plug & play instruments
oscilloscopes

Phone +64 9 524 7456
Fax +64 9 524 7457
Email info@cleverscope.com
28 Ranfurly Rd, Epsom
P.O. Box 26-527
Auckland 1023
New Zealand

5 April 2009 v1.0

Cleverscope Triggering and Protocol example

Summary

Cleverscope has two independent triggers that can be used to find features of interest in the waveforms you are exploring. The two triggers can be combined to trigger on a combination of a time duration between the two triggers, and a count of one of the triggers. In addition we can trigger on either trigger happening.

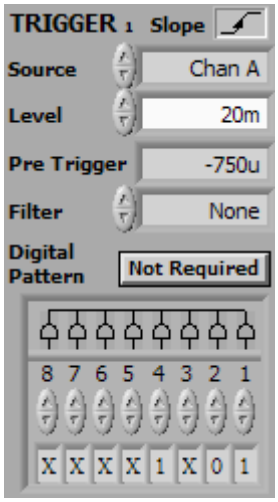
To illustrate the triggering capability we explore the Cleverscope signal generator operation. The signal generator uses an SPI (Serial Peripheral Interface) to set the output frequency. For debugging purposes the signal generator can send its set frequency out a serial port, in ascii format. We use the protocol system to decode the results. We make use of the large 4 Mega sample buffer to capture once and zoom from then on.

Example

The example waveform used throughout this document can be loaded from '*Trigger - protocol example.apc*'.

Triggering Capability

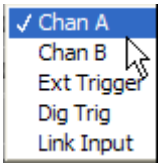
The Cleverscope includes two identical trigger systems, called Trigger 1 and Trigger 2.
For each trigger you can select:



Slope

Select a rising or falling trigger for Chan A, Chan B, Ext Trigger or Link Input. The digital inputs each have their own individual slopes.

Source

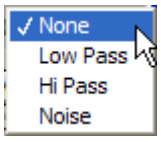


You can choose the trigger source – Link Input is the Rear Link connector.

Level

You can set the trigger voltage level. Separate voltage levels are maintained for Chan A, Chan B, External Trigger and Digital Inputs. In all cases hysteresis is applied to minimize the effects of noise. The hysteresis can be increased using the Trigger Filter (see below).

Filter



The trigger filter changes the way the trigger system responds to the input signal. If None, a small amount (1 minor division) of hysteresis is used to reduce the effects of signal noise. If Low Pass, high frequency signals are ignored, while for Hi Pass, low frequency signals are ignored. The Noise setting uses a hysteresis value equal to one major division on the graph.

Digital Pattern

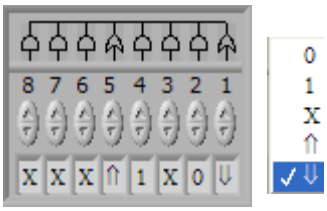
If you set the Digital Pattern qualifier as Required, then for Chan A, Chan B, Ext Trigger and Link Input triggers, the Digital Input Pattern acts as an enable on the trigger system – if the pattern is met, the trigger can happen, otherwise the trigger is disabled. The example shows Digital Inputs 1 and 4 required high (1), Digital Input 2

required low (0), and all the rest don't care. The digital pattern is useful in setting the state in which a system is before triggers are recognized. For example when triggering on SPI communications, only trigger for the correct SPI address.

When using the digital inputs as the trigger source (Source = Dig Trig) each individual Digital Input can be configured to require a low level (0), a high level (1), or be don't care (X), or be a triggering input with a rising edge, or a falling edge.

In addition individual input conditions can be AND or OR combined. In this example, to generate a trigger, we require:

(Input 1 to be falling OR Input 5 to be rising)
AND input 4 to be high (1)
AND input 2 to be low (0),
AND all other inputs are don't care.



Trigger 2

Trigger 2 is identical to trigger 1. It can be setup using the View/Display Trigger 2 Settings menu item.

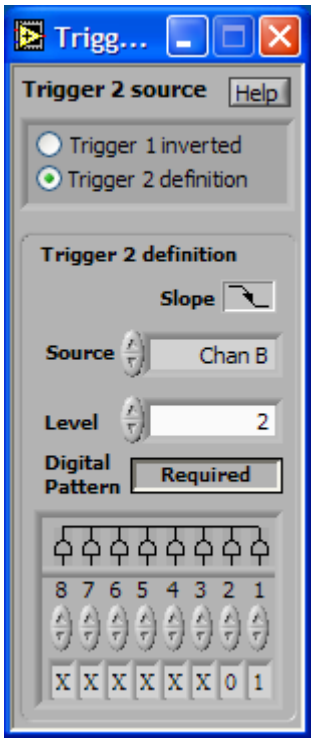
The Filter setting above applies to both Trigger 1 and Trigger 2.

The Trigger 2 window is shown to the right.

Trigger 2 source

The Trigger 2 source defines how the Trigger 2 definition is derived:

- If Trigger 1 inverted, Trigger 2 is set the same as Trigger 1, except all edge directions are inverted.
- If Trigger 2 definition, Trigger 2 is set as the specified in the 'Trigger 2 definition'

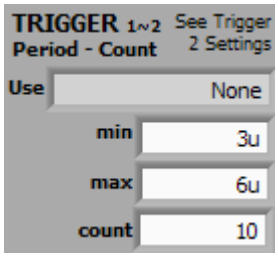


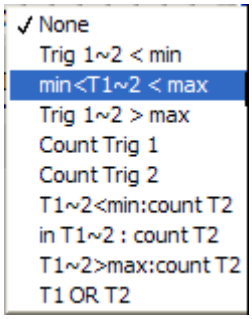
In the example, we are using a separate Trigger 2 definition, with a falling edge trigger on Channel B. The trigger level is 2V. A digital pattern is required, with Digital Input 1 being 1 and Digital Input 2 being 0. All the other Digital inputs are don't care.

Using the Trigger Definitions

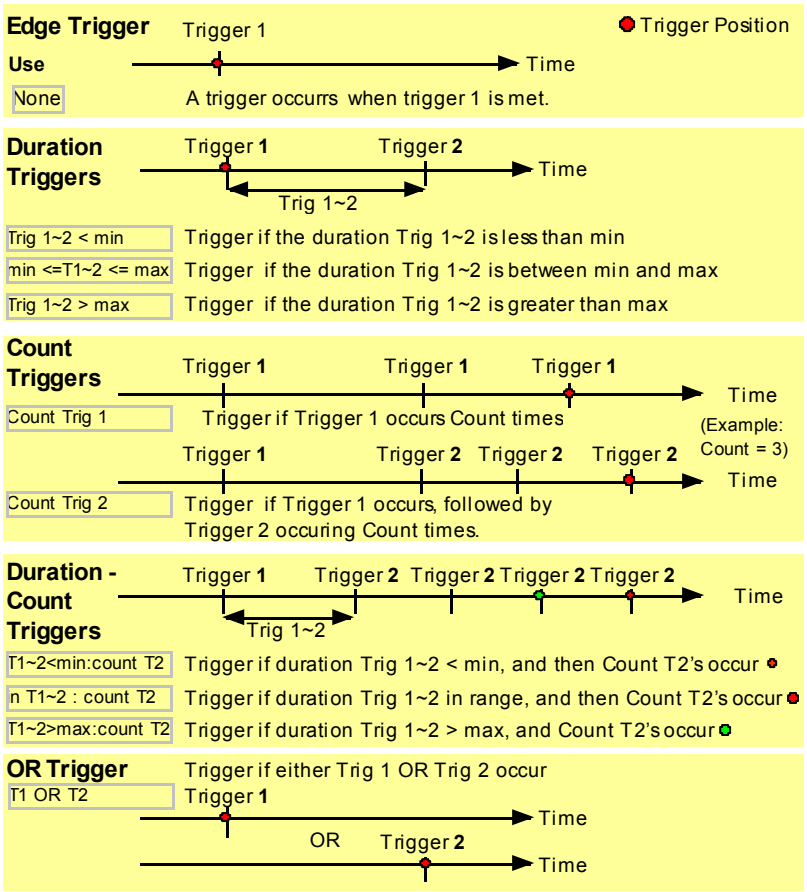
The two trigger definitions are combined according to the lower part of the trigger panel on the Cleverscope Control Panel.

The **Use** drop down list defines how the two triggers are combined. If the setting is 'none' then only Trigger 1 is used.





For other combinations this diagram shows the available Trigger 1~2 combinations:



In the table above the small red and green filled circles represent a trigger occurring, and the trigger position on the resulting graph.

As an example, if **Use** is 'T1~2>max', and max = 230us, then if the duration between Trigger 1 and Trigger 2 is greater than 230 us, or even if there is no Trigger 2 at all, the trigger system will trigger. The trigger position is the point at which Trigger 1 occurred.

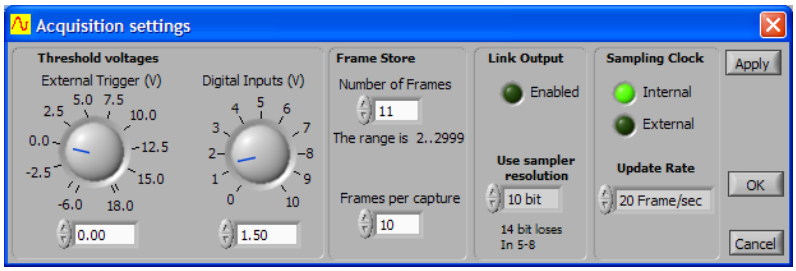
If we then change **Use** to 'T1~2>max: count T2, and max = 230 us, and count = 12, we wait for a Trigger 1 ~ Trigger 2 duration of more than 230us, and then make a trigger when Trigger 2 occurs for the 12th time.

Capture Example

Acquisition setup and signal capture

This example explores the outputs from the Cleverscope signal generator. The signal generator uses an SPI controlled AD9834 DDS signal generator chip. The acquisition unit was programmed to include a diagnostic serial output for displaying the current frequency. We used two Cleverscopes, and two copies of the application to do these tests.

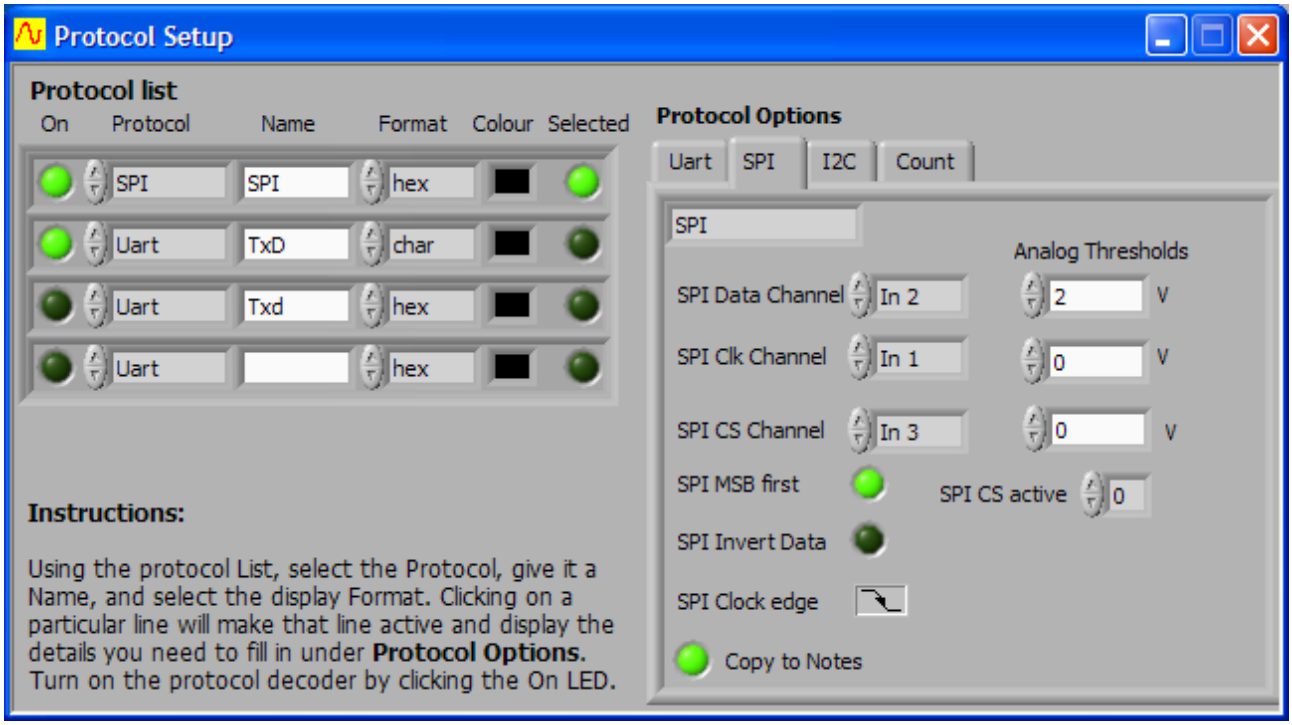
The Sig Gen Cleverscope was setup with a sweep method of 'Sweep in acquisition unit', with a base frequency of 1kHz, and a step frequency of 1 kHz. The acquisition settings were as shown, meaning that 10 frequency steps were performed per capture.



The Capture Cleverscope was set up with the following connections:

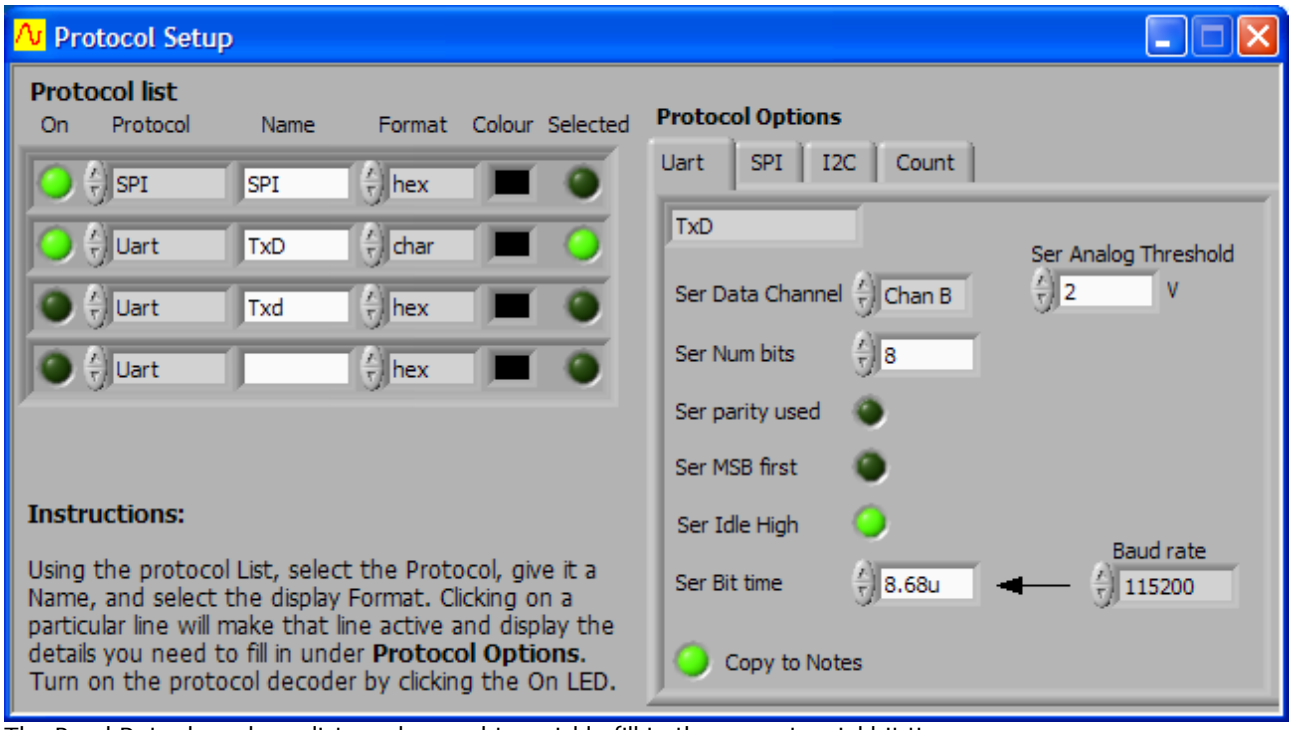
- Channel A – The signal generator output
- Channel B – the serial link TxD output
- Dig In 1 – the SPI clock channel (the falling edge is active)
- Dig In 2 – the SPI data out channel (with Most Significant Bit first)
- Dig In 3 – the SPI chip select (active low)

Here is the Protocol setup for the SPI:



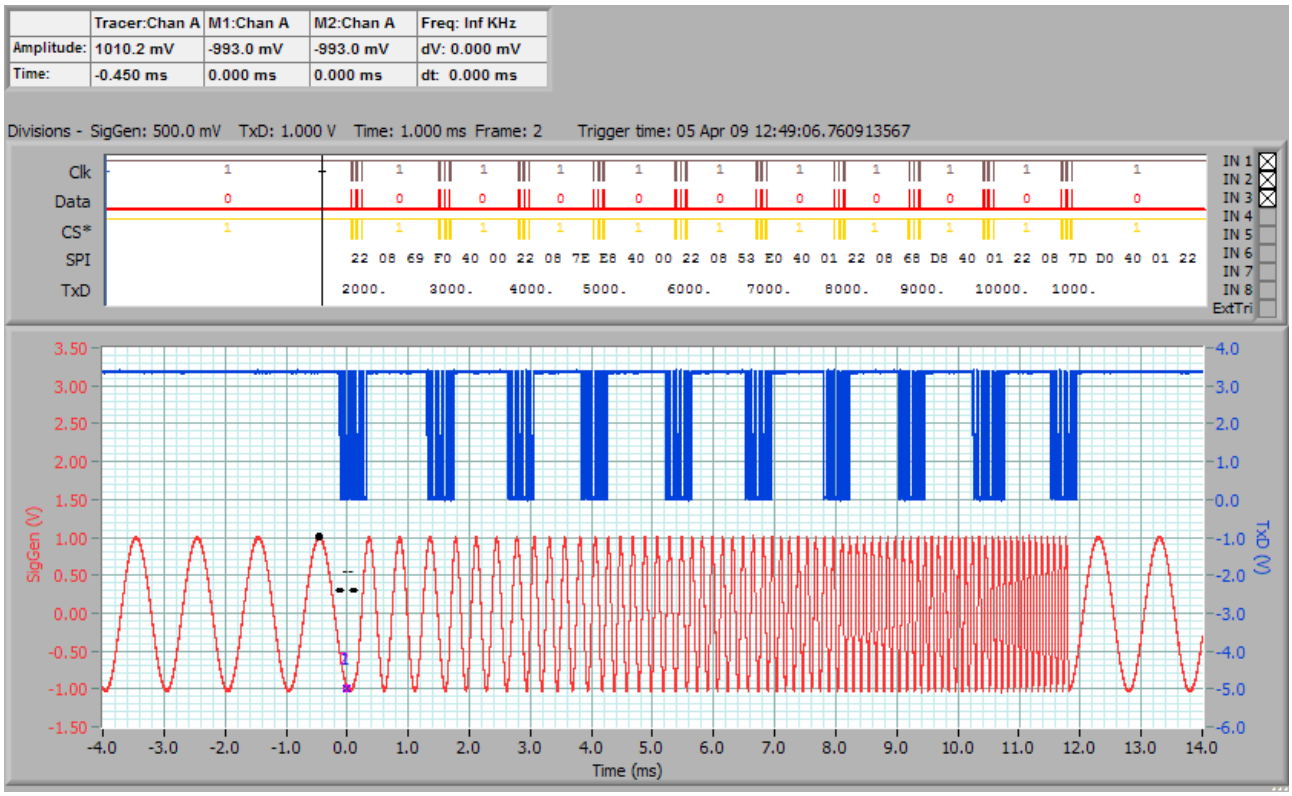
We used Copy to Notes to get a text record of the decode (which is also formatted to be easily readable in Excel).

The serial channel (labelled TxD) was setup to receive a 115 Kbaud serial data stream, connected to Channel B. The protocol decoder works using either digital or analog signals. The setup was:



The Baud Rate drop down list can be used to quickly fill in the correct serial bit time.

We setup triggering as detailed later, and captured one succession of frame captures, using the Capture Cleverscope:



You can see the frequency starting at 1 kHz, and then stepping up at 1kHz intervals. There were 10 frames captured (including the first 1 kHz setting) before the sig gen was returned to 1 kHz.

Note that the sig gen outputs a signal without discontinuities in the signal, smoothly changing from one frequency to the next.

The decode above was achieved after clicking the 'Get Frame' button. With a view 18ms wide, the application does not normally have enough samples in the display buffer to decode a 1us (or higher) SPI signal. Even though the Cleverscope Acquisition Unit (CAU) always captures the maximum number of samples at maximum time resolution (18ms with 10ns resolution in our example), the application only transfers a limited set of samples to the display to keep things fast. The protocol decoder system does automatically increase the number of samples transferred in order to correctly decode a signal, but only up to a maximum of 65000 samples. With 18 ms screen width, this results in 280 ns/sample. With a 1us clock rate, and a requirement for 5 samples per SPI clock cycle (ie 200 ns/sample), the resolution is insufficient to capture the SPI, but sufficient to capture the TxD Uart. To improve the resolution further, we click on the 'Get Frame' button, which automatically transfers the full buffer, and then re-runs the decoder, now with 10ns resolution. This is good enough to correctly decode the SPI.

After 'Get Frame':

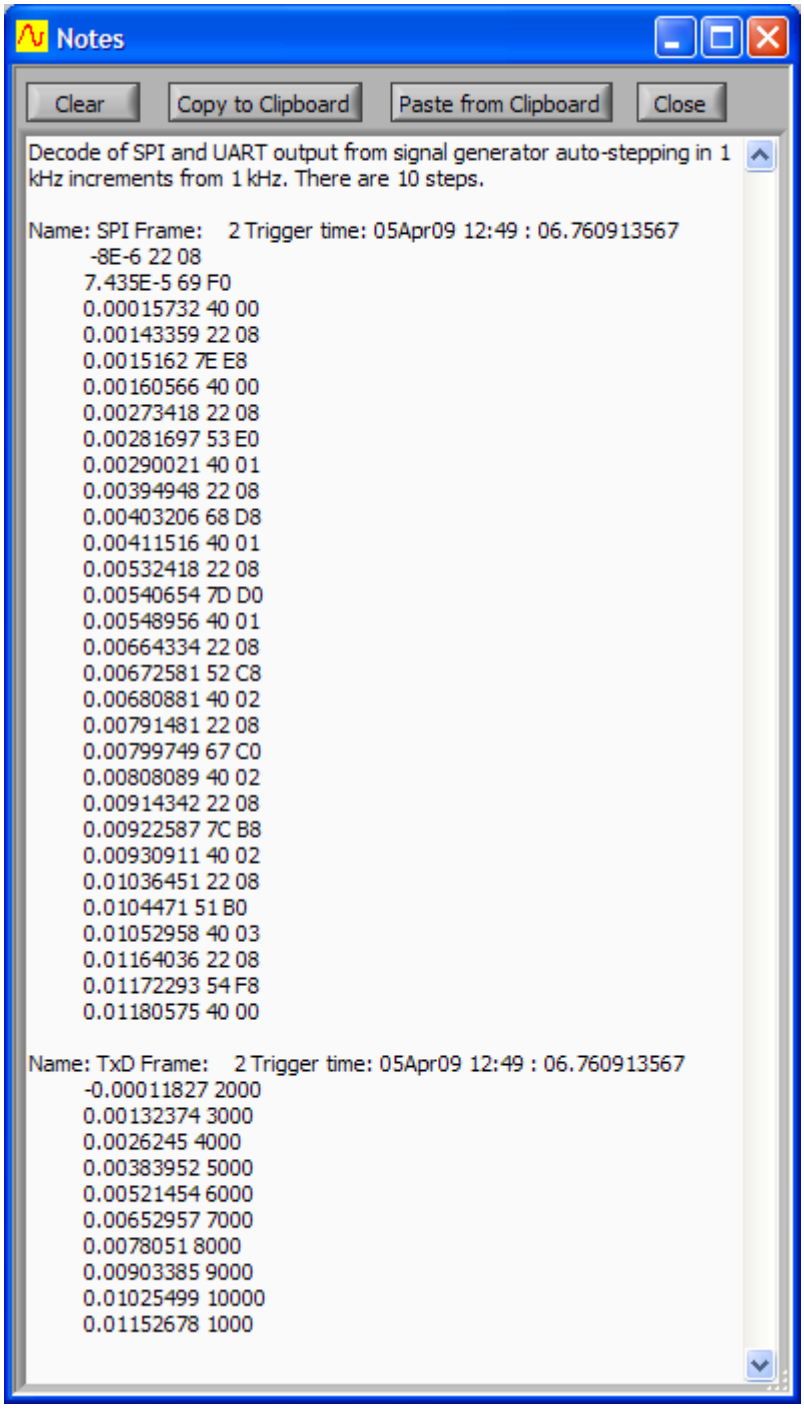
Duration	18m
Resolution	10n
N Display	1800001
F Sample	100M

We selected the decode to be copied to Notes (the Cleverscope facility for keeping notes about whatever is pertinent to the capture). The Notes output can be useful for documenting the signals decoded, including timing analysis.

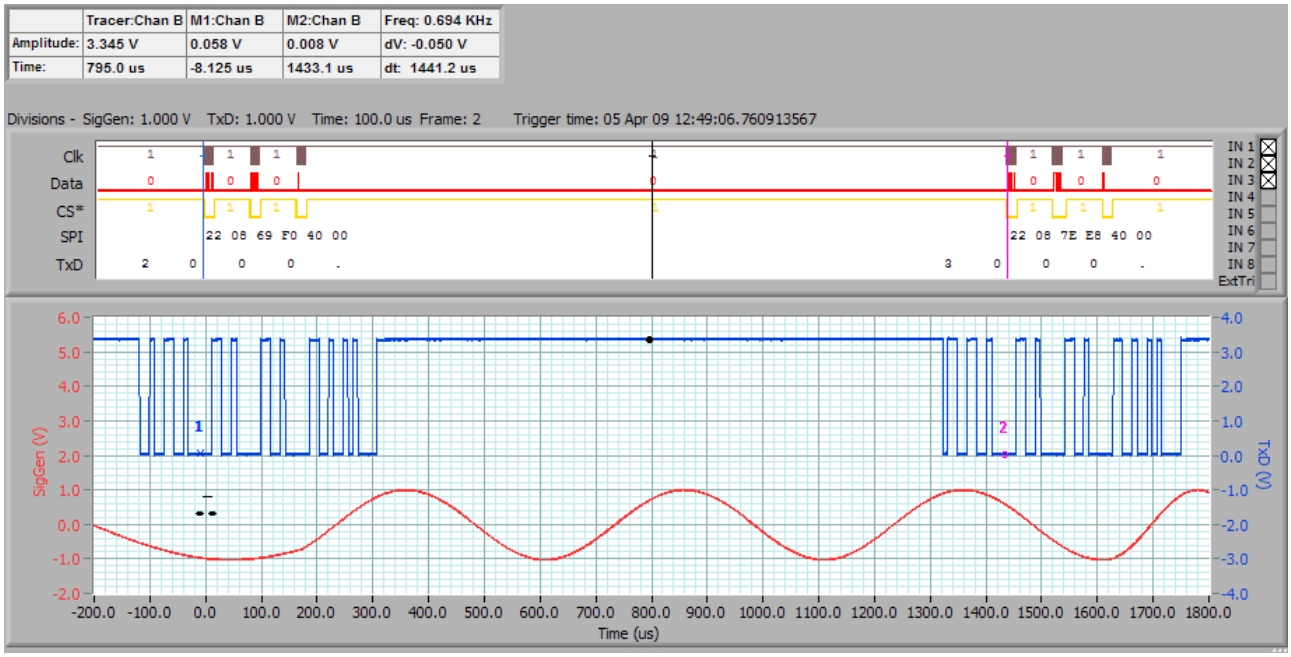
Here we can see that the trigger occurred on 5 April 09, at 12:49:06.760913567 secs. The trigger time is maintained with 10ns resolution, and so is useful for measuring the precise trigger – trigger time between successive frames or captures. (Cleverscope can be configured to capture up to 3000 frames, each triggered by a separate trigger event).

We can also see the time at which each decoded message was output. The 22 08 sequence is used to program the AD9834 to output a new frequency. So we can see that the time between the first and second updates was 1433.59us – (-8 us) = 1,441.59 us (the update rate is dependant on how wide the Sig Gen Cleverscope capture period is).

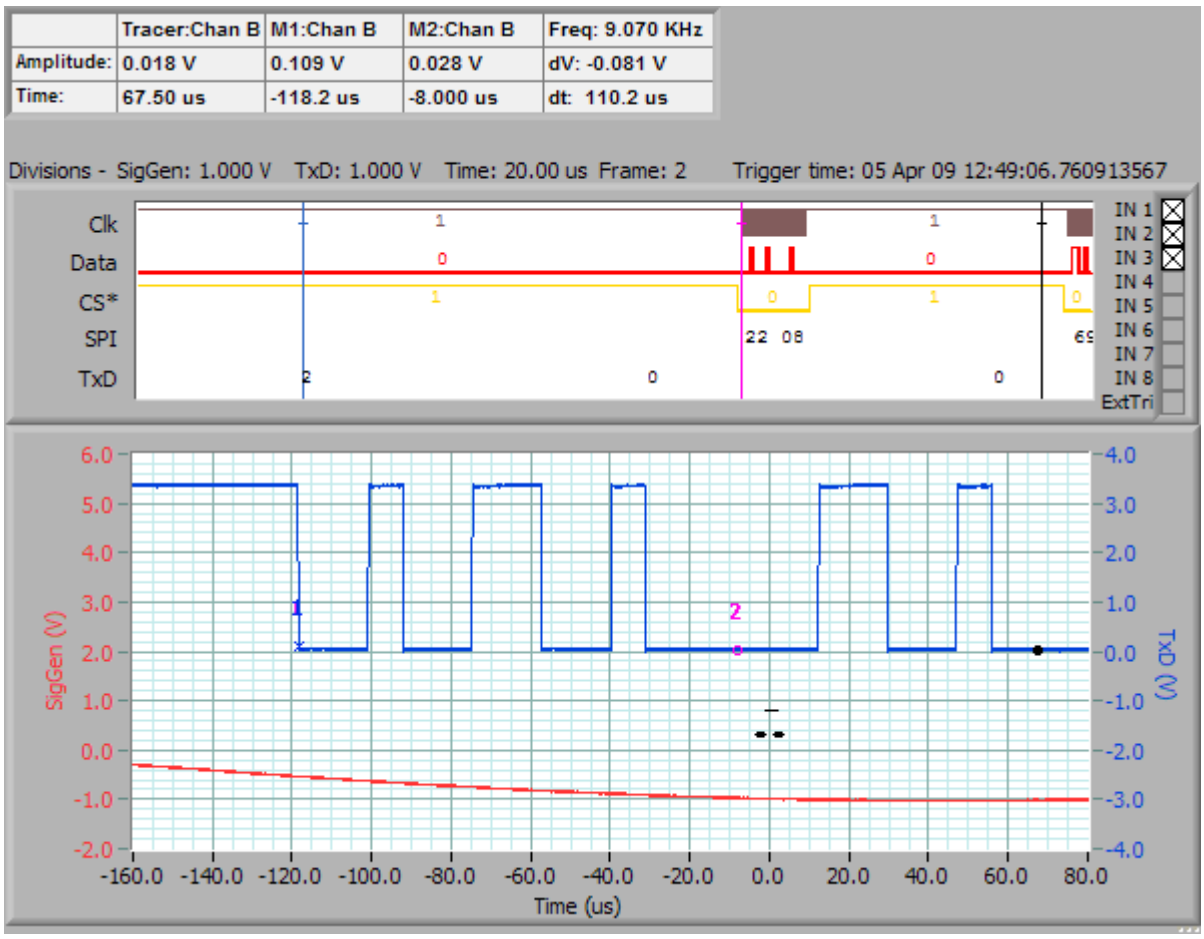
We can also see the time relationship between the procedure outputting the serial data message, and that setting the sig gen. The serial data message started output
-8 - (-118.27) = 110.27 us before the sig gen program started.



We can verify this timing by zooming in a little.
Here we zoom to show the first message to the second message timing. The Marker dT shows the difference between the two messages is as the decoder found it:



Zooming in again we can see the difference between the start of the TxD (M1) message and the SPI (M2) message agrees with the protocol decoder (the small differences are because of on-screen resolution):



We can copy and paste from Notes into Word or Excel, as illustrated here. Tab characters are embedded in the protocol output so that the Excel copy can be quickly used.

Decode of SPI and UART output from signal generator auto-stepping in 1 kHz increments from 1 kHz. There are 10 steps.

Name:	SPI	Frame:	2	Trigger time:	05Apr09 12:49	:	06.760913567	
							-8E-6	22 08
							7.435E-5	69 F0
							0.00015732	40 00
							0.00143359	22 08
							0.0015162	7E E8
							0.00160566	40 00
							0.00273418	22 08
							0.00281697	53 E0
							0.00290021	40 01
							0.00394948	22 08
							0.00403206	68 D8
							0.00411516	40 01
							0.00532418	22 08
							0.00540654	7D D0
							0.00548956	40 01
							0.00664334	22 08
							0.00672581	52 C8
							0.00680881	40 02
							0.00791481	22 08
							0.00799749	67 C0
							0.00808089	40 02
							0.00914342	22 08
							0.00922587	7C B8
							0.00930911	40 02
							0.01036451	22 08
							0.0104471	51 B0
							0.01052958	40 03
							0.01164036	22 08
							0.01172293	54 F8
							0.01180575	40 00
Name:	TxD	Frame:	2	Trigger time:	05Apr09 12:49	:	06.760913567	
							-0.00011827	2000
							0.00132374	3000
							0.0026245	4000
							0.00383952	5000
							0.00521454	6000
							0.00652957	7000
							0.0078051	8000
							0.00903385	9000
							0.01025499	10000
							0.01152678	1000

Triggering

In many circumstances we want to use the triggering system to prove the correct operation of the unit under test. For example, we want to verify that the timing relationship between signals is correct.

For this example, we have specific SPI setup requirements to meet. The example is arbitrary, but applies to many circumstances.

In our example we are interested in the time between the Chip Select (CS*) going low, and the first Clock Pulse becoming active (going low in our example). Secondly we would like to verify the value of the least significant byte of the 16 bit word of the Sig Gen control word. In many situations the device data sheet places limits on these values. In addition, we can search for variability in these values as a sign of other activity happening (for example overloading of the processor by interrupt activity).

First we look for what we want, and verify that the design is actually working. Then we look for what we don't want, and if we get occurrences of it, we now have evidence on which we can base a solution.

Our design is based on a CS* to Clock falling edge time of 1.35 usecs. So to prove correct operation, we will allow ±50 ns on this value. In addition we want to find the 8th falling edge after the trigger – this is the first bit of the least significant byte of the 16 bit sig gen control word. We want this byte to be 08, and we can use the trigger position to stabilize the message on the screen to make it easy to see this particular byte.

Trigger Setups

The trigger setups are as to the right.
We use two independent triggers (Trigger 2 is using its own definition).

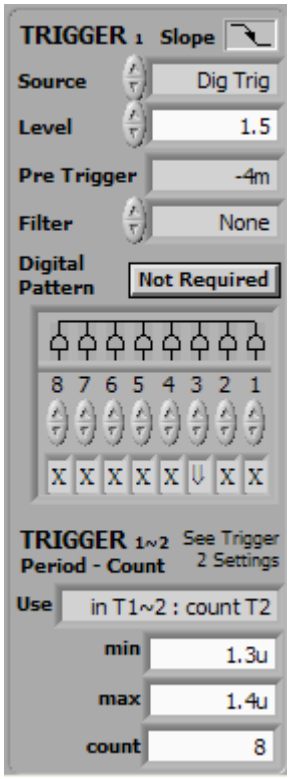
Trigger 1 is triggering on the falling edge of CS* (Dig In 3). We use a Dig Trig source. The Digital Trigger threshold is 1.5V (this applies to all digital inputs).

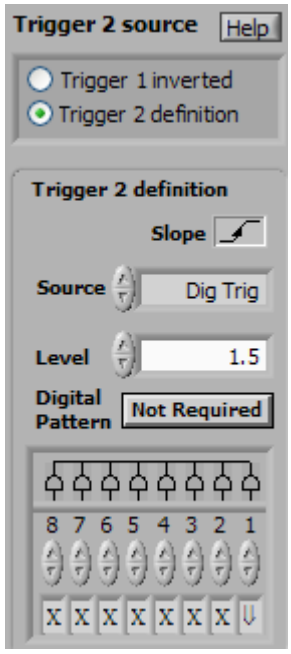
Trigger 2 is triggering on the falling edge of the clock input (Dig In 1).

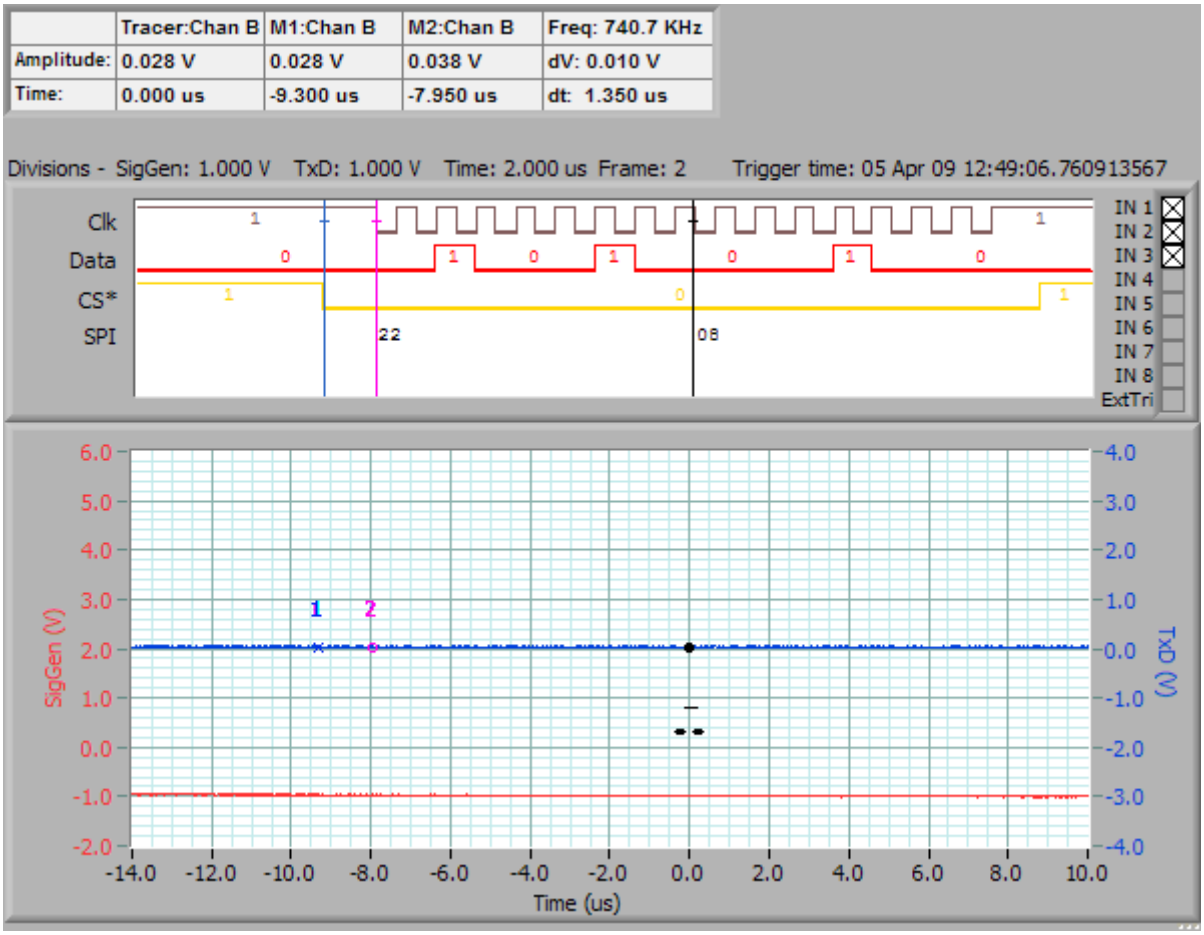
The Trigger 1~2 usage is trigger if the time duration between Trigger 1 and Trigger 2 is between 1.3usecs (min) and 1.4 usecs (max), and then wait for 8 occurrences of Trigger 2 – ie, 8 falling clock edges.

You can see the result below – M1 and M2 show the CS* and clock falling edges, and the time duration. We have our trigger positioned at the start of the LSB.

The analog values are changing slowly, so look flat.

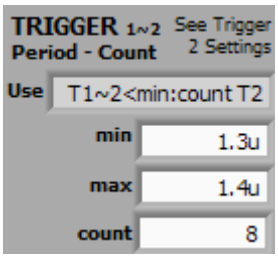






Now we can proceed to verify that the system is not behaving badly.

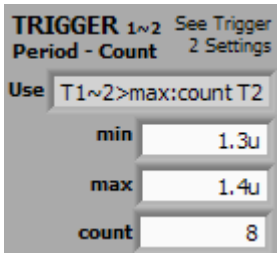
First we set the Trigger 1~2 use to be:



We look for CS* to Clock being less than 1.3 usecs.

We run the application for a long period of time, varying use of the system, and varying operating temperature. If we get no triggers, we know we are not failing the CS* to Clock setup time, over these operating conditions.

Next we check for the period being too large:



We look for CS* to Clock being greater than 1.4usecs.

If we get triggers, we know other system activity maybe impacting onto performance. From that point we can start testing to find what is causing the increase in time.

In addition, when we do get a trigger, we can see what actually happened, this may be of use in solving the problem, or perhaps refining our limits.

Conclusion

We have illustrated capturing a mixed signal waveform, decoding it, and verifying that the timing parameters we have designed for are being met. We have shown that it is easy to document the design process by copy and paste of waveforms, and text information. We have shown that the Cleverscope deep memory buffer is very useful in carrying out these functions. Finally we can see it is much easier to work with recognizable values using the protocol decoder, than manually decoding the signals.