Details on Decoding Options Package

Serial Decode Specifications

Number of Buses for Decoding: 2

Decoding Type:

Parallel (standard) RS232 /UART (option), I2C (option), SPI (MSO4XX4/DS4XX4 option), CAN (option), FlexRay (option)

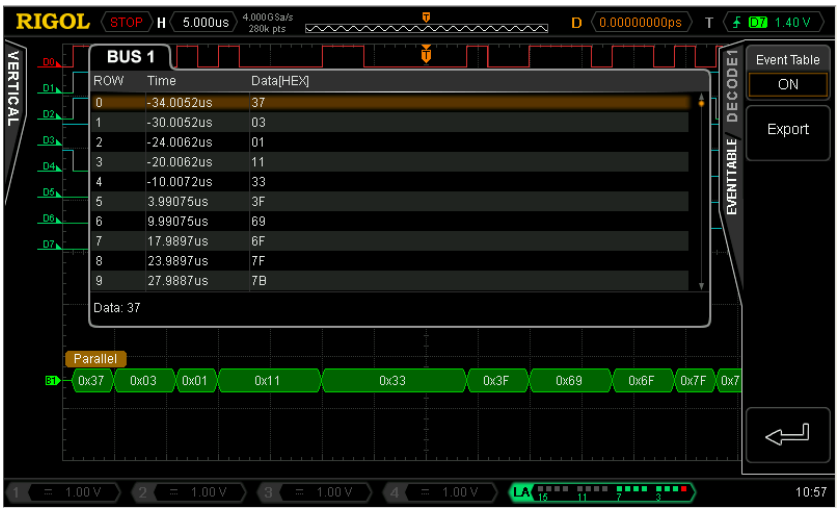
Serial Bus Triggering Specifications

RS232/UART Trigger	
Trigger Condition	Start, Error, Check Error, Data
Polarity	Normal, Invert
Baud	2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115200 bps, User
Data Bits	5 bit, 6 bit, 7 bit, 8 bit
I2C Trigger	
Trigger Condition	Start, Restart, Stop, Missing Ack, Address, Data, A&D
Address Bits	7 bit, 8 bit, 10 bit
Address Range	0 to 127, 0 to 255, 0 to 1023
Byte Length	1 to 5
SPI Trigger	
Trigger Condition	CS, TimeOut
Timeout Value	100 ns to 1 s
Data Bits	4 bit to 32 bit
Data Line Setting	H, L, X
Clock Edge	Rising edge, Falling edge
CAN Trigger	
Signal Type	Rx, Tx, CAN_H, CAN_L, Differential
Trigger Condition	SOF, EOF, Frame Type, Frame Error
Baud	10 kbps, 20 kbps, 33.3 kbps, 50 kbps, 62.5 kbps, 83.3 kbps, 100 kbps, 125 kbps, 250 kbps, 500 kbps, 800
	kbps, 1 Mbps, User
Sample Point	5% to 95%
Frame Type	Data, Remote, Error, OverLoad
Error Type	Bit Fill, Answer Error, Check Error, Format Error, Random Error
FlexRay Trigger	
Baud	2.5 Mb/s, 5 Mb/s, 10 Mb/s
Trigger Condition	Frame, Symbol, Error, TSS

5. Decoding Table

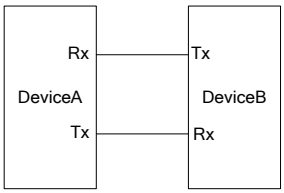
The decoding table displays the decoded data and the corresponding line number and time in table format. This can be useful when observing longer data transimissions by presenting the data in a tabular format.

Press **Event Table** → **Event Table** to select “ON” (note that this operation is only available when **BusStatus** is set to “ON”) to enter the decoding table interface as shown in the figure below. The decoding table lists the decoded data in time order. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.

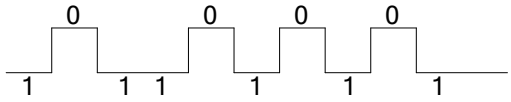


RS232 Decoding (Option)

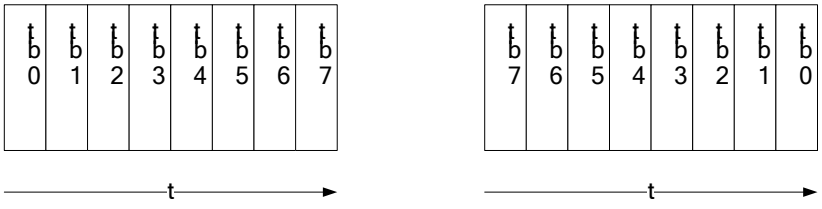
RS232 serial bus consists of the transmitting data line (TX) and the receiving data line (RX).



Industry standard RS232 uses “Negative Logic”, namely high level is logic “0” and low level is logic “1”.



By default RS232 also uses LSB (Least Significant Bit) transmission sequence, namely the lowest bit of the data is transmitted first. While for MSB (Most Significant Bit), the highest bit of the data is transmitted first.



Endian (LSB) Endian (MSB)

In RS232, baud rate is used to represent the transmitting rate (namely bits per second) of the data. The commonly used baud rates include 2400bps, 4800bps, 9600bps, 19200bps, 38400bps, 57600bps and 115200bps.

In RS232, you need to set the start bit, data bits, check bit (optional) and stop bit of each frame of data.

Start Bit: represents when the data begins. Setting the **Polarity** is equivalent to specifying the "Start Bit".

Data Bits: represents the number of data bits actually contained in each frame of data.

Even-OddCheck: check the correctness of the data transmission.

- Odd-Check: the number of "1" in the data bit and check bit is an odd. For example, when 0x55 (01010101) is sent, a 1 needs to be filled in the check bit to make the number of 1 to be an odd.
- Even Check: the number of "1" in data bit and check bit is an even. For example, when 0x55 (01010101) is sent, a 0 should be filled in the check bit.
- None: no check bit during the transmission.

Press **Decode1** → **Decode** to select "RS232" to open the RS232 decoding function menu.

1. TX and RX Channel Setting

Press **TX** to select any channel (CH1 to CH4 or D0-D15) as the transmitting channel and when "OFF" is selected, no transmitting channel is set. Use the same method the set the **RX** channel. What's more, you need to set the thresholds of the input channels of **TX** and **RX**. Switch the menu page and press **TXThreshold** and **RXThreshold** respectively to input the desired threshold values.

2. Polarity Setting

Press **Polarity** to select "Normal" or "Invert" and the default is normal. The oscilloscope will select the rising or falling edge as the start position during decoding.

3. Endian Setting

Press **Endian** to select "LSB" or "MSB" and the default is "LSB".

4. Baud Rate Setting

Press **Baud** to select the desired baud rate and the default is 9600bps.

5. Data Packet Setting

As mentioned before, in RS232, you need to set the start bit, data bits, check bit (optional) and stop bit of each frame of data. "Start Bit" is specified by the "Polarity Setting". The setting methods of other parameters are as follows:

- Press **Data Bits** to set the data width of each frame. It can be set to 5, 6, 7, 8 or 9 and the default is 8.
- Press **Stop Bit** to set the stop bit after each frame of data. It can be set to 1 bit, 1.5 bits or 2 bits.
- Press **Even-OddCheck** to set the even-odd check mode of the data transmission. It can be set to None, Odd Check or Even Check.
- Press **Packet** to enable or disable the packet end. When packet end is enabled, several data blocks are combined according to the packet end.
- Press **PacketEnd** to set the packet end during data transmission and it can be set to 00 (NULL), 0A (LF), 0D (CR), 20 (SP) or FF.

6. Display-related Setting

Press **Format** to set the display format of the bus to Hex, Decimal, Binary or ASCII.

Press **Offset** and use ↻ to adjust the vertical display position of the bus.

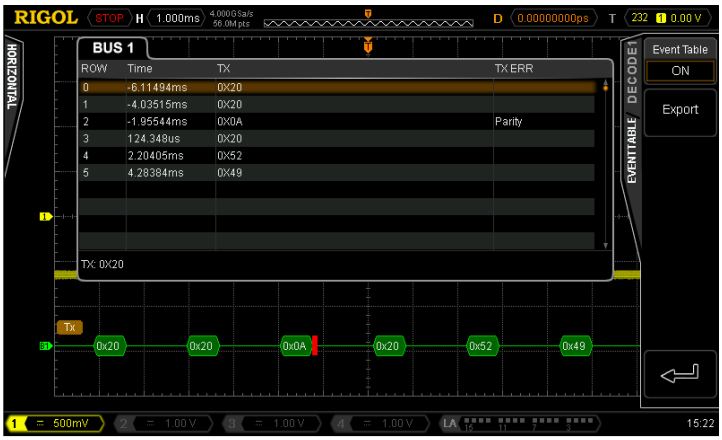
Press **BusStatus** to turn the bus display on or off.

7. Decoding Table

The decoding table displays the decoded data, the corresponding line number, time and error information on TX and RX data lines in table format. This can be useful when observing longer data transimissions by presenting the data in a tabular format.

Press **Event Table** → **Event Table** to select "ON" (note that this operation is only available when **BusStatus** is set to "ON") to enter the decoding table interface as shown in the figure below. The decoding table lists the decoded data in time order. If an error occurs during the decoding, the corresponding

error information is displayed. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.

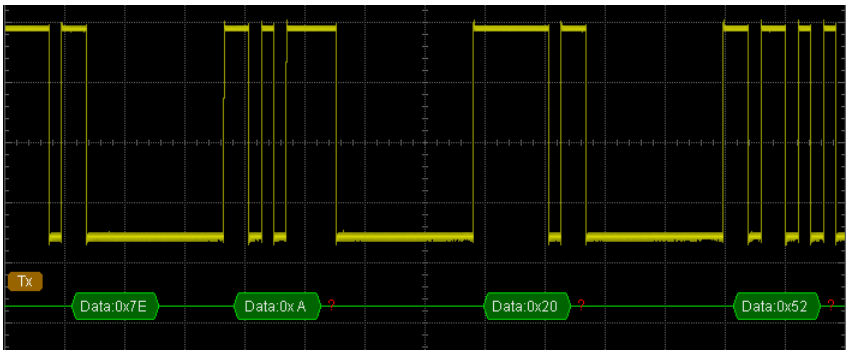


8. The Error Expression during Decoding

The MSO4000/DS4000 makes full use of the resources such as color and view to express the results of the protocol decoding effectively so as to let users find the desired information quickly.

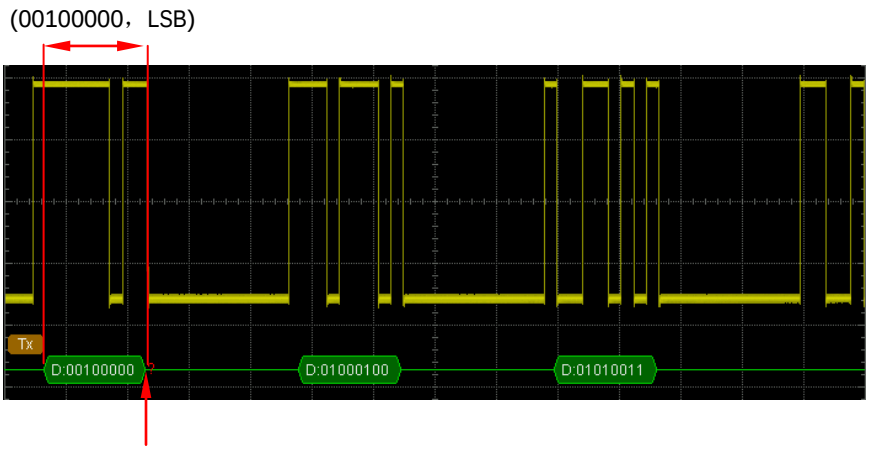
End Frame Error:

This error is generated when the end frame condition is not met. When the stop bit is set to 1.5, a red error mark is displayed (note that the red mark is displayed in different modes according to the horizontal time base setting. When the horizontal time base is small,  is displayed. Otherwise,  is displayed) is displayed.



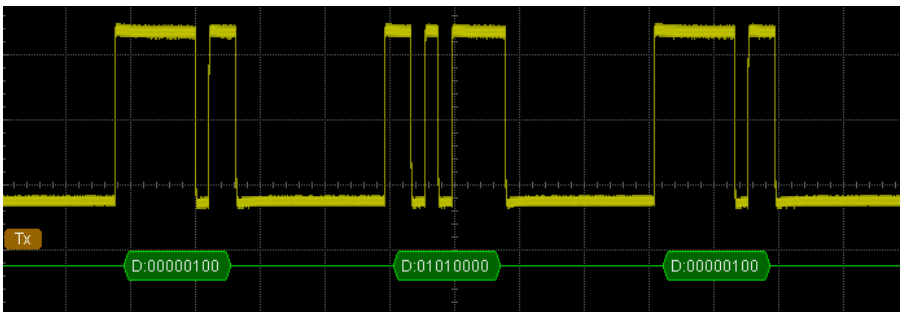
Check Error:

When a check bit error is detected during the decoding, a red error mark will be displayed. For example, when the transmitting terminal is set to none check and the decoder is set to odd check, the following check error occurs:



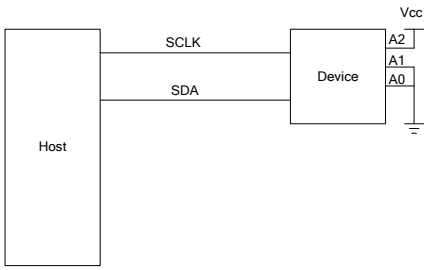
Wherein, there is an odd number (1) of 1 in the 8 bits data 00100000 and the check bit should be 0. The check bit detected on the TX is 1, thus a check error occurs.

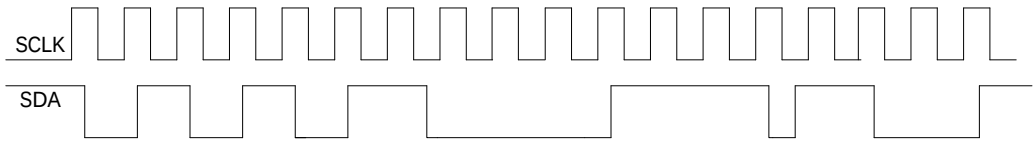
After the decoder is set to none check, the decoding shows no error.



I2C Decoding (Option)

I2C serial bus consists of the clock line (SCLK) and the data line (SDA).





SCLK: sample the SDA on the clock rising edge or falling edge.
SDA: denotes the data channel.

Press **Decode1** → **Decode** to select "I2C" and open the I2C decoding function menu.

1. SCLK Setting

Press **SCLK** to select any channel (CH1-CH4 or D0-D15) as the clock channel.
Press **SCLKThreshold** to set the threshold of the clock channel.

2. SDA Setting

Press **SDA** to select any channel (CH1-CH4 or D0-D15) as the data channel.
Press **SDAThreshold** to set the threshold of the data channel.

3. Display-related Setting

Press **Format** to set the display format of the bus to Hex, Decimal, Binary or ASCII.
Press **Offset** and use ↻ to adjust the vertical display position of the bus.
Press **BusStatus** to turn the bus display on or off.

4. Decoding Table

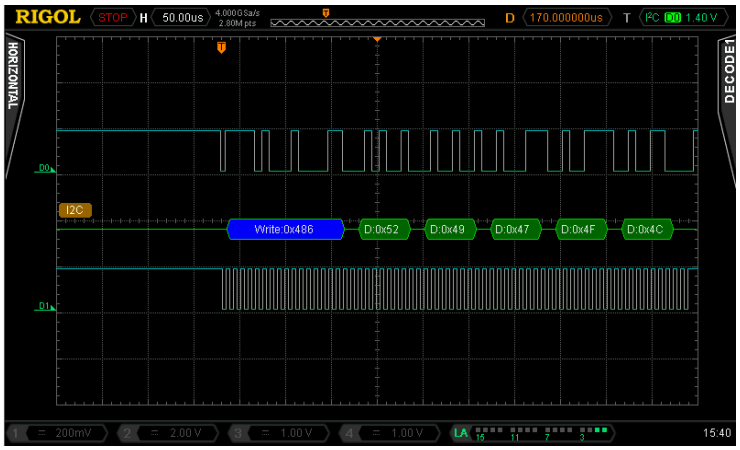
The decoding table displays the decoded data, the corresponding line number, time, data direction, ID and ACK information in table format.

Press **Event Table** → **Event Table** to select “ON” (note that this operation is only available when **BusStatus** is set to “ON”) to enter the decoding table interface as shown in the figure below. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.

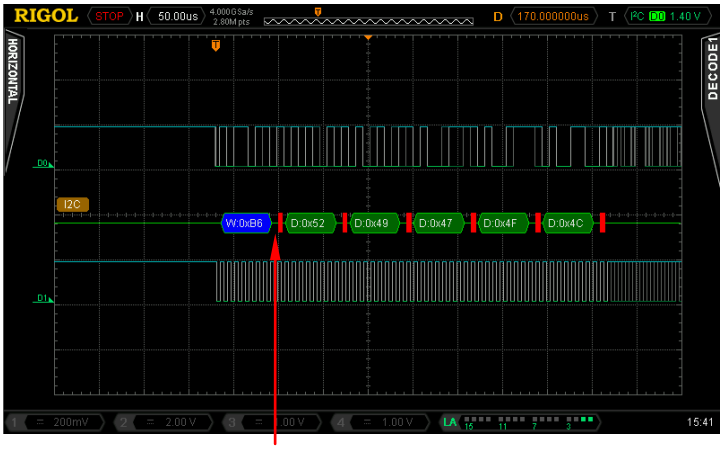
5. Error Expressions during Decoding

In I2C bus, the front part of each frame of data contains the address information and blue patches are used to represent address ID. In the ID, “Write” is used to represent writing address and “Read” is used to represent reading address.

Press **Include R/W** to select open and "R/W" will be the part of the address value in the **AddrBits**.



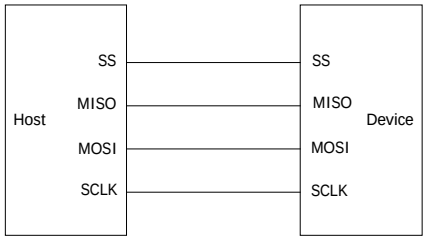
When the ACK (ACKnowledge Character) is not met, the red error marks as shown in the figure below will be displayed.

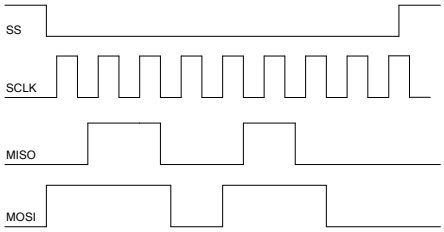


ACK=1

SPI Decoding (Option)

SPI serial bus consists of chip select line (SS), clock line (SCLK), MISO and MOSI.





Press **Decode1** → **Decode** to select "SPI" and open the SPI decoding function menu.

1. Decoding mode

Press **Mode** to set the desired decoding mode.

- CS: press **SS** to enter the chip select line setting interface. Press **Channel** to select D0-D15 or CH1-CH4 as the chip select channel. When CH1-CH4 is selected, press **Threshold** to set the threshold of the selected channel. When D0-D15 is selected, the threshold does not need to be set. Press **Polarity** to set the polarity of the chip select channel to (positive polarity) or (negative polarity). Note that the **SS** softkey is only valid when this mode is selected.
- TimeOut: press **TimeOut** to set the timeout time and the range available is from 1 ns to 1 s. Note that at this point, the **SS** softkey is invalid (not displayed).

2. SCLK Setting

Press **SCLK** to turn on the clock line setting interface.

- Press **Channel** to select any channel (CH1-CH4 or D0-D15) as the clock channel.
- Press **Slope** to set the instrument to sample MISO and MOSI on the rising edge or falling edge of SCLK.
- Press **Threshold** to set the threshold of the clock channel.

3. MISO Setting

Press **MISO** to enter the MISO data line setting interface.

- Press **Channel** to select any channel (CH1-CH4 or D0-D15) as the MISO data channel. When "None" is selected, this data line is not set.
- Press **Polarity** to set the polarity of the MISO data line to  (the high level is 1) or  (the low level is 1).
- Press **Threshold** to set the threshold of the MISO data channel.

4. MOSI Setting

Press **MOSI** to enter the MOSI data line setting interface.

- Press **Channel** to select any channel (CH1-CH4 or D0-D15) as the MOSI data channel. When "OFF" is selected, this data line is not set.
- Press **Polarity** to set the polarity of the MOSI data line  (the high level is 1) or  (the low level is 1).
- Press **Threshold** to set the threshold of the MOSI data channel.

5. Data Bits Setting

Press **Data Bits** to set the number of bits of each frame of data. The range available is from 4 to 32.

6. Endian Setting

Press **Endian** to select "LSB" or "MSB" and the default is "MSB".

7. Display-related Setting

Press **Format** to set the display format of the bus to Hex, Decimal, Binary or ASCII.

Press **Offset** and use  to adjust the vertical display position of the bus.

Press **BusStatus** to turn the bus display on or off.

8. Decoding Table

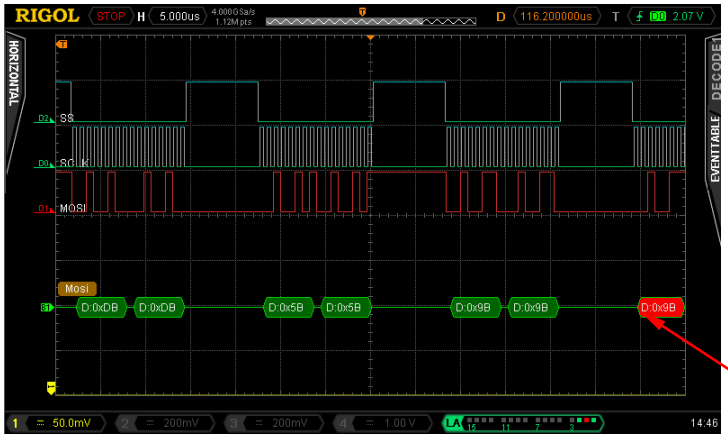
The decoding table displays the decoded data, the corresponding line number, time and error information on the MOSI or MISO data line in table format. This can be useful when observing longer data transimissions by presenting the data in a tabular format.

Press **Event Table** → **Event Table** to select "ON" (note that this operation is only available when **BusStatus** is set to "ON") to enter the decoding table

interface as shown in the figure below. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.

9. Error Expressions during Decoding

When the clock for a frame in SPI does not contain enough visible waveform to cover the data frame, the data will be filled with red patches. For example, when Data Bits is set to 7 and SCLK slope is set to rising edge, a decoding error will be generated.



CAN Decoding (Option)

Press **Decode1** → **Decode** and select "CAN" to open the CAN decoding function menu.

1. Source

Press **Source** and select any channel (CH1-CH4 or D0-D15) as the source channel.

2. Signal Type

Press **Signal Type** to select the desired signal type.

- **CAN_H:** the actual CAN_H bus signal.
- **CAN_L:** the actual CAN_L bus signal.
- **Differential:** the CAN differential bus signals connected to an analog channel using a differential probe. The positive lead of the probe connects CAN_H and the negative lead connects CAN_L.

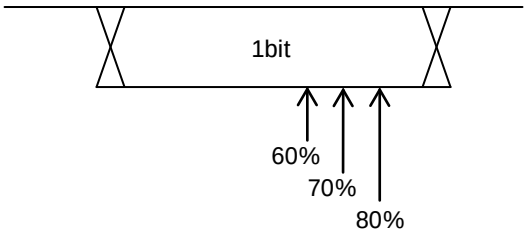
3. Baud

Press **Baud** to select a baud rate (100 kb/s, 125 kb/s, 250 kb/s, 400 kb/s, 500 kb/s, 800 kb/s, 1 Mb/s or User) that matches the CAN bus signal. When "User" is selected, press **Setup** and use ↻ to enter the desired rate, the range is from 10 kb/s to 1 Mb/s.

4. Sample Point

The Sample point is a point within a bit's time. The oscilloscope samples the bit level at this point. "Sample point" is represented by the percentage of "the time from the start of the bit's time to the sample point time" in the "bit's time".

Press **Sample Point** and use ↻ to adjust this parameter with a step of 1%. The range is from 5% to 95%.



5. Threshold

Refer to the introduction in "**Parallel Decoding**".

6. Display-related Setting

Press **Format** to set the bus display format to Hex, Decimal, Binary or ASCII.

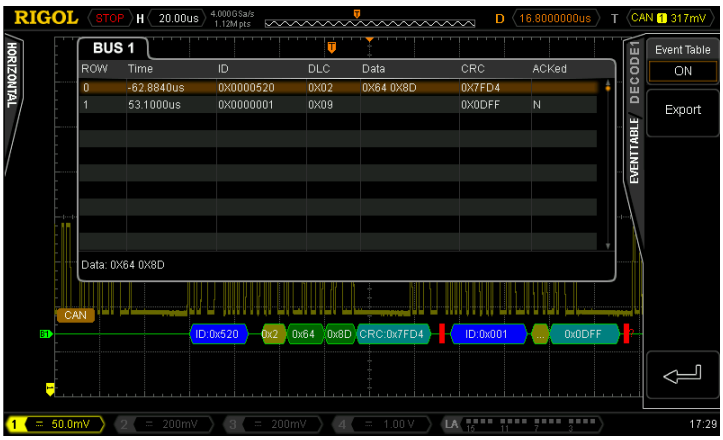
Press **Offset** and use ↻ to adjust the vertical display position of the bus.

Press **BusStatus** to enable or disable bus display.

7. Decoding Table

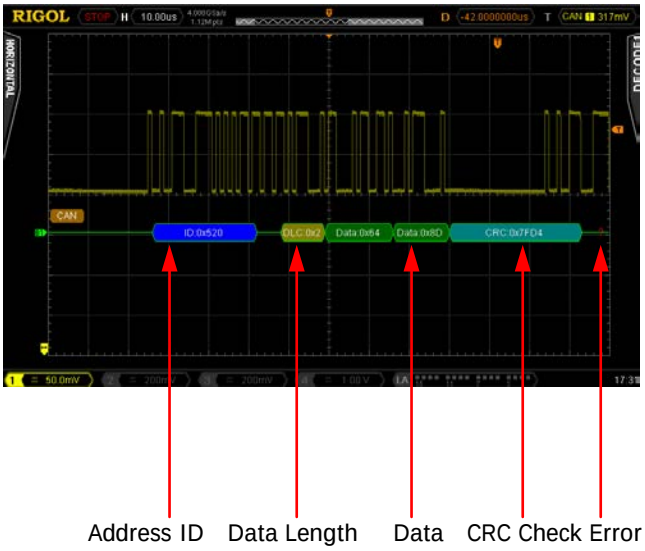
The decoding table displays the decoded data, the corresponding line number, time, frame ID, DLC, CRC and ACK information in table format.

Press **Event Table** → **Event Table** to select "ON" (note that this operation is only available when **BusStatus** is set to "ON") to enter the decoding table interface as shown in the figure below. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.



8. Decoded CAN Data Interpretation

- Frame ID: display as hex digits in blue.
- Data Length code (DLC): displayed as a chartreuse patch.
- Data Frame: displayed as green patches if data is successfully decoded. The frames appear as red patches if the data frame is lost.
- Cyclic Redundancy Check (CRC): displayed in a light blue patch when valid and red error mark is displayed when error occurs.



FlexRay Decoding (Option)

Press **Decode1** → **Decode** and select "FlexRay" to open the FlexRay decoding function menu.

1. Source

Press **Source** to select any channel (CH1-CH4 or D0-D15) as the signal source channel.

2. Signal Path

Press **Signal Path** to select the signal path (A or B) that matches the FlexRay bus signal.

3. Signal Type

Press **Signal Type** to select the type of signal that matches the FlexRay bus. The signal types available include BP, BM and TX/RX.

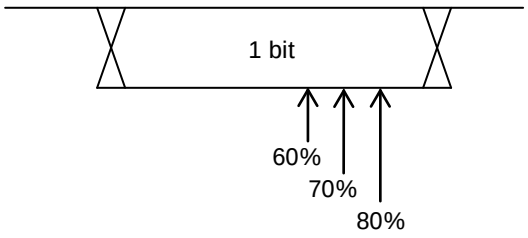
4. Baud

Press **Baud** to set the signal rate (2.5Mb/s, 5Mb/s or 10Mb/s) that matches the FlexRay bus signal.

5. Sample Point Position

The sample point is a point within a bit's time.The oscilloscope samples the bit level at this point. "Sample Point" is expressed by the percentage of "the time from the start of bit to the sample bit time" in "bit's time".

Press **Sample Point** and use ↻ to adjust this parameter with a step of 1% and the range is from 5% to 95%.



6. Threshold

Refer to the introduction in "**Parallel Decoding**".

7. Display-related Setting

Press **Format** to set the display format of the bus to Hex, Decimal, Binary or ASCII.

Press **Offset** and use ↻ to adjust the vertical display position of the bus.

Press **BusStatus** to enable or disable bus display.

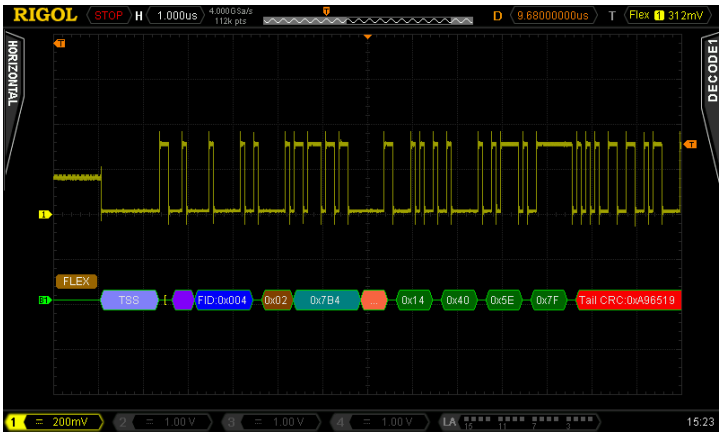
8. Decoding Table

The decoding table lists the decoded data, the corresponding line number, time and error information in table format. This can be useful when observing longer data transimissions by presenting the data in a tabular format.

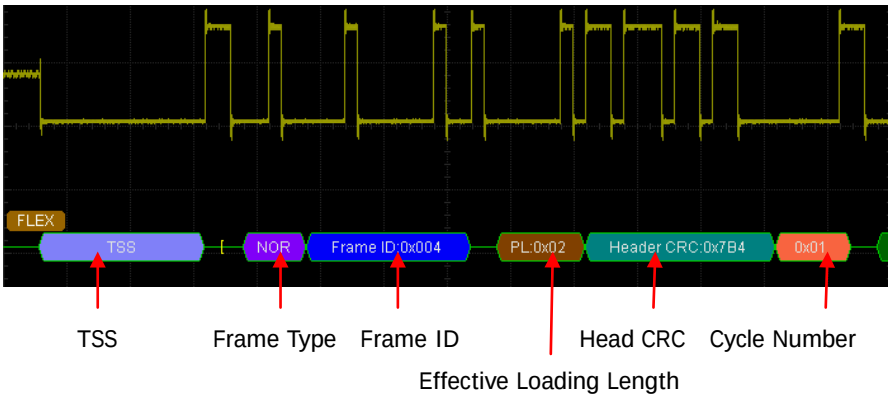
Press **Event Table** → **Event Table** to select "ON" (note that this operation is only available when **BusStatus** is set to "ON") to enter the decoding table interface as shown in the figure below. If a USB storage device is currently connected to the instrument, press **Export** to export the data table to the external USB storage device in CSV format.

9. Explanation of the Decoded FlexRay Frame Data

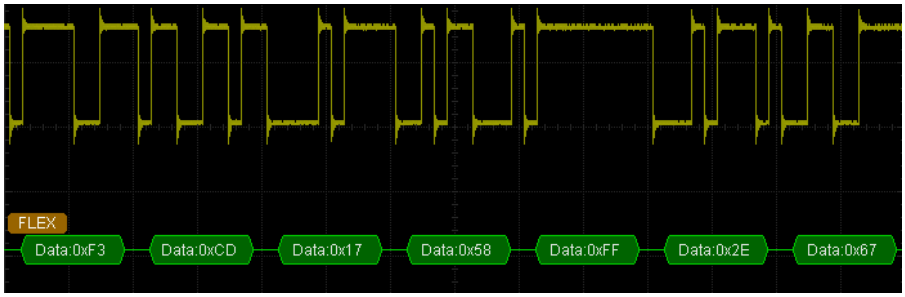
The decoded FlexRay frame data is as shown in the figure below.



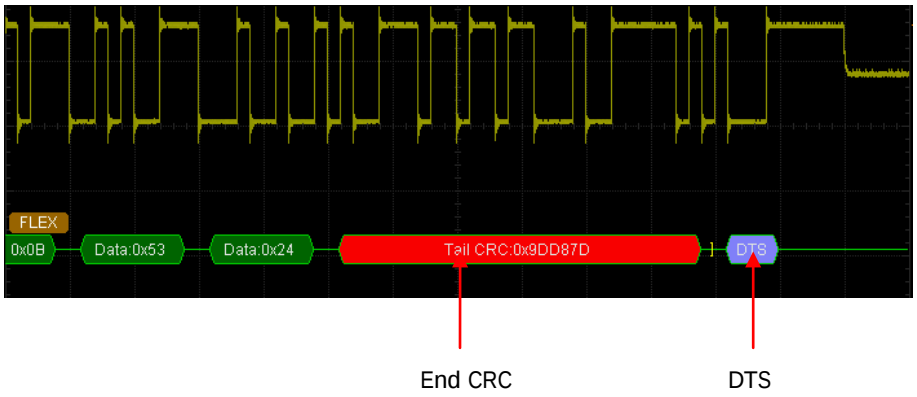
- TSS: transmission start sequence and is expressed by a light purple patch.
- Frame Type: FlexRay frame can be NORMAL, SYNC, SUP or NULL. The frame type in the figure above is “NOR” (namely NORMAL) and is expressed by a purple patch.
- Frame ID: decimal number and is expressed by a blue patch.
- Effective Loading Length: a decimal number and is expressed by a brown patch.
- Head CRC: a hexadecimal number and is expressed by a blue-green patch. When CRC is invalid, it is expressed by red patch.
- Cycle Number: a decimal number and is expressed by a pink patch.



- Data: displayed in the format (Hex, Decimal, Binary or ASCII) specified in **Format** and expressed by a green patch.



- End CRC: a hexadecimal number and is expressed by a red patch. When CRC is invalid, it is expressed by a red patch.
- DTS: dynamic end sequence and is expressed by a light purple patch.



RIGOL
Beyond Measure

Meeting Embedded Design Challenges with

Mixed Signal Oscilloscopes

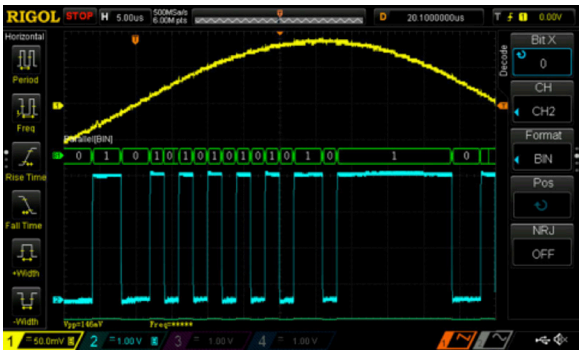


Figure 1: DAC output and input bit on a Rigol DS1074Z Oscilloscope

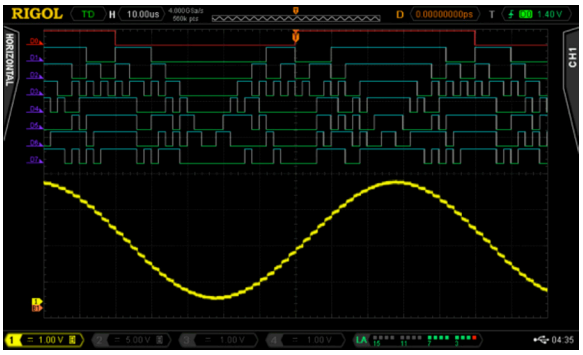


Figure 2: DAC output and 8 bit input bus on a Rigol MSO4034Z Oscilloscope

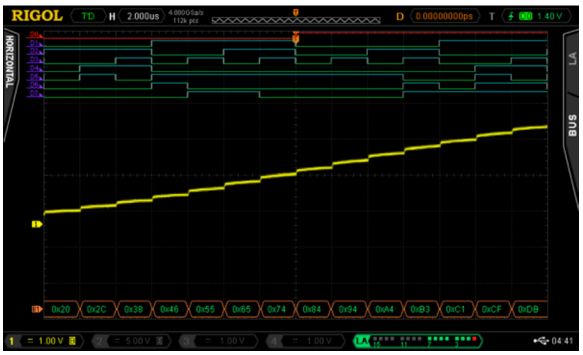


Figure 3: DAC output and 8 bit input bus with Hex decode on a Rigol MSO4034Z Oscilloscope

Introduction

Embedded design and especially design work utilizing low speed serial signaling is one of the fastest growing areas of digital electronics design. The need to communicate between modules, FPGAs, and processors within a wide array of consumer and industrial electronics is increasing at an astounding rate. Customized communication protocol and bus usage is critical to design efficiency and time to market, but comes with the risk of being sometimes difficult to analyze and debug. The most common sources and types of problems when using low speed serial data in an embedded application include timing, noise, signal quality, and data. We will recommend debug tips and features available in modern oscilloscopes that will make debugging these complex systems faster and easier.

Types of Errors

Timing

Timing is critical in any serial data system, but finding the system timing limitations related to components, transmission length, processing time, and other variables can be difficult. Let's start with a simple 16 bit DAC circuit. First, make sure you understand the data and timing specifications for the protocol in use. Does it sample data right on the clock edge? How far off can the clock and data be when we still expect good data? In other words: do we have a clock sync error budget defined? Once we understand these timing requirements then we can experimentally verify both the Tx and Rx hardware subsystems. Now we can analyze the system level timing delays and the overall accuracy of the conversions because we can make direct measurements of both the logic and analog channels in a time correlated fashion. We will also be able to simultaneously view the decoded bit patterns numerically and graphically all on an oscilloscope that comes in well below your budget limits.

To the left is a simple example of measuring a bit on channel 2 (blue) that is driving the DAC output that is creating the Sine wave on channel 1 (yellow).

Utilizing the parallel bus decoding **(figure 1)** we can get a quick look at the transitions of this single line. But this doesn't give us all the information we need since the DAC is utilizing a number of data lines to set its output level. Getting more complete data requires a different approach. Let's move all the DAC lines **(figure 2)** over to the MSO's digital inputs. Now we can see how the digital lines really coordinate with the DAC output.

To investigate further we can simplify the decoding to show Hex values **(figure 3)** and zoom in so we can view the decoded data.

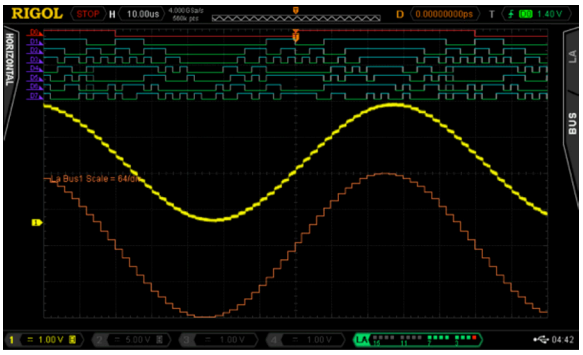


Figure 4: DAC output and 8 bit input bus with Plot of decode data on a Rigol MSO4034Z Oscilloscope

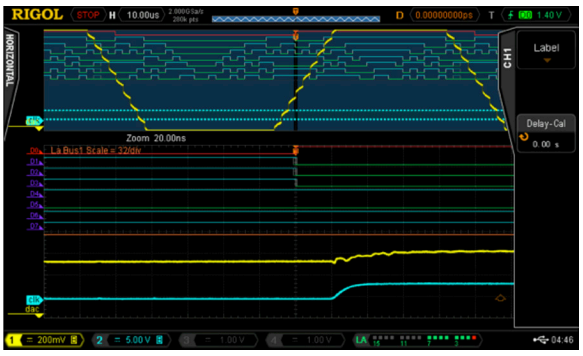


Figure 5: DAC output and 8 bit input bus zoomed in on a Rigol MSO4034Z Oscilloscope

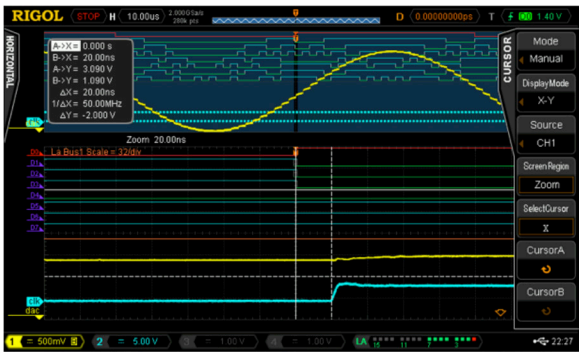


Figure 6: DAC output and 8 bit input bus zoomed in with cursors shown on a Rigol MSO4034Z Oscilloscope

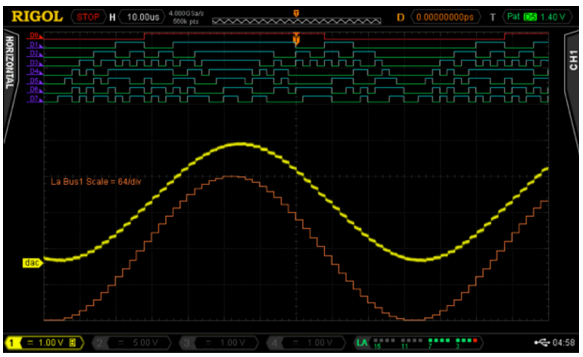


Figure 7: DAC output and 8 bit input bus triggered by a digital pattern on a Rigol MSO4034Z Oscilloscope

Additionally, if we want to view the changes in the bus graphically we can use a function in the Logic Analyzer bus menu called plot **(figure 4)**. This can graphically render the bit patterns for easy visual analysis. This is perfect for working with DACs and A2Ds because you can get immediate feedback if there is an issue with your encoding or decoding schemes.

Now we can use the zoom feature to clearly see the relationship between the bit and DAC transitions **(figure 5)**. For the zoom we have turned on analog channel 2 in blue that is on the DAC clock. Zoomed in by a factor of 500x from 10 usec per division to 20 nsec per division allows us to see that the bit transitions are occurring 20 nsec before the clock transition. The clock transitions in under 5 nsec and the DAC output starts changing in sync with the clock.

We can also utilize the scope cursors to make the timing in the transitions more clear and well defined **(figure 6)**.

Verifying timing in mixed signal systems can be made easier with the right tools. Select a modern scope with the correct set of channels and options to make sure you can easily view what you need on the display when you need it. From digital buses to processing delays, get the full picture of the device's operation and delve into details as needed to verify timing issues on your device.

We can also trigger on the digital patterns instead of the analog signal **(figure 7**). Triggering on a digital pattern can be critical for debugging when there is a problem. There isn't always a good way to track events from the analog side of a system. When using a digital trigger method make sure to set the additional trigger parameters. These may include start bits or even address and data for some protocols. Even for a simple parallel bus like this you need to define and arrange the channels in the bus for the results to be easiest to interpret.

Accurate timing of Low Speed Serial signals is critical to system stability. Therefore, making sure your measurement tools are up to the task of precise and easy triggering, monitoring, and analyzing of your waveforms is vital to improved R&D efficiency and ultimately time to market.

Noise

One of the most common issues in correct serial data measurements is the handling of system noise. Noise in these measurements can come from a number of sources including poor grounding, bandwidth issues, crosstalk, electromagnetic immunity (EMI) problems. Sometimes the problem is in the device, but improved probing and measurement techniques can also improve the results significantly without changing the device under test. A good first step is always to make sure we are using best measurement practices.

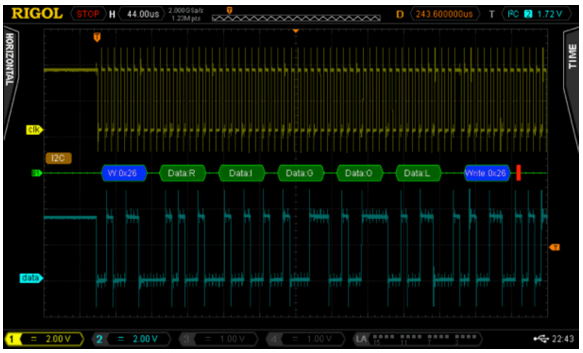


Figure 8: I2C clock and data with noise from poor grounding using a Rigol MSO4034Z Oscilloscope

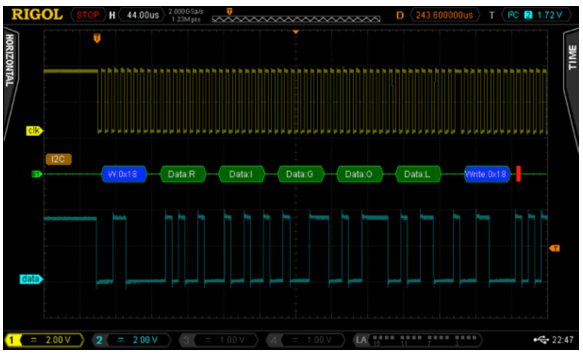


Figure 9: I2C clock and data with ground noise improved using a Rigol MSO4034Z Oscilloscope

Figure 10: RP1100D 100 MHz Differential Probe

Figure 11: RP7150 1.5 GHz Differential Probe

Here is a decoded I2C bus segment using a 4000 series oscilloscope **(figure 8)**. In the first example we have extremely poor grounding on our probes. Because the scope's ground is tied directly to power ground signals that need to float or simply use a different or noisy ground plane can cause results like this. It is also possible for high current draw through ground in local power to create ground loops that can cause noise to be injected in your system.

We solve these problems in order from easy to difficult. First, we can look at our probe connections. Normally, we would use the alligator clip ground strap that connects on the probe to make a ground connection. Assuming we are doing that correctly and still having a problem we may need to use the ground spring instead. The ground spring connects closer to the probe tip and significantly reduces the loop area of the connection. This can significantly improve noise and signal quality **(figure 9)** especially for high speed signals or signals sensitive to capacitance or coupled voltages. All Rigol probes come with both the standard ground strap and the ground spring for these types of measurements.

If ground noise is still an issue try isolating your device from ground. The scope operates best grounded to AC power ground via the plug. If the rest of the device or system being tested can be isolated from ground this eliminates ground loops. If ground noise is still an issue you may consider a differential probe like the RP1100D **(figure 10)** which enables measurements without reference to ground on the scope. Differential measurements may be the only way to clearly view some low speed serial data such as a LVDS bus (Low Voltage Differential Signaling). Buses like this purposely move the reference line to maximize bandwidth and increase communication distances, but it may require true differential probing or the use of multiple channels of your scope together to view the signal correctly. Rigol has several different probe types for these measurements including the RP1000D series differential probes **(**typically used for high voltage floating applications and the RP7150 1.5 GHz differential probe **(figure 11)** or high speed data applications.

Now that we have improved our signal to noise ratio by decreasing noise injected from the ground we can turn our attention to bandwidth filtering. High frequency noise can also enter your measurements via channel to channel cross talk or other high frequency sources nearby or within your device. One way to address this is to utilize the channel bandwidth limits **(figure 12)**. Every Rigol scope channel can limit the bandwidth to the ADC. A 20 MHz limit is pretty standard.
Some scopes will have higher options as well. Some scopes can even digitally set notch filters to limit other noise sources.

Additionally, there are a few acquisition mode and triggering settings that can improve performance in the face of noise. Many trigger types have a menu item allowing you to turn on noise rejection for the triggering scheme. The 1000Z series even includes HFR and LFR (high and low frequency rejection) as options in how to couple the signal triggered on. All UltraVision Rigol scopes have an acquisition mode called High Res or high resolution **(figure13)** . This feature uses extra oversampling that is being done behind the scenes on many

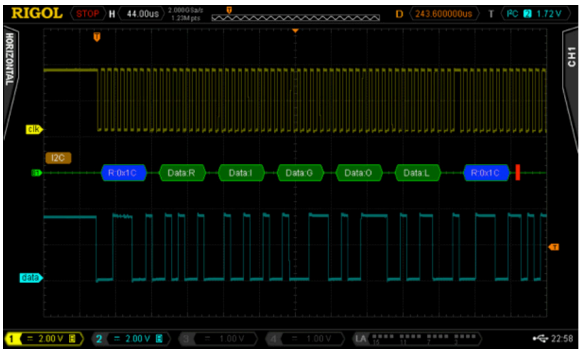


Figure 12: I2C clock and data with reduced using bandwidth limit on a Rigol MSO4034Z Oscilloscope

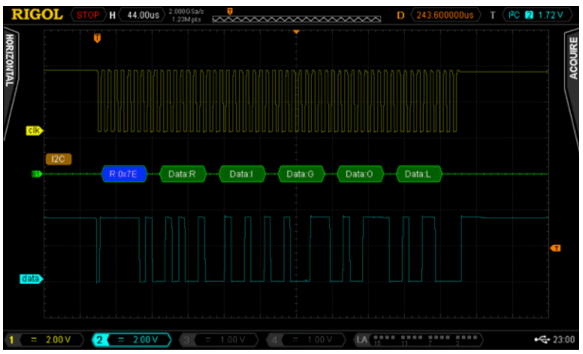


Figure 13: I2C clock and data using high resolution mode on a Rigol MSO4034Z Oscilloscope

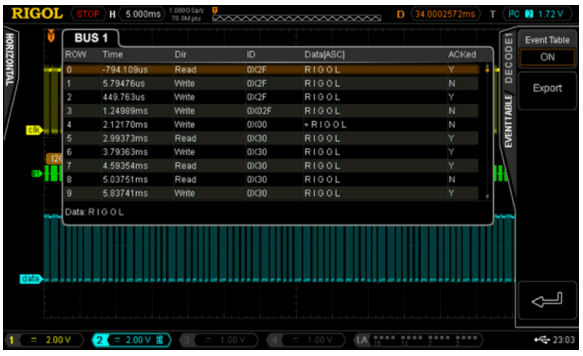


Figure 14: I2C clock and data event table view using a Rigol MSO4034Z Oscilloscope

measurements to provide an average that results in less noise. This is best to use if you are able to set the sampling to take at least 200 samples per time division. This will average rather than reject high frequency signals, so be sure to understand your potential error sources and how they may interact with your measurement setup. Finally, the 1000Z series scope also has a NRJ (noise reject) feature directly within the decoding menu. This removes noise that appears in bursts and can be set in time rather than frequency. To further isolate and locate sources of noise within your system you may want to focus on EMC or EMI related issues. To further investigate these error sources please download the EMC Precompliance app note for use with the Rigol DSA815 Spectrum Analyzer at ***http://www.rigolna.com/EMC***

Noise is always a concern when working with low speed serial signals. By definition these signals continue to go to higher speeds, more advanced encoding, farther transmission distances, and lower voltage and power levels. All of these trends make hardware more susceptible to noise. Making careful measurements that limit or eliminate adding noise to our system then enables us to focus on noise in the system that may still cause long term design issues.

Signal Quality

Monitoring and improving the quality of low speed serial signals is a critical part of the debugging process. Issues like impedance mismatches, bandwidth, and loading errors can all effect the quality of signals even when noise isn't present. Now that we are looking more closely at the exact nature of these signals it is important to verify the way we are using our oscilloscope for these tests. For signal quality tests we will be using the analog channels because they provide the best look at what is actually occurring with our signals, but we will still be doing some decoding. This requires some additional forethought. To clearly see data transitions we should definitely use a sampling rate that is as high as possible. Sampling at 5x the bit rate of the digital bus should be considered the minimum because of the high frequency components that we need to visualize. Sampling at 10 times the bit rate should enable us to see any issues. But when we decode the signal the scope likely uses a subset of the full memory data to handle the decode analysis. This can be important because you don't necessarily want decode being done at too high of a rate. That can mask problems you will find when a more nominal receiver is used to decode the data. On Rigol scopes the decode is done on 1 Mpts of memory spread across the acquisition. By setting the memory depth and the time per division the user can determine whether they want the decode to be done directly from the analog points or from a subset. Decoding is also shown across the display region. To capture more decoded bytes than you can view on the display use the event table function **(figure 14).** You can also export the table results to a text file from the event table menu for record keeping or offline timing analysis.

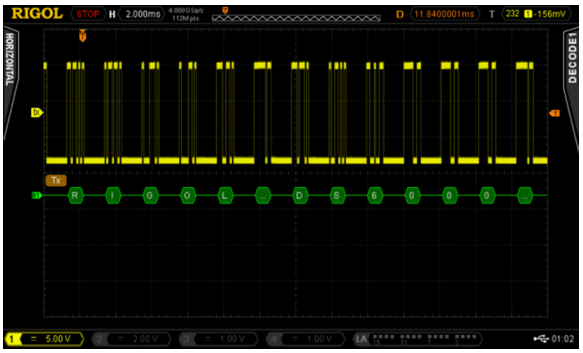


Figure 15: RS232 optimal decoded data view settings using a Rigol MSO4034Z Oscilloscope

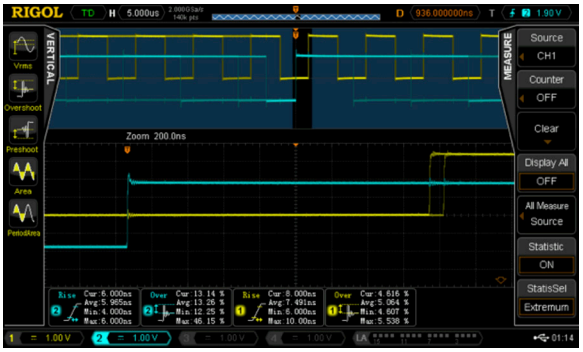


Figure 16: I2C clock and data min/max signal measurements using a Rigol MSO4034Z Oscilloscope

Here is an example table of how the memory depth, time per division, and sample rate effect the actual decode sample rate on a DS2000 Series oscilloscope. Based on your serial data speeds and the receiver you will eventually use to collect the serial data you can optimize your serial decode rate.

LA Saple Rate (per s)	Memory Depth (pts)	Analog Sample Rate (per s)	Time Per Horizontal Division of Display	Decode Sample Rate on Analog Channel (per s)	Ideal for Decoding this Speed (bits per s) based on 5x oversampling on Analog Channels
500,000	56,000,000	2,000,000	1 sec	35,714	7K
500,000	14,000,000	1,000,000	1 sec	71,429	14K
50,000	1,400,000	100,000	1 sec	71,429	14K
5,000	140,000	10,000	1 sec	10,000	2K
500	14,000	1,000	1 sec	1,000	200
500,000,000	56,000,000	2,000,000,000	1 msec	35,714,286	7M
500,000,000	14,000,000	1,000,000,000	1 msec	71,428,571	14M
25,000,000	1,400,000	100,000,000	1 msec	71,428,571	14M
5,000,000	140,000	10,000,000	1 msec	10,000,000	2M
500,000	14,000	1,000,000	1 msec	1,000,000	200K
500,000,000	56,000,000	2,000,000,000	1 usec	35,714,286	7M
500,000,000	14,000,000	2,000,000,000	1 usec	142,857,143	28M
500,000,000	1,400,000	2,000,000,000	1 usec	1,428,571,429	285M
500,000,000	140,000	2,000,000,000	1 usec	2,000,000,000	400M
500,000,000	14,000	2,000,000,000	1 usec	2,000,000,000	400M

Table of memory depth, time per division, sample rate, decode sample rate, and optinal decode speed based on sample rate on a Rigol 2000 Series Oscilloscope.

Now that we have set and verified our sampling times for best analog and decoding results, we also want to set our display up for optimal triggering conditions. When triggering on the rising edge of an analog signal make sure to keep the trigger level at least 1 division away from the signal low state. This separation allows for consistent triggering action without any false triggers. When visualizing digital signals with the analog channels use more screen real estate when possible. Using about 2 vertical divisions and about ½ to 1 horizontal division per decode character will allow you to see any major overshoot or impedance issues as well as some of the other types of error we will be looking at. Here is the setup **(figure 15)** I prefer to monitor decoded data on a bus like RS232.

On a more complex bus like I2C we view both clock and data lines on the screen. The timing correlation between multiline buses is, of course, vital to successful decoding. Making critical measurements on the screen like risetime and overshoot for each line makes reliability tests simple to setup. We can view the measurements in max/min or using standard deviation notation for more advanced statistical testing. These measurements can be accessed from the left side menus on all Rigol MSOs **(figure 16)**.

Figure 17: I2C zoom in on data risetime measurement using a Rigol MSO4034Z Oscilloscope

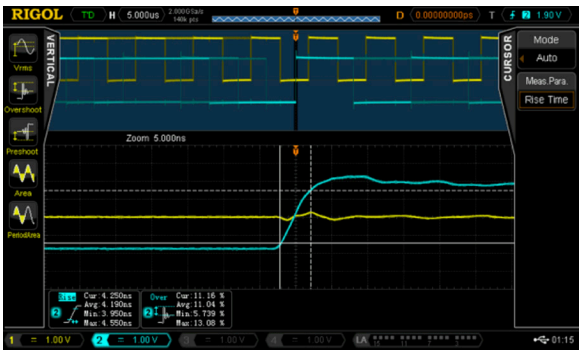


Figure 18: I2C data measurements with auto cursor visualization using a Rigol MSO4034Z Oscilloscope

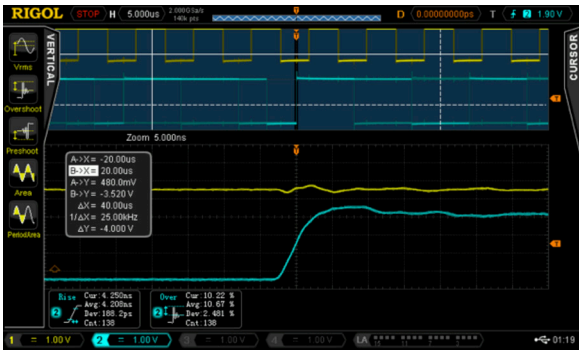


Figure 19: I2C data measurements with standard deviation using a Rigol MSO4034Z Oscilloscope

Figure 20: RS232 optimal decoded data view settings using a Rigol MSO4034Z Oscilloscope

In addition to the risetime and overshoot for the 2 serial bus lines you can also see the jitter in the clock when compared to the data transition. This test device appears to have the clock transition occur about 1.6 microseconds after the data transition. There appears to be about 100 nanoseconds of jitter on the clock when viewed, as above, in reference to the triggering data transition. In **(figure 17)** we zoom in on the data transition to get a more accurate measurement of the risetime and overshoot.

An additional measurement capability combines cursors and automated measurements. Using the auto cursor setting in the cursor menu the cursor is automatically shown on the screen in the positions being used to make the latest measurement **(figure 18)**. This is a great technique for visualizing overshoot or risetime.

Signal quality encompasses many of the types of issues you find on LSS buses. Efficient debug means making the most of your embedded analysis capabilities to find signal discrepancies that can lead you to design changes as early as possible in your design process. A mixed signal oscilloscope is the perfect tool for measuring signal quality **(figure 19)** from risetime variance to ASCII packet data.

Data

The key to any Low Speed Serial application is the ability to quickly and easily look at the data being transmitted. This means adding the capability to do embedded decoding on your oscilloscope. Decoding affects both the triggering and display on the scope. It adds a decoded bus display to the instrument's screen. You can decode values as ASCII or just as hex, octal, or binary data depending on what you want to look at. You can also now trigger on these values to make sure you are looking at the packets of most interest to you. There are several methods to make sure you are getting the quickest and easiest view of exactly the data you need. If you just want to look at a single packet then trigger on a value in the packet and set the time per division to display it like **(figure 20)**.

If you are interested in timing between packets or evaluating more than one or two consecutive packets of data, use the event table mode **(figure 21)** to generate a list of data packets on a wider timebase.

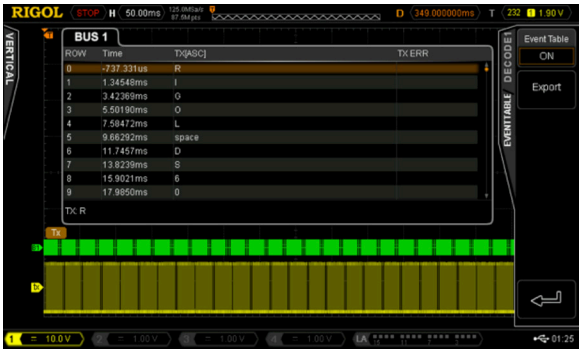


Figure 21: IRS232 event table capture of multiple transmissions using a Rigol MSO4034Z Oscilloscope

Figure 22: RS232 packet view using zoom mode on a Rigol MSO4034Z Oscilloscope

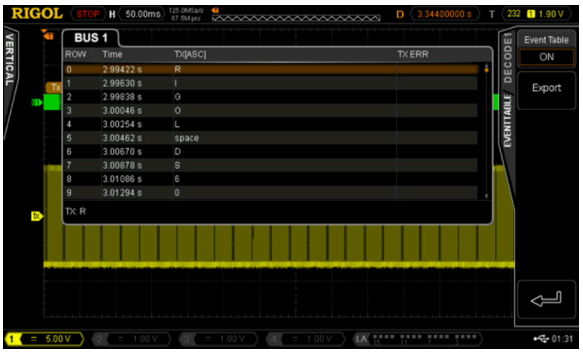


Figure 23: RS232 delayed view of transmissions 3 seconds after the trigger point using a Rigol MSO4034Z Oscilloscope

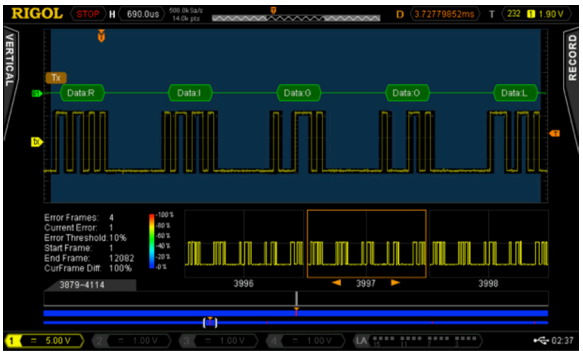


Figure 24: RS232 record mode with pass/fail analysis enabled using a Rigol MSO4034Z Oscilloscope

Using this method you can always then utilize the zoom function to see that signals and data **(figure 22)** from an individual packet in that series.

Setting the trigger offset to view packets listed on the event table with a timestamp is a great way to view specific packets that you aren't triggering on but need to investigate further **(figure 23)**.

Lastly, to view how decoded segments differ over time or compare between triggered events when other signals might be affecting the results the best analysis method is often to use record mode. Rigol's record mode enables you to capture thousands of frames around a trigger event. Once captured, you can use pass/fail or a trace difference analysis mode to visualize changes from frame to frame **(figure 24)**.

These recordings can be stored and played back as a movie as well, but the analysis features let you search for failures or outliers while also viewing decoded data for comparison **(figure 25)**.

Data errors as well as the debugging process are always closely tied to the protocol and its specifications. To be efficient with your test equipment make sure you are utilizing the best analysis method to easily view the data you need to see without extraneous results getting in your way.

Keys to Look Out For

Proper Oversampling & Bandwidth

As discussed, proper sampling is critical to making correct measurements as well as completely debugging your low speed serial interface. A good rule of thumb for analog signals is 5x the bandwidth of the signal you want to measure. This limits your risetime error to about 2%. To view the best detail on high frequency signal components set up your scope to achieve 5-10X over sampling as well. When digital signals this means sampling 5 times in the width of one bit. When sampling on digital lines or for sampling that will be used for decoding oversampling is less important, but set up your measurement device so it is as similar as the LSS receiver you will ultimately be using. This gives you the best chance to focus on material errors that will cause problems down the road.

Grounding, Noise, and Differential Signaling

Proper probing and understanding the use of differential vs. ground referenced signals is important to debugging. If your data lines are not ground referenced make sure to understand the impact of ground loops and ground coupled noise on your measurements. Use proper probe techniques and advanced noise cancelling features on the scope to limit noise sources. If necessary, add differential probes to your measurement system to improve measurement quality.

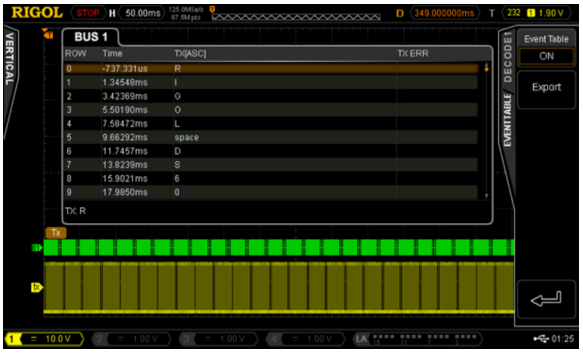


Figure 25: RS232 transmission record mode in playback using a Rigol MSO4034Z Oscilloscope

How Best to View Low Speed Serial Signals

There are a number of methods for analyzing, viewing, and evaluating LSS bus activity on a modern oscilloscope. The best way differs depending on whether you want to look at a single bit transition for noise, speed, or synchronization; whether you want to look at a complete packet of data; or if you want to compare packets and packet timing over a longer time period. Make sure your bench tools allow you to see everything you need and familiarize yourself with features like zoom, record mode, event tables, deep memory, and automatic measurements as well as how they interact and how best to transition between them when considering your test plan. Ideally, an oscilloscope empowers you to view all the results you need and quickly switch modes to acquire additional information.

Conclusions

Embedded design and debugging of digital data is a growing test requirement in a broad range of consumer and industrial applications. Having the right mixed signal oscilloscope can make viewing, analyzing, and resolving issues including timing, noise, signal quality, and data easier and faster. This improves engineering efficiency and time to market. Rigol's line of UltraVision enabled oscilloscopes that include mixed signal options from 70 to 500 MHz as well as standard or optional capabilities for the methods and measurements discussed here are powerful benchtop instruments that provide uncompromising performance at unprecedented value.

For more information on our oscilloscopes please go to **rigolna.com** or contact us directly at *applications@rigoltech.com* or call us toll free at 877-4-RIGOL-1.

Rigol Technologies USA

10200 SW Allen Blvd, Suite C
Beaverton, OR 97005
877.474.4651