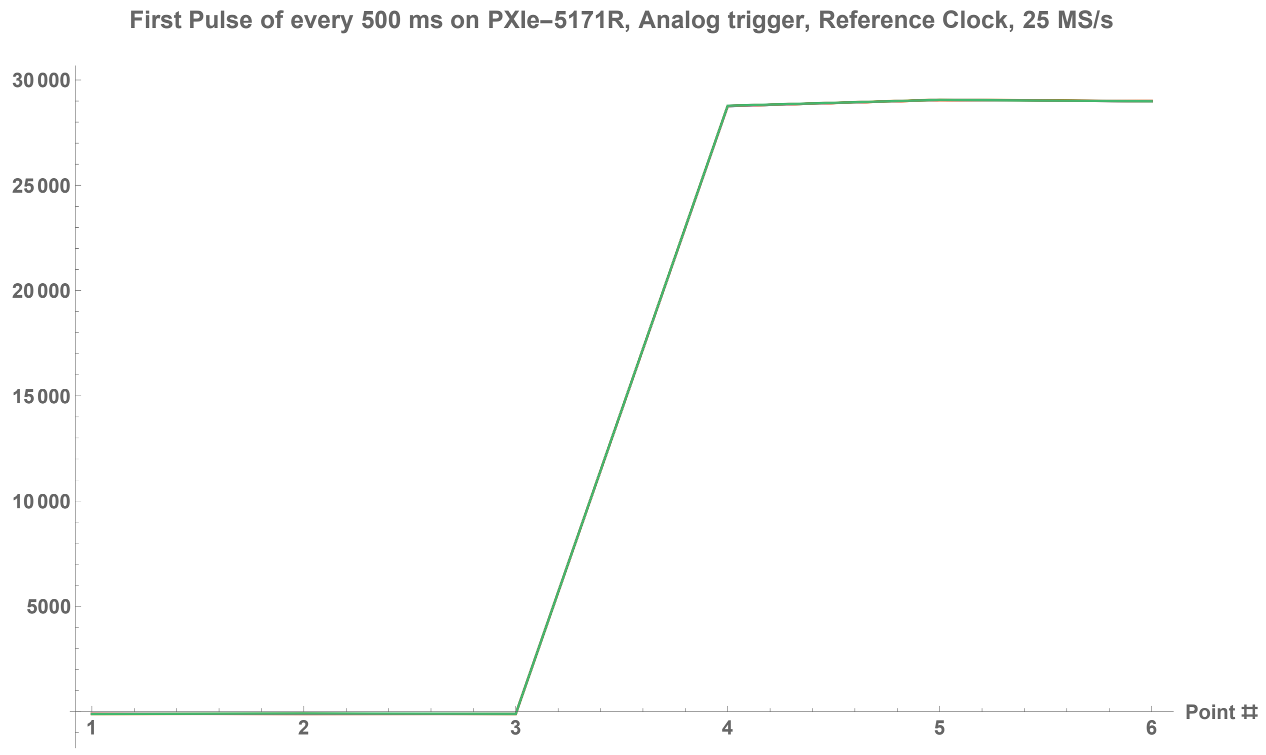

Figure 10: The rising edges of the 200kHz square wave within a single 10 ms pulse train overlapped, sampling at 25 MS/s, and using an analog trigger with the NI-SCOPE drivers. No timing jitter is visible.