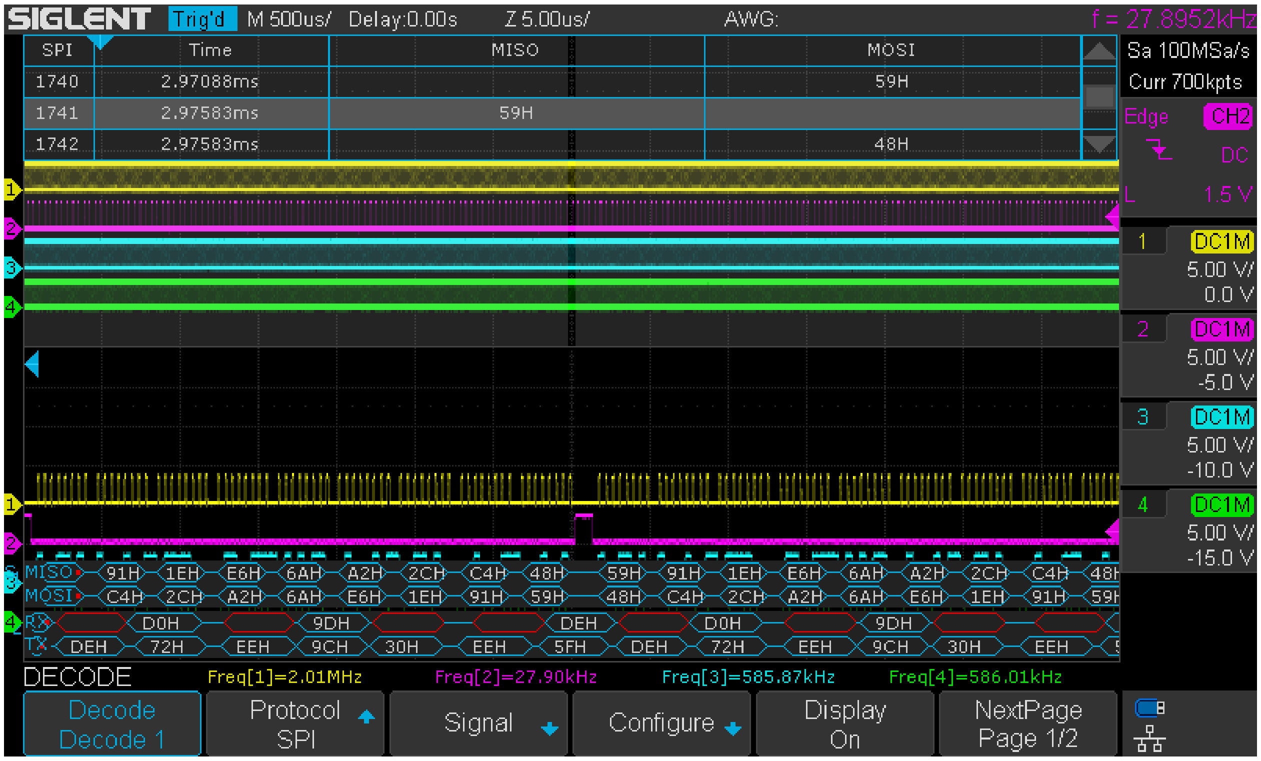
Serial Decode

General

The Siglent SDS1104X-E has the potential to be an excellent tool for low level debugging of serial communication links. It has 4 analog channels, reasonably deep memory and it supports the most popular serial protocols. Most importantly, two independent full duplex decoders are available that can work in parallel. The screenshot below demonstrates such a situation with zoom view centered on a 700kpts record, containing 3484 8bit data words decoded from SPI at 2.35Mbit/s and slightly less for UART at 2Mbaud.



Trig'd M 500us/ Delay:0.00s Z 5.00us/ AWG: f = 27.8952kHz

SPI	Time	MISO	MOSI
1740	2.97088ms		59H
1741	2.97583ms	59H	
1742	2.97583ms		48H

Sa 100MSa/s
Curr 700kpts
Edge CH2 DC
L 1.5 V

1	DC1M	5.00 V/	0.0 V
2	DC1M	5.00 V/	-5.0 V
3	DC1M	5.00 V/	-10.0 V
4	DC1M	5.00 V/	-15.0 V

DECODE Freq[1]=2.01MHz Freq[2]=27.90kHz Freq[3]=585.87kHz Freq[4]=586.01kHz

SIGLENT
MISO 91H 1EH E6H 6AH A2H 2CH C4H 48H 59H 91H 1EH E6H 6AH A2H 2CH C4H 48H
MOSI C4H 2CH A2H 6AH E6H 1EH 91H 59H 48H C4H 2CH A2H 6AH E6H 1EH 91H 59H
RX D0H 9DH DEH D0H 9DH
TX DEH 72H EEH 9CH 30H EEH 5FH DEH 72H EEH 9CH 30H EEH
Decode Decode 1
Protocol SPI
Signal
Configure
Display On
NextPage Page 1/2

SDS1104X-E_SPI_UART_Zoom_2

The serial decoding has been redesigned for the SDS1000X-E series; there are still a few decoder specific issues (which will be mentioned in the related chapters) and I'd like to see some refinement in general – but this list is quite short by now:

- When in zoom view, changing the position of the zoom window scrolls the corresponding excerpt of the decode list into view quite nicely. But the alignment isn't perfect, we would like the data at the center of the decoder bar to correspond with the entry in the middle of the list window.
- Decode list cannot be used for navigation, i.e. scrolling through the list doesn't update the zoom window position.

As mentioned before, the SDS1104X-E has two full duplex decoders, referred to as Decode 1 and Decode 2. Each of these two decoders can be set up for any of the available protocols and both can work in parallel. In an embedded system, we'd be able to monitor the internal SPI and I^{2}C bus at the same time or we could just as well use both decoders on different UART interfaces.

In order to set up a serial decoder, the signals and possibly other parameters have to be configured in the *Signal* and *Configure* submenus. After that there are a few common steps for all decoders.

After signals and protocol options for a decoder are configured, proceed as follows:

- Switch *Display* menu item on.
- In the *List* submenu, the list display can be turned on for one of the two decoders if so desired.
- If the list display has been turned on, set the number of display lines between 1 and 7 (the height of the list display window) and there is a soft menu button for scrolling through the list.
- In the *Format* submenu, select between binary, decimal, hex and ASCII representation of the data. This setting applies to the list as well as the decoder bar at the bottom of the screen.
- The *Copy Setting* menu item sets the trigger type to serial and copies the decoder settings to the trigger, which is quite convenient.

Be aware that there is a decoder-specific limit for the number of list entries which is also the maximum number of decoded data words in general. Since serial decoders usually don't require excessive oversampling, it looks like the SDS1104X-E would have enough acquisition memory to decode more than that. Even though this might actually be true, the limits are still sensible and high enough to not pose serious limitations to the kind of work a DSO/MSO is supposed to do. For most applications it might be a good idea to limit the record length to 700kpts in the *Acquire* menu, which speeds up operation and also gives us the opportunity to keep up to 35 previous records in the history.

The following chapters deal with the individual decoders. Please bear in mind that these are just examples and it is next to impossible to cover all possible scenarios. I have to admit that I personally rarely ever use serial decoders, as they aren't really necessary for a signal integrity check, which would be the main purpose of a DSO. As an MSO, it is also useful to trigger on a certain serial telegram and check the reaction of the DUT, which would usually be digital and/or analog signals changes, as well as the corresponding time delays. For high level protocol analysis though, I prefer dedicated tools that let me work on a large PC screen comfortably. But then I know there are also some folks who seem to look at serial decoding as the single most important feature of a DSO…

Anyway, I'm trying to provide a brief demonstration of the basic capabilities for the individual decoders.

SPI

With 4 analog channels, we can have one full duplex SPI decoding with SCK, MISO, MOSI and CS. Triggering on the falling edge of the active low CS signal is usually all that’s needed to get the messages properly displayed and decoded.

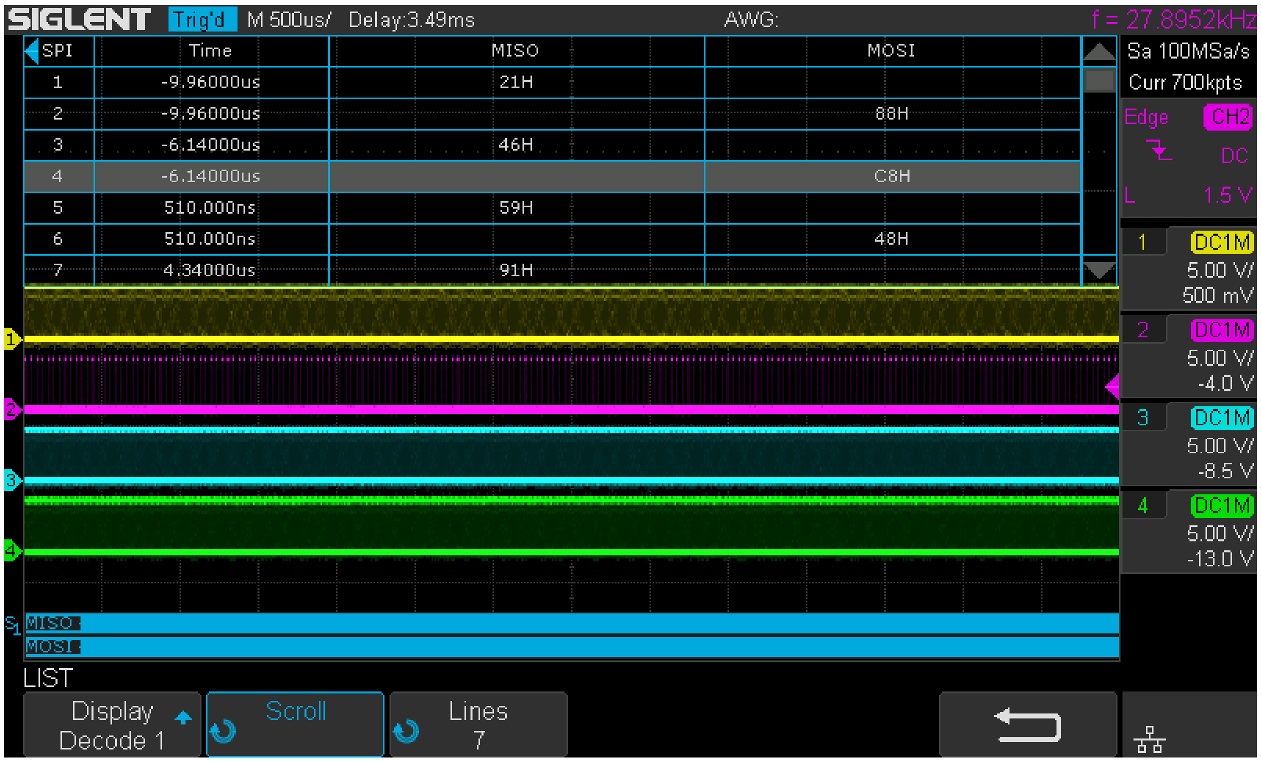
SPI Decoder

To set up an SPI decoder, proceed as follows:

- Go to the *Signal* menu and configure the channels and signal thresholds for CLK, MISO, MOSI and CS.
- For CLK, select rising or falling edge accordingly.
- For CS, we have the choice of positive/negative logic or clock timeout, i.e. no chip select signal at all.
- Go to the *Configure* menu and set the *Bit Order* (MSB/LSB first) and the *Data Length* (4-32 bits).

There is a limit of 6000 decoded data words regardless of the number of bits, so that limit applies to 32bit data words as well. As a consequence, we can capture up to 24k bytes or 192k bits with a single record. We don’t need excessive oversampling, a sample rate of ten times the bit rate has shown to be plenty enough for a symmetrical clock signal.

The following screenshot shows a full duplex SPI transfer for 8bit data words at a bit rate of 2.35Mbit/s. Time base is 500µs/div, so we’re looking at a time window of 7ms at a sample rate of 100MSa/s, resulting in a record length of 700kpts. Of course the signal traces as well as the decoder bar at the bottom of the screen are way too compressed to show anything meaningful at that time scale, but the list view contains all the decoded data.



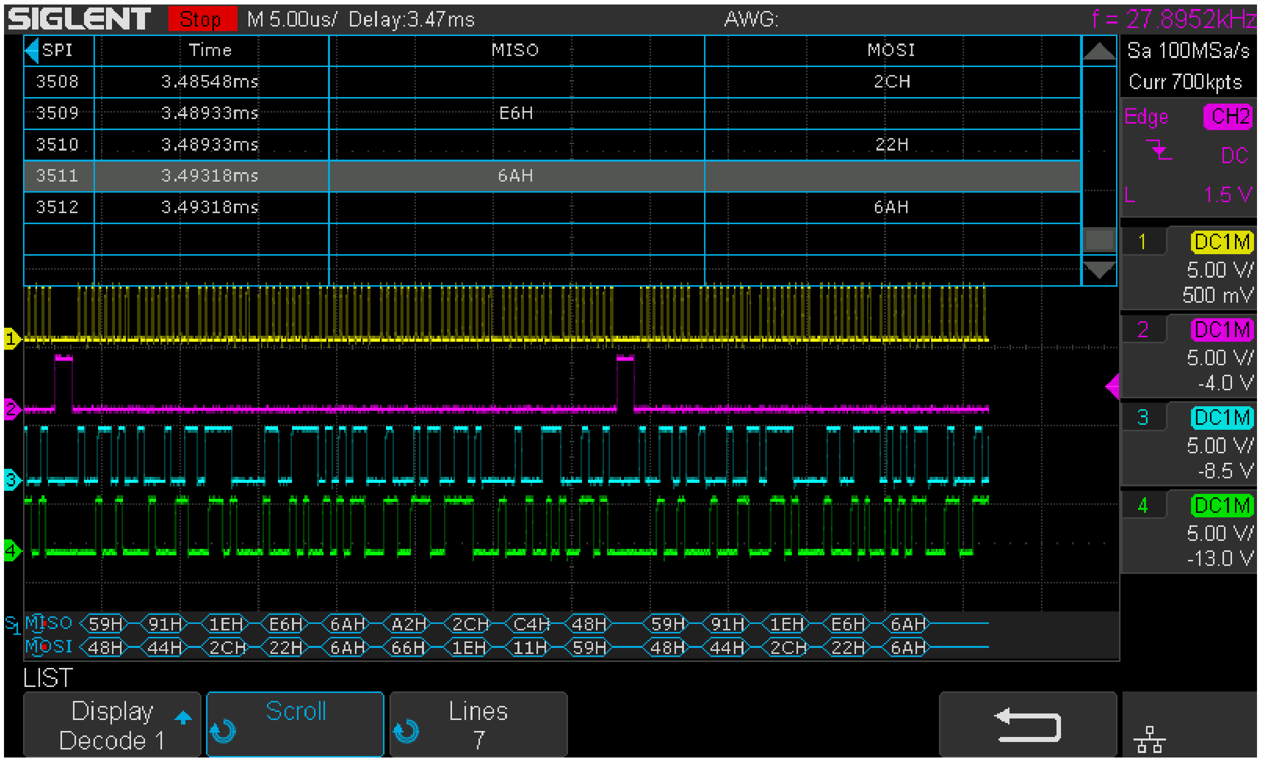
M 500us/ Delay:3.49ms f = 27.8952kHz

SPI	Time	MISO	MOSI
1	-9.96000us	21H	
2	-9.96000us		88H
3	-6.14000us	46H	
4	-6.14000us		C8H
5	510.000ns	59H	
6	510.000ns		48H
7	4.34000us	91H	

Sa 100MSa/s
Curr 700kpts
Edge CH2 DC
L 1.5 V
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V
MISO
MOSI

SDS1104X-E_Serial_SPI_8bit

We can stop the acquisition, lower the time base in order to zoom into the waveform until the data in the decoder bar become readable and of course we can also scroll through the record as we like. The screenshot below shows this situation where the view has been positioned to the end of the record.

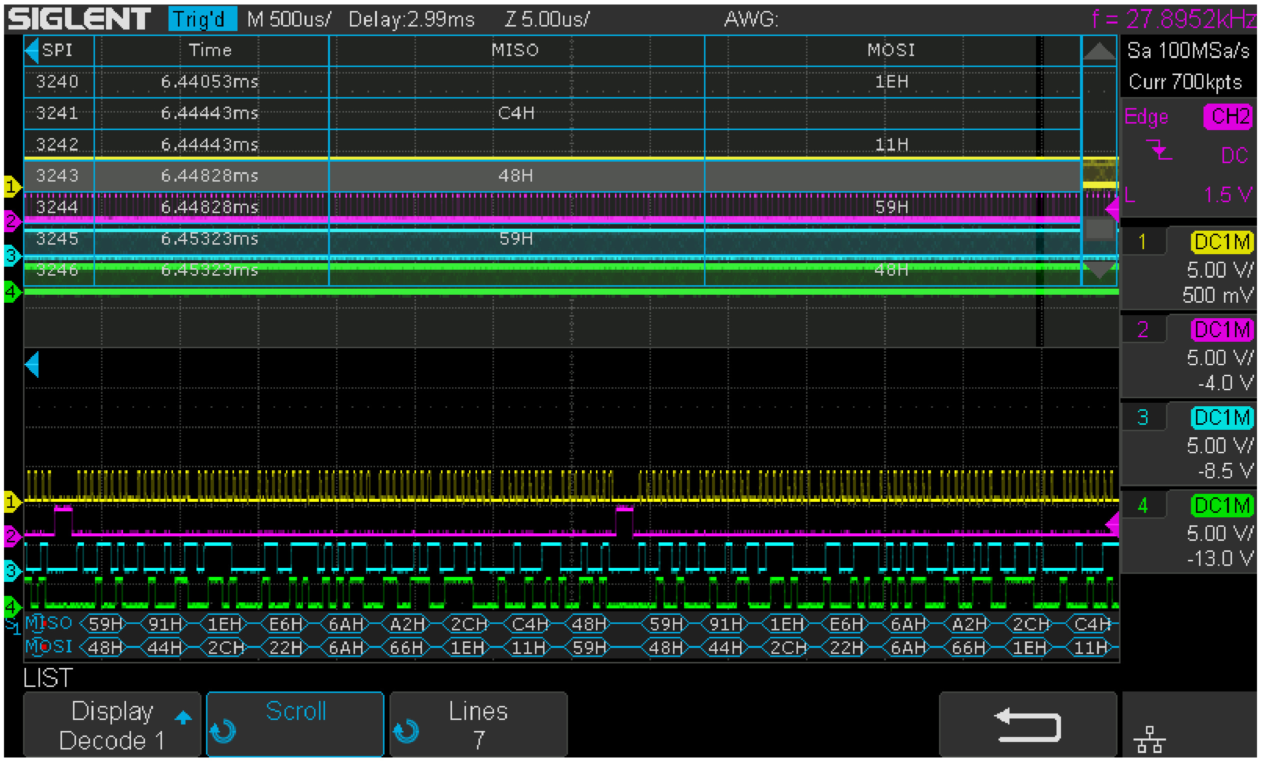


SPI	Time	MISO	MOSI
3508	3.48548ms		2CH
3509	3.48933ms	E6H	
3510	3.48933ms		22H
3511	3.49318ms	6AH	
3512	3.49318ms		6AH

SDS1104X-E_Serial_SPI_8bit_Stop5us_last

The list view in the upper part of the screen is automatically centered on a data word near the center of the decoding bar and if necessary we can scroll the list a few positions down in order to see the very last data entry, which is the index 3512 for the MOSI line in this example.

We can also inspect the signals and decoded data while acquisition is still running if we use zoom mode. Of course we can zoom in and out anytime and also scroll through the record as we like. The following screenshots demonstrate this situation for a zoomed time base of 5µs and 500ns respectively, with the zoom view approximately centered on index 3245 of the decoder list.



SIGLENT Trig'd M 500us/ Delay:2.99ms Z 5.00us/ AWG: f = 27.8952kHz

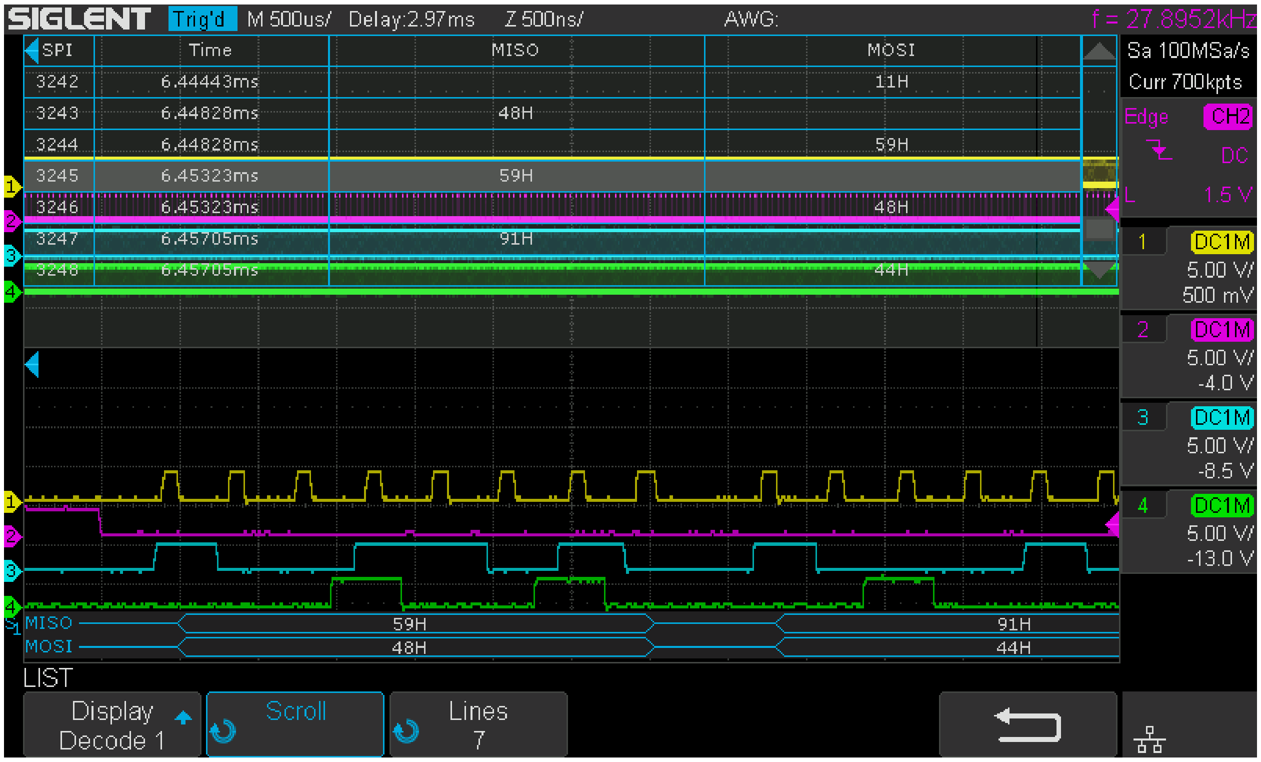
Sa 100MSa/s Curr 700kpts Edge CH2 DC L 1.5 V

1 DC1M 5.00 V/ 500 mV; 2 DC1M 5.00 V/ -4.0 V; 3 DC1M 5.00 V/ -8.5 V; 4 DC1M 5.00 V/ -13.0 V

SPI	Time	MISO	MOSI
3240	6.44053ms		1EH
3241	6.44443ms	C4H	
3242	6.44443ms		11H
3243	6.44828ms	48H	
3244	6.44828ms		59H
3245	6.45323ms	59H	
3246	6.45323ms		48H

MISO 59H 91H 1EH E6H 6AH A2H 2CH C4H 48H 59H 91H 1EH E6H 6AH A2H 2CH C4H
MOSI 48H 44H 2CH 22H 6AH 66H 1EH 11H 59H 48H 44H 2CH 22H 6AH 66H 1EH 11H
LIST
Display Decode 1
Scroll
Lines 7

SDS1104X-E_Serial_SPI_8bit_zoom5us



SIGLENT Trig'd M 500us/ Delay:2.97ms Z 500ns/ AWG: f = 27.8952kHz

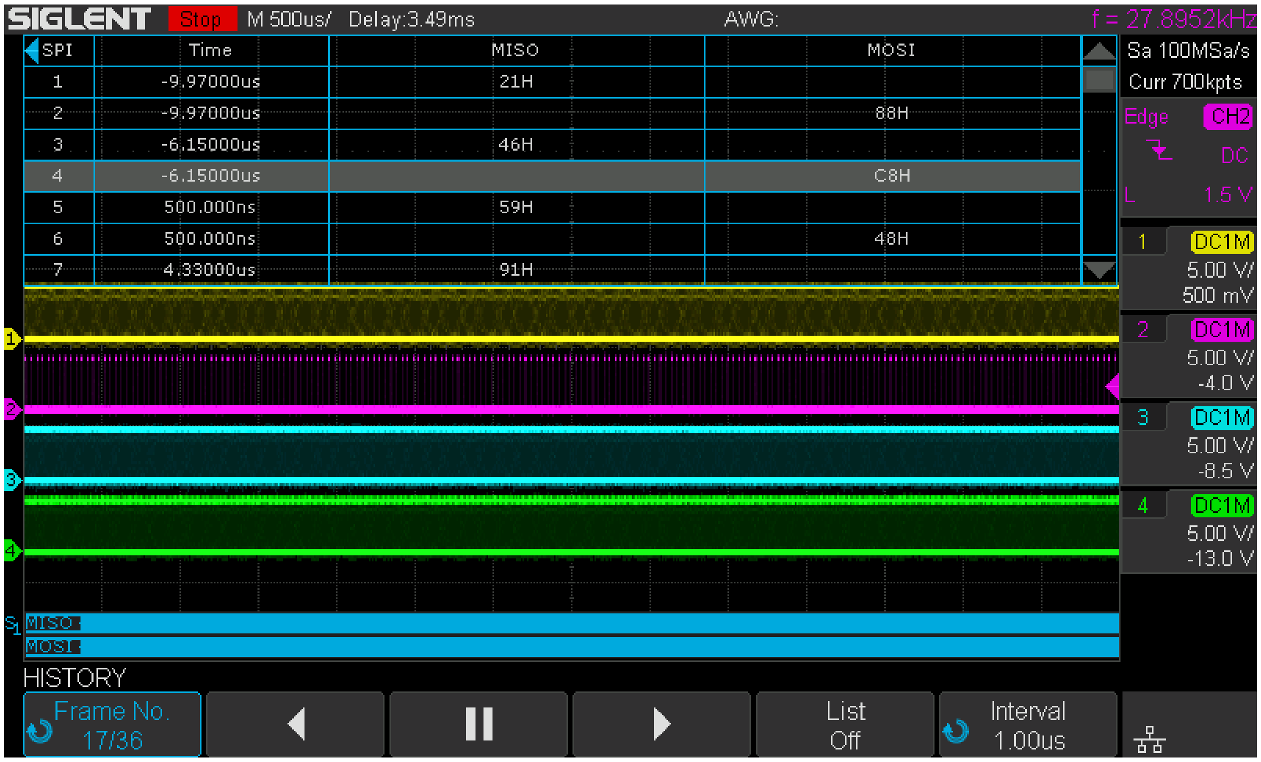
Sa 100MSa/s Curr 700kpts Edge CH2 DC L 1.5 V

1 DC1M 5.00 V/ 500 mV; 2 DC1M 5.00 V/ -4.0 V; 3 DC1M 5.00 V/ -8.5 V; 4 DC1M 5.00 V/ -13.0 V

SPI	Time	MISO	MOSI
3242	6.44443ms		11H
3243	6.44828ms	48H	
3244	6.44828ms		59H
3245	6.45323ms	59H	
3246	6.45323ms		48H
3247	6.45705ms	91H	
3248	6.45705ms		44H

SDS1104X-E_Serial_SPI_8bit_zoom500ns

We can also stop the acquisition and enter history in order to analyze the current or any previous record stored in the history, e.g. number 17 out of 36 as shown below.

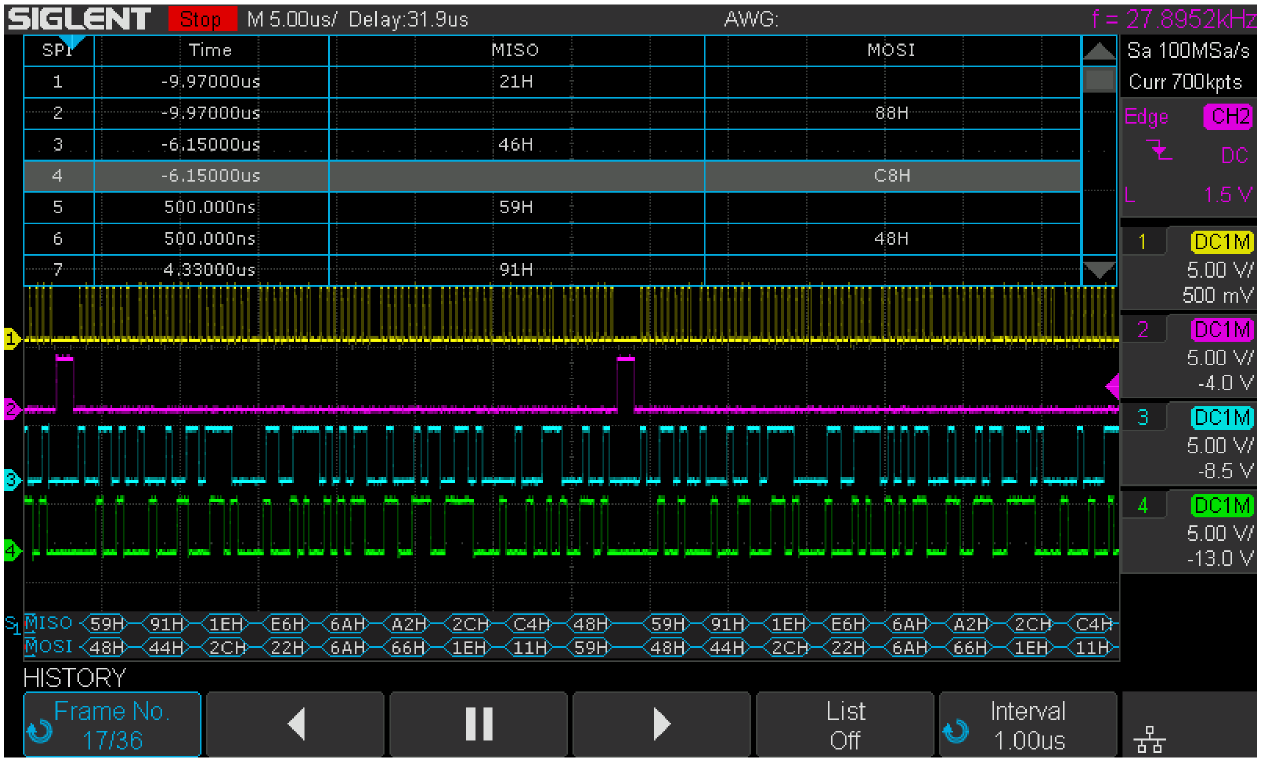

SIGLENT Stop M 500us/ Delay:3.49ms AWG: f = 27.8952kHz

SPI	Time	MISO	MOSI
1	-9.97000us	21H	
2	-9.97000us		88H
3	-6.15000us	46H	
4	-6.15000us		C8H
5	500.000ns	59H	
6	500.000ns		48H
7	4.33000us	91H	

Sa 100MSa/s
Curr 700kpts
Edge CH2 DC L 1.5 V
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V

SDS1104X-E_Serial_SPI_8bit_History.png

Once again we can lower the time base and scroll through the record for closer inspection of certain parts of the message, like 5µs/div in the example below.


SIGLENT Stop M 5.00us/ Delay:31.9us AWG: f = 27.8952kHz

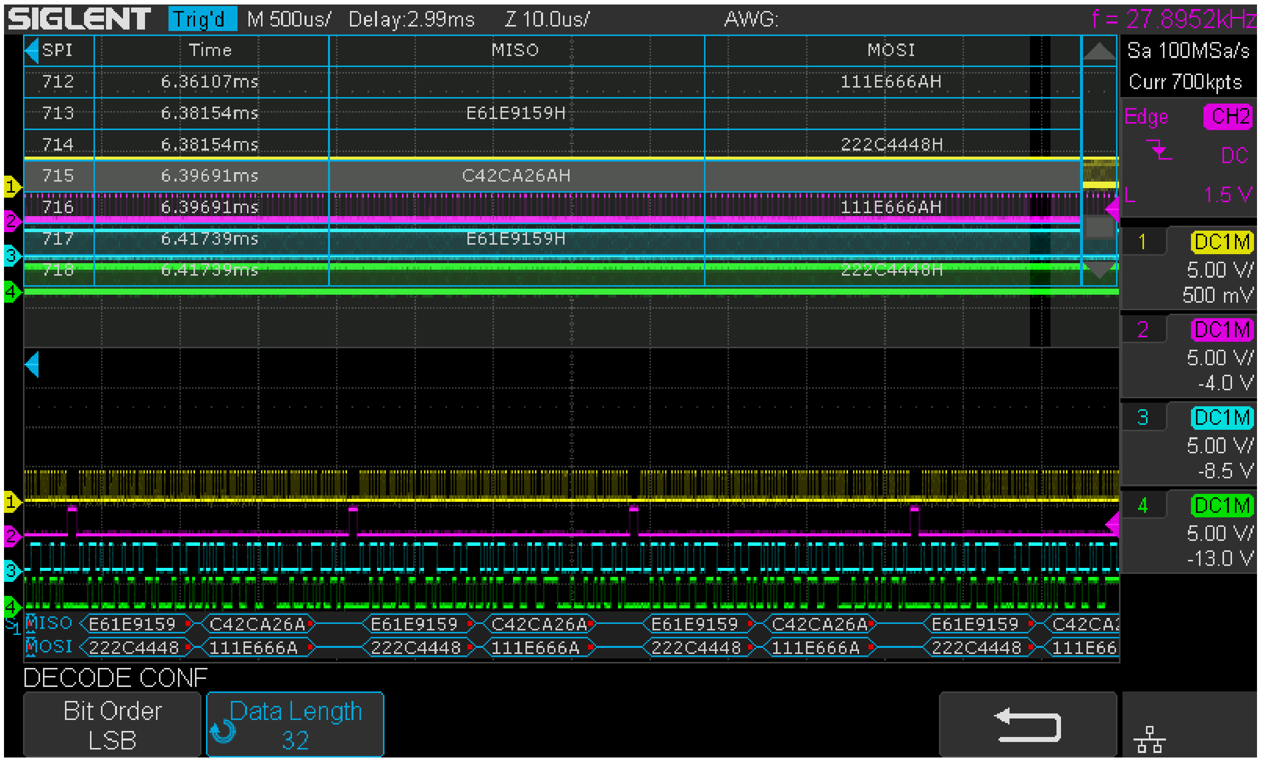
SPI	Time	MISO	MOSI
1	-9.97000us	21H	
2	-9.97000us		88H
3	-6.15000us	46H	
4	-6.15000us		C8H
5	500.000ns	59H	
6	500.000ns		48H
7	4.33000us	91H	

Sa 100MSa/s
Curr 700kpts
Edge CH2 DC L 1.5 V
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V

MISO 59H 91H 1EH E6H 6AH A2H 2CH C4H 48H 59H 91H 1EH E6H 6AH A2H 2CH C4H
MOSI 48H 44H 2CH 22H 6AH 66H 1EH 11H 59H 48H 44H 2CH 22H 6AH 66H 1EH 11H

SDS1104X-E_Serial_SPI_8bit_History_5us

Finally there is an example for 32 bit data words. Same time base and record length as before, zoomed in at 10µs/div and centered on index 713 of the list during acquisition.



SIGLENT Trig'd M 500us/ Delay:2.99ms Z 10.0us/ AWG: f = 27.8952kHz

SPI	Time	MISO	MOSI
712	6.36107ms		111E666AH
713	6.38154ms	E61E9159H	
714	6.38154ms		222C4448H
715	6.39691ms	C42CA26AH	
716	6.39691ms		111E666AH
717	6.41739ms	E61E9159H	
718	6.41739ms		222C4448H

Sa 100MSa/s
Curr 700kpts
Edge CH2
DC
L 1.5 V
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V

MISO E61E9159 C42CA26A E61E9159 C42CA26A E61E9159 C42CA26A E61E9159 C42CA2
MOSI 222C4448 111E666A 222C4448 111E666A 222C4448 111E666A 222C4448 111E66

DECODE CONF
Bit Order LSB
Data Length 32

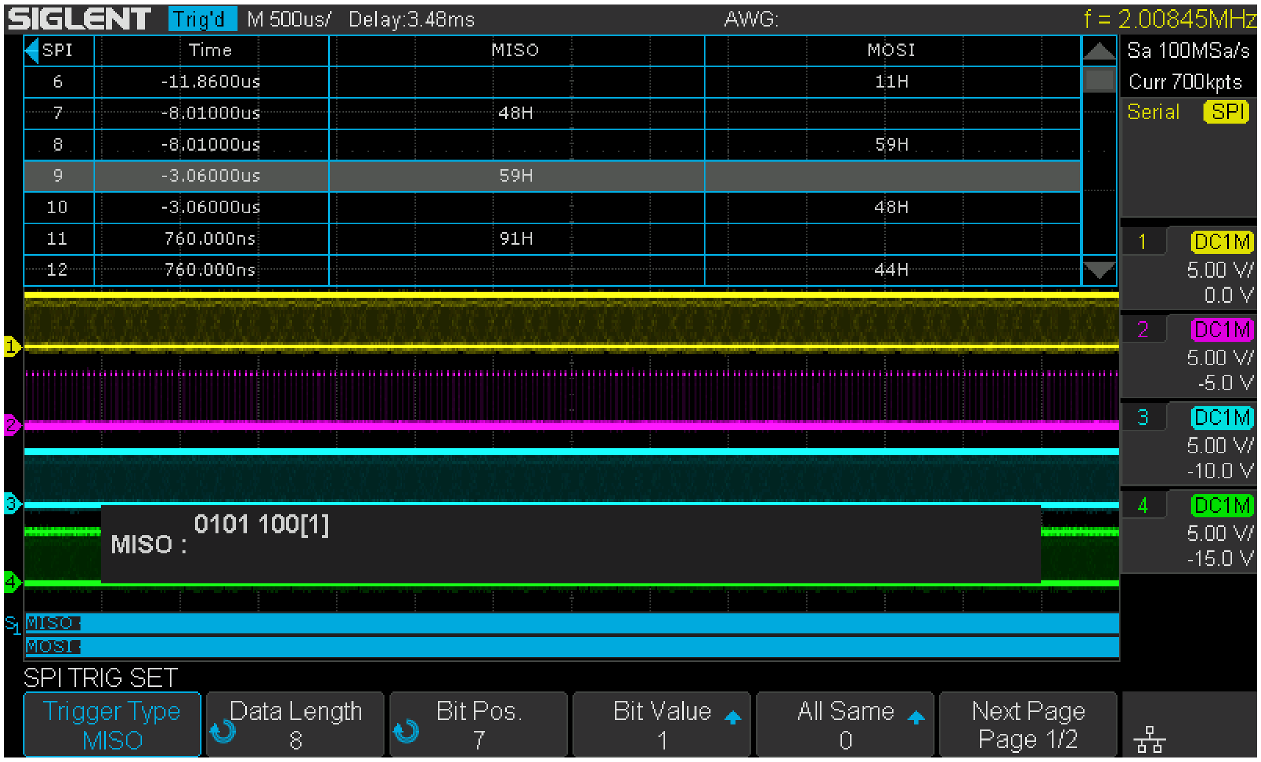
SDS1104X-E_Serial_SPI_32bit_zoom10us

SPI Trigger

The following screenshots show examples for some SPI triggers:

1. 8bit Trigger on 8bit MISO data 0101 1001b = 59h
2. 8bit Trigger on 8bit MOSI data 0100 1000b = 48h
3. 32bit Trigger on 32bit MOSI data 0010 0010 0010 1100 0100 0100 0100 1000b = 222C4448h
4. 40bit Trigger on 32bit MOSI data 0110 1010 0010 0010 0010 1100 0100 0100 0100 1000b = 6A222C4448h

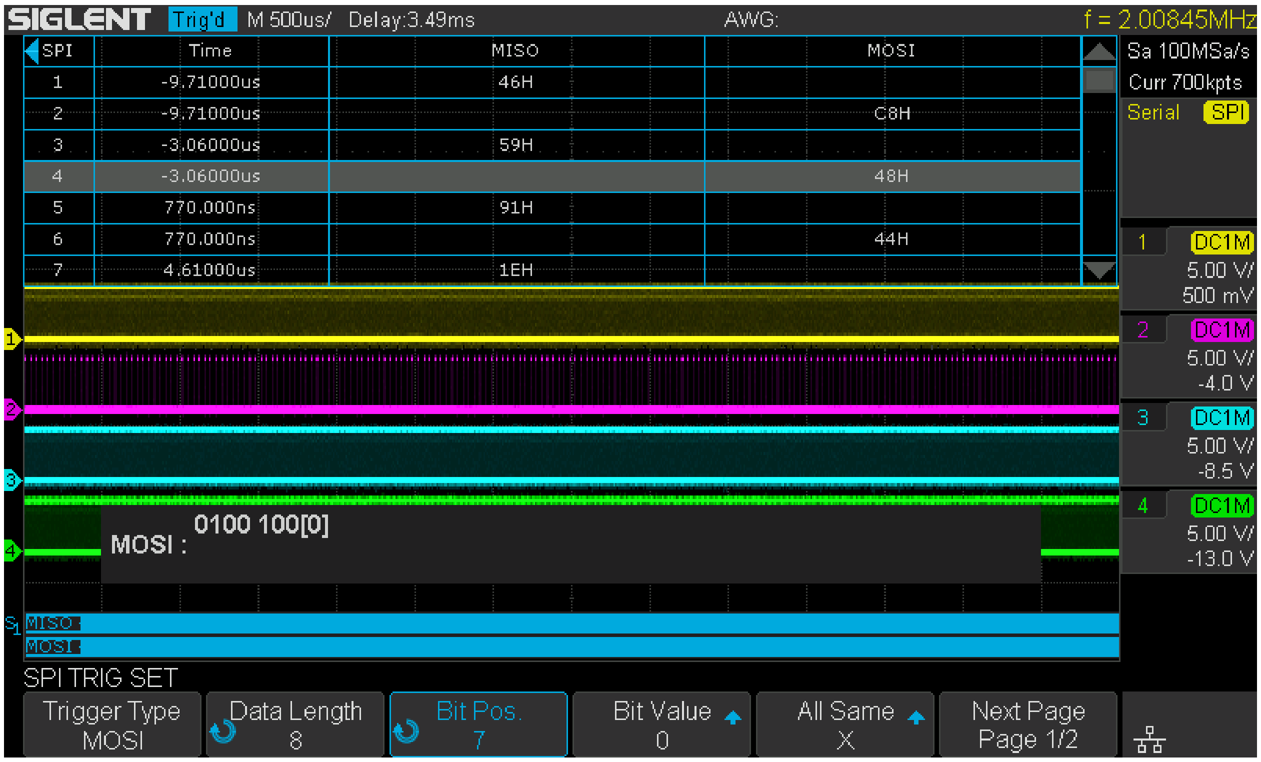
The 4th example demonstrates triggering on a 40bit sequence even though a data word cannot exceed 32 bits. The trigger pattern can be up to 96 bits and is totally independent of the data word size.



SPI	Time	MISO	MOSI
6	-11.8600us		11H
7	-8.01000us	48H	
8	-8.01000us		59H
9	-3.06000us	59H	
10	-3.06000us		48H
11	760.000ns	91H	
12	760.000ns		44H

MISO : 0101 100[1]

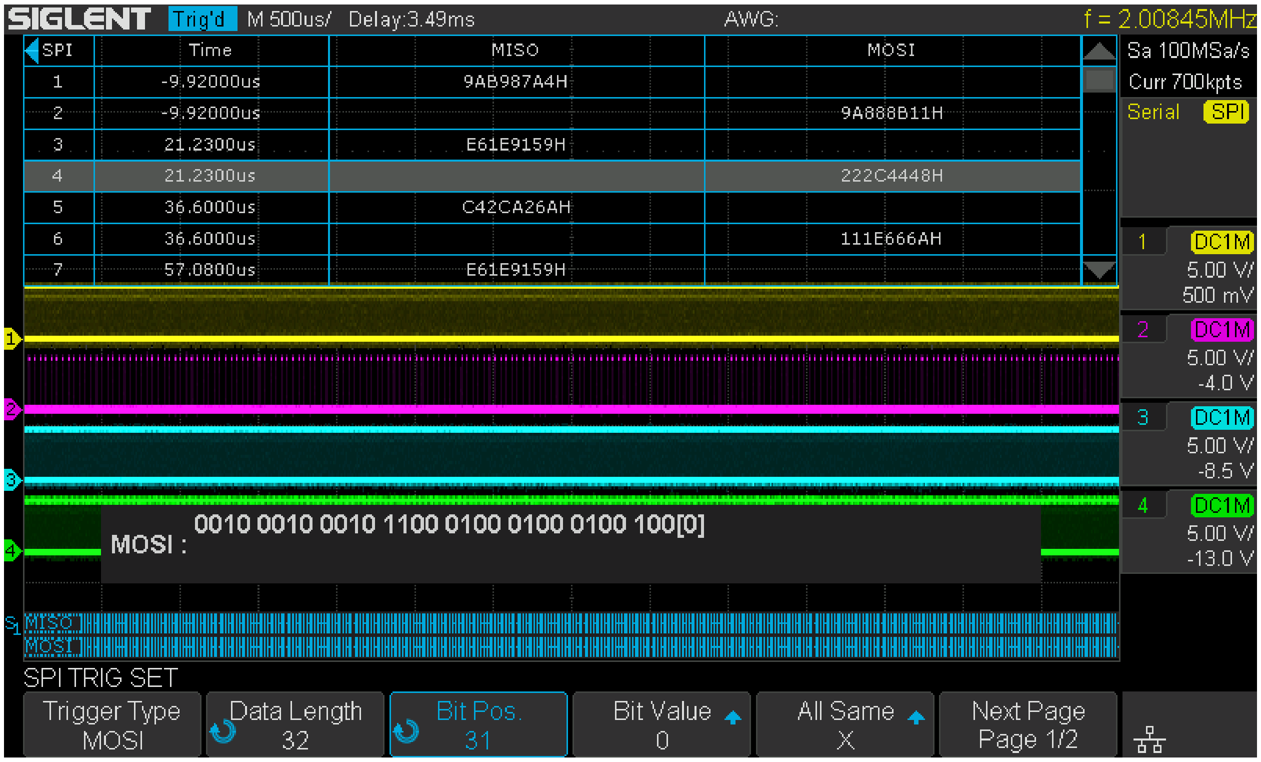
SDS1104X-E_SPI_Trigger_MISO_8bit



SPI	Time	MISO	MOSI
1	-9.71000us	46H	
2	-9.71000us		C8H
3	-3.06000us	59H	
4	-3.06000us		48H
5	770.000ns	91H	
6	770.000ns		44H
7	4.61000us	1EH	

MOSI : 0100 100[0]

SDS1104X-E_SPI_Trigger_MOSI_8bit


SIGLENT Trig'd M 500us/ Delay:3.49ms AWG: f = 2.00845MHz

SPI	Time	MISO	MOSI
1	-9.92000us	9AB987A4H	
2	-9.92000us		9A888B11H
3	21.2300us	E61E9159H	
4	21.2300us		222C4448H
5	36.6000us	C42CA26AH	
6	36.6000us		111E666AH
7	57.0800us	E61E9159H	

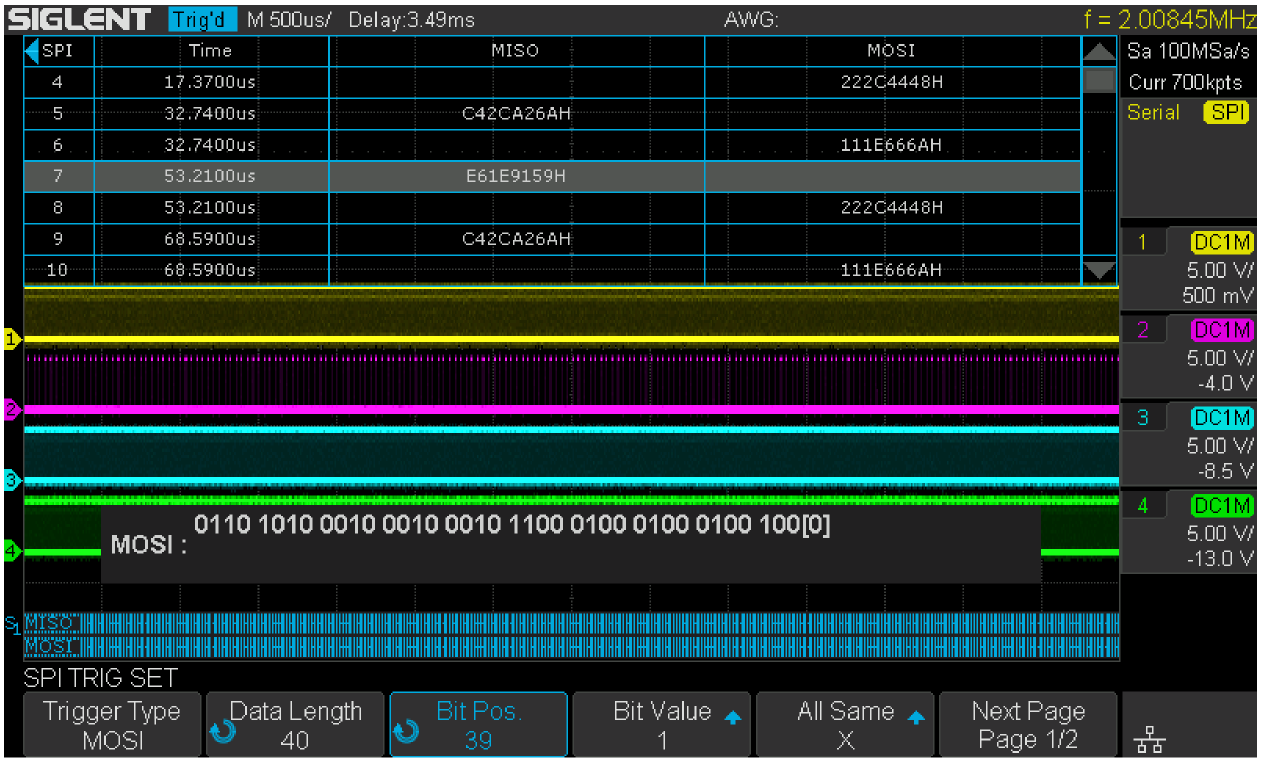
Sa 100MSa/s
Curr 700kpts
Serial SPI

1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V

MOSI : 0010 0010 0010 1100 0100 0100 0100 100[0]

SPI TRIG SET
Trigger Type MOSI | Data Length 32 | Bit Pos. 31 | Bit Value 0 | All Same X | Next Page Page 1/2

SDS1104X-E_SPI_Trigger_MOSI_32bit


SIGLENT Trig'd M 500us/ Delay:3.49ms AWG: f = 2.00845MHz

SPI	Time	MISO	MOSI
4	17.3700us		222C4448H
5	32.7400us	C42CA26AH	
6	32.7400us		111E666AH
7	53.2100us	E61E9159H	
8	53.2100us		222C4448H
9	68.5900us	C42CA26AH	
10	68.5900us		111E666AH

Sa 100MSa/s
Curr 700kpts
Serial SPI

1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V
3 DC1M 5.00 V/ -8.5 V
4 DC1M 5.00 V/ -13.0 V

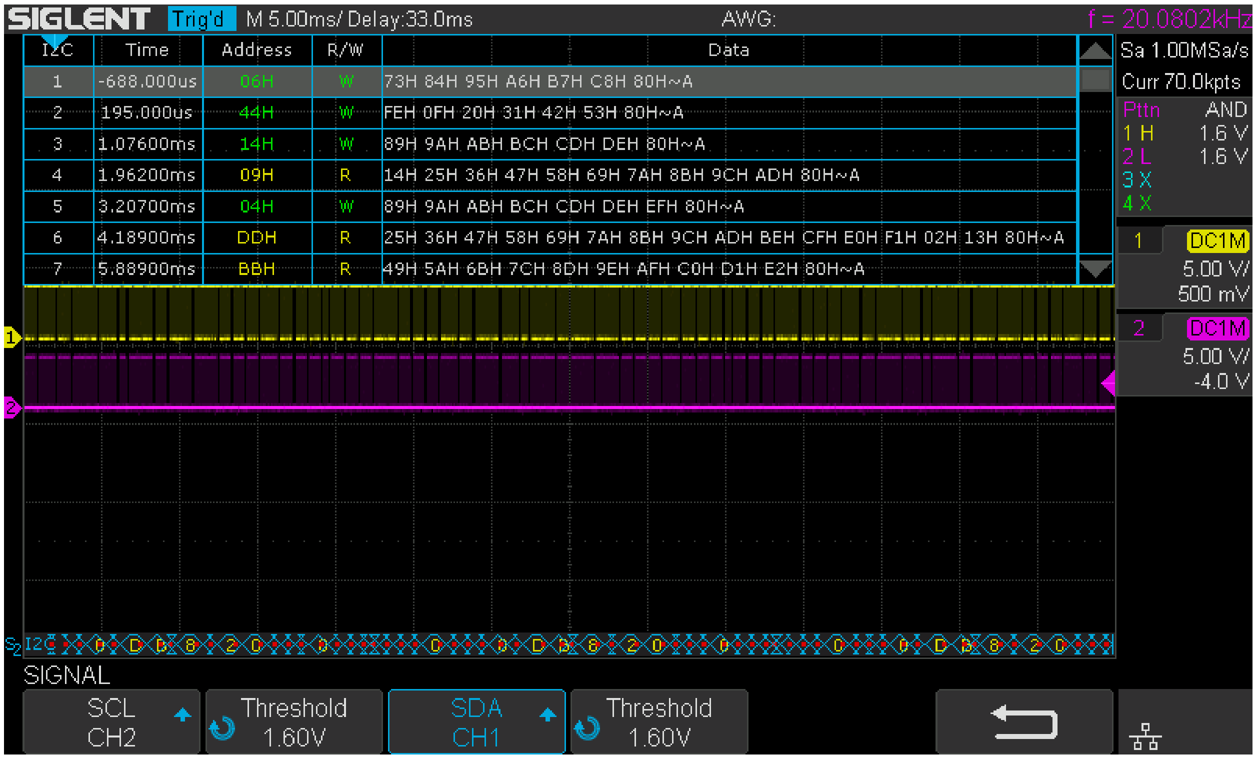
MOSI : 0110 1010 0010 0010 0010 1100 0100 0100 0100 100[0]

SPI TRIG SET
Trigger Type MOSI | Data Length 40 | Bit Pos. 39 | Bit Value 1 | All Same X | Next Page Page 1/2

SDS1104X-E_SPI_Trigger_MOSI_40bit

I^{2}C

By using 2 channels, we can have half duplex I^{2}C decoding with SCL and SDA. Triggering behind a stop condition can be accomplished by setting up a pattern trigger for the break before the following start condition, when SCL is low and SDA is high and this condition is stable for a certain time that should be slightly longer than one period of the SCL clock.



I2C	Time	Address	R/W	Data
1	-688.000us	06H	W	73H 84H 95H A6H B7H C8H 80H~A
2	195.000us	44H	W	FEH 0FH 20H 31H 42H 53H 80H~A
3	1.07600ms	14H	W	89H 9AH ABH BCH CDH DEH 80H~A
4	1.96200ms	09H	R	14H 25H 36H 47H 58H 69H 7AH 8BH 9CH ADH 80H~A
5	3.20700ms	04H	W	89H 9AH ABH BCH CDH DEH EFH 80H~A
6	4.18900ms	DDH	R	25H 36H 47H 58H 69H 7AH 8BH 9CH ADH BEH CFH E0H F1H 02H 13H 80H~A
7	5.88900ms	BBH	R	49H 5AH 6BH 7CH 8DH 9EH AFH C0H D1H E2H 80H~A

SDS1104X-E_Serial_I2C_Setup

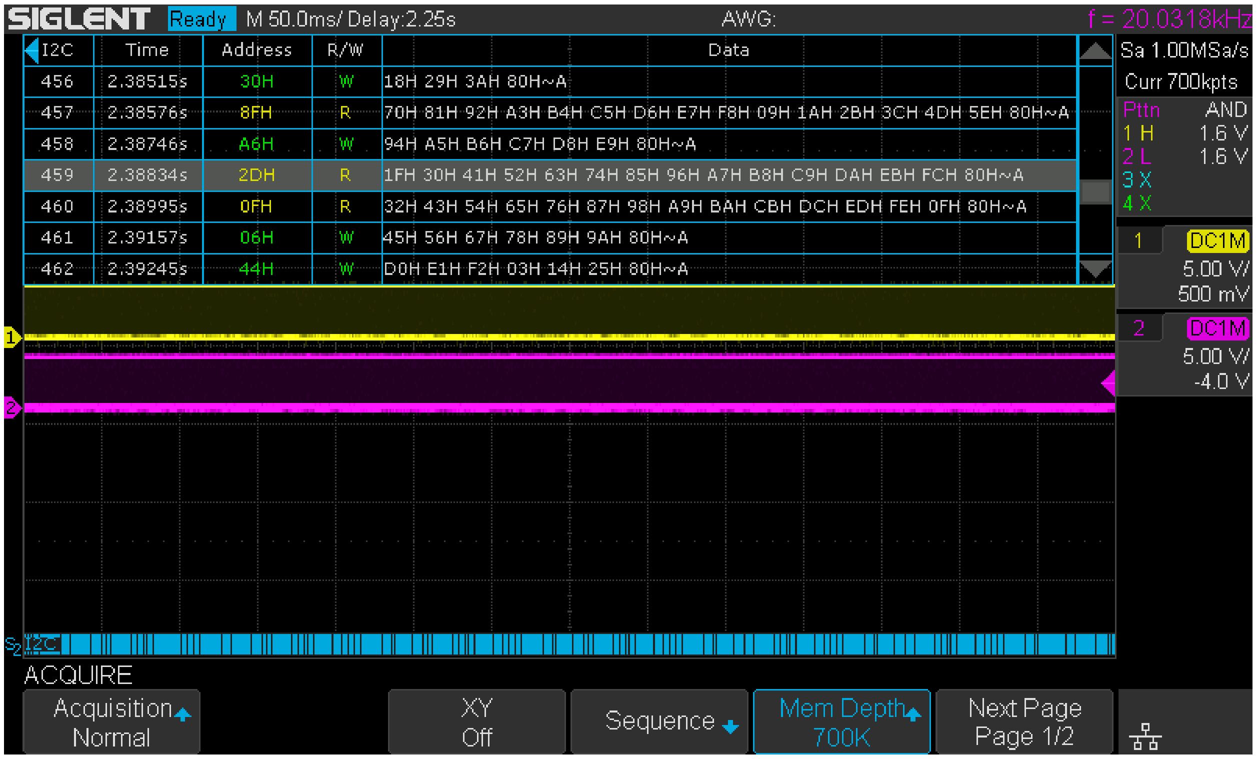
I^{2}C Decoder

To set up an I^{2}C decoder, proceed as follows:

- Go to the *Signal* menu and configure the channels and signal thresholds for SCL and SDA.
- In the *Configure* menu there is an *Include R/W bit* option that can be turned on or off. It is used for in/excluding the r/w bit into the address byte as displayed in the decoder bar and the list view.

There is a limit of 1000 list entries, i.e. decoded messages, where the list view can display the address together with up to 16 data bytes. This means we can decode and display up to 17k bytes or 136kbits in the list view. We don't need excessive oversampling, a sample rate of ten times the bit rate is usually more than sufficient.

The following screenshot shows a half duplex I^{2}C transfer for 8bit data words at a bit rate of ~100kbit/s. Time base is 50ms/div, so we're looking at a time window of 700ms at a sample rate of 1MSa/s, resulting in a record length of 700kpts. Of course the signal traces as well as the decoder bar at the bottom of the screen are way too compressed to show anything meaningful at that time scale, but the list view contains all the decoded messages.

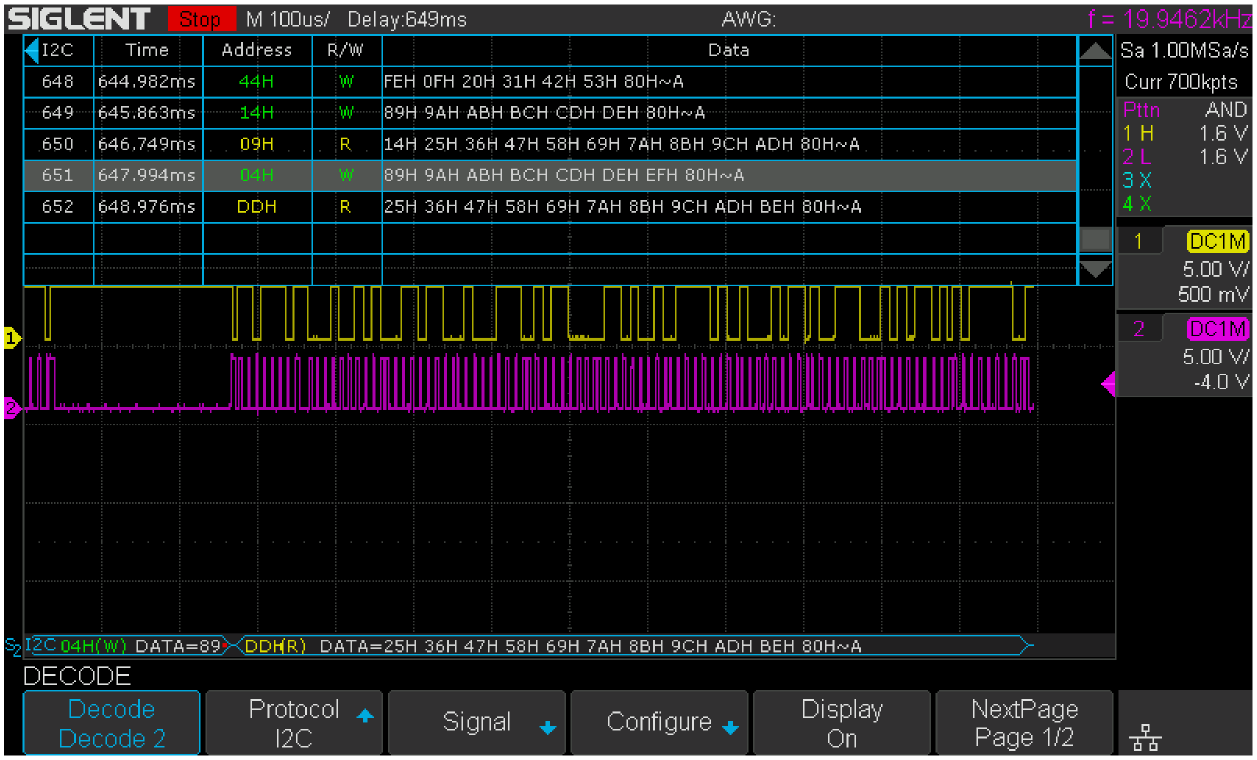


M 50.0ms/ Delay:2.25s f = 20.0318kHz

I2C	Time	Address	R/W	Data
456	2.38515s	30H	W	18H 29H 3AH 80H~A
457	2.38576s	8FH	R	70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 2BH 3CH 4DH 5EH 80H~A
458	2.38746s	A6H	W	94H A5H B6H C7H D8H E9H 80H~A
459	2.38834s	2DH	R	1FH 30H 41H 52H 63H 74H 85H 96H A7H B8H C9H DAH EBH FCH 80H~A
460	2.38995s	0FH	R	32H 43H 54H 65H 76H 87H 98H A9H BAH CBH DCH EDH FEH 0FH 80H~A
461	2.39157s	06H	W	45H 56H 67H 78H 89H 9AH 80H~A
462	2.39245s	44H	W	D0H E1H F2H 03H 14H 25H 80H~A

SDS1104X-E_Serial_I2C

We can stop the acquisition, lower the time base in order to zoom into the waveform until the data in the decoder bar become readable and of course we can also scroll through the record as we like. The screenshot below shows this situation where the view has been positioned to the end of the record.



M 100us/ Delay:649ms f = 19.9462kHz

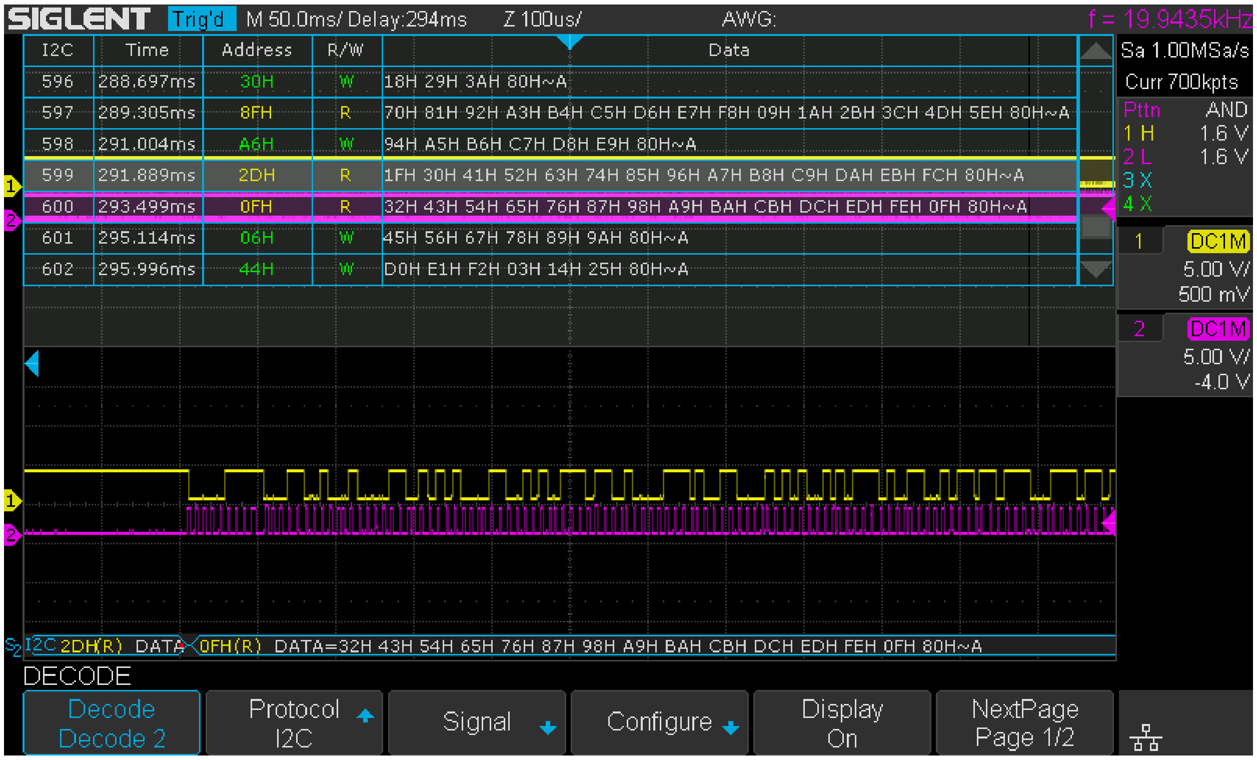
I2C	Time	Address	R/W	Data
648	644.982ms	44H	W	FEH 0FH 20H 31H 42H 53H 80H~A
649	645.863ms	14H	W	89H 9AH ABH BCH CDH DEH 80H~A
650	646.749ms	09H	R	14H 25H 36H 47H 58H 69H 7AH 8BH 9CH ADH 80H~A
651	647.994ms	04H	W	89H 9AH ABH BCH CDH DEH EFH 80H~A
652	648.976ms	DDH	R	25H 36H 47H 58H 69H 7AH 8BH 9CH ADH BEH 80H~A

I2C 04H(W) DATA=89 DDH(R) DATA=25H 36H 47H 58H 69H 7AH 8BH 9CH ADH BEH 80H~A

SDS1104X-E_Serial_I2C_Stop100us_last

The list view in the upper part of the screen is automatically adjusted to show the message(s) visible in the decoding bar and we can see the very last message with the index 652.

We can also inspect the signals and decoded data while acquisition is still running if we use zoom mode. Of course we can zoom in and out and scroll through the record as we like. The following screenshot demonstrates this situation for a zoomed time base of 100µs, with the zoom view centered on index 599 of the decoder list.



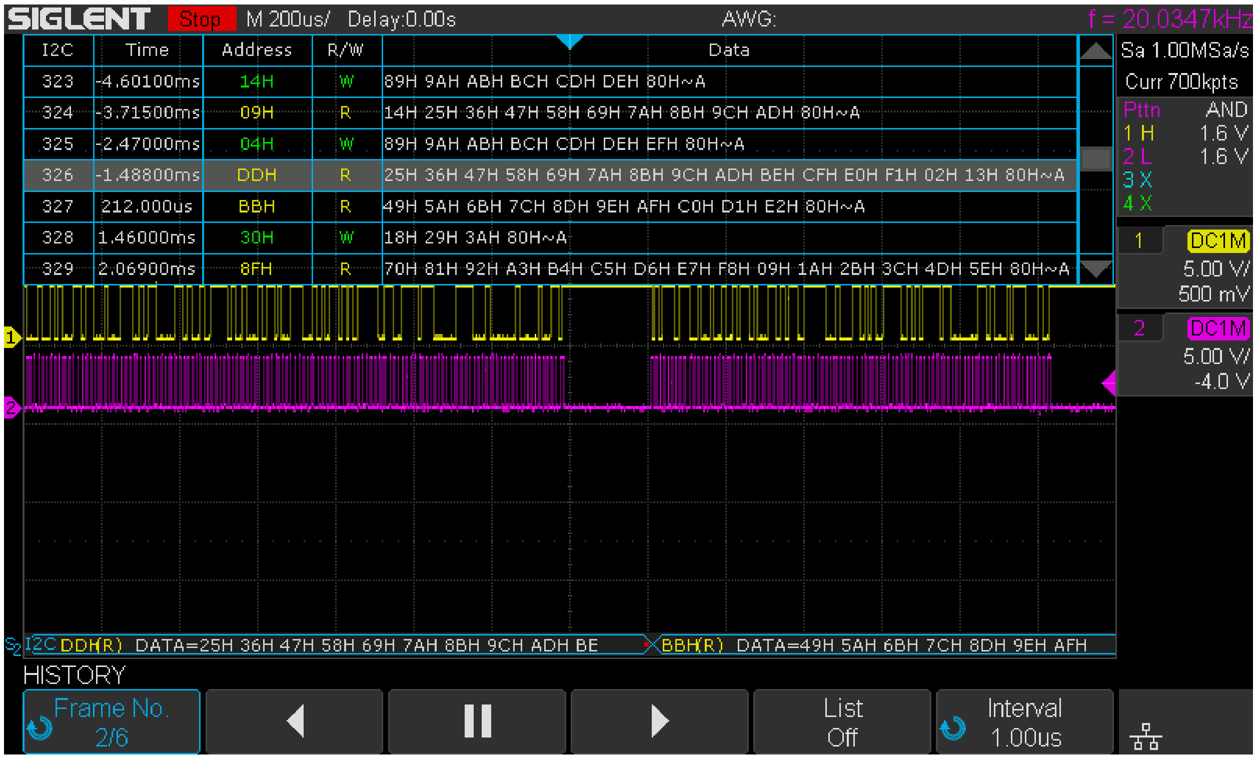
Trig'd M 50.0ms/ Delay:294ms Z 100us/ AWG: f = 19.9435kHz

I2C	Time	Address	R/W	Data
596	288.697ms	30H	W	18H 29H 3AH 80H~A
597	289.305ms	8FH	R	70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 2BH 3CH 4DH 5EH 80H~A
598	291.004ms	A6H	W	94H A5H B6H C7H D8H E9H 80H~A
599	291.889ms	2DH	R	1FH 30H 41H 52H 63H 74H 85H 96H A7H B8H C9H DAH EBH FCH 80H~A
600	293.499ms	0FH	R	32H 43H 54H 65H 76H 87H 98H A9H BAH CBH DCH EDH FEH 0FH 80H~A
601	295.114ms	06H	W	45H 56H 67H 78H 89H 9AH 80H~A
602	295.996ms	44H	W	D0H E1H F2H 03H 14H 25H 80H~A

Sa 1.00MSa/s
Curr 700kpts
Pttn AND
1 H 1.6 V
2 L 1.6 V
3 X
4 X
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V

I2C 2DH(R) DATA 0FH(R) DATA=32H 43H 54H 65H 76H 87H 98H A9H BAH CBH DCH EDH FEH 0FH 80H~A

SDS1104X-E_Serial_I2C_Zoom100us



Stop M 200us/ Delay:0.00s AWG: f = 20.0347kHz

I2C	Time	Address	R/W	Data
323	-4.60100ms	14H	W	89H 9AH ABH BCH CDH DEH 80H~A
324	-3.71500ms	09H	R	14H 25H 36H 47H 58H 69H 7AH 8BH 9CH ADH 80H~A
325	-2.47000ms	04H	W	89H 9AH ABH BCH CDH DEH EFH 80H~A
326	-1.48800ms	DDH	R	25H 36H 47H 58H 69H 7AH 8BH 9CH ADH BEH CFH E0H F1H 02H 13H 80H~A
327	212.000us	BBH	R	49H 5AH 6BH 7CH 8DH 9EH AFH C0H D1H E2H 80H~A
328	1.46000ms	30H	W	18H 29H 3AH 80H~A
329	2.06900ms	8FH	R	70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 2BH 3CH 4DH 5EH 80H~A

Sa 1.00MSa/s
Curr 700kpts
Pttn AND
1 H 1.6 V
2 L 1.6 V
3 X
4 X
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V

I2C DDH(R) DATA=25H 36H 47H 58H 69H 7AH 8BH 9CH ADH BE BBH(R) DATA=49H 5AH 6BH 7CH 8DH 9EH AFH

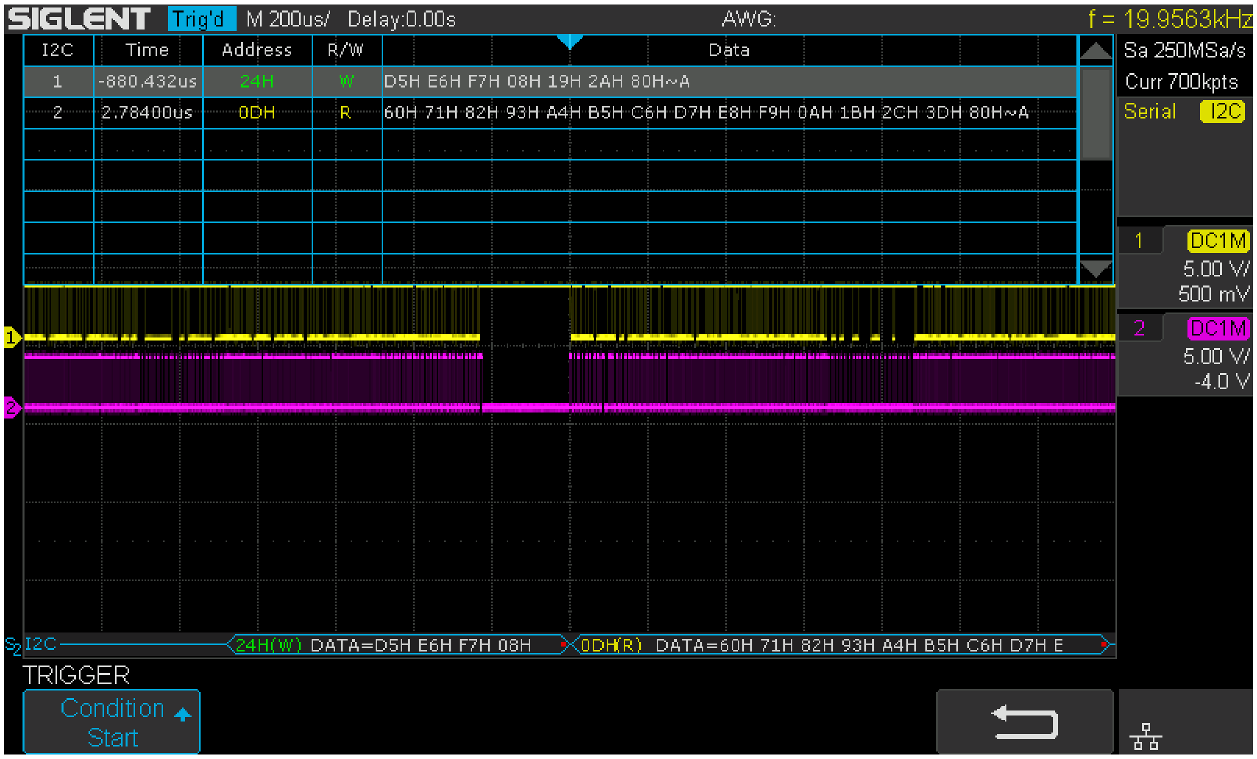
SDS1104X-E_Serial_I2C_History_200us

We can also stop the acquisition and enter history to analyze the current or any previous record stored in the history. Once again we can lower the time base and scroll through the record for closer inspection of certain parts of the message. The example above shows this for history record 2 of 6 and 200µs/div.

I^{2}C Trigger

The following screenshots show examples for some I^{2}C triggers:

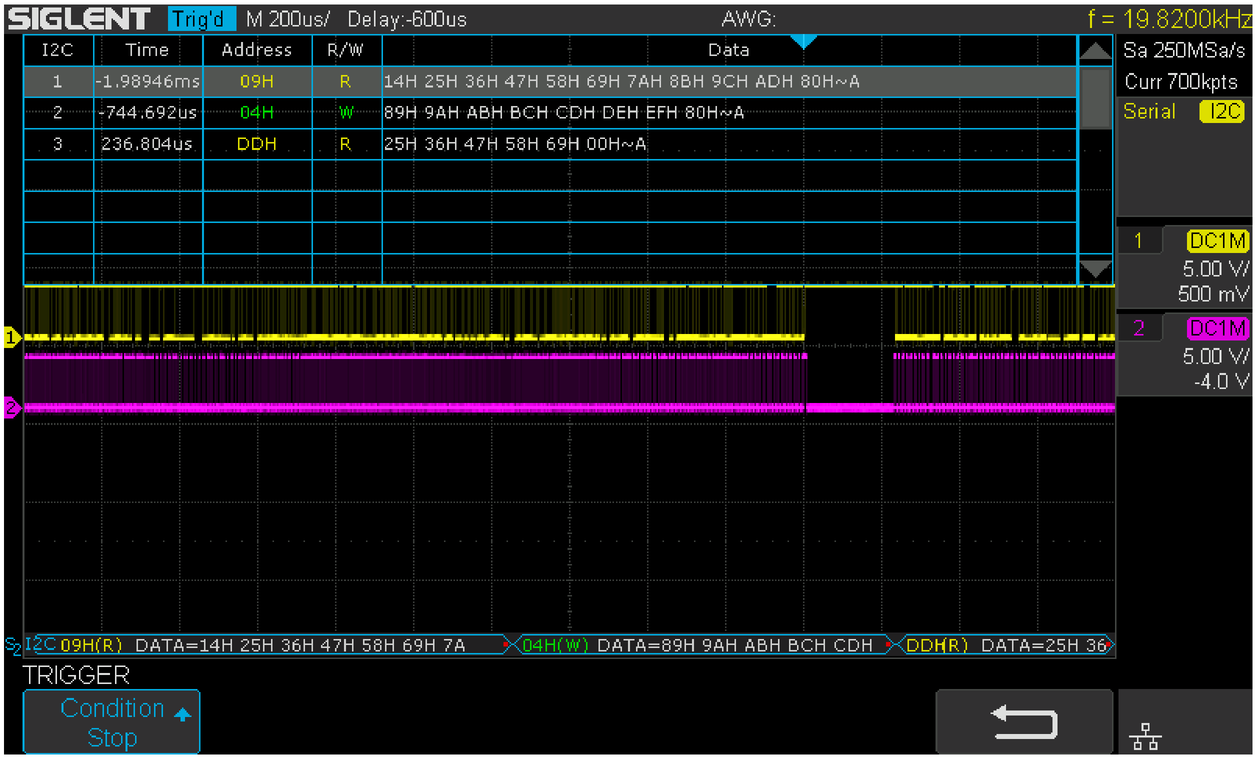
1. Trigger on I^{2}C start condition
2. Trigger on I^{2}C stop condition
3. Trigger on I^{2}C address bit length & data byte count 7bit / 2bytes
4. Trigger on I^{2}C 7bit address 47h & 2 data bytes = 2Bh, 3Ch (these two consecutive bytes can be anywhere within the message and it's the 12th and 13th byte in the example)
5. Trigger on I^{2}C EEPROM access, data = 94h



I2C	Time	Address	R/W	Data
1	-880.432us	24H	W	D5H E6H F7H 08H 19H 2AH 80H~A
2	2.78400us	0DH	R	60H 71H 82H 93H A4H B5H C6H D7H E8H F9H 0AH 1BH 2CH 3DH 80H~A

S2 I2C 24H(W) DATA=D5H E6H F7H 08H 0DH(R) DATA=60H 71H 82H 93H A4H B5H C6H D7H E

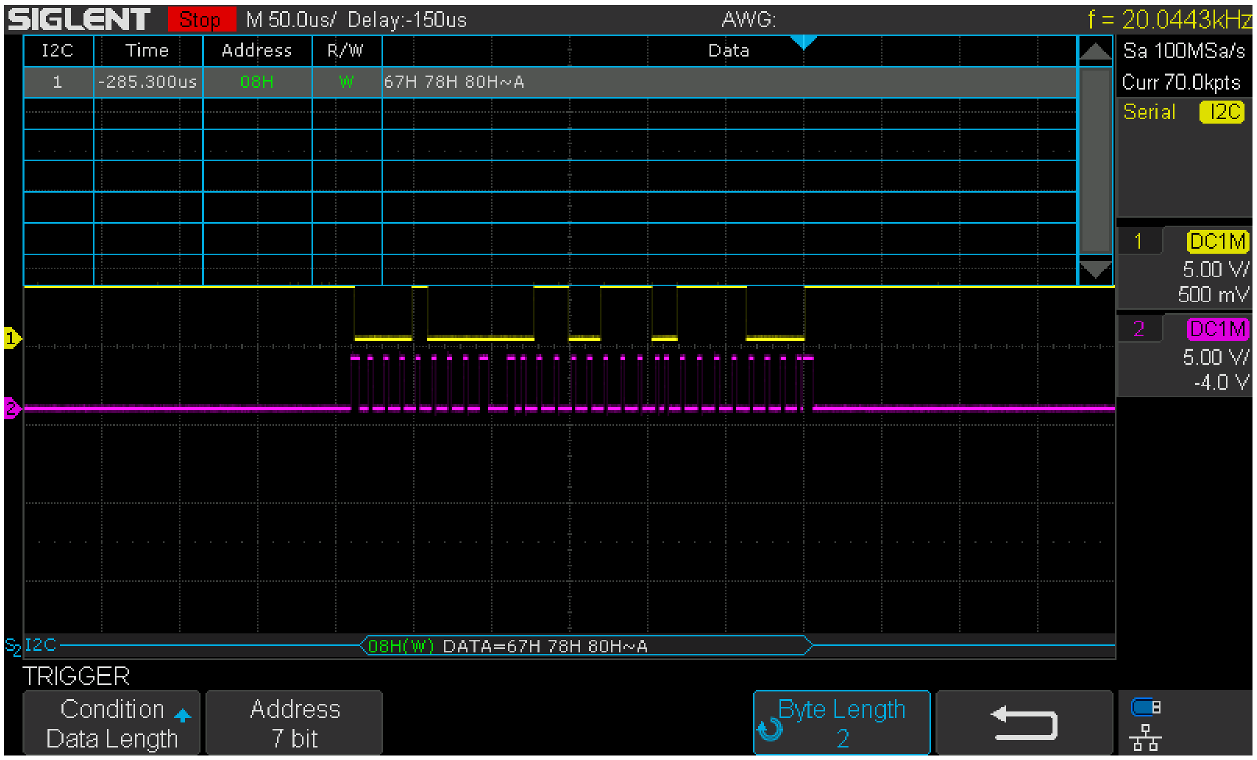
SDS1104X-E_Serial_I2C_Trigger_Start



I2C	Time	Address	R/W	Data
1	-1.98946ms	09H	R	14H 25H 36H 47H 58H 69H 7AH 8BH 9CH ADH 80H~A
2	-744.692us	04H	W	89H 9AH ABH BCH CDH DEH EFH 80H~A
3	236.804us	DDH	R	25H 36H 47H 58H 69H 00H~A

```
I2C 09H(R) DATA=14H 25H 36H 47H 58H 69H 7A  04H(W) DATA=89H 9AH ABH BCH CDH  DDH(R) DATA=25H 36
```

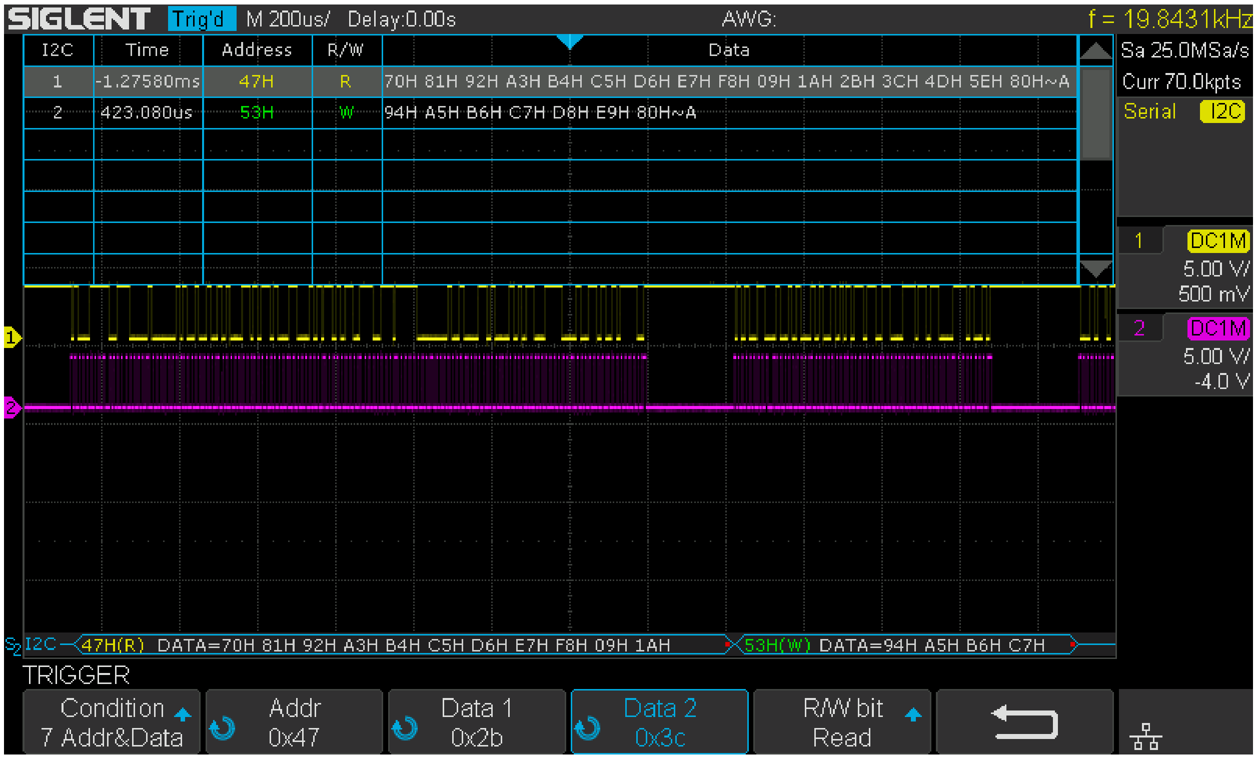
SDS1104X-E_Serial_I2C_Trigger_Stop



I2C	Time	Address	R/W	Data
1	-285.300us	08H	W	67H 78H 80H~A

```
I2C 08H(W) DATA=67H 78H 80H~A
```

SDS1104X-E_Serial_I2C_Trigger_Length_2


SIGLENT Trig'd M 200us/ Delay:0.00s AWG: f = 19.8431kHz

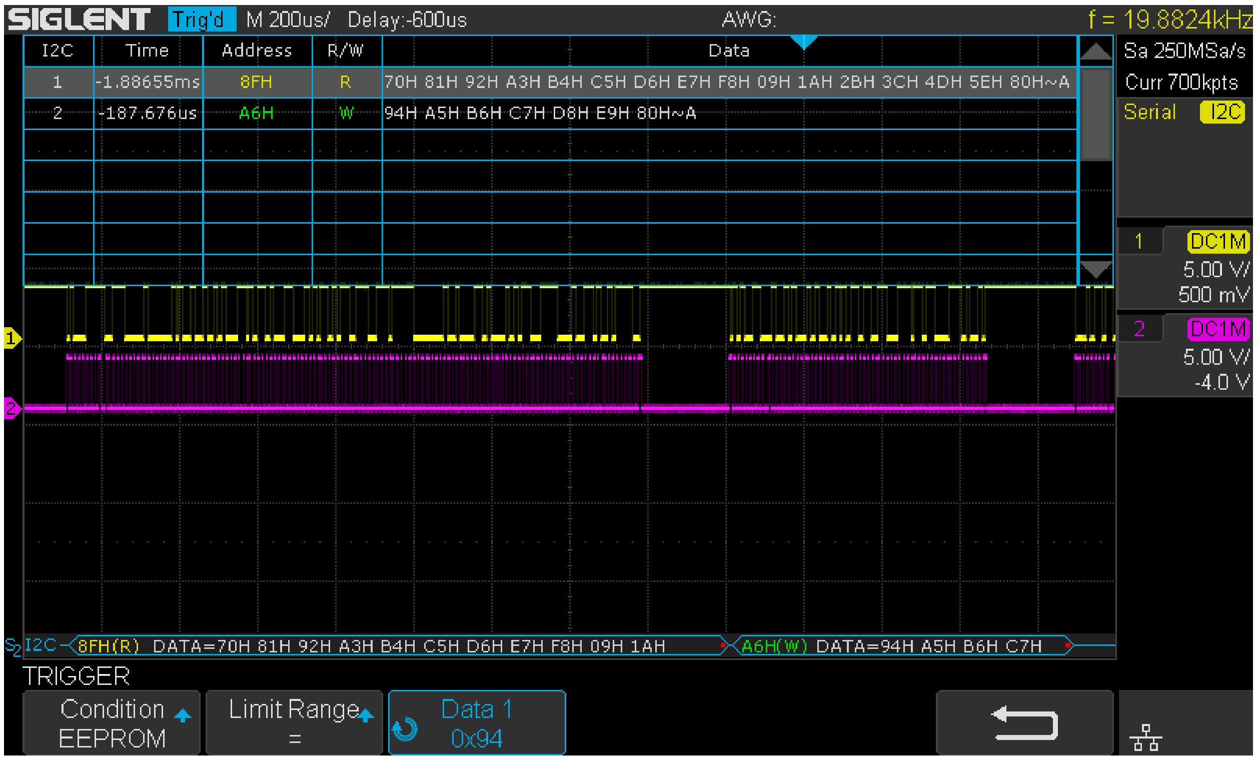
I2C	Time	Address	R/W	Data
1	-1.27580ms	47H	R	70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 2BH 3CH 4DH 5EH 80H~A
2	423.080us	53H	W	94H A5H B6H C7H D8H E9H 80H~A

Sa 25.0MSa/s
Curr 70.0kpts
Serial I2C
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V

I2C 47H(R) DATA=70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 53H(W) DATA=94H A5H B6H C7H

TRIGGER
Condition 7 Addr&Data | Addr 0x47 | Data 1 0x2b | Data 2 0x3c | R/W bit Read

SDS1104X-E_Serial_I2C_Trigger_A7&D_47h_2Bh_3Ch


SIGLENT Trig'd M 200us/ Delay:-600us AWG: f = 19.8824kHz

I2C	Time	Address	R/W	Data
1	-1.88655ms	8FH	R	70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH 2BH 3CH 4DH 5EH 80H~A
2	-187.676us	A6H	W	94H A5H B6H C7H D8H E9H 80H~A

Sa 250MSa/s
Curr 700kpts
Serial I2C
1 DC1M 5.00 V/ 500 mV
2 DC1M 5.00 V/ -4.0 V

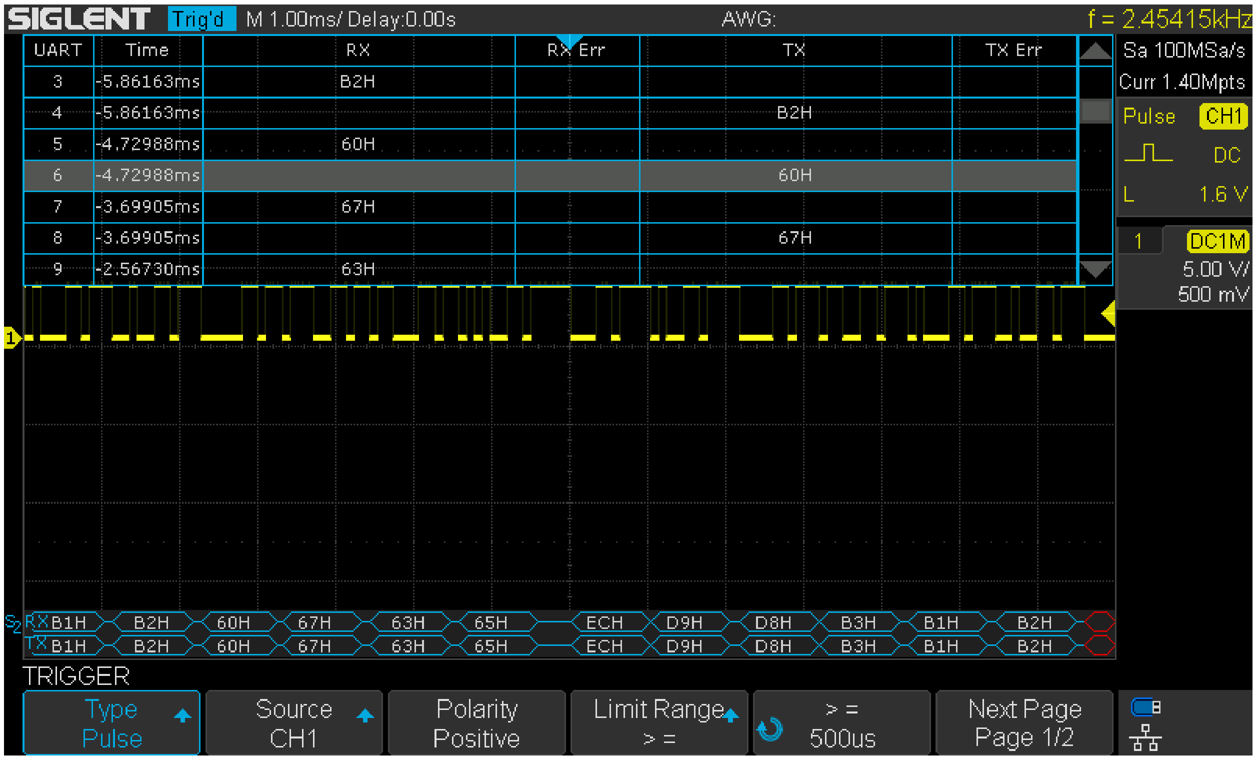
I2C 8FH(R) DATA=70H 81H 92H A3H B4H C5H D6H E7H F8H 09H 1AH A6H(W) DATA=94H A5H B6H C7H

TRIGGER
Condition EEPROM | Limit Range = | Data 1 0x94

SDS1104X-E_Serial_I2C_Trigger_EEPROM_94h

UART

By using 2 channels, we can have full duplex UART decoding with RX and TX. Triggering on the gap between messages can be achieved by using a pulse trigger for the idle level with a duration greater than a byte length which is about 8/baudrate (e.g. positive pulse >833µs for 9600 baud and idle level = high).



SIGLENT Trig'd M 1.00ms/ Delay:0.00s AWG: f = 2.45415kHz

UART	Time	RX	RX Err	TX	TX Err
3	-5.86163ms	B2H			
4	-5.86163ms			B2H	
5	-4.72988ms	60H			
6	-4.72988ms			60H	
7	-3.69905ms	67H			
8	-3.69905ms			67H	
9	-2.56730ms	63H			

Sa 100MSa/s
Curr 1.40Mpts
Pulse CH1
DC
L 1.6 V
1 DC1M
5.00 V/
500 mV

RX B1H B2H 60H 67H 63H 65H ECH D9H D8H B3H B1H B2H
TX B1H B2H 60H 67H 63H 65H ECH D9H D8H B3H B1H B2H

TRIGGER
Type Pulse | Source CH1 | Polarity Positive | Limit Range > = | > = 500us | Next Page Page 1/2

SDS1104X-E_Serial_UART_Trig

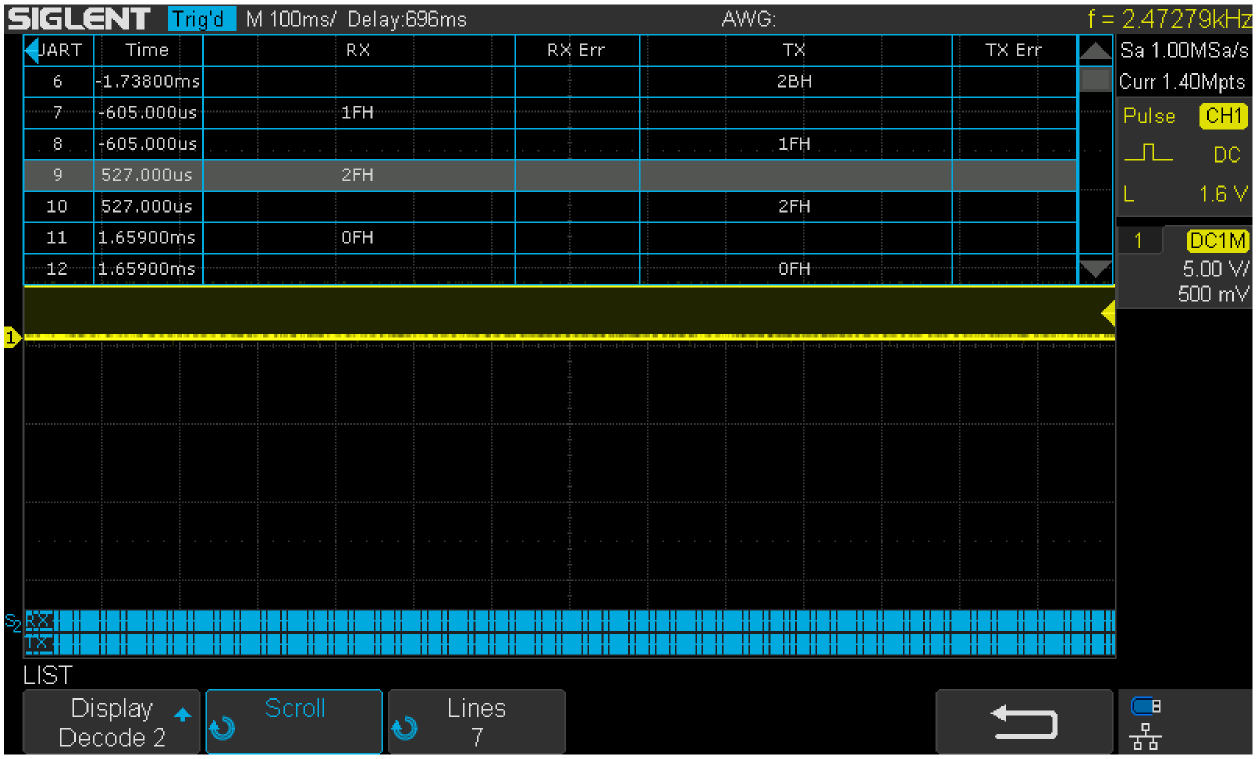
UART Decoder

To set up an UART decoder, proceed as follows:

- Go to the *Signal* menu and configure the channels and signal thresholds for RX and TX.
- In the *Configure* menu page 1, set the *Baud* rate (up to 5Mbaud), *Data Length* (5~8), *Parity* (even/odd/none) and *Stop Bit* (1/1.5/2) according to the frame format.
- In the *Configure* menu page 2, set the *Idle Level* (high/low) and the *Bit Order* (MSB/LSB first).

There is a limit of 6000 list entries, i.e. decoded bytes. We don't need excessive oversampling, a sample rate of ten times the baud rate is usually more than sufficient.

The following screenshot shows a half duplex UART transfer for 8N1 data frames at a bit rate of 9600 baud. Time base is 100ms/div, so we're looking at a time window of 1.4s at a sample rate of 1MSa/s, resulting in a record length of 1.4Mpts. Of course the signal traces as well as the decoder bar at the bottom of the screen are way too compressed to show anything meaningful at that, but the list view contains all the decoded bytes.



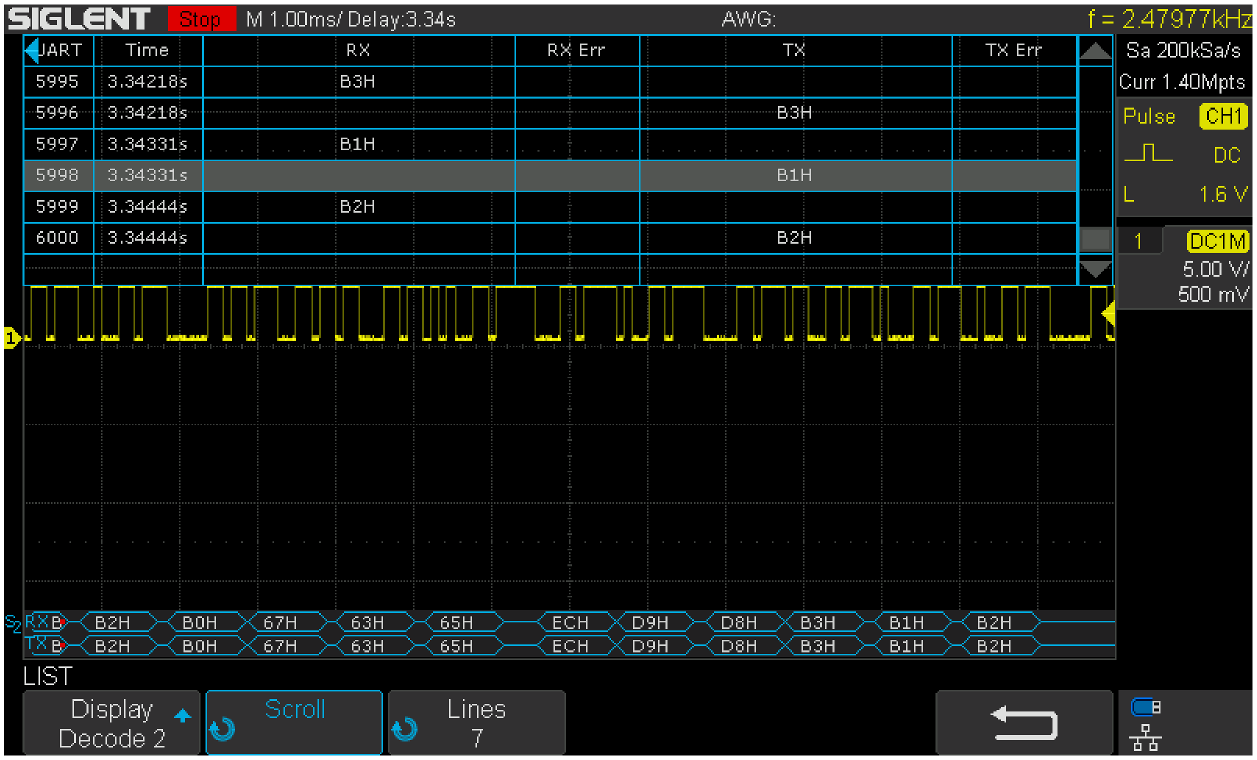
SIGLENT Trig'd M 100ms/ Delay:696ms AWG: f = 2.47279kHz

UART	Time	RX	RX Err	TX	TX Err
6	-1.73800ms			2BH	
7	-605.000us	1FH			
8	-605.000us			1FH	
9	527.000us	2FH			
10	527.000us			2FH	
11	1.65900ms	0FH			
12	1.65900ms			0FH	

Sa 1.00MSa/s
Curr 1.40Mpts
Pulse CH1
DC
L 1.6 V
1 DC1M
5.00 V/
500 mV

SDS1104X-E_Serial_UART

We can stop the acquisition, lower the time base in order to zoom into the waveform until the data in the decoder bar become readable and of course we can also scroll through the record as we like. The screenshot below shows this situation where the view has been positioned to the end of the record.



SIGLENT Stop M 1.00ms/ Delay:3.34s AWG: f = 2.47977kHz

UART	Time	RX	RX Err	TX	TX Err
5995	3.34218s	B3H			
5996	3.34218s			B3H	
5997	3.34331s	B1H			
5998	3.34331s			B1H	
5999	3.34444s	B2H			
6000	3.34444s			B2H	

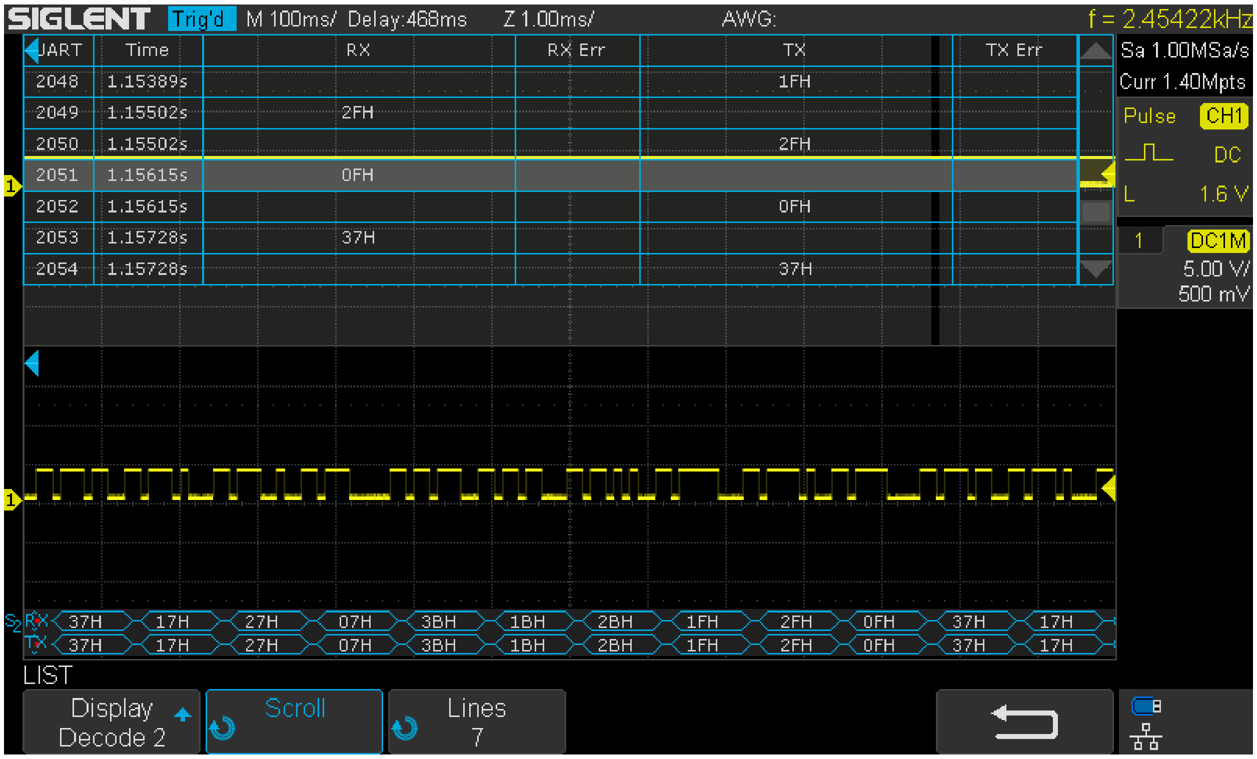
Sa 200kSa/s
Curr 1.40Mpts
Pulse CH1
DC
L 1.6 V
1 DC1M
5.00 V/
500 mV

RX B B2H B0H 67H 63H 65H ECH D9H D8H B3H B1H B2H
TX B B2H B0H 67H 63H 65H ECH D9H D8H B3H B1H B2H

SDS1104X-E_Serial_UART_Stop1ms_last

The list view in the upper part of the screen is automatically centered on a data byte near the center of the decoding bar and if necessary we can scroll the list a few positions down in order to see the very last data entry, which is the index 6000 for TX byte B2h in this example.

We can also inspect the signals and decoded data while acquisition is still running if we use zoom mode. Of course we can zoom in and out and scroll through the record as we like. The following screenshot demonstrates this situation for a zoomed time base of 1ms, with the zoom view centered on index 2051 of the decoder list.


SIGLENT Trig'd M 100ms/ Delay:468ms Z 1.00ms/ AWG: f = 2.45422kHz

JART	Time	RX	RX Err	TX	TX Err
2048	1.15389s			1FH	
2049	1.15502s	2FH			
2050	1.15502s			2FH	
2051	1.15615s	0FH			
2052	1.15615s			0FH	
2053	1.15728s	37H			
2054	1.15728s			37H	

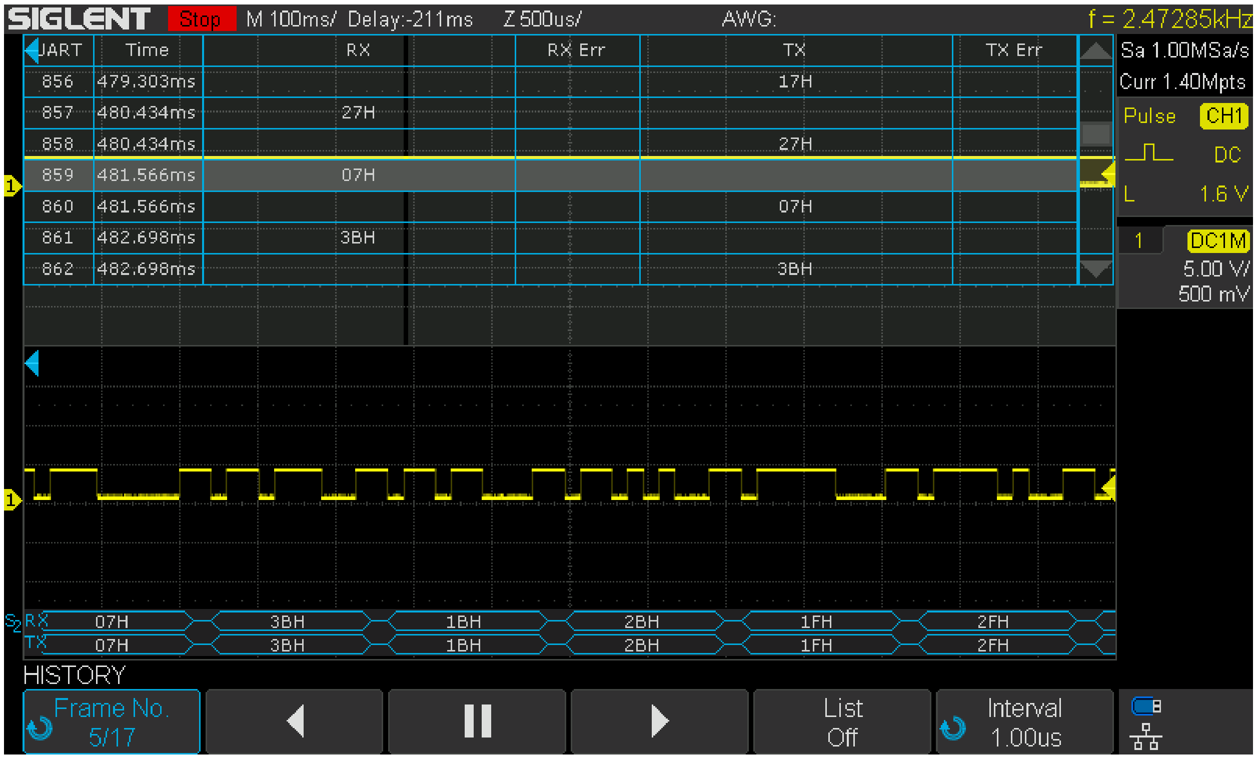
Sa 1.00MSa/s
Curr 1.40Mpts
Pulse CH1
DC
L 1.6 V
1 DC1M
5.00 V/
500 mV

RX: 37H 17H 27H 07H 3BH 1BH 2BH 1FH 2FH 0FH 37H 17H
TX: 37H 17H 27H 07H 3BH 1BH 2BH 1FH 2FH 0FH 37H 17H

LIST
Display Decode 2 | Scroll | Lines 7

SDS1104X-E_Serial_UART_Zoom1ms

We can also stop the acquisition and enter history to analyze the current or any previous record stored in the history. Once again we can lower the time base and scroll through the record for closer inspection of certain parts of the message. The example below shows this for history record 5 of 17 and 500µs/div.



SIGLENT Stop M 100ms/ Delay:-211ms Z 500us/ AWG: f = 2.47285kHz

UART	Time	RX	RX Err	TX	TX Err
856	479.303ms			17H	
857	480.434ms	27H			
858	480.434ms			27H	
859	481.566ms	07H			
860	481.566ms			07H	
861	482.698ms	3BH			
862	482.698ms			3BH	

Sa 1.00MSa/s
Curr 1.40Mpts
Pulse CH1 DC
L 1.6 V
1 DC1M 5.00 V/ 500 mV

```
RX  07H  3BH  1BH  2BH  1FH  2FH
TX  07H  3BH  1BH  2BH  1FH  2FH
```

HISTORY
Frame No. 5/17 | List Off | Interval 1.00us

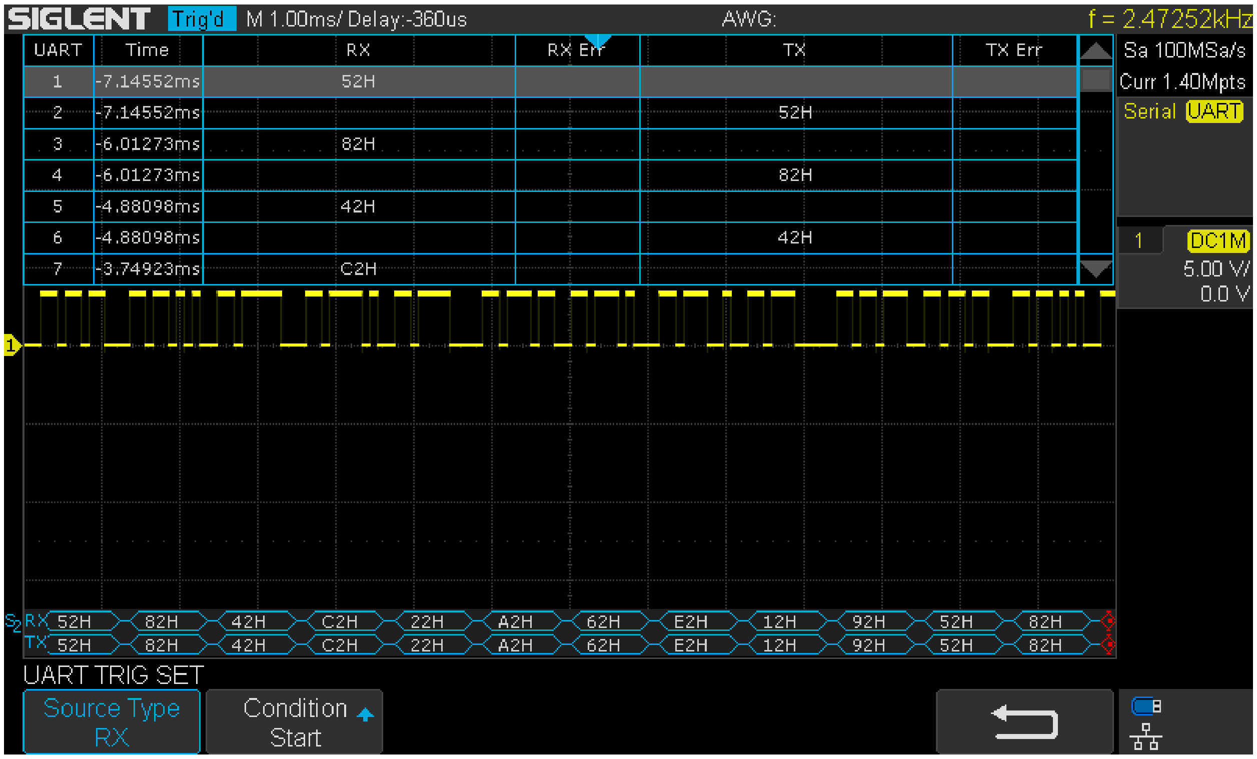
SDS1104X-E_Serial_UART_History_Zoom500us

UART Trigger

The following screenshots show examples for some UART triggers:

1. Trigger on UART start bit
2. Trigger on UART stop bit
3. Trigger on UART data byte = 1Bh

Be aware that the start bit detection is not very reliable and obviously different for trigger and decoder, so a mismatch between trigger condition and decoded data can happen, depending on the trigger position. I do hope Siglent will be able to improve that in a future update. There are no problems if the data stream contains brief idle periods longer than 10/baudrate (and the record includes such an idle period near the beginning), because both trigger and decoder will certainly be in sync after an idle state.



M 1.00ms/ Delay:-360us AWG: f = 2.47252kHz

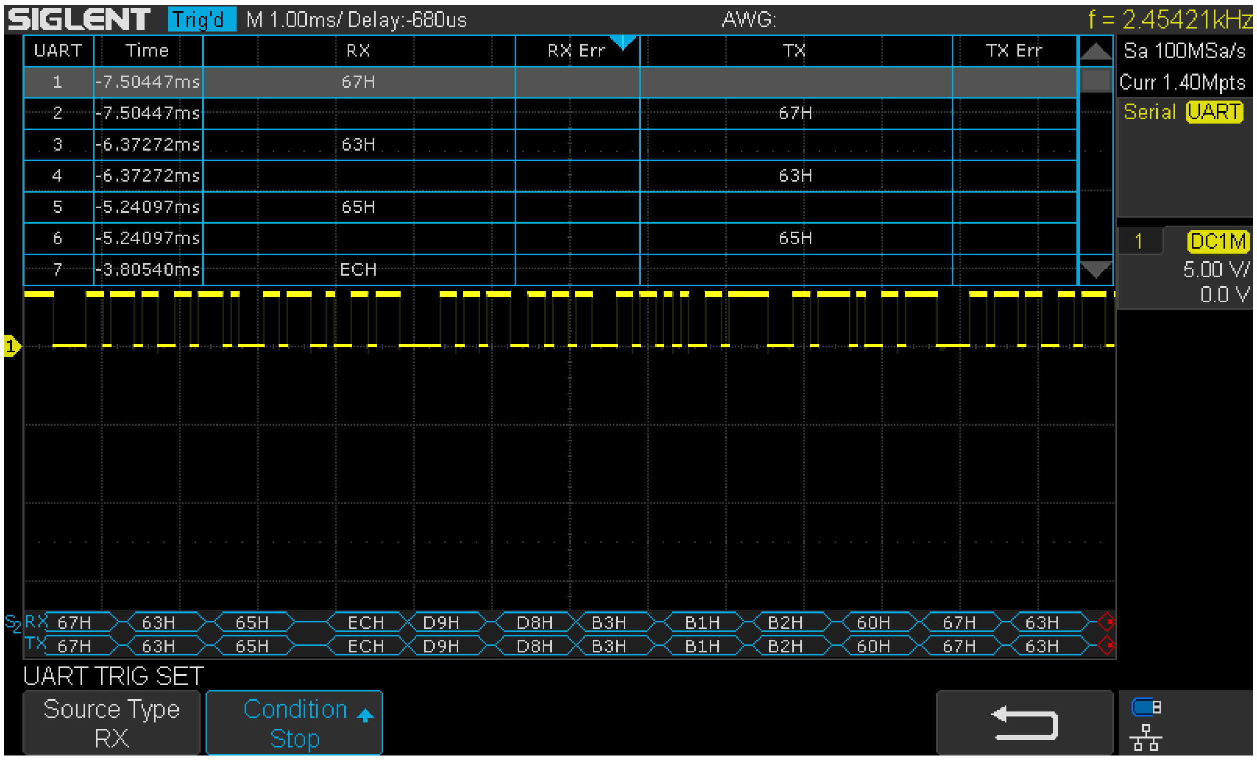
UART	Time	RX	RX Err	TX	TX Err
1	-7.14552ms	52H			
2	-7.14552ms			52H	
3	-6.01273ms	82H			
4	-6.01273ms			82H	
5	-4.88098ms	42H			
6	-4.88098ms			42H	
7	-3.74923ms	C2H			

Sa 100MSa/s
Curr 1.40Mpts
Serial UART
1 DC1M
5.00 V/
0.0 V

```
RX 52H 82H 42H C2H 22H A2H 62H E2H 12H 92H 52H 82H
TX 52H 82H 42H C2H 22H A2H 62H E2H 12H 92H 52H 82H
```

UART TRIG SET
Source Type RX
Condition Start

SDS1104X-E_Serial_UART_Trigger_Start



M 1.00ms/ Delay:-680us AWG: f = 2.45421kHz

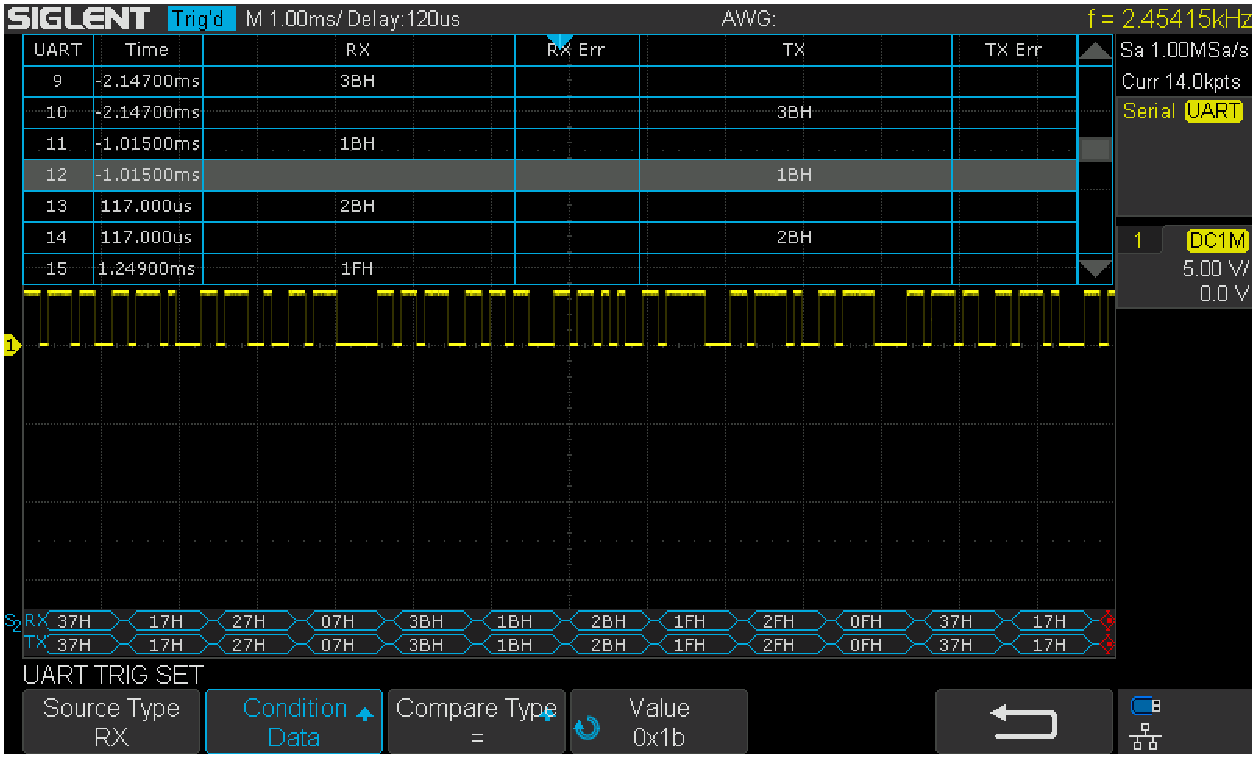
UART	Time	RX	RX Err	TX	TX Err
1	-7.50447ms	67H			
2	-7.50447ms			67H	
3	-6.37272ms	63H			
4	-6.37272ms			63H	
5	-5.24097ms	65H			
6	-5.24097ms			65H	
7	-3.80540ms	ECH			

Sa 100MSa/s
Curr 1.40Mpts
Serial UART
1 DC1M
5.00 V/
0.0 V

```
RX 67H 63H 65H ECH D9H D8H B3H B1H B2H 60H 67H 63H
TX 67H 63H 65H ECH D9H D8H B3H B1H B2H 60H 67H 63H
```

UART TRIG SET
Source Type RX
Condition Stop

SDS1104X-E_Serial_UART_Trigger_Stop



UART	Time	RX	RX Err	TX	TX Err
9	-2.14700ms	3BH			
10	-2.14700ms			3BH	
11	-1.01500ms	1BH			
12	-1.01500ms			1BH	
13	117.000us	2BH			
14	117.000us			2BH	
15	1.24900ms	1FH			

```
RX 37H 17H 27H 07H 3BH 1BH 2BH 1FH 2FH 0FH 37H 17H
TX 37H 17H 27H 07H 3BH 1BH 2BH 1FH 2FH 0FH 37H 17H
```

SDS1104X-E_Serial_UART_Trigger_Data1Bh

CAN

By using 2 channels, we can have half duplex CAN decoding with CAN_H and CAN_L. Triggering on the gap between messages can be achieved by using a positive pulse trigger for the CAN_L signal with a duration greater than a byte length which is about 8/bitrate (e.g. positive pulse >64µs for 125 kbit/s).

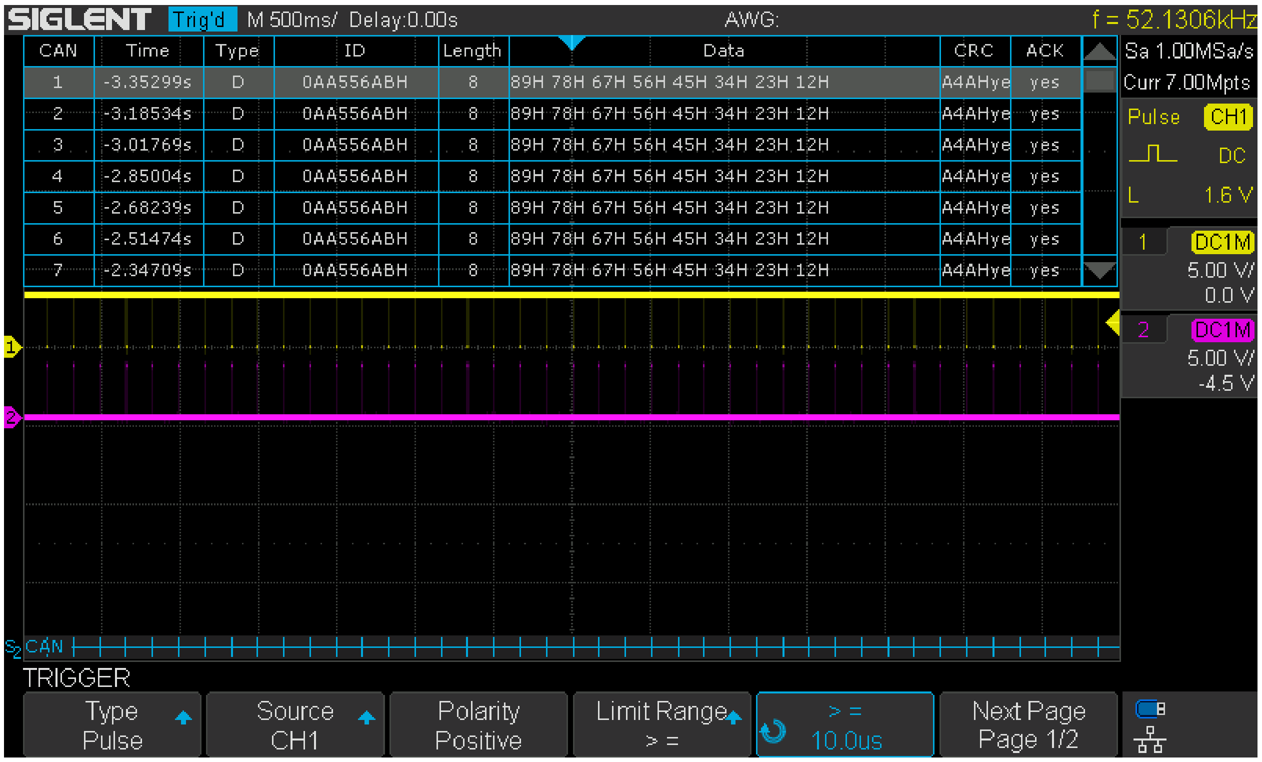
CAN Decoder

To set up a CAN decoder, proceed as follows:

- Go to the *Signal* menu and configure the channels and signal thresholds for CAN_H and CAN_L. For best results, select CANH-CANL as *Source* whenever feasible.
- Go to the *Configure* menu and set the *Baud* rate (up to 1Mbit/s).

There is a limit of 2000[1] list entries, i.e. decoded messages. We don't need an excessive oversampling, a sample rate of ten times the baud rate is usually more than sufficient.

The following screenshot shows a half duplex CAN transfer at a bit rate of 125kbit/s. Time base is 500ms/div, so we're looking at a time window of 7s at a sample rate of 1MSa/s, resulting in a record length of 7Mpts. Of course the signal traces as well as the decoder bar at the bottom of the screen can't show anything meaningful at that time scale, but the list view contains all the decoded messages.


SIGLENT Trig'd M 500ms/ Delay:0.00s AWG: f = 52.1306kHz

CAN	Time	Type	ID	Length	Data	CRC	ACK
1	-3.35299s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
2	-3.18534s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
3	-3.01769s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
4	-2.85004s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
5	-2.68239s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
6	-2.51474s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
7	-2.34709s	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes

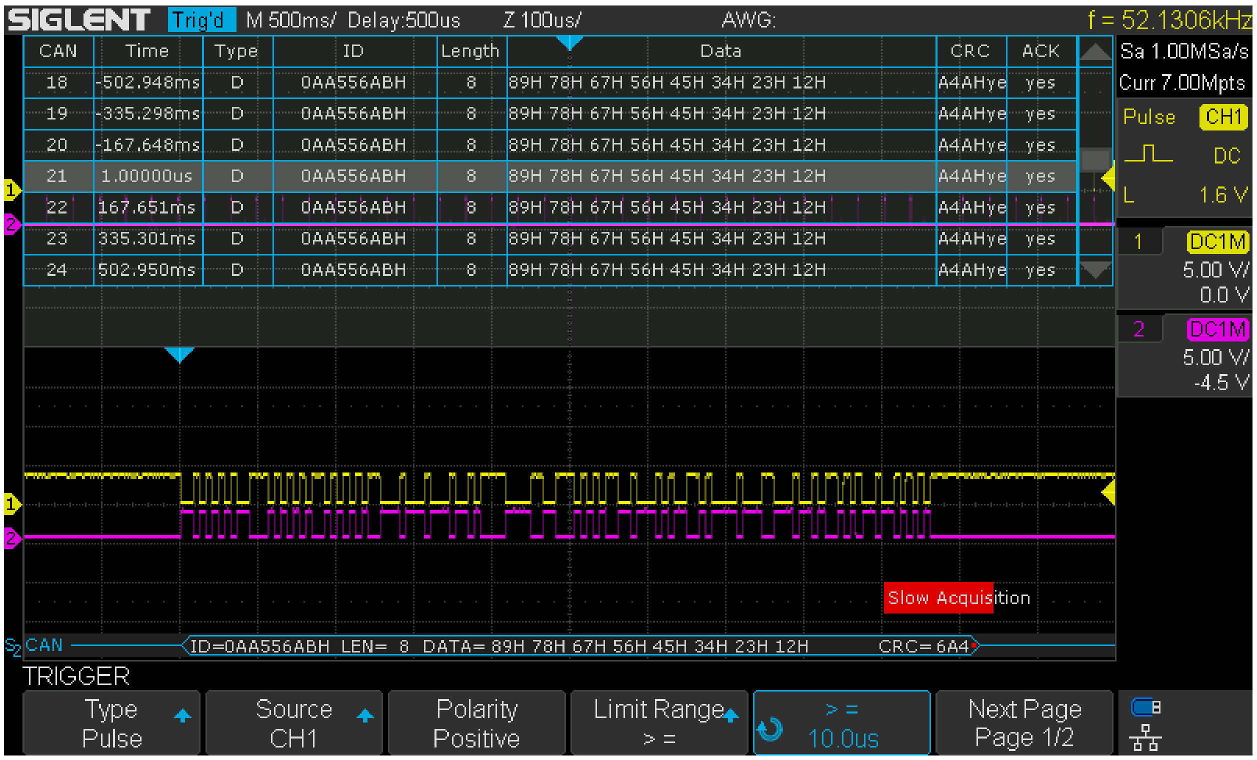
Sa 1.00MSa/s
Curr 7.00Mpts
Pulse CH1
DC
L 1.6 V
1 DC1M 5.00 V/ 0.0 V
2 DC1M 5.00 V/ -4.5 V
S2 CAN

TRIGGER
Type Pulse | Source CH1 | Polarity Positive | Limit Range > = | > = 10.0us | Next Page Page 1/2

SDS1104X-E_Serial_CAN

We can stop the acquisition, lower the time base in order to zoom into the waveform until the data in the decoder bar become readable and then scroll through the record as we like. We can also inspect the signals and decoded data while acquisition is still running if we use zoom mode. The following screenshot demonstrates this for a zoomed time base of 100µs, with the zoom view centered on list index 21.

[1] This is an estimation only. I have not been able to test this.



SIGLENT Trig'd M 500ms/ Delay:500us Z 100us/ AWG: f = 52.1306kHz

CAN	Time	Type	ID	Length	Data	CRC	ACK
18	-502.948ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
19	-335.298ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
20	-167.648ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
21	1.00000us	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
22	167.651ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
23	335.301ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes
24	502.950ms	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes

Sa 1.00MSa/s
Curr 7.00Mpts
Pulse CH1 DC
L 1.6 V
1 DC1M 5.00 V/ 0.0 V
2 DC1M 5.00 V/ -4.5 V

Slow Acquisition

S2 CAN ID=0AA556ABH LEN= 8 DATA= 89H 78H 67H 56H 45H 34H 23H 12H CRC= 6A4

TRIGGER
Type Pulse | Source CH1 | Polarity Positive | Limit Range > = | > = 10.0us | Next Page Page 1/2

SDS1104X-E_Serial_CAN_Zoom100us

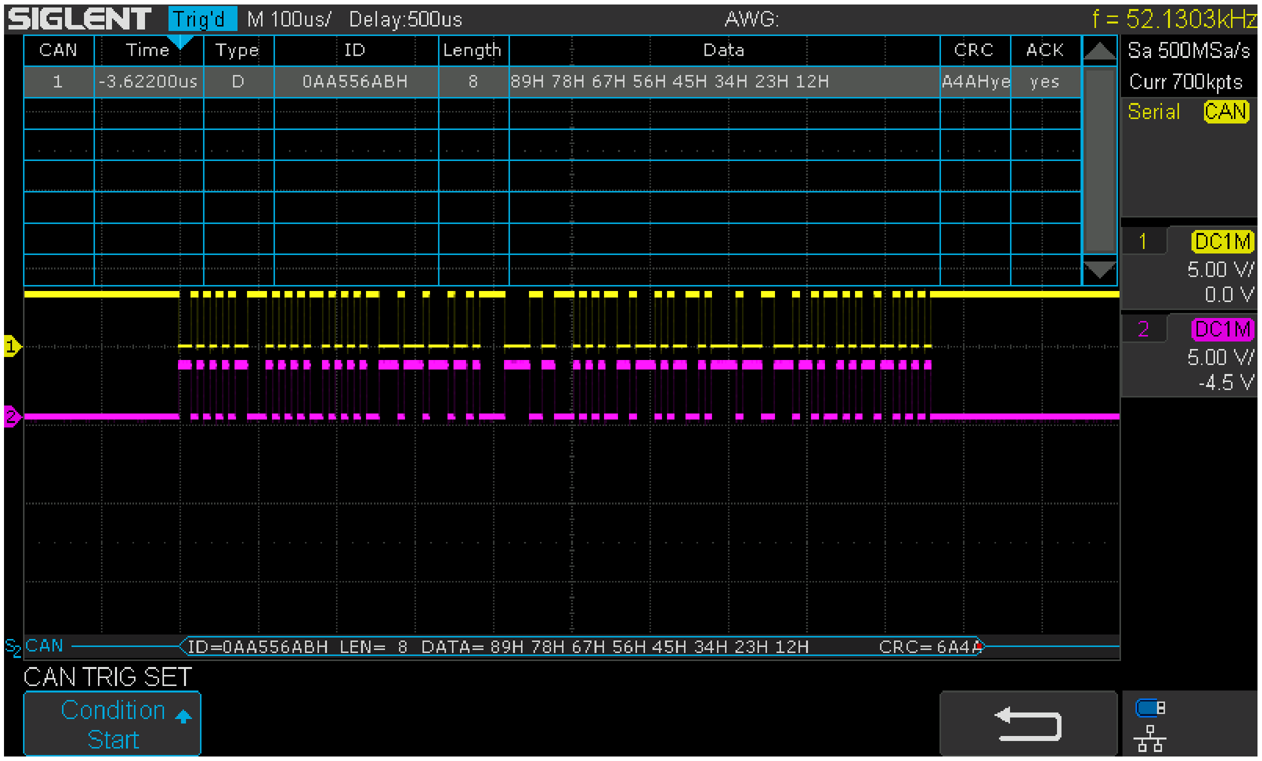
Of course we can do the same as with the other decoders, so it's also possible to analyze each individual record stored in the history.

Bug: As can be seen from the screenshots, the CRC is a mess as it is merged with the acknowledgement and doesn't fit into the table column because of this. The CRC is also truncated in the decoder bar for some weird reason. This has been tested with an early firmware and I expect this to be fixed by now.

CAN Trigger

The following screenshots show examples for some CAN triggers:

1. Trigger on CAN start
2. Trigger on CAN error
3. Trigger on CAN ID = 0AA556ABh
4. Trigger on CAN ID = 0AA556ABh & data = 67h, 56h (these two consecutive bytes can be anywhere within the message and it's the 2nd and 3rd byte in the example)



SIGLENT Trig'd M 100us/ Delay:500us AWG: f = 52.1303kHz

CAN	Time	Type	ID	Length	Data	CRC	ACK
1	-3.62200us	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes

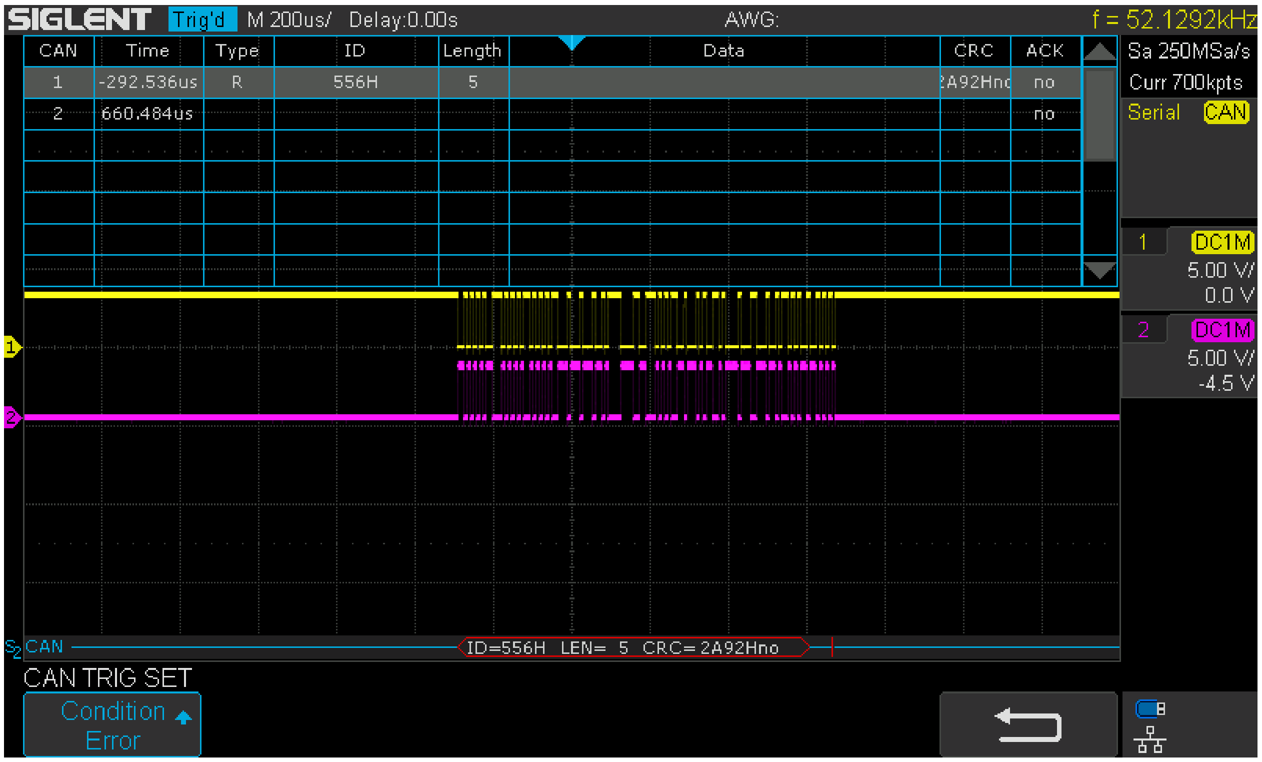
Sa 500MSa/s
Curr 700kpts
Serial CAN

1 DC1M 5.00 V/ 0.0 V
2 DC1M 5.00 V/ -4.5 V

S2 CAN ID=0AA556ABH LEN= 8 DATA= 89H 78H 67H 56H 45H 34H 23H 12H CRC= 6A4A

CAN TRIG SET
Condition Start

SDS1104X-E_Serial_CAN_Trigger_Start



SIGLENT Trig'd M 200us/ Delay:0.00s AWG: f = 52.1292kHz

CAN	Time	Type	ID	Length	Data	CRC	ACK
1	-292.536us	R	556H	5		2A92Hno	no
2	660.484us						no

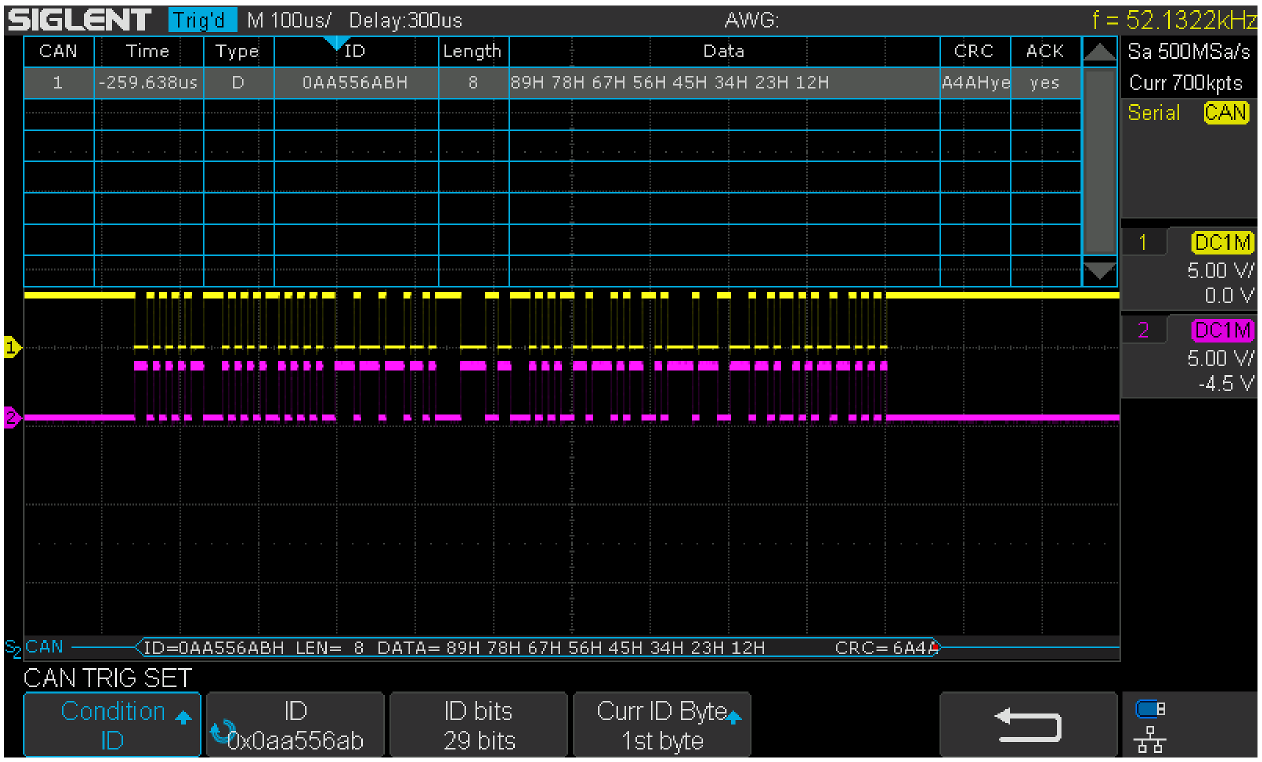
Sa 250MSa/s
Curr 700kpts
Serial CAN

1 DC1M 5.00 V/ 0.0 V
2 DC1M 5.00 V/ -4.5 V

S2 CAN ID=556H LEN= 5 CRC= 2A92Hno

CAN TRIG SET
Condition Error

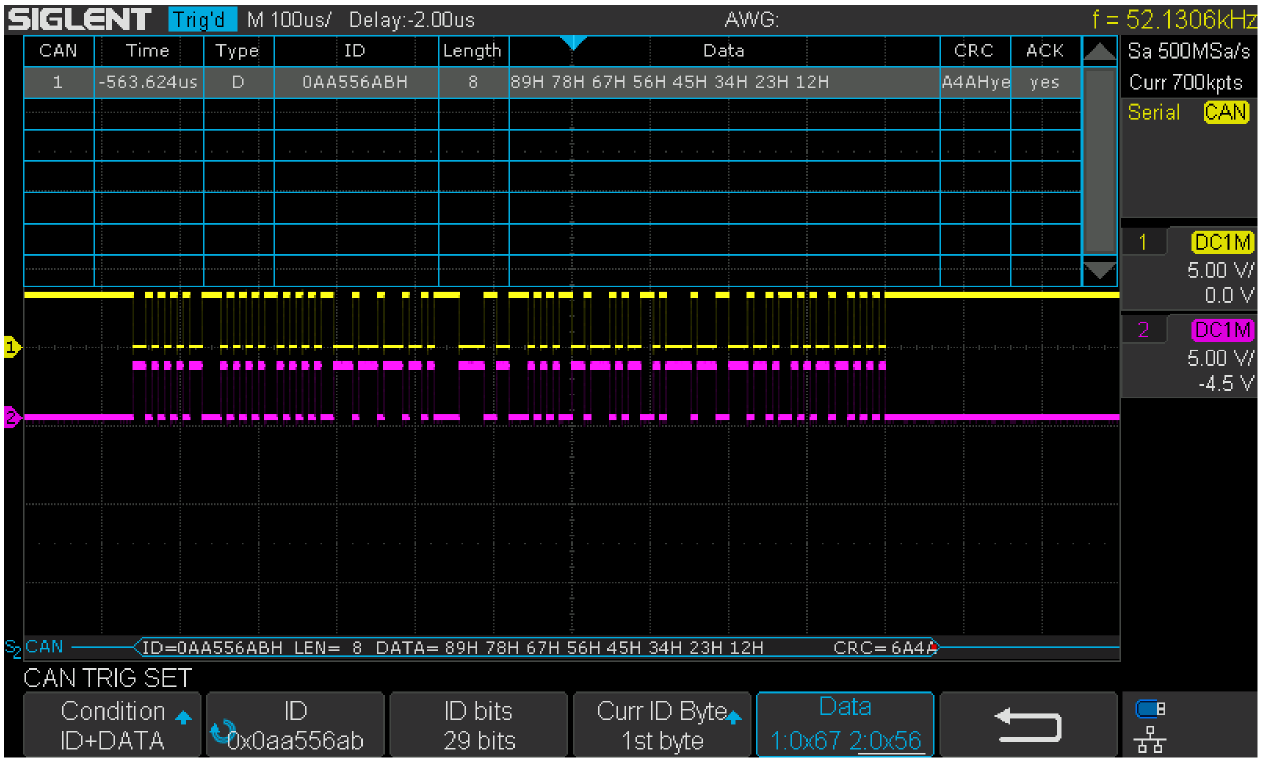
SDS1104X-E_Serial_CAN_Trigger_Error



CAN	Time	Type	ID	Length	Data	CRC	ACK
1	-259.638us	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes

```
CAN ID=0AA556ABH LEN= 8 DATA= 89H 78H 67H 56H 45H 34H 23H 12H    CRC= 6A4A
```

SDS1104X-E_Serial_CAN_Trigger_ID



CAN	Time	Type	ID	Length	Data	CRC	ACK
1	-563.624us	D	0AA556ABH	8	89H 78H 67H 56H 45H 34H 23H 12H	A4AHye	yes

```
CAN ID=0AA556ABH LEN= 8 DATA= 89H 78H 67H 56H 45H 34H 23H 12H    CRC= 6A4A
```

SDS1104X-E_Serial_CAN_Trigger_ID+Data

LIN

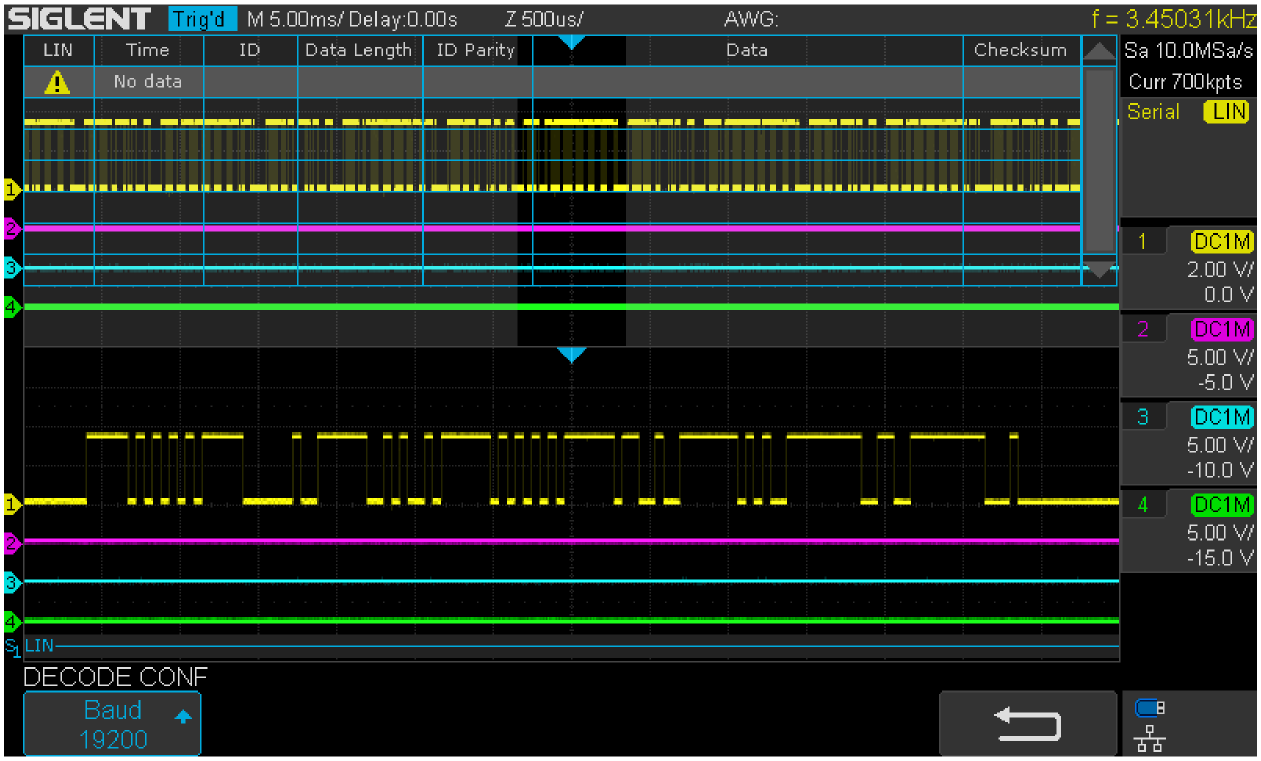
By using just one channel, we can have half duplex LIN decoding. Triggering on the gap between messages can be achieved by using a negative pulse trigger for the LIN signal with a duration greater than a byte length which is about 8/bitrate (e.g. negative pulse >417µs for 19.2 kbit/s).

LIN Decoder

To set up a LIN decoder, proceed as follows:

- Go to the *Signal* menu and configure the channel and signal threshold for LIN.
- Go to the *Configure* menu and set the *Baud* rate (up to 20kbit/s).

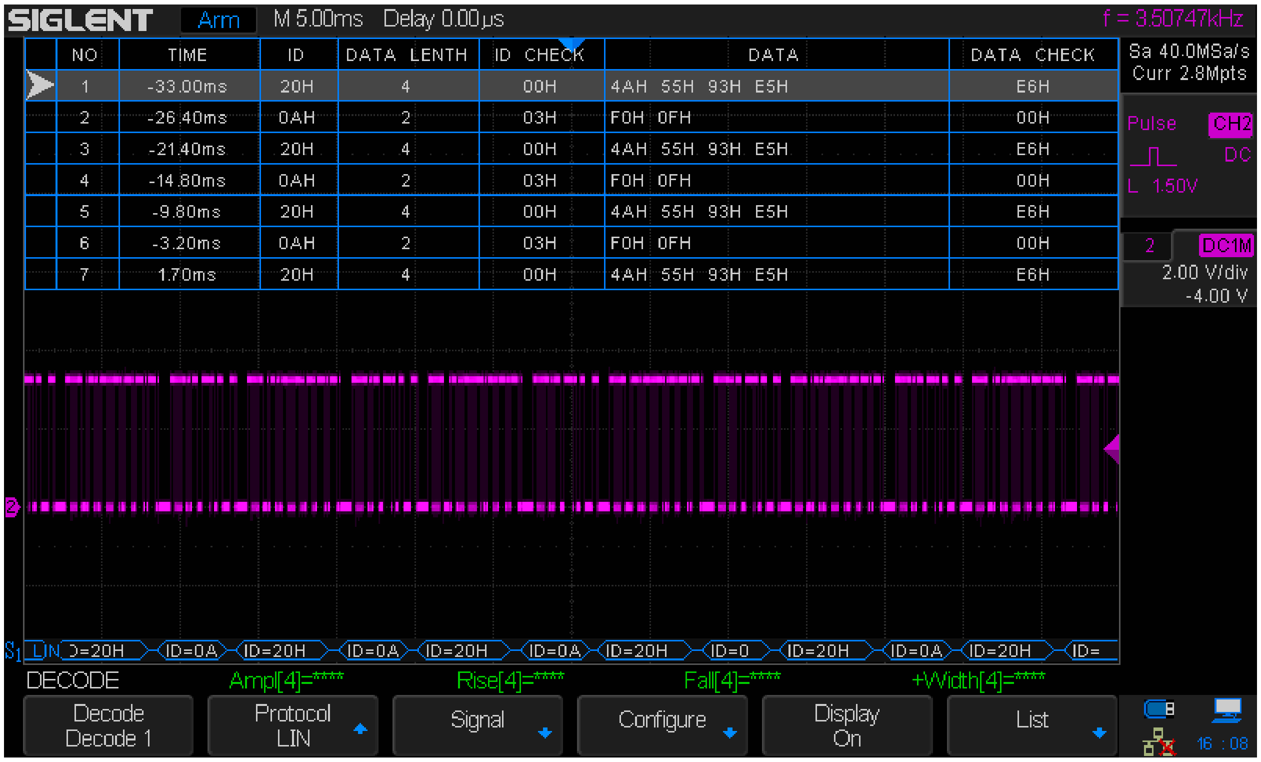
Bug: The LIN decoder doesn't seem to like my signals, even though LIN ID & data trigger does. This has been tested with an early firmware and I expect this to be fixed by now.



LIN	Time	ID	Data Length	ID Parity	Data	Checksum
⚠	No data					

SDS1104X-E_LIN_Zoom_10M

For the time being, the decoding from a SDS2304X is shown instead:



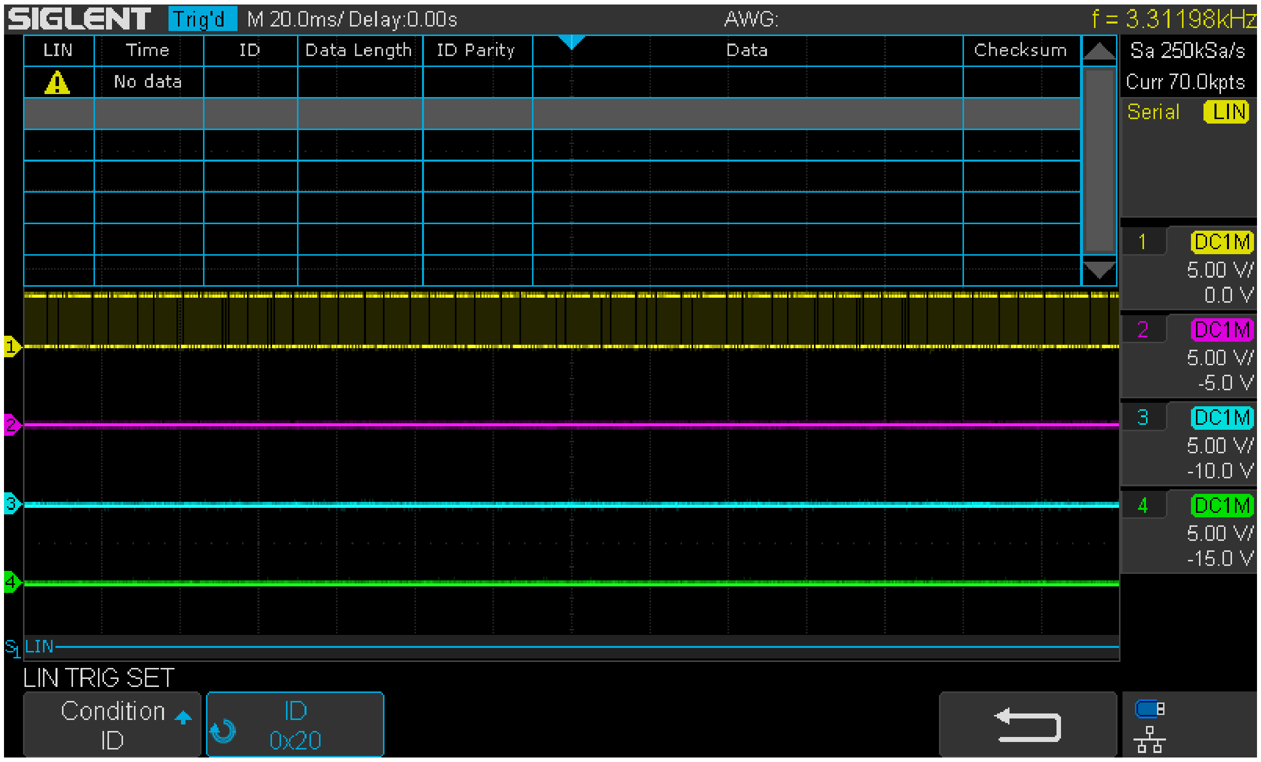
NO	TIME	ID	DATA LENTH	ID CHECK	DATA	DATA CHECK
1	-33.00ms	20H	4	00H	4AH 55H 93H E5H	E6H
2	-26.40ms	0AH	2	03H	F0H 0FH	00H
3	-21.40ms	20H	4	00H	4AH 55H 93H E5H	E6H
4	-14.80ms	0AH	2	03H	F0H 0FH	00H
5	-9.80ms	20H	4	00H	4AH 55H 93H E5H	E6H
6	-3.20ms	0AH	2	03H	F0H 0FH	00H
7	1.70ms	20H	4	00H	4AH 55H 93H E5H	E6H

SDS2304X_LIN

LIN Trigger

The following screenshots show examples for some LIN triggers:

1. Trigger on LIN ID = 20h
2. Trigger on LIN ID = 20h & data = 4Ah, 55h (these two consecutive bytes can be anywhere within the message and it's the 1st and 2nd byte in the example)



M 20.0ms/ Delay:0.00s
f = 3.31198kHz

LIN	Time	ID	Data Length	ID Parity	Data	Checksum
⚠	No data					

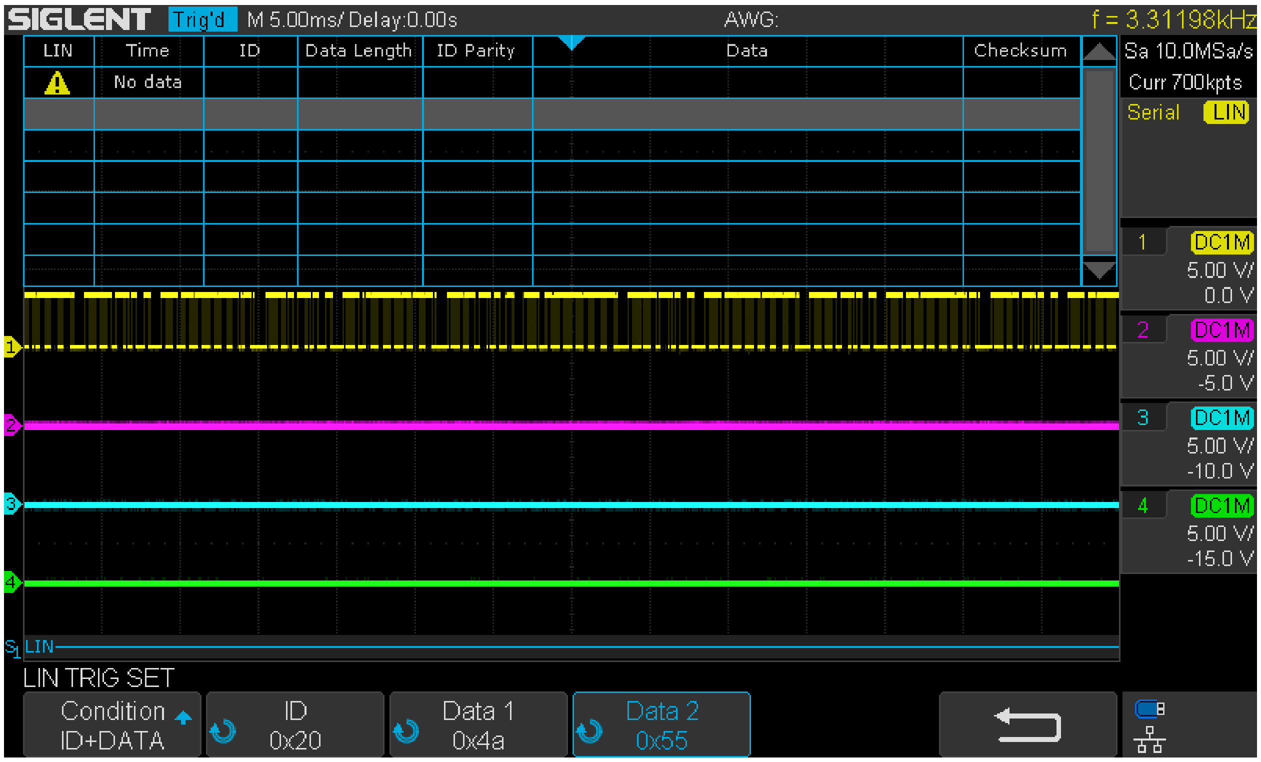
Sa 250kSa/s
Curr 70.0kpts

1	DC1M	5.00 V/	0.0 V
2	DC1M	5.00 V/	-5.0 V
3	DC1M	5.00 V/	-10.0 V
4	DC1M	5.00 V/	-15.0 V

LIN TRIG SET

Condition ID
ID 0x20

SDS1104X-E_LIN_Trig_ID



M 5.00ms/ Delay:0.00s
f = 3.31198kHz

LIN	Time	ID	Data Length	ID Parity	Data	Checksum
⚠	No data					

Sa 10.0MSa/s
Curr 700kpts

1	DC1M	5.00 V/	0.0 V
2	DC1M	5.00 V/	-5.0 V
3	DC1M	5.00 V/	-10.0 V
4	DC1M	5.00 V/	-15.0 V

LIN TRIG SET

Condition ID+DATA
ID 0x20
Data 1 0x4a
Data 2 0x55

SDS1104X-E_LIN_Trig_ID_Data_10M