TELEDYNE LECROY
Everywhereyoulook™

Trigger, Decode, Measure/Graph, Eye Diagrams

for Automotive Protocols:
CAN and **CAN FD**
LIN
FlexRay

Contents

About This Manual

Teledyne LeCroy offers a wide array of toolsets for decoding and debugging serial data streams. These toolsets may be purchased as optional software packages, or are provided standard with some oscilloscopes.

This manual explains the basic procedures for using serial data decoder and trigger software options. There are also sections pertaining to the measure and graphing capabilities and eye diagram tests.

While some of the images in this manual may not exactly match what is on your oscilloscope display—or may show an example taken from another standard—be assured that the functionality is identical, as much functionality is shared. Product-specific exceptions will be noted in the text.

It is assumed that you have a basic understanding of the serial data physical and protocol layer specifications, and a basic understanding of how to use an oscilloscope, specifically the Teledyne LeCroy oscilloscope on which the option is installed. Only features directly related to serial data triggering and decoding are explained in this manual.

Teledyne LeCroy is constantly expanding coverage of serial data standards and updating software. Some capabilities described in this documentation may only be available with the latest version of our firmware. You can download the free firmware update from:

About the Options

Teledyne LeCroy decoders apply software algorithms to extract serial data information from physical layer waveforms measured on your oscilloscope. The extracted information is displayed over the actual physical layer waveforms, color-coded to provide fast, intuitive understanding of the relationship between message frames and other, time synchronous events.

Trigger *and* decode (-TD) options enable you to trigger the oscilloscope acquisition upon finding specific message frames, data patterns, or errors in serial data streams. Conditional filtering at different levels enables you to target the trigger to a single message or a range of matching data.

The installation of any -DME or -TDME option adds a set of measurements designed for serial data analysis and protocol-specific eye diagram tests to the standard trigger and decoder capabilities. See Measuring for instructions on using the measure and graphing capabilities. See Eye Diagram Tests for instructions on using the eye diagram tests.

About the CANbus and CAN FDbus Options

CAN is a vehicle bus designed specifically for automotive applications, but it is now found in other applications, as well. The various CAN specifications are maintained by the International Organization for Standards (ISO) and can be obtained at www.iso.org.

CANbus TD supports decoding standard CAN (11-bit and extended 29-bit).

CAN FDbus TD supports decoding standard CAN as well as the advanced features of CAN FD: bitrate increase for the Data segment and the extension of the possible Data payload from 8 bytes to 64 bytes. It also supports ISO and non-ISO frames.

Both options enable triggering on CAN messages, Frame IDs, serial data patterns, or errors. Frame-level and value-level filtering enable you to target the trigger to a certain type of CAN message, a single frame ID or data pattern, or to a range of data.

The TDME variants of these options add a set of CAN-specific measurement parameters, plots, and eye diagram tests.

The TDME SYMBOLIC variants of these options add symbolic decoding, triggering, and measuring based on user-defined .DBC or .ARXML (AutoSAR) symbol files.

About the LINbus Options

LIN is a low-cost master/slave system designed for implementation in vehicles, typically in what is commonly referred to as body electronics. The LIN specification is maintained by the International Standards Organization (ISO) and can be obtained at www.iso.org.

The LINbus TD and TDME options decode and enable triggering on specific LIN frames, data patterns, or errors. Other features include:

- Ability to decode LIN Version 1.3, 2.x, and SAE J2602 formats, even when LINbus traffic contains mixed versions.
- Triggering on Checksum, Header Parity, and Sync Byte Errors.

About the FlexRaybus Options

FlexRay is an automotive network communications protocol developed by the FlexRay Consortium to govern on-board automotive computing. The FlexRay consortium disbanded in 2009, and FlexRay is now defined in ISO standards 17458-1 to 17458-5.

The FlexRaybus TD and TDME options decode FlexRay protocol version 2.1 at 10 Mb/s, 5 Mb/s or 2.5 Mb/s. Features include:

- Ability to trigger on TSS, Frame characteristics, protocol Errors, or various Symbols (such as wakeup patterns).
- Frame triggers may be set on individual Frame IDs or ranges and be further conditionalized using Cycle Counts or Frame Qualifiers.

The FlexRay TDP and TDMP options also provide special physical layer eye diagram tests as specified by the FlexRay standard and physical-layer measurement parameters.

Serial Decode

The algorithms described here at a high level are used by all Teledyne LeCroy serial decoders sold for oscilloscopes. They differ slightly between serial data signals that have a clock embedded in data and those with separate clock and data signals.

Bit-level Decoding

The first software algorithm examines the embedded clock for each message based on a default or user-specified vertical threshold level. Once the clock signal is extracted or known, the algorithm examines the corresponding data signal at the predetermined vertical level to determine whether a data bit is high or low. The default vertical level is set to 50%, as determined from a measurement of peak amplitude of the signals acquired by the oscilloscope. For most decoders, it can also be set to an absolute voltage level, if desired. The algorithm intelligently applies a hysteresis to the rising and falling edge of the serial data signal to minimize the chance of perturbations or ringing on the edge affecting the data bit decoding.

Note: Although the decoding algorithm is based on a clock extraction software algorithm using a vertical level, the results returned are the same as those from a traditional protocol analyzer using sampling point-based decode.

Logical Decoding

After determining individual data bit values, another algorithm performs a decoding of the serial data message after separation of the underlying data bits into logical groups specific to the protocol (Header/ ID, Address Labels, Data Length Codes, Data, CRC, Parity Bits, Start Bits, Stop Bits, Delimiters, Idle Segments, etc.).

Message Decoding

Finally, another algorithm applies a color overlay with annotations to the decoded waveform to mark the transitions in the signal. Decoded message data is displayed in tabular form below the grid. Various compaction schemes are utilized to show the data during a long acquisition (many hundreds or thousands of serial data messages) or a short acquisition (one serial data message acquisition). In the case of the longest acquisition, only the most important information is highlighted, whereas in the case of the shortest acquisition, all information is displayed with additional highlighting of the complete message frame.

User Interaction

Your interaction with the software in many ways mirrors the order of the algorithms. You will:

- Assign a protocol/ encoding scheme, an input source, and a clock source (if necessary) to one of the four decoder panels using the Serial Data and Decode Setup dialogs.
- Complete the remaining dialogs required by the protocol/ encoding scheme.
- Work with the decoded waveform, result table, and measurements to analyze the decoding.

Decoding Workflow

We recommend the following workflow for effective decoding:

1. Connect your data and strobe/clock lines (if used) to the oscilloscope.
2. Set up the decoder using the lowest level decoding mode available (e.g., Bits or Nibbles).
3. Acquire a sufficient burst of relevant data. The data burst should be reasonably well centered on screen, in both directions, with generous idle segments on both sides.

   Note: See Failure to Decode for more information about the required acquisition settings. A burst might contain at most 100000 transitions, or 32000 bits/1000 words, whichever occurs first. This is more a safety limit for software engineering reasons than a limit based on any protocol. We recommend starting with much smaller bursts.

4. Stop the acquisition, then run the decoder.
5. Use the various decoder tools to verify that transitions are being correctly decoded. Tune the decoder settings as needed.
6. Once you know you are correctly decoding transitions in one mode, continue making small acquisitions of five to eight bursts and running the decoder in higher level modes (e.g., Words). The decoder settings you verify on a few bursts will be reused when handling many packets.
7. Run the decoder on acquisitions of the desired length.

When you are satisfied the decoder is working properly, you can disable/enable the decoder as desired without having to repeat this set up and tuning process, provided the basic signal characteristics do not change.

Decoder Set Up

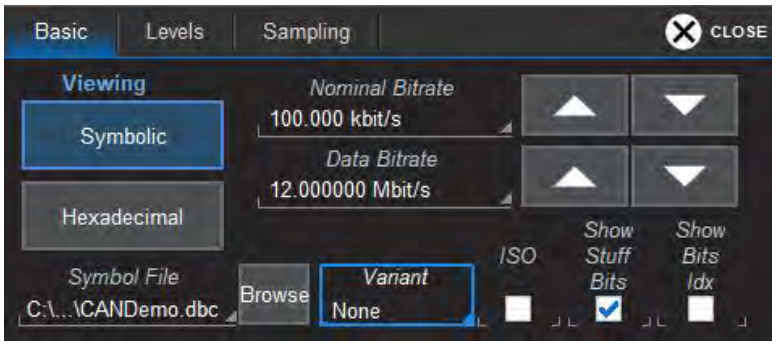
Use the Decode Setup dialog and its protocol-related subdialogs to preset decoders for future use. Each decoder can use different protocols and data sources, or have other variations, giving you maximum flexibility to compare different signals or view the same signal from multiple perspectives.

1. Touch the **Front Panel Serial Decode button** (if available on your oscilloscope), or choose **Analysis > Serial Decode** from the oscilloscope menu bar. Open the **Decode Setup** dialog.
2. From the buttons at the left, select the **Decode #** to set up.
3. Select the **data source (Src 1)** to be decoded and the **Protocol** to decode.
4. If required by the protocol, also select the **Strobe** or **Clock** source. (These controls will simply not appear if not relevant.)
5. Define the bit- and protocol-level decoding on the subdialogs next to the Decode Setup dialog.

Tip: After completing setup for one decoder, you can quickly start setup for the other decoders by using the buttons at the left of the Decode Setup dialog to change the Decode # .

CAN/ CAN FD Decoder Settings

Basic Subdialog



Enter the **Nominal Bitrate** (Arbitration bitrate) of the bus to which you are connected as precisely as you know it. The value should be correct within 5%. A mismatched bit rate will cause various confusing side effects on the decoding, so it is best to take time to correctly adjust this fundamental value. Use the Up and Down arrow buttons to scroll a list of standard bitrates, or touch the field to enter the value using the Adjust knob or the Virtual Keypad.

Tip: If unsure of the bitrate, apply the Bit Rate measurement parameter to a short acquisition.

Select the **CAN Variant** in use: GM LAN, J1939, J1939+1, J1939+2 or None (all others).

Check **Show Stuff Bits** to display the stuff bits on each CAN message frame.

Check **Show Bits Idx** to display a bit index for each bit, which can assist with analysis or debugging of the decode. The bit index starts at 0 at the beginning of the message and increases monotonously to the end of the message. Note that the bit index skips the stuffbits.

CAN FD decoder, also:

Enter the **Data Bitrate**. The same precautions apply as for the Nominal Bitrate.

Check **ISO** if your signal contains ISO frames.

Symbolic Decoding

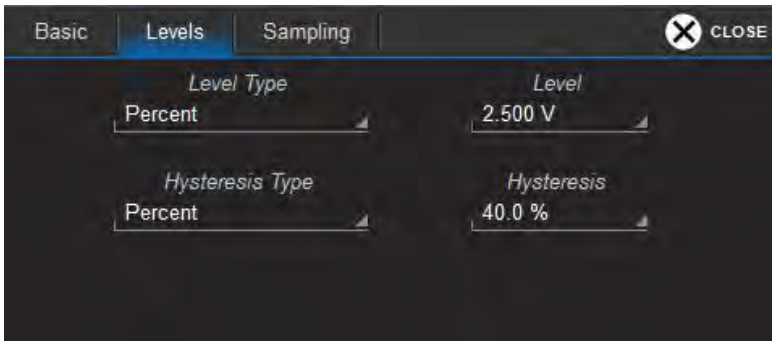
If you have purchased one of the SYMBOLIC options, these additional controls related to symbolic decoding will appear on the decoder set up dialogs:

- Choose to view the decoder results with just **Hexadecimal** or with **Symbolic** translation.
- If Symbolic, select the **Symbol File** to use.

Note: CAN and CAN FD decoders and triggers support standard .DBC and .ARXML (AutoSAR) symbol file types.

The default symbol file is located in D:\Applications\<protocol>. You may copy your own symbol file to this directory and browse to it for easy selection. Symbol files must have the .dbc or .arxml extension. We recommend that you copy and modify the default .dbc file to ensure the symbol file is properly formatted.

Levels Subdialog



Enter the vertical **Level** used to determine the edge crossings of the signal. This value will be used to determine the bit-level decoding.

Optionally, enter a **Hysteresis** band value. Hysteresis represents the amount the signal may rise or fall from the crossing Level without affecting the bit transition.

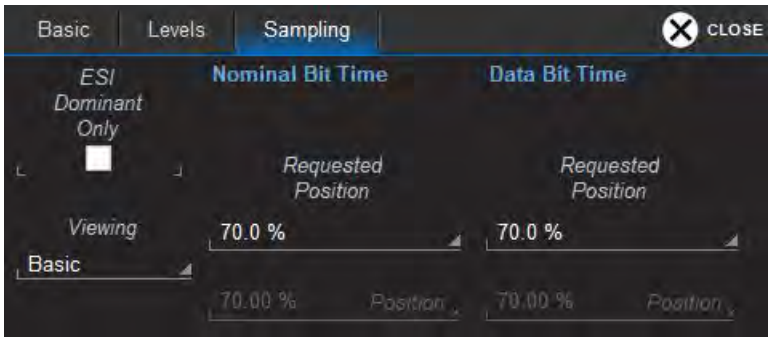
Sampling Subdialog

This dialog appears only when you are setting up a CAN FD decoder. It is used to determine the sampling position for both the arbitration phase (Nominal Bit Time) and data phase (Data Bit Time). This may be done using either the Basic Viewing format or the Advanced Viewing format.

Note: For more detailed information about how these settings function, see CAN FD Serial Trigger Setup. All these settings are ignored when decoding a Standard CAN frame.

By default, **ESI Dominant Only** is selected, which indicates the signal always contains an ESI bit set to dominant. This allows the decoder to operate in a mode where it doesn't need sampling point information. When it is possible that signal can contain a recessive ESI bit, then enter the sampling point information manually. Clear ESI Dominant Only to enable the Nominal Bit Time and Data Bit Time fields and set the sampling point.

For the Basic setup, enter the sampling point as a percent of the UI in the **Requested Position** fields.



Note: For most users, the default Basic sampling point works sufficiently well. However, an incorrect Requested Sampling Point percentage may result in false error determination. If you see false error triggers or error frames in the decoding, use the Advanced format to enter the values manually.

For the Advanced setup, manually enter the number of time quanta for the:

- **Prop_Seg**, part of the bit time used to compensate for the physical network delay times.
- **Phase_Seg1** Number of time quantums before the sampling point.
- **Phase_Seg2** Number of time quantums after the sampling point.

There is no field for entering the Sync_Seg, because the software always uses the value 1.

The sum of the Sync_Seg, Prop_Seg, Phase_Seg1 and Phase_Seg2 must be equal to the available time quantas in each phase for the settings to work correctly.

When the decoder is enabled, the **Actual Sampling Point** levels used are displayed on the dialog in either format. In the Advanced menu, you may see this value change as you change the values in each field.

LIN Decoder Settings

Basic Subdialog

Under **Viewing**, choose to view/enter data in either **Binary** or **Hex**(adecimal) formats.

Enter the vertical **Level** used to determine the edge crossings of the signal. This value will be used to determine the bit-level decoding. For guidelines, see Setting Level and Hysteresis.

Enter the **Bitrate** of the bus to which you are connected as precisely as you know it. The value should be correct within 5%. A mismatched bit rate will cause various confusing side effects on the decoding, so it is best to take time to correctly adjust this fundamental value. Use the Up and Down arrows to scroll a list of standard bit rates, or enter any value between 1 and 20 kbit/s.

Tip: If you are unsure of the bitrate, apply the Bit Rate measurement parameter to a short acquisition.

Choose the **LIN Version** (standard) used to encode the input signal. If you are unsure or don't care, choose ALL.

FlexRay Decoder Settings

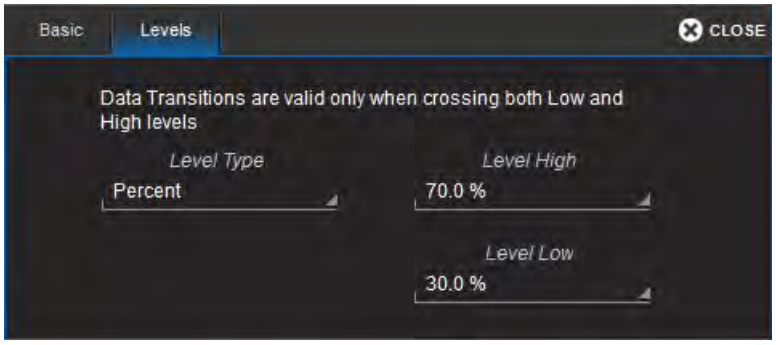
Basic Subdialog

Adjust the **Bitrate** value to match the one on the bus to which you are connected. Touch the arrow buttons to scroll through standard bitrates (2.5, 5.0 or 10.0 Mb/s) and make a selection. Or, touch Bitrate and enter the value using the ADJUST knob or the Virtual Keypad.

Select the appropriate **Channel** to decode, Channel A or Channel B of the FlexRay bus.

Note: Decoding will still occur if the wrong channel is selected, but it will result in CRC errors being shown on the overlay.

Levels Subdialog



In **Level High** and **Level Low**, enter the signal crossing values. FlexRay is a tri-level signal and requires 2 levels for the oscilloscope to distinguish between 1 and 0. As indicated on the dialog, data transitions are valid only when crossing both Low and High levels.

Level is normally entered as percent and defaults to 70% (of amplitude) for Level High and 30% for Level Low. For guidelines, see Setting Level and Hysteresis.

Setting Level and Hysteresis

The **Level** setting represents the logical level for bit transition, corresponding to the physical Low and High distinction. Level is normally set as 50% of waveform amplitude, but can alternatively be set as an absolute voltage (with reference to the waveform 0 level) by changing the **Level Type** to Absolute.

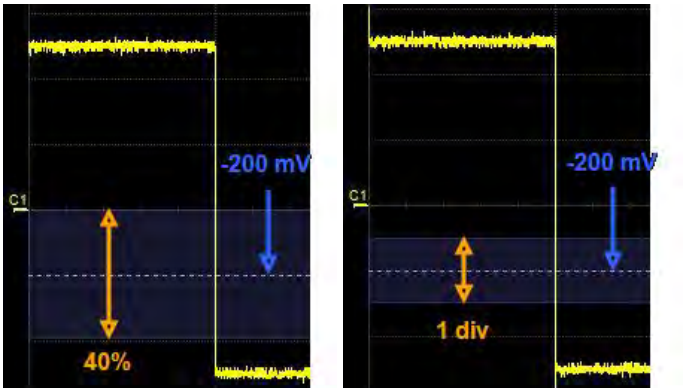
Percent mode is easy to set up because the software immediately determines the optimal threshold, but in some cases it might be beneficial to switch to Absolute mode:

- On poor signals, where Percent mode can fail and lead to bad decodes
- On noisy signals or signals with a varying DC component
- On very long acquisitions, where Percent mode adds computational load

The transition Level appears as a dotted, horizontal line across the oscilloscope grid. If your initial decoding indicates that there are a number of error frames, make sure that Level is set to a reasonable value.

The optional **Hysteresis** setting imposes a limit above and below the measurement level that precludes measurements of noise or other perturbations within this band.

A blue marker around the Level line indicates the area of the hysteresis band. As with Level Type, **Hysteresis Type** may be either a percentage of amplitude or an absolute number of vertical grid divisions.



Hysteresis set as 40 percent of total waveform amplitude (left) and Hysteresis set as equivalent of 1 grid division (right) around an absolute -200mV Level setting.

Note: Usually, you can set the Level and Hysteresis in the same or different modes. For a few protocols, Hysteresis can only be set as a number of mV plus/minus the Level.

Observe the following when setting Hysteresis:

- Hysteresis must be larger than the maximum noise spike you wish to ignore.
- The largest usable hysteresis value must be less than the distance from the level to the closest extreme value of the waveform.

Failure to Decode

Three conditions in particular may cause a decoder to fail, in which case a failure message will appear in the first row of the summary result table, instead of in the message bar as usual.

All decoders will test for the condition **Too small amplitude**. If the signal's amplitude is too small with respect to the full ADC range, the message "Decrease V/Div" will appear. The required amplitude to allow decoding is usually one vertical division.

If the decoder incorporates a user-defined bit rate (usually these are protocols that do not utilize a dedicated clock/strobe line), the following two conditions are also tested:

- **Under sampled**. If the sampling rate (SR) is insufficient to resolve the signal adequately based on the bit rate (BR) setup or clock frequency, the message "Under Sampled" will appear. The minimum SR:BR ratio required is 4:1. It is suggested that you use a slightly higher SR:BR ratio if possible, and use significantly higher SR:BR ratios if you want to also view perturbations or other anomalies on your serial data analog signal.
- **Too short acquisition**. If the acquisition window is too short to allow any meaningful decoding, the message "Too Short Acquisition" will appear. The minimum number of bits required varies from one protocol to another, but is usually between 5 and 50.

In all the above cases, the decoding is turned off to protect you from incorrect data. Adjust your acquisition settings accordingly, then re-enable the decoder.

Note: It is possible that several conditions are present, but you will only see the first relevant message in the table. If you continue to experience failures, try adjusting the other settings.

Serial Decode Dialog

To first set up a decoder, go to the Decode Setup dialog. Once decoders have been configured, use the Serial Decode dialog to quickly turn on/off a decoder or make minor modifications to the settings.

To turn on decoders:

1. On the same row as the **Decode #**, check **On** to enable the decoder.

   As long as On is checked (and there is a valid acquisition), a result table and decoded waveform appear. The number of rows of data displayed will depend on the **Table #Rows** setting (on the Decode Setup dialog).

2. Optionally, modify the:
   - **Protocol** associated with the decoder.
   - **Data (Source)** to be decoded.
3. Check Link To Trigger **On** to tie this decoder setup to a serial trigger setup.

To turn off decoders: deselect the On boxes individually, or touch **Turn All Off**.

Reading Waveform Annotations

When a decoder is enabled, an annotated waveform appears on the oscilloscope display, allowing you to quickly see the relationship between the protocol decoding and the physical layer. A colored overlay marks significant bit-sequences in the source signal: Header/ ID, Address, Labels, Data Length Codes, Data, CRC, Parity Bits, Start Bits, Stop Bits, Delimiters, Idle segments, etc. Annotations are customized to the protocol or encoding scheme.

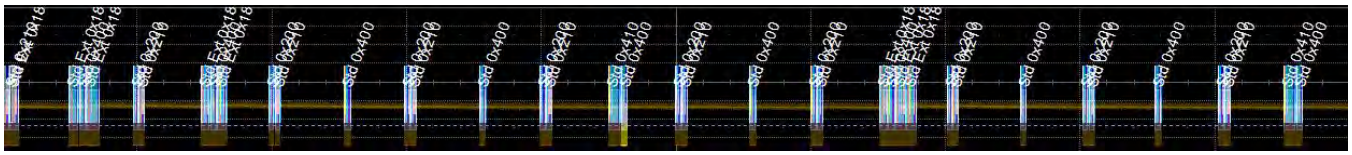
The amount of information shown on an annotation is affected by the width of the rectangles in the overlay, which is determined by the magnification (scale) of the trace and the length of the acquisition. Zooming a portion of the decoder trace will reveal the detailed annotations.

CAN Waveform Annotations

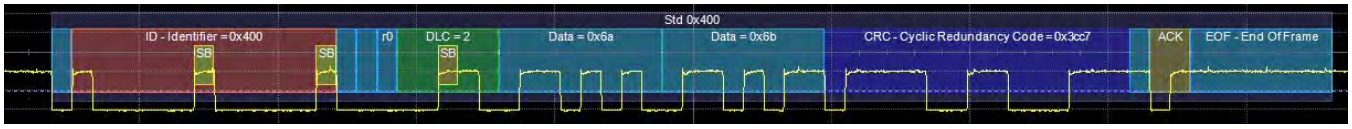
These overlays appear on a decoded standard CAN waveform or its zoom trace.

Annotation	Overlay Color (1)	Overlay Text (2) (3)
Index	Navy Blue (behind other fields)	< Std | FD > <*value*>
Frame ID	Brick Red	ID=<*value*>
Reserved Bits	Aqua Blue	< r0 | r1 >
Data Length Code	Green	DLC=<*value*>
Payload Data	Aqua Blue	Data = <*value*> (in Hexadecimal) <*interpretation*> (in Symbolic)
Cyclic Redundancy Check	Royal Blue	CRC=<*value*>
Stuff Bits	Grey	SB
ACK	Tan	ACK = <*value*>
Start/End of Frame	Aqua Blue	< SOF | EOF >
Protocol Error	Bright Red (behind other fields)	Error=<*error type*>

1. Combined overlays affect the appearance of colors.
2. Text in brackets < > is variable. The amount of text shown depends on your zoom factors.
3. Data values are shown in symbolic or hexadecimal depending on your decoder selection.



Initial decoding. At this resolution, little information appears on the overlay.



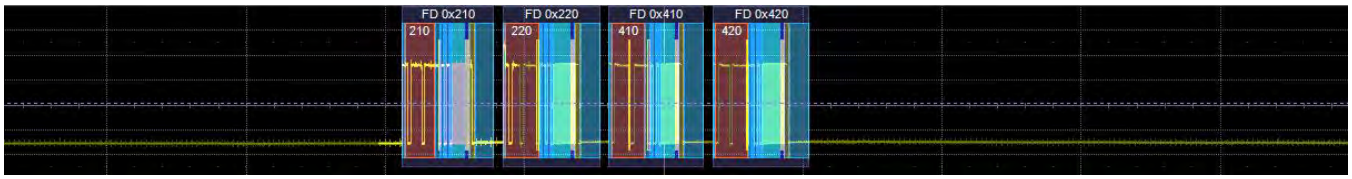
Zoom of single index showing annotation details.

CAN FD Waveform Annotations

These overlays appear on a decoded CAN FD waveform or its zoom trace.

Annotation	Overlay Color (1)	Overlay Text (2) (3)
Index	Navy Blue (behind other fields)	< Std | FD > <*value*>
Frame ID	Brick Red	ID = <*value*>
Reserved Bits	Aqua Blue	< r1 | r0 >
Arbitration Field Identifier Extension	Aqua Blue	IDE
Extended Data Length Bits	Aqua Blue	EDL
Bit Rate Switch	Aqua Blue	BRS
Data Length Code	Bright Green	DLC = <*value*>
Payload Data	Aqua Blue	Data = <*value*> (in Hexadecimal) <*interpretation*> (in Symbolic)
Cyclic Redundancy Check	Royal Blue	CRC = <*value*>
Stuff Bits	Grey	SB
ACK	Tan	ACK
Start/End of Frame	Aqua Blue	< SOF | EOF >
Error	Bright Red (behind other fields)	Error = <*error type*>

1. Combined overlays affect the appearance of colors.
2. Text in brackets < > is variable. The amount of text shown depends on your zoom factors.
3. Data values are shown in symbolic or hexadecimal depending on your decoder selection.



Initial decoding.

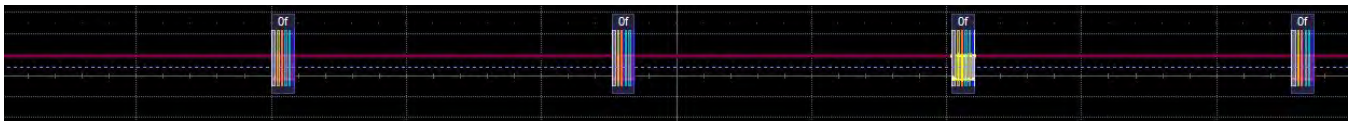
Zoom of single index showing annotation details.

LIN Waveform Annotations

These overlays appear on a decoded LIN waveform or its zoom trace.

Annotation	Overlay Color (1)	Overlay Text (2) (3)
Index	Navy Blue (behind other fields)	*<type> = <ID value>*
Protocol error	Bright Red (behind other fields)	*<error type>*
Breaks	Grey	Break
Start/Stop bits	Grey	< S | T >
Synch byte	Olive Green	Synch = *<value>*
Message ID	Brick Red	ID = *<value>*
Parity bits	Royal Blue	Parity = *<value>*
Payload data	Aqua Blue	Data = *<value>*
Checksum bits	Royal Blue	Checksum = *<value>*

1. Combined overlays affect the appearance of colors.
2. Text in brackets < > is variable. The amount of text shown depends on your zoom factors.
3. Data values are shown in binary or hexadecimal depending on your decoder selection.



Initial decoding.

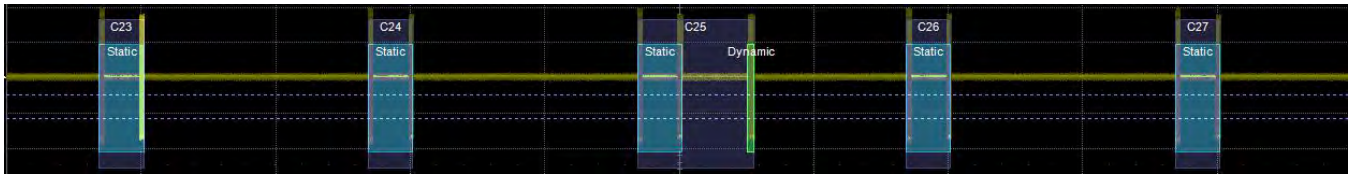
Zoom of single index.

FlexRay Waveform Annotations

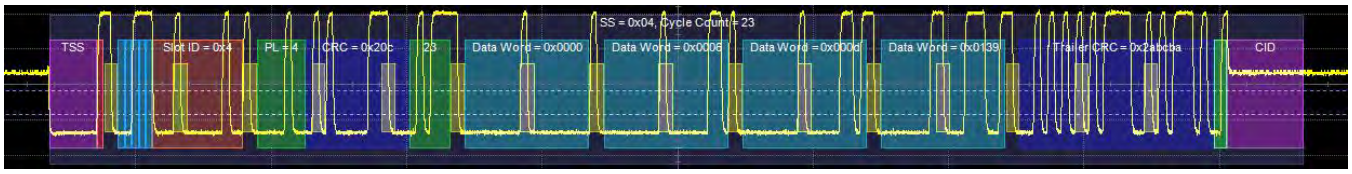
These overlays appear on a decoded FlexRay waveform or its zoom trace.

Annotation	Overlay Color (1)	Overlay Text (2)
Index	Navy (behind other fields)	<*type*> = <*ID*>, Cycle Count = <*value*>
Protocol error	Bright Red (behind other fields)	<*error type*>
Transmit Start Sequence and Channel Idle Delimiter	Purple	< TSS | CID >
Frame Start Sequence and Dynamic Trailing Sequence	Cranberry Red	< FSS | DTS >
Byte Start Sequence	Tan	BSS
Reserved, Preamble, Null Frame, Sync, and StartUp bits	Cyan Blue	< R | P | N | S | SU >
Slot ID	Brick Red	Slot ID = <*value*>
Payload data	Aqua Blue	Data Word = <*value*>
Payload Length, Cycle Count, and Frame End Sequence	Green	< PL | CC | FES > = <*value*>
Cyclic Redundancy Check	Royal Blue	< Header | Trailer > CRC = <*value*>

1. Combined overlays affect the appearance of colors.
2. Text in brackets < > is variable. The amount of text shown depends on your zoom factors.



Initial decoding.



Zoom of single index.

Serial Decode Result Table

When **View Decode** is checked on the Decode Setup Dialog *and* a source signal has been decoded using that protocol, a table summarizing the decoder results appears below the grids. This result table provides a view of data as decoded during the most recent acquisition, even when there are too many bursts for the waveform annotation to be legible.

You can export result table data to a .CSV file. See also Automating the Decoder.

Tip: If any downstream processes such as measurements reference a decoder, the result table does not have to be visible in order for the decoder to function. Hiding the table can improve performance when your aim is to export data rather than view the decoding.

Table Rows

Each row of the table represents one index of data found within the acquisition, numbered sequentially. Exactly what this represents depends on the protocol and how you have chosen to "packetize" the data stream when configuring the decoder (frame, message, packet, etc.).

Note: For some decoders, it is even possible to turn off packetization, in which case all the decoded data appears on one row of the table.

When multiple decoders are run at once, the index rows are combined in a summary table, ordered according to their acquisition time. The Protocol column is colorized to match the input source that resulted in that index.

You can change the number of rows displayed on the table at one time. The default is five rows.

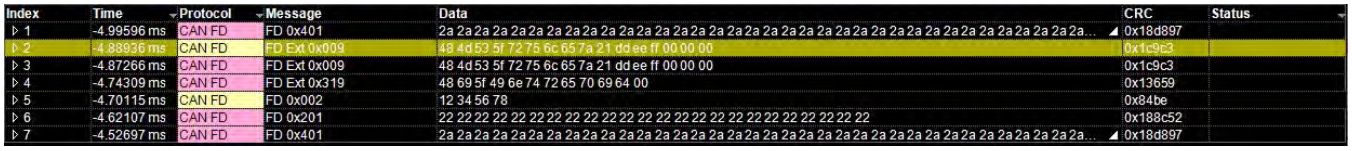
Swipe the table up/down or use the scrollbar at the far right to navigate the table. See Using the Result Table for more information about how to interact with the table rows to view the decoding.

Table Columns

When a single decoder is enabled, the result table shows the protocol-specific details of the decoding. This **detailed result table** may be customized to show only selected columns.

A **summary result table** combining results from two decoders always shows these columns.

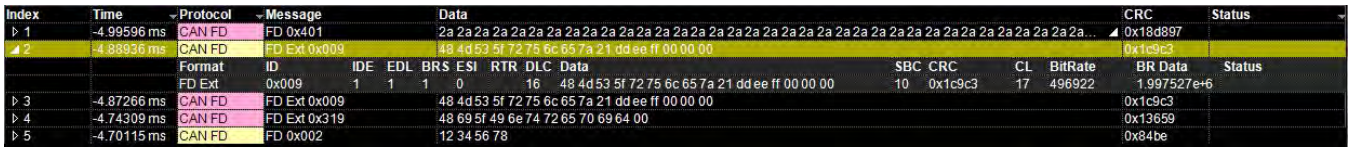
Column	Extracted or Computed Data
Index	Number of the line in the table
Time	Time elapsed from start of acquisition to start of message
Protocol	Protocol being decoded
Message	Message identifier bits
Data	Data payload
CRC	Cyclic Redundancy Check sequence bits
Status	Any decoder messages; content may vary by protocol



Index	Time	Protocol	Message	Data	CRC	Status
1	-4.99596 ms	CAN FD	FD 0x401	2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a...	0x18d897	
2	-4.88936 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	
3	-4.87266 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	
4	-4.74309 ms	CAN FD	FD Ext 0x319	48 69 5f 49 6e 74 72 65 70 69 64 00	0x13659	
5	-4.70115 ms	CAN FD	FD 0x002	12 34 56 78	0x84be	
6	-4.62107 ms	CAN FD	FD 0x201	22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22	0x188c52	
7	-4.52697 ms	CAN FD	FD 0x401	2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a...	0x18d897	

Example summary result table, with results from two decoders combined on one table.

When you select the Index number from the summary result table, the detailed results for that index drop-in below it.



Index	Time	Protocol	Message	Data	CRC	Status
1	-4.99596 ms	CAN FD	FD 0x401	2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a...	0x18d897	
2	-4.88936 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	

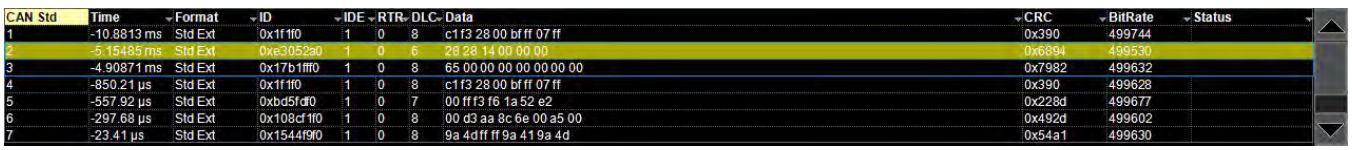
Format	ID	IDE	EDL	BRS	ESI	RTR	DLC	Data	SBC	CRC	CL	BitRate	BR Data	Status
FD Ext	0x009	1	1	1	0		16	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	10	0x1c9c3	17	496922	1.997527e+6	

Index	Time	Protocol	Message	Data	CRC	Status
3	-4.87266 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	
4	-4.74309 ms	CAN FD	FD Ext 0x319	48 69 5f 49 6e 74 72 65 70 69 64 00	0x13659	
5	-4.70115 ms	CAN FD	FD 0x002	12 34 56 78	0x84be	

Example summary result table showing drop-in detailed result table.

CAN Result Table

Column	Extracted or Computed Data
Index	Number of the line in the table
Time	Time elapsed from start of acquisition to Start of Frame
Format	Frame format, Standard (Std) or Flexible Data Rate (FD)
ID	Standard or Extended Identifier bits (aka Frame ID)
IDE	Arbitration Field Identifier Extension; Dominant (0) = 11-bit, Recessive (1) = 29-bit
RTR	Remote Transmission Request bits
DLC	Real number of Data Length Code bytes (not the value encoded in the DLC)
Data (Hexadecimal)	Payload data bytes
Symbols (Symbolic)	Symbolic interpretation of the signals in the message (e.g., Vehicle Speed = 12.4 m/s, or Aircraft Pitch=12.3 degrees)
Signal (Symbolic)	Input signal physical characteristics
CRC	Cyclic Redundancy Check sequence bits
BitRate	Actual bitrate (BR) for this message, the average BR recomputed by dividing the entire Message time span by the total number of bits in the message
Status	Description of all the errors in this message detected by the decoder

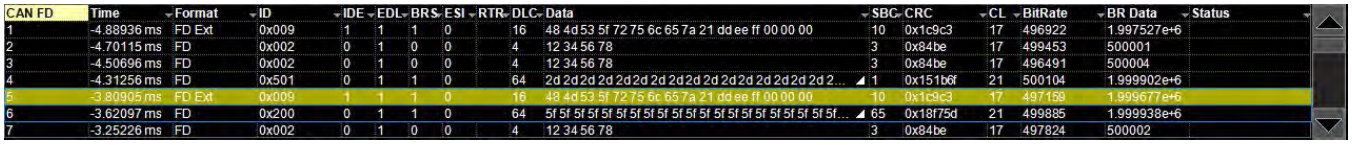


CAN Std	Time	Format	ID	IDE	RTR	DLC	Data	CRC	BitRate	Status
1	-10.8813 ms	Std Ext	0x1f1f0	1	0	8	c1 f3 28 00 bf ff 07 ff	0x390	499744	
2	-5.15485 ms	Std Ext	0xe3052a0	1	0	6	28 28 14 00 00 00	0x6894	499530	
3	-4.90871 ms	Std Ext	0x17b1fff0	1	0	8	65 00 00 00 00 00 00 00	0x7982	499632	
4	-850.21 µs	Std Ext	0x1f1f0	1	0	8	c1 f3 28 00 bf ff 07 ff	0x390	499628	
5	-557.92 µs	Std Ext	0xbd5fdf0	1	0	7	00 ff f3 f6 1a 52 e2	0x228d	499677	
6	-297.68 µs	Std Ext	0x108cf1f0	1	0	8	00 d3 aa 8c 6e 00 a5 00	0x492d	499602	
7	-23.41 µs	Std Ext	0x1544f9f0	1	0	8	9a 4d ff ff 9a 41 9a 4d	0x54a1	499630	

Section of typical CAN detailed result table.

CAN FD Result Table

Column	Extracted or Computed Data
Idx	Number of the line in the table
Time	Time elapsed from start of acquisition to Start of Frame
Format	Frame format, Standard (Std) or Flexible Data Rate (FD)
ID	11- or 29-bit message ID (ID of the message also governs its priority on the bus, based on the CSMA/CD scheme)
IDE	Arbitration Field Identifier Extension; Dominant (0) = 11-bit, Recessive (1) = 29-bit
EDL	Extended Data Length bits
BRS	Bit Rate Switch when Arbitration Frame and Data Frame configured to have different bit rate; Dominant (0) = no bit rate change, Recessive (1) = change to higher bit rate
ESI	Error State Indicator; Dominant (0) = node is error active, Recessive (1) = node is error passive
RTR	Remote Transmission Request bit. Dominant (0) = Data Frame; Recessive (1) = Remote Frame. It is only used in Standard CAN, not in CAN FD, but listed in case Standard CAN frames are included in the signal. It will appear empty for CAN FD frames
DLC	Data Length Code bits
Data (Hexadecimal)	Payload data bytes
Symbols (Symbolic)	Symbolic interpretation of the signals in the message (e.g., Vehicle Speed = 12.4 m/s, or Aircraft Pitch=12.3 degrees)
Signal (Symbolic)	Input signal physical characteristics
SBC	Stuff Bit Counter. This column only appears when ISO frames are supported and ISO is checked
CRC	Cyclic Redundancy Check Sequence bits
CL	CRC Length, depending on the number of bytes in the payload either 17 bits for data frames up to 16 bytes or 21 bits for data frames over 16 bytes
BitRate	Computed transmission bit rate
BR Data	Data bit rate dynamically recomputed for every message, similar to the nominal BR
Status	Description of all the errors in this message detected by the decoder



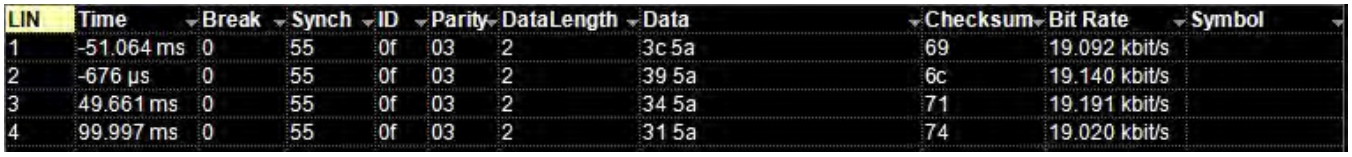
CAN FD	Time	Format	ID	IDE	EDL	BRS	ESI	RTR	DLC	Data	SBC	CRC	CL	BitRate	BR Data	Status
1	-4.88936 ms	FD Ext	0x009	1	1	1	0		16	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	10	0x1c9c3	17	496922	1.997527e+6	
2	-4.70115 ms	FD	0x002	0	1	0	0		4	12 34 56 78	3	0x84be	17	499453	500001	
3	-4.50696 ms	FD	0x002	0	1	0	0		4	12 34 56 78	3	0x84be	17	496491	500004	
4	-4.31256 ms	FD	0x501	0	1	1	0		64	2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2...	1	0x151b6f	21	500104	1.999902e+6	
5	-3.80905 ms	FD Ext	0x009	1	1	1	0		16	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	10	0x1c9c3	17	497159	1.999677e+6	
6	-3.62097 ms	FD	0x200	0	1	1	0		64	5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f...	65	0x18f75d	21	499885	1.999938e+6	
7	-3.25226 ms	FD	0x002	0	1	0	0		4	12 34 56 78	3	0x84be	17	497824	500002	

Section of CAN FD detailed result table.

Note: When there is a CRC error, the value in the CRC column is the value transmitted at the end of the (detected) frame, while the value following "CRC Error:" in the Status column is the value calculated from the (expected) frame bits.

LIN Result Table

Column	Extracted or Computed Data
Idx	Number of the line in the table
Time	Time elapsed from start of acquisition to Start of Frame
Break	Break (inter-message) bits
Synch	Synch bits
ID	Frame ID
Parity	Parity bits
Data Length	Data field length
Data	Data field bytes
Checksum	Checksum bits
BitRate	Transmission bit rate
Symbol	Status of any LIN symbols found in the frame

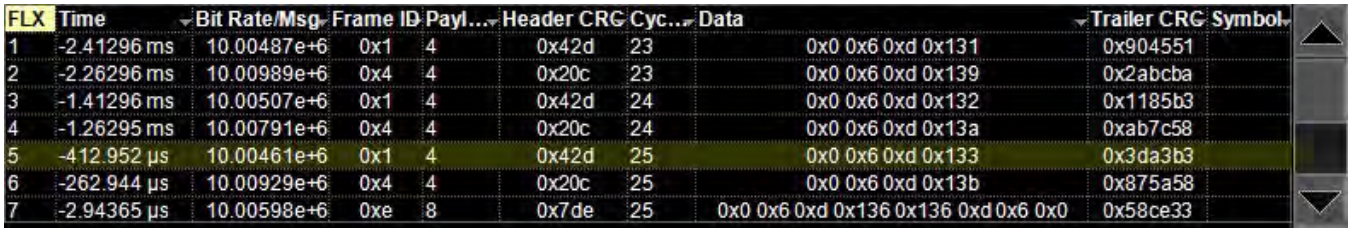


LIN	Time	Break	Synch	ID	Parity	DataLength	Data	Checksum	Bit Rate	Symbol
1	-51.064 ms	0	55	0f	03	2	3c 5a	69	19.092 kbit/s	
2	-676 µs	0	55	0f	03	2	39 5a	6c	19.140 kbit/s	
3	49.661 ms	0	55	0f	03	2	34 5a	71	19.191 kbit/s	
4	99.997 ms	0	55	0f	03	2	31 5a	74	19.020 kbit/s	

Section of typical LIN detailed result table.

FlexRay Result Table

Column	Extracted or Computed Data
Index	Number of the line in the table
Time	Time elapsed from start of acquisition to Start of Frame
Bit Rate/Msg	Actual bit rate (BR) for this message, the average BR recomputed by diving the entire Message time span by the total number of bits in the message
Frame ID	Frame identifier bits
Payload Length	Number of words in data payload
Header CRC	Cyclic Redundancy Check sequence bits
Cycle Count	Cycle number
Data	Data Field bytes
Trailer CRC	Cyclic Redundancy Check sequence bits
Symbol	Symbol type, if found: Wakeup Pattern (WUS), Channel Idle (CID), Collision Avoidance (CAS), or Media Access Test (MTS)



FLX	Time	Bit Rate/Msg	Frame ID	Payl...	Header CRC	Cyc...	Data	Trailer CRC	Symbol
1	-2.41296 ms	10.00487e+6	0x1	4	0x42d	23	0x0 0x6 0xd 0x131	0x904551	
2	-2.26296 ms	10.00989e+6	0x4	4	0x20c	23	0x0 0x6 0xd 0x139	0x2abcba	
3	-1.41296 ms	10.00507e+6	0x1	4	0x42d	24	0x0 0x6 0xd 0x132	0x1185b3	
4	-1.26295 ms	10.00791e+6	0x4	4	0x20c	24	0x0 0x6 0xd 0x13a	0xab7c58	
5	-412.952 µs	10.00461e+6	0x1	4	0x42d	25	0x0 0x6 0xd 0x133	0x3da3b3	
6	-262.944 µs	10.00929e+6	0x4	4	0x20c	25	0x0 0x6 0xd 0x13b	0x875a58	
7	-2.94365 µs	10.00598e+6	0xe	8	0x7de	25	0x0 0x6 0xd 0x136 0x136 0xd 0x6 0x0	0x58ce33	

Section of typical FlexRay detailed result table.

Using the Result Table

Besides displaying the decoded serial data, the result table helps you to inspect the acquisition.

Zoom & Search

Touching any cell of the table opens a zoom centered around the part of the waveform corresponding to the index. The Z*n* dialog opens to allow you to rescale the zoom, or to Search the acquisition. This is a quick way to navigate to events of interest in the acquisition.

Tip: When in a summary table, touch any data cell *other than* Index and Protocol to zoom.

The table rows corresponding to the zoomed area are highlighted, as is the zoomed area of the source waveform. The highlight color reflects the zoom that it relates to (Z1 yellow, Z2 pink, etc.). As you adjust the zoom scale, the highlighted area may expand to several rows of the table, or fade to indicate that only a part of that Index is shown in the zoom.

When there are multiple decoders running, each can have its own zoom of the decoding highlighted on the summary table at the same time.

Note: The zoom number is no longer tied to the decoder number. The software tries to match the numbers, but if it cannot it uses the next zoom that is not yet turned on.

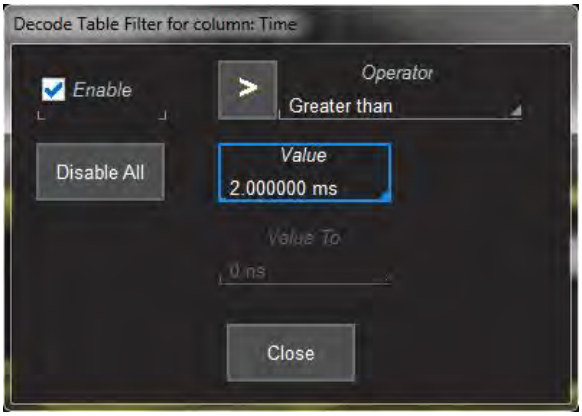
Index	Time	Protocol	Message	Data	CRC	Status
1	-4.99596 ms	CAN FD	FD 0x401	2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a...	0x18d897	
2	-4.88936 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	
3	-4.87266 ms	CAN FD	FD Ext 0x009	48 4d 53 5f 72 75 6c 65 7a 21 dd ee ff 00 00 00	0x1c9c3	
4	-4.74309 ms	CAN FD	FD Ext 0x319	48 69 5f 49 6e 74 72 65 70 69 64 00	0x13659	
5	-4.70115 ms	CAN FD	FD 0x002	12 34 56 78	0x84be	
6	-4.62107 ms	CAN FD	FD 0x201	22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22	0x188c52	
7	-4.52697 ms	CAN FD	FD 0x401	2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a...	0x18d897	

Example multi-decoder summary table, both zoomed indexes highlighted.

Filter Results

Those columns of data that have a drop-down arrow in the header cell can be filtered: 

Touch the **header cell** to open the Decode Table Filter dialog.



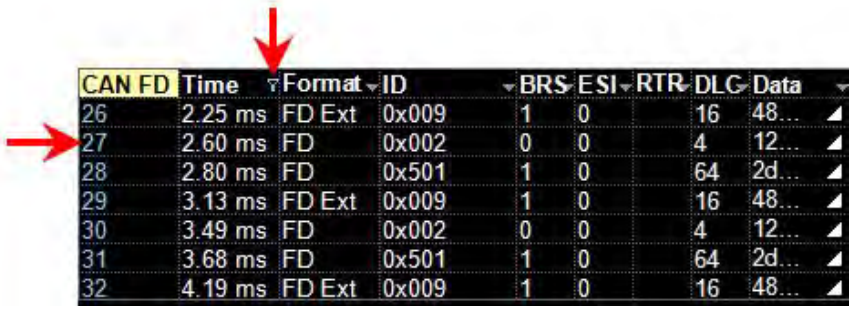
Select a filter **Operator** and enter a **Value** that satisfies the filter condition.

Operators	Data Types	Returns
=, ≠	Numeric or Text	Exact matches only
>, ≥, <, ≤	Numeric	All data that satisfies the operator
In Range, Out Range	Numeric	All data within/without range limits
Equals Any (on List), Does Not Equal Any (on List)	Text	All data that is/is not an exact match to any full value on the list. Enter a comma-delimited list of values, no spaces before or after the comma, although there may be spaces within the strings.
Contains, Does Not Contain	Text	All data that contains or does not contain the string

Note: Once the Operator is selected, the dialog shows the format that may be entered in Value for that column of data. Numeric values must be within .01%tolerance of a result to be considered a match. Text values are case-sensitive, including spaces within the string.

Select **Enable** to turn on the column filter; deselect it to turn off the filter. Use the **Disable All** button to quickly turn off multiple filters. The filter settings remain in place until changed and can be re-enabled on subsequent decodings.

Those columns of data that have been filtered will have a funnel icon (similar to Excel) in the header cell, and the index numbers will be colorized.



CAN FD	Time	Format	ID	BRS	ESI	RTR	DLC	Data
26	2.25 ms	FD Ext	0x009	1	0		16	48...
27	2.60 ms	FD	0x002	0	0		4	12...
28	2.80 ms	FD	0x501	1	0		64	2d...
29	3.13 ms	FD Ext	0x009	1	0		16	48...
30	3.49 ms	FD	0x002	0	0		4	12...
31	3.68 ms	FD	0x501	1	0		64	2d...
32	4.19 ms	FD Ext	0x009	1	0		16	48...

Example filtered decoder table.

On summary tables, only the Time, Protocol, and Status columns can be filtered.

If you apply filters to a single decoder table, the annotation is applied to only that portion of the waveform corresponding to the filtered results, so you can quickly see where those results occurred. Annotations are not affected when a summary table is filtered.

Also, eye diagrams are modified to represent only the filtered results, which can help to identify exactly which indices of data are the cause of signal integrity problems.

View Details

When viewing a summary table, touch the **Index number** in the first column to drop-in the detailed decoding of that record. Touch the Index cell again to hide the details.

If there is more data than can be displayed in a cell, the cell is marked with a white triangle in the lower-right corner. Touch this to open a pop-up showing the full decoding.

Navigate

In a single decoder table, touch the **Index column header** (top, left-most cell of the table) to open the Decode Setup dialog. This is especially helpful for adjusting the decoder during initial tuning.

When in a summary table, the Index column header cell opens the Serial Decode dialog, where you can enable/disable all the decoders. Touch the **Protocol** cell to open the Decode Setup dialog for the decoder that produced that index of data.

Customizing the Result Table

Performance may be enhanced if you reduce the number of columns in the result table to only those you need to see. It is also especially helpful if you plan to export the data.

1. On the Decode Setup tab, touch the **Configure Table** button.
2. On the **View Columns** pop-up dialog, mark the columns you want to appear and clear those you wish to remove. Only those columns selected will appear on the oscilloscope display.

   Note: If a column is not relevant to the decoder as configured, it will not appear.

   To return to the preset display, touch **Default**.
3. Touch the **Close** button when finished.

On some decoders, you may also use the View Columns pop-up to set a **Bit Rate Tolerance** percentage. When implemented, the tolerance is used to flag out-of-tolerance messages (messages outside the user-defined bitrate +- tolerance) by colorizing in red the Bitrate shown in the table.

You may customize the size of the result table by changing the **Table # Rows** setting on the Decode Setup dialog. Keep in mind that the deeper the table, the more compressed the waveform display on the grid, especially if there are also measurements turned on.

Exporting Result Table Data

You can manually export the detailed result table data to a .CSV file:

1. Press the Front Panel **Serial Decode button**, or choose **Analysis > Serial Decode**, then open the **Decode Setup tab**.
2. Optionally, touch **Browse** and enter a new **File Name** and output folder.
3. Touch the **Export Table** button.

Export files are by default created in the D:\Applications\<protocol> folder, although you can choose any other folder on the oscilloscope or any external drive connected to a host USB port. The data will overwrite the last export file saved, unless you enter a new filename.

Note: Only rows and columns displayed are exported. When a summary table is exported, a combined file is saved in D:\Applications\Serial Decode. Separate files for each decoder are saved in D:\Applications\<protocol>.

The Save Table feature will automatically create tabular data files with each acquisition trigger. The file names are automatically incremented so that data is not lost. Choose **File > Save Table** from the oscilloscope menu bar and select **Decode*x*** as the source.

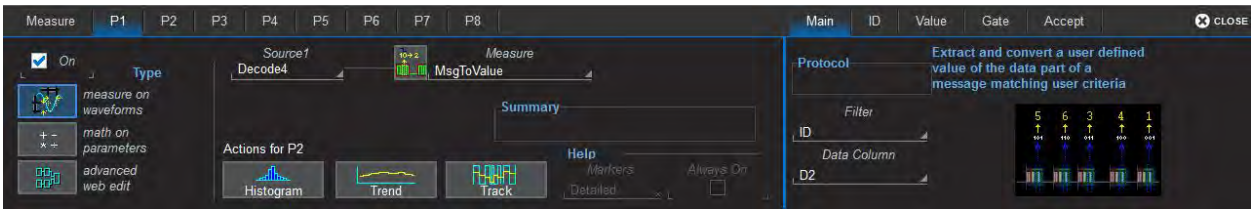
Using MsgToValue

MsgToValue enables you to apply oscilloscope features to a subset of the result table and is aimed at protocols with addressed packets containing varying types of data, like CAN, LIN, MIL1553 and many others. With it, you can filter the table by a particular ID to extract and convert decoded data values into a

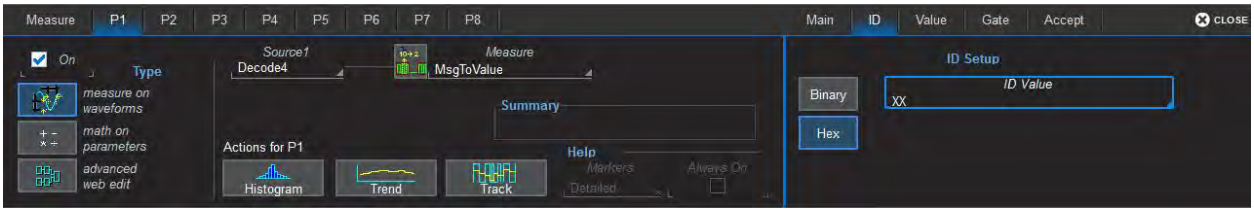
parameter that can be used for other math or measurement processes, in particular the Track function. The track of the Msg to Value parameter is, in effect, a Digital to Analog Converter (DAC) that can display digitally-encoded sensor data as an analog waveform.

MsgToValue requires several selections from the parameter set up subdialogs:

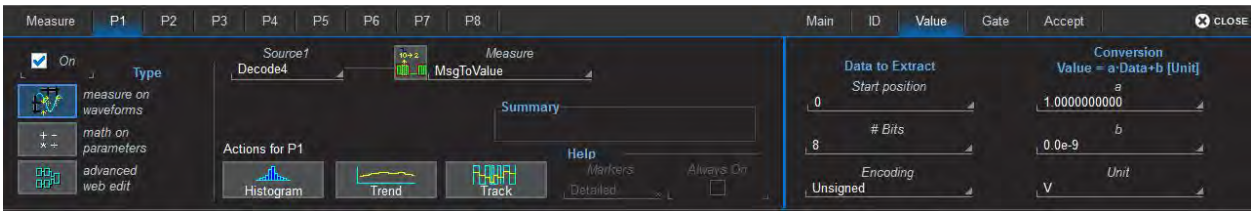
- Choose whether or not you wish to **Filter** by **ID** or accept **Any** packets.



- If you are filtering by ID, enter the desired ID on the ID subdialog.



- On the Value subdialog, enter the **Data to Extract** and any **Conversion** to be made.



Follow these steps to define the values to extract:

1. SYMBOLIC users: touch **Browse DBC**. Expand the .dbc file, then click on the desired symbol to filter on occurrences of that symbol.

   Note: This button will not appear if your installation does not support symbolic decoding.

2. Under Data to Extract, begin by entering the **Start position** and the **# Bits** to extract.
3. Choose the **Encoding** type if the signal uses encoding, otherwise leave it Unsigned.
4. Under Conversion, enter the **a. Coefficient** and **b. Term** that satisfy the formula: *Value = Coefficient * Raw Value + Term*.
5. Optionally, enter a **Unit** for the extracted decimal value.

Searching Decoded Waveforms

Touching the Action toolbar **Search button** button on the Decode Setup dialog creates a 10:1 zoom of the center of the decoder source trace and opens the Search subdialog.

Touching the **any cell** of the result table similarly creates a zoom and opens Search, but of only that part of the waveform corresponding to the index (plus any padding).

Tip: In summary table mode, touch any cell *other than* Index and Protocol to create the zoom.

Basic Search

On the Search subdialog, select what type of data element to **Search for**. These basic criteria vary by protocol, but generally correspond to the columns of data displayed on the detailed decoder result table.

Optionally:

- Check **Use Value** and enter the **Value** to find in that column. If you do not enter a Value, Search goes to the beginning of the next data element of that type found in the acquisition.
- Enter a **Left/ Right Pad**, the percentage of horizontal division around matching data to display on the zoom.
- Check **Show Frame** to mark on the overlay the frame in which the event was found.

After entering the Search criteria, use the **Prev** and **Next** buttons to navigate to the matching data in the table, simultaneously shifting the zoom to the portion of the waveform that corresponds to the match.

The touch screen message bar shows details about the table row and column where the matching data was found.

Advanced Search

Advanced Search allows you to create complex criteria by using Boolean AND/ OR logic to combine up-to-three different searches. On the Advanced dialog, choose the **Col(umns) to Search 1 - 3** and the **Value** to find just as you would a basic search, then choose the **Operator(s)** that represent the relationship between them.

Decoding in Sequence Mode

Not supported on legacy WaveSurfer3000 models.

Decoders can be applied to Sequence Mode acquisitions. In this case, the index numbers on the result table are followed by the segment in which the index was found and the number of the sample within that segment: *index* (*segment-sample*).

Note: For some protocols, the Serial Trigger does not support Sequence Mode acquisitions, although you could still decode Sequence Mode acquisitions made using a different trigger type.

CAN Std	Time	Format	ID	IDE	RTR	DLC	Data
2 (2-1)	9.72882 ms	Std	0x400	0	0	2	6a 6b
3 (3-1)	19.7527 ms	Std	0x400	0	0	2	6a 6b
4 (4-1)	30.2558 ms	Std	0x400	0	0	2	6a 6b
5 (5-1)	40.1663 ms	Std	0x400	0	0	2	6a 6b
6 (6-1)	49.8284 ms	Std	0x400	0	0	2	6a 6b
7 (7-1)	59.8595 ms	Std	0x400	0	0	2	6a 6b
8 (8-1)	69.8913 ms	Std	0x400	0	0	2	6a 6b
9 (9-1)	80.4032 ms	Std	0x400	0	0	2	6a 6b
10 (10-1)	89.9384 ms	Std	0x400	0	0	2	6a 6b
11 (11-1)	99.9688 ms	Std	0x400	0	0	2	6a 6b

Example filtered result table for a sequence mode acquisition.

In the example above, each segment was triggered on the occurrence of ID 0x400, which occurred only once per segment, so there is only one sample per segment. The Time shown for each index in a Sequence acquisition is absolute time from the first segment trigger to the beginning of the sample segment.

Otherwise, the results are the same as for other types of acquisitions and can be zoomed, filtered, searched, or used to navigate. When a Sequence Mode table is filtered, the waveform annotation appears on only those segments and samples corresponding to the filtered results.

Note: Waveform annotations can only be shown when the Sequence Display Mode is Adjacent. Annotations are not adjusted when a Sequence Mode summary table is filtered, only the result table data.

Multiple decoders can be run on Sequence Mode acquisitions, but in a summary table, each decoder will have a first segment, second segment, etc., and there may be any number of samples in each. As in any summary table, the samples will be interleaved and indexed according to their actual acquisition time. So, you may find (3-2) of one decoder before (1-1) of another. Filter on the Protocol column to see the sequential results for only one decoder.

Improving Decoder Performance

Digital oscilloscopes repeatedly capture "windows in time". Between captures, the oscilloscope is processing the previous acquisition.

The following suggestions can improve decoder performance and enable you to better exploit the long memories of Teledyne LeCroy oscilloscopes.

Where possible, **decode Sequence Mode acquisitions.** By using Sequence mode, you can take many shorter acquisitions over a longer period of time, so that memory is targeted on events of interest.

Note: For some protocols, the Serial Trigger does not support Sequence Mode acquisitions, although you could still decode Sequence Mode acquisitions made using a different trigger type.

Parallel test using multiple oscilloscope channels. Up-to-four decoders can run simultaneously, each using different data or clock input sources. This approach is statistically interesting because multi-channel acquisitions occur in parallel. The processing is serialized, but the decoding of each input only requires 20%additional time, which can lessen overall time for production validation testing, etc.

Avoid oversampling. Too many samples slow the processing chain.

Optimize for analysis, not display. The oscilloscope has a preference setting (Utilities > Preference Setup > Preferences) to control how CPU time is allocated. If you are primarily concerned with quickly processing data for export to other systems (such as Automated Test Equipment) rather than viewing it personally, it can help to switch the Optimize For: setting to Analysis.

Turn off tables, annotations, and waveform traces. As long as downstream processes such as measurements or Pass/ Fail tests reference a decoder, the decoder can function without actually displaying results. If you do not need to see the results but only need the exported data, you can deselect View Decode, or minimize the number of lines in a table. Closing input traces also helps.

Decrease the number of columns in tables. Only the result table rows and columns shown are exported. It is best to reduce tables to only the essential columns if the data is to be exported, as export time is proportional to the amount of data exchanged.

Automating the Decoder

As with all other oscilloscope settings, decoder features such as result table configuration and export can be configured remotely using COM Automation.

Note: The examples shown here were taken from a CAN FD decoding, but all decoder result tables share the same Automation structure.

Configuring the Decoder

The object path to the decoder Control Variables (CVARs) is:

app.SerialDecode.Decode*n*

Where *n* is the decoder number, 1 to 4. All relevant decoder objects will be nested under this. Use the XStreamBrowser utility (installed on the oscilloscope desktop) to view the entire object hierarchy.

Accessing the Result Table

The decoder Result Table is a complex matrix with secondary tables nested within some of its cells. The table data can be accessed using the Automation object:

app.SerialDecode.Decode*n*.out.Result.cellvalue(*RowA*, *ColA*)(*RowB*, *ColB*)

Where:

n:= 1 to 4

RowA:= 0 to K (0=Row Index Number)

ColA:= 0 to L (0=Column Header)

RowB:= 0=MeasuredValue, 1=StartTime, 2=StopTime

ColB:= 0 to M

Complicating the matter of accessing the table is that there are two types of cell that may appear in the Result table, Simple Cell and Table Cell, which are accessed in slightly different ways, and that some columns are always hidden from view, yet they are still counted among the columns when querying.

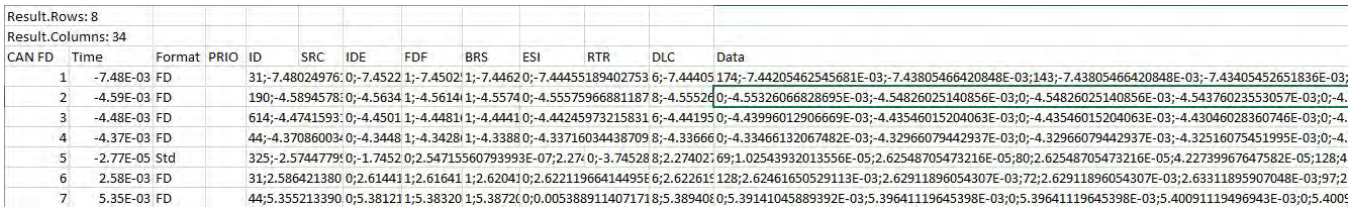
Reading the Structure of the Result Table

In order to successfully access the data, it is necessary to first ascertain how many rows and columns are actually in your decoder result table, and what cell type is used for the column of data you wish to read.

To do this, we have provided the script, **ExampleTableSerialDecode.vbs**, which by default installs into oscilloscope: C:\LeCroy\XStream\Scripts\Automation\ExampleTableSerialDecode.vbs.

Tip: This script may also be used as a basis for your own remote control programs, or used as is to read decoder table data.

With the decoder table populated, run the script from the oscilloscope (or a remote PC if you have a DCOM connection to the oscilloscope). The script will generate the comma-delimited file, **ExampleTableSerialDecode.txt**, which may be imported into Excel or other spreadsheet software to show the table structure.



Result.Rows: 8												
Result.Columns: 34												
CAN FD	Time	Format	PRIO	ID	SRC	IDE	FDF	BRS	ESI	RTR	DLC	Data
1	-7.48E-03	FD		31;-7.480249766	0;-7.4522	1;-7.4502	1;-7.4462	0;-7.444551894027536		6;-7.44405	174;-7.44205462545681E-03;-7.43805466420848E-03;143;-7.43805466420848E-03;-7.43405452651836E-03;	
2	-4.59E-03	FD		190;-4.589457867	0;-4.5634	1;-4.5614	1;-4.5574	0;-4.555759668811877		8;-4.55526	0;-4.553260668286955E-03;-4.54826025140856E-03;0;-4.54826025140856E-03;-4.54376023553057E-03;0;-4.	
3	-4.48E-03	FD		614;-4.474159337	0;-4.4501	1;-4.4481	1;-4.4441	0;-4.442459732158316		6;-4.44195	0;-4.43996012906669E-03;-4.43546015204063E-03;0;-4.43546015204063E-03;-4.43046028360746E-03;0;-4.	
4	-4.37E-03	FD		44;-4.370860034	0;-4.3448	1;-4.3428	1;-4.3388	0;-4.337160344387098		8;-4.33666	0;-4.33466132067482E-03;-4.32966079442937E-03;0;-4.32966079442937E-03;-4.32516075451995E-03;0;-4.	
5	-2.77E-05	Std		325;-2.574477959	0;-1.7452	0;2.54715560793993E-07		2.274	0;-3.74528	8;2.274027	69;1.025439320135566E-05;2.62548705473216E-05;80;2.62548705473216E-05;4.22739967647582E-05;128;4	
6	2.58E-03	FD		31;2.586421380	0;2.61441	1;2.61641	1;2.62041	0;2.622119664144956		6;2.62261	128;2.62461650529113E-03;2.62911896054307E-03;72;2.62911896054307E-03;2.63311895907048E-03;97;2	
7	5.35E-03	FD		44;5.355213390	0;5.38121	1;5.38320	1;5.38720	0;0.005388911407171		8;5.38940	0;5.39141045889392E-03;5.39641119645398E-03;0;5.39641119645398E-03;5.40091119496943E-03;0;5.400	

Example spreadsheet after importing ExampleTableSerialDecode.txt.

The first two rows of the imported file will show the total number of rows and columns in the table, in this example 8 rows and 34 columns. This indicates the range of your *RowA* and *ColA* keys.

The third row of the imported file will replicate the column headers of the Result Table (0), with individual records (frames, messages, etc., depending on how you have "packetized" the decoding) appearing in subsequent rows (1-*n*).

Counting from 0 at the far left (Row Index Number), find the column of the data you wish to access. That will be the *ColA* key in your script.

Note: Do not confuse the number/ letter of the cells in the imported file with the rows/ columns of the Result Table.

Hidden columns (whether hidden by you or the software) must still be counted, so, in the example above, PRIO is column 3, making ID column 4, and so forth. So, if you wished to access the ID of record 6, the first argument of your query would be: (6,4)

Within each column, Simple Cells contain a single value that appears at the specified location in the table. In the above example, columns 0 through 2 are Simple Cells. Simple Cell VBS access syntax is:

vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(*RowA*,*ColA*)'

However, many cells of the Result Table are the Table Cell type, nested tables that may contain multiple "B" columns and always three "B" rows that, when coupled with the column key, each return a different component of the measurement: (0,*ColB*) = MeasuredValue, (1,*ColB*)= StartTime, (2,*ColB*) = StopTime. These cells can be identified by the list of semi-colon delimited values within them. The first three values in the list are Col0, the second three values are Col1, and so forth.

To access Table Cells, the (*RowB*,*ColB*) argument is sent in a second parenthesis, following the A "locators":

vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(*RowA*,*ColA*)(*RowB*, *ColB*)'

Although the image above does now show it, the ID and IDE columns each contain a single-column, three-row nested table. To read the *values* from such columns, you would add the argument (0,0) following your "locators": (*RowA*,4),(0,0) and (*RowA*,6),(0,0) respectively.

Reading the Data column (*RowA*,12) is more complicated, because it contains a *multi-column,* three-row nested table, as indicated by the longer list of values. To access the full Data column value for each record, all *ColB*s must be called by your script.

For example, if these were your decoder results:

CAN FD	Time	Format	ID	IDE	FDF	BRS	ESI	RTR	DLC	Data
1	-7.4822 ms	FD	0x01f	0	1	1	0		6	ae 8f a0 a3 00 06
2	-4.5915 ms	FD	0x0be	0	1	1	0		8	00 00 00 00 00 00 00 00
3	-4.4762 ms	FD	0x266	0	1	1	0		6	00 00 00 00 00 00
4	-4.3729 ms	FD	0x02c	0	1	1	0		8	00 00 00 00 00 00 00 00
5	-27.74 µs	Std	0x145	0	0			0	8	45 50 80 00 00 00 00 00
6	2.58442 ms	FD	0x01f	0	1	1	0		6	80 48 61 44 00 06
7	5.35321 ms	FD	0x02c	0	1	1	0		8	00 00 00 00 00 00 00 00

The following table shows example VBS queries you might add to a remote control program to read data from the decoder result table.

Remote Queries	Returned Value (s)	What Is Read by Query
vbs? 'return=app.SerialDecode.Decode1.out.Result.rows'	8	Number of table rows (incl. header Row 0)
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,0)' vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,1)' vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,2)'	6 2.58442...E-03 FD	Value in first 3 columns of Row 6, including: Index # in Row 6 Col 0 Time in Row 6 Col 1 Format in Row 6 Col 2
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,12)(0,0)'	128	Data value in ColB0 of Row 6 Col 12
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,12)(1,0)'	2.62461...E-03	StartTime of Data in ColB0 of Row 6 Col 12 (hidden)
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,12)(2,0)'	2.62911...E-03	StopTime of Data in ColB0 of Row 6 Col 12
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,12)(0,1)'	72	Data value in ColB1 of Row 6 Col 12
vbs? 'return=app.SerialDecode.Decode1.out.Result.cellvalue(6,12)(1,1)'	2.62911...E-03	StartTime of Data in ColB1 of Row 6 Col 12 (hidden)

Modifying the Result Table

The CVAR app.SerialDecode.Decode*n*.Decode.ColumnState contains a pipe-delimited list of all the table columns and their current state (visible=on, hidden=off). For example:

app.SerialDecode.Decode1.Decode.ColumnState = "Idx=On|Time=On|Data=On|..."

If you wish to hide or display table columns, send the full string with the state changed from "on" to "off", or vice versa, rather than remove any column from the list.

Serial Trigger

Note: These instructions pertain only to the -TD and -TDME options for those protocols and encoding schemes where serial trigger is supported: 8b10b, 64b66b, 80-bit NRZ, AudioBus (I2S/RJ/LJ), CAN, CAN FD, FlexRay, I2C, I3C, LIN, MIL-1553, SENT, SMBus, SPI, SPMI, UART-RS232 and USB2.

TD options provide advanced serial data triggering in addition to decoding. Serial data triggering is implemented directly within the hardware of the oscilloscope acquisition system. The serial data trigger scrutinizes the data stream in real time to recognize "on-the-fly" the user-defined serial data conditions. When the desired pattern is recognized, the oscilloscope takes a real-time acquisition of all input signals as configured in the instrument's acquisition settings. This allows decode and analysis of the signal being triggered on, as well as concomitant data streams and analog signals.

The serial trigger supports fairly simple conditions, such as "trigger at the beginning of any packet," but the conditions can be made more restrictive depending on the protocol and the available filters, such as "trigger on packets with ID = 0x456". The most complex triggers incorporate a double condition on the ID and data, for example "trigger on packets with ID = 0x456 and when data in position 27 exceeds 1000".

Note: The trigger and decode systems are independent, although they are seamlessly coordinated in the user interface and the architecture. It is therefore possible to use the serial trigger without decoding the acquisition, or to decode acquisitions made without using the serial trigger.

Requirements

Serial trigger options require the appropriate hardware (please consult support), an installed option key, and the latest firmware release.

Restrictions

The serial trigger only operates on one protocol at a time. It is therefore impossible to express a condition such as "trigger on CAN frames with ID = 0x456 followed by LIN packet with Adress 0xEBC."

Linking Trigger and Decoder

A quick way to set up a serial trigger is to link it to a decoder by checking the **Link to Trigger** ("On") box on the Serial Decode dialog. Linking trigger and decoder allows you to configure the trigger with the exact same values that are used for decoding the signal (in particular the bit rate), saving the extra effort needed to re-enter values on the serial trigger set up dialogs.

While the decoder and the trigger have distinct sets of controls, when the link is active, a change to the bit rate in the decoder will immediately propagate to the trigger and vice-versa.

CAN/ CAN FD Serial Trigger Setup

To access the serial trigger dialogs:

- Touch the **Trigger descriptor box** or choose **Trigger > Trigger Setup** from the Menu Bar.
- On the Trigger dialog, touch the **Serial** Type button, then the **CAN Std** or **CAN FD** Standard button.

Working from left to right, make the desired selections from the trigger setup dialog.

Source Setup

In **DATA**, select the data source input channel.

Use the **Threshold** control to adjust the vertical level for the trigger. Much like an Edge trigger, you must specify the level at which to process the incoming signal to determine whether the serial data pattern meets the trigger condition.

Tip: One way to test that the threshold is set correctly is to initially set the Trigger Type to All. If you see a decoded message appear each time the oscilloscope triggers, the level is correct.

Frame Type

These controls appear only for CAN FD triggers. They allow you to specify information about the frame types to be found in the data stream.

1. Choose whether the EDL bit **Type** is Both (X), CAN Std (0) only, or CAN FD (1) only. Both enables triggering on streams with mixed legacy and extended frame types.
2. If using CAN FD Type, also:
   - Check **ISO Frame** if the stream supports ISO frames.
   - Choose a **BR Select** of Both (X), Normal (0), or FD (1). Both examines frames with or without a bit rate switch. Normal and FD examine only frames with the respective BRS bit values.

CAN Setup

If you have not linked the trigger to a pre-set decoder, enter the **Nominal Bitrate** used during the Arbitration sequence. Use the Up and Down arrows to scroll the list of standard bit rates and make a selection, or touch Bitrate and use the Adjust knob or the Virtual Keypad to enter the value.

Also enter the **Data Bitrate** used during the Data sequence.

Note: When the trigger is linked, this value is dynamically linked to the decoding bitrate; they are always the same.

Trigger Type

These buttons determine which frames/fields are included in the trigger search:

- **All** triggers upon finding the first CAN frame.
- **Remote** triggers upon finding matching Frame ID values in Remote frames (only). Complete the Frame ID Setup described below.
- **ID** triggers upon finding matching Frame ID values in any type of frame. Complete the Frame ID Setup described below.
- **ID+Data** triggers upon finding matching Frame ID values *and* data patterns. Complete both the Frame ID Setup and Data Pattern Setup described below.
- **Error** triggers when a protocol error occurs. Select all the errors that can produce a trigger from below **Error Type**.

Viewing Format

Choose to display/enter values in **Binary**, **Hex**(adecimal) or **Sym**(bolic) format (depending on your capabilities). This selection propagates throughout the trigger setup. Toggling formats does not result in loss of information, but will transform the appearance of values.

Note: SYMBOLIC users, follow the Symbolic Setup below for Remote, ID, and ID+Data triggers instead of Frame ID Setup and Data Pattern Setup.

Frame ID Setup

Frame ID Setup is used to trigger upon encountering either a specific ID value or any value relative to a reference ID value (e.g., greater than *x*).

1. Choose the **ID Condition** (Boolean operator) that describes the relationship to the Frame ID value. To use a range of values, choose In Range or Out Range.
2. In **ID Bits**, choose to trigger on 11-bit (Standard CAN) messages, 29-bit (Extended CAN) messages, or ALL.
3. Enter the reference **Frame ID** value.

   When setting a range, also enter the stop value in **To Frame ID**.

Data Pattern Setup

Create a condition statement that describes the Data field pattern upon which to trigger. This condition is added to the Frame ID condition.

1. Choose the **Data Condition** (Boolean operator) that describes the relationship to the reference Data Value. To use a range of values, choose In Range or Out Range.
2. Set **DLC** (Data Length Code) to any integer value from 0 to 8. It should match the DLC of the CAN message(s) on which to trigger.
3. Choose from either Motorola (default) or Intel **Byte Order**.
4. Use **Start Bit** and # **Data Bits** together to define a string of up-to-64 contiguous data bits (8 data bytes) starting from any location in the message data field. The Start Bit can be any value from 0 to 63, it is not limited to the start of a full byte or a nibble. The Start Bit value is always in LSB format, the bit numbering shown on the decoded waveform, with bit 0 at the far left and bit 63 at the far right. Make sure the Start Bit value makes sense in relation to the DLC value (e.g., a Start Bit value of 32 with a DLC Value of 4 is not going to result in a successful trigger). The total # Data Bits can be any value from 1 to 64.
5. Choose a **Sign Type** of signed or unsigned integer format.
6. Enter the reference **Data Value**. When using a range, enter the start Data Value. For Hexadecimal format values, if desired, you can precede the ID value with **0x**, but this is not necessary. Be sure to enter a Data Value that matches the DLC Value.

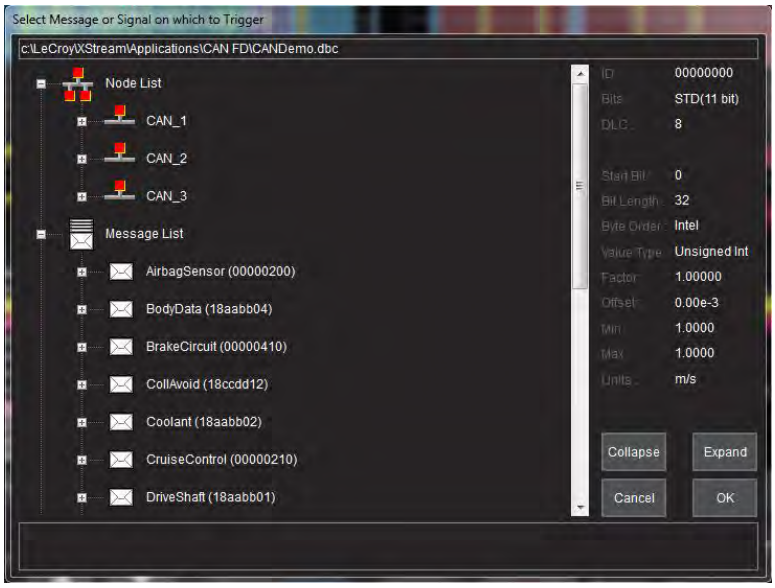
   When using a range, enter the stop value in **Data Value To**.

Symbolic Setup

This feature is only enabled when a TD Symbolic or TDME Symbolic option is installed.

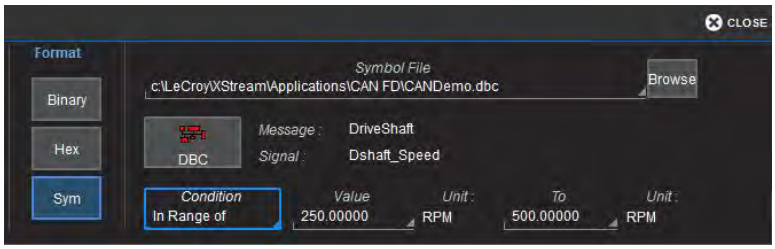
Symbolic triggering sets the trigger conditions on values defined in a symbol file.

1. Browse to and select the **Symbol File**.
2. Touch the **DBC** button to view the selected symbol file. A pop-up dialog displays a hierarchy of Nodes, Messages, and Signals.
3. Expand the list to see the values defined for each.



4. Select the element on which to trigger, then touch **OK**.
5. If you are setting up a symbolic ID+Data trigger, also create a condition statement that describes the data value upon which to trigger. All values that meet the condition will cause a trigger.
   - Choose the **Condition** (Boolean operator) that describes the relationship to the reference value, then enter the **Value**.
   - To use a range of values, choose In Range or Out Range and also enter the **To** value.

   The Units assigned to the values are taken directly from the symbol file.



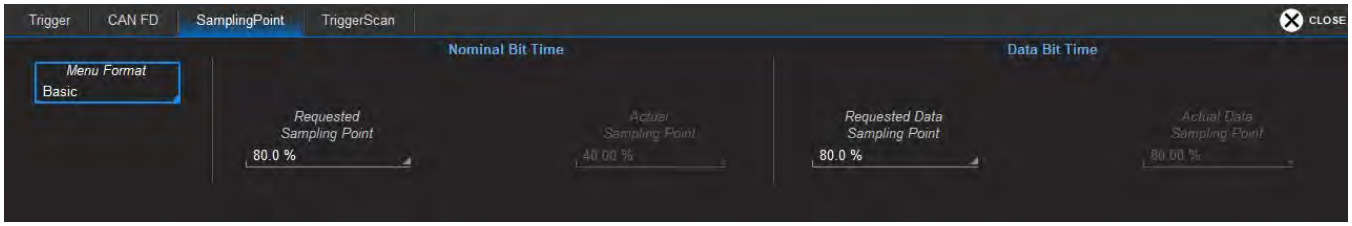
Sampling Point

The Sampling Point dialog appears behind the trigger setup dialog for entering the Nominal Bit Time(s) so that the software can determine the best sampling point. CAN FD users should also set the data phase Data Bit Time(s), which can run at different data rates than the arbitration phase.

There are two **Menu Formats** for setting the sampling point.

Note: If the trigger is linked to a decoder, these fields will already reflect the entries on the decoder Sampling subdialog. Changing values here will also change the decoder.

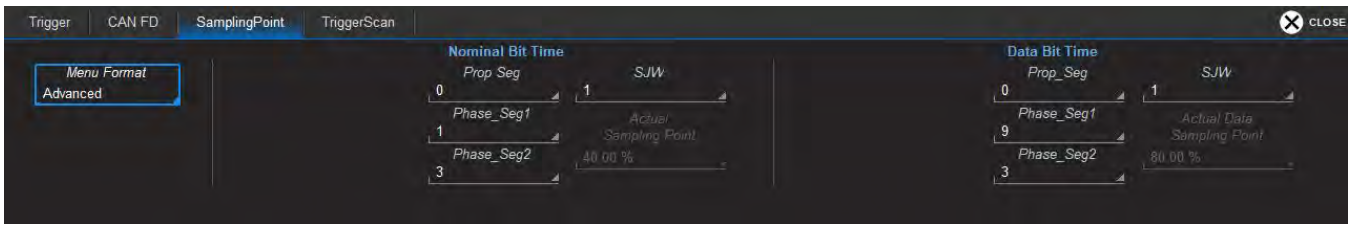
In the **Basic** format, you need only set the **Requested Sampling Point** position(s) within the bit used by your nodes, expressed as a percentage of the UI. According to the percentage(s) entered, Prop_Seg, Phase_Seg1, and Phase_Seg2 times are estimated by the software.



Note: For most users, the default Basic sampling point works sufficiently well. However, an incorrect Requested Sampling Point percentage may result in false error determination. If you see false error triggers or error frames in the decoding, use the Advanced menu format to enter the values manually.

In the **Advanced** format, you enter the number of time quantas for the following segments, according to your DUT's specification:

- **Prop_Seg**, part of the bit time used to compensate for physical network delay times.
- **Phase_Seg1**, number of time quantums before the sampling point.
- **Phase_Seg2**, number of time quantums after the sampling point.
- **SJW** (Synchronization Jump Width).



There is no field for entering the Sync_Seg, because the software always uses the value 1.

The sum of the Sync_Seg, Prop_Seg, Phase_Seg1 and Phase_Seg2 must be equal to the available time quantas in each phase for the settings to work correctly.

Tip: The length in time of one quanta depends on the clock rate of your node. For a 40 MHz clock, one quanta is 1/40 MHz, or 25 ns long. A system with a 500 kHz arbitration rate and an 8 MHz data rate has 80 time quantas for one arbitration phase bit and 5 time quantas for one data phase bit.

When the trigger is enabled, the **Actual Sampling Point** levels used are displayed on the dialog in either format. In the Advanced menu, you may see this value change as you change the values in each field.

LIN Serial Trigger Setup

To access the serial trigger dialogs:

- Touch the Trigger descriptor box or choose **Trigger > Trigger Setup** from the Menu Bar.
- Touch the **Serial** Type button, and the **LIN** Standard button.

Then, working from left to right, make the desired selections from the LIN dialog.

Source Setup

In **DATA**, select the data source input channel.

Use the **Threshold** control to adjust the vertical level for the trigger. Much like an Edge trigger, you must specify the level at which to process the incoming signal to determine whether the serial data pattern is meeting the set trigger condition.

LIN Setup

If you have not linked the trigger to a pre-set decoder, enter the **Bitrate** of the bus to which you are connected. This bit rate selection is dynamically linked to the decoding bit rate (they are always the same value). Use the Up or Down arrows to scroll the list of standard bit rates and make a selection, or touch the control and enter the value as close as possible to the actual bus rate.

Trigger Type

The oscilloscope can be configured to trigger upon a simple LIN Start of Frame (**Break**), a Frame ID value, a combination of Frame ID values and Data patterns, or error. Choose the desired Trigger Type and complete the remaining fields required to set the trigger condition:

- **Error**, complete the Checksum Error Setup below.
- **Frame ID**, complete the Frame ID Setup below.
- **ID + Data**, complete both Frame ID Setup and Data Pattern Setup below.

Setup Format

Choose to display/ enter values in **Binary** or **Hex**(adecimal) format. The selection propagates throughout the entire trigger setup. Toggling between formats does not result in loss of information, but will transform the appearance of values.

Frame ID Setup

Frame ID Setup is used to trigger upon encountering either a specific ID value or any value relative to a reference ID value (e.g., greater than *x*). Use these controls to create a condition statement that describes the trigger criteria.

Choose the **ID Condition** (Boolean operator) that describes the relationship to the Frame ID value. To use a range of values, choose In Range or Out Range.

Enter the reference **Frame ID** value. When setting a range, enter the start Frame ID.

When setting a range, enter the stop value in **To Frame ID**.

Data Pattern Setup

Create a condition statement that describes the Data field pattern upon which to trigger. This condition is added to the Frame ID condition.

Choose the **Data Condition** (Boolean operator) that describes the relationship to the reference Data Value. To use a range of values, choose In Range or Out Range.

Enter the reference **Data Value**. When using a range, enter the start Data Value. In Hexadecimal format, data must be entered as full bytes even though the minimum required acceptable entry is a nibble. If less than a full byte is entered, then a "don't care" (wildcard) X precedes the pattern values entered. Up to 8 bytes of data can be entered as a pattern value. If less than 8 bytes of data is entered for the pattern value, the data is assumed to begin at Data Byte 1 in the LIN message. If this is not desired, then add preceding or trailing X nibbles to the pattern value.

When using a range, enter the stop value in **Data Value To**.

Data Bytes defaults to the length, in bytes, of the pattern set in the Data Value. If you change the length to be less than this value, it would truncate the beginning of the pattern value. If you were to increase the pattern length, it would add "don't care" XX byte values to the beginning of the pattern value. The maximum number of data bytes is 8, per the LIN standard.

Checksum Error Setup

Enter the **Error Frame ID** to search.

Check all the error types on which to trigger: **Checksum Error**, **Header Parity**, or **Sync Byte**.

If Checksum Error is selected, also enter the **LIN Spec.** (standard) used to encode the data and the **# Data Bytes** in a message.

FlexRay Serial Trigger Setup

To access the serial trigger dialogs:

- Touch the Trigger descriptor box or choose **Trigger > Trigger Setup** from the Menu Bar.
- Touch the **Serial** Type button, and the **FlexRay** Standard button.

Then, working from left to right, make the desired selections from the FlexRay dialog.

Source Setup

These controls are use by all trigger types.

In **DATA**, select the data source input channel.

Use the **Threshold** controls to set the High and Low vertical level for the trigger. Much like an Edge trigger, you must specify the level at which to process the incoming signal to determine whether the serial data pattern is meeting the set trigger condition. FlexRay is a tri-level signal and requires two voltage threshold settings which enable the oscilloscope to distinguish between 1 and 0.

If desired, use the **Find Threshold** button to set appropriate thresholds based on the input signal characteristics.

FlexRay Setup

If you have not linked the trigger to a pre-set decoder, enter the **Bitrate** of the bus to which you are connected. This value is dynamically linked to the decoding bitrate; they are always the same. Touch the arrow buttons to scroll through standard bitrates (2.5, 5.0 or 10.0 Mb/s), or touch Bitrate and enter the value using the Adjust knob or the Virtual Keypad.

Trigger Type

The oscilloscope can be configured to trigger upon the occurrence of a simple Transmit Start Sequence (TSS), a Symbol, various Frame characteristics (ID, Cycle Count, or Qualifiers), or a FlexRay error. Choose the desired Trigger Type and complete the remaining fields required to set the trigger condition:

- **TSS**, complete Setup Format below.
- **Frame**, complete Setup Format and Frame Trigger below.
- **Symbol**, complete the Symbol Trigger below.
- **Error**, complete Error Trigger below.

Setup Format

Choose to display/enter values in **Binary** or **Hex**(adecimal) format. The selection propagates throughout the entire trigger setup. Toggling between formats does not result in loss of information, but will transform the appearance of values.

Frame Trigger Setup

Frame ID

Frame ID Setup is used to trigger upon encountering either a specific ID value or any value relative to a reference ID value (e.g., greater than *x*). Use these controls to create a condition statement that describes the Frame ID criteria.

Choose the **Condition** (Boolean operator) that describes the relationship to the Frame ID value. To use a range of values, choose In Range or Out Range.

Enter the reference **Value**. When setting a range, enter the start Frame ID in Value and the stop Frame ID value in **To**.

Tip: Use X as a wildcard ("Don't Care") in any position. To effectively exclude the Frame ID as a trigger condition, choose the Equal condition and the default value of all Xs.

Cycle Count

The Cycle Count is a decimal value between 0 and 63 correlating to the FlexRay Cycle Count numbering system. The default Cycle Count is 0. You may also specify a Repetition Factor to allow for triggering when Cycle multiplexing is being used. This condition is added to any Frame ID conditions.

As with Frame ID, create a Boolean statement describing the Cycle Count criteria. Omit the Cycle Count criteria by selecting Condition "Don't Care".

When the Cycle Count condition is set to Equal, set a **Repetition Factor** of 1, 2, 4, 8, 16, 32 or 64.

Frame Qualifiers

Defined in the FlexRay specification, the occurrence of **Payload Preamble**, **Null Frame**, **Sync Frame**, or **Startup Frame** bits may be set as trigger conditions. These conditions are added to any Frame ID or Cycle Count conditions.

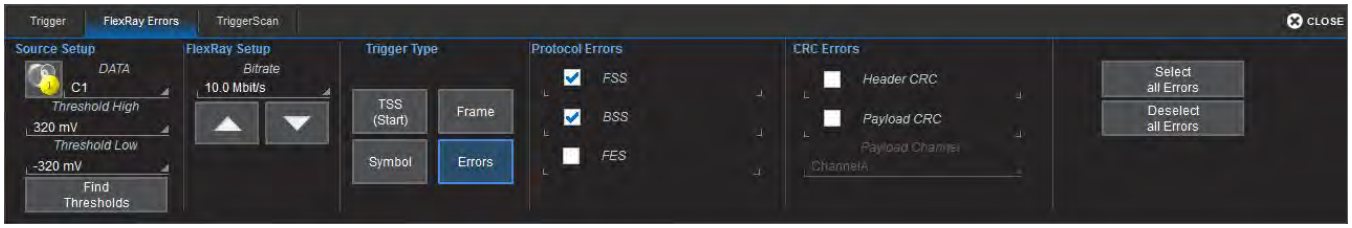
Each qualifier can be set to one or zero, or omitted by selecting "Don't Care". The trigger will be restricted to frames with matching values in the respective field.

Symbol Trigger Setup

The Symbol selection triggers upon finding any frame that includes any one of the selected symbols. Mark all the desired symbol types.

Error Trigger Setup

The Error selection triggers upon the occurence of any FlexRay protocol or CRC error. Mark all the desired error types. If you choose Payload CRC, also select the Payload input channel.



Using the Decoder with the Trigger

A key feature of Teledyne LeCroy trigger and decode options is the integration of the decoder functionality with the trigger. While you may not be interested in the decoded data per se, using the decoded waveform can help with understanding and tuning the trigger.

Stop and Look

Decoding with repetitive triggers can be very dynamic. Stop the acquisition and use the decoder tools such as Search, or oscilloscope tools such as TriggerScan, to inspect the waveform for events of interest. Touch and drag the paused trace to show time pre- or post-trigger.

Optimize the Grid

The initial decoding may be very compressed and impossible to read. Try the following:

- Increase the height of the trace by *decreasing* the gain setting (V/Div) of the decoder source channel. This causes the trace to occupy more of the available grid.
- Change your Display settings to turn off unnecessary grids. The Auto Grid feature automatically closes unused grids. On many oscilloscopes, you can manually move traces to consolidate grids.
- Close setup dialogs.

Use Zoom

The default trigger point is at zero (center), marked by a small triangle of the same color as the input channel at the bottom of the grid. Zoom small areas around the trigger point. The zoom will automatically expand to fit the width of the screen on a new grid. This will help you to see that your trigger is occurring on the bits you specified.

If you drag a trace too far left or right of the trigger point, the message decoding may disappear from the grid. You can prevent "losing" the decode by creating a zoom of whatever portion of the decode interests you. The zoom trace will not disappear when dragged and will show much more detail.

Saving Trigger Data

The message decoding and the result table are dynamic and will continue to change as long as there are new trigger events. As there may be many trigger events in long acquisitions or repetitive waveforms, it can be difficult (if not impossible) to actually read the results on screen unless you stop the acquisition. You can preserve data concurrent with the trigger by using the **AutoSave** feature.

- AutoSave Waveform creates a .trc file that copies the waveform at each trigger point. These files can be recalled to the oscilloscope for later viewing. Choose **File > Save Waveform** and an Auto Save setting of **Wrap** (overwrite when drive full) or **Fill** (stop when drive full). The files are saved in D:\Waveforms.
- AutoSave Table creates a .csv file of the result table data at each trigger point. Choose **File > Save Table** and an Auto Save setting of **Wrap** or **Fill**. The files are saved in D:\Tables.

Caution: If you have frequent triggers, it is possible you will eventually run out of hard drive space. Choose Wrap only if you're not concerned about files persisting on the instrument. If you choose Fill, plan to periodically delete or move files out of the directory.

Measure/ Graph

The installation of the Measure/ Graph package (included with any -DME or -TDME option) adds a set of measurements and plots designed for serial data analysis to the oscilloscope's standard measurement capabilities. Measurements can be quickly applied without having to leave the waveform or tabular views of the decoding.

Note: This functionality was formerly offered as part of -TDM options and the ProtoBus MAG software option. The features described in this section should be present If you have either of these installed on your oscilloscope.

Serial Data Measurements

These measurements designed for debugging serial data streams can be applied to the decoded waveform. Measurements appear in a tabular readout below the grid (the same as for any other measurements) and are in addition to the result table that shows the decoded data. You can set up as many measurements as your oscilloscope has parameter locations.

Note: Measurements appear in the Serial Decode sub-menu of the Measure Setup menu and may have slightly different names. For example, the CAN sub-menu has measurements for CANtoValue instead of MsgToValue, etc. The measurements are the same.

Measurement	Filters	Description
AnalogToMsg	ID, Data, Analog	Computes time from crossing threshold on an analog signal to start of first message that meets conditions. If the message condition precedes the analog condition, no measurement is performed.
BusLoad	ID, Data	Computes the load of selected messages on the bus (as a percent).
DeltaMsg	ID, Data	Computes time difference between two messages on a single decoded line.
MsgBitrate	ID, Data	Computes the bitrate of selected messages within the decoded stream.
MsgToAnalog	ID, Data, Analog	Computes time from start of first message that meets conditions to crossing threshold on an analog signal. If the analog condition precedes the message condition, no measurement is performed.
MsgToMsg	ID, Data	Computes time from start of first message that meets conditions to start of the next message that meets conditions.
MsgToValue	ID, Value	Extracts a selected portion of the data to a measurement parameter location, with optional conversion of value. Data may be selected by ID and/or data field position.
NumMessages	ID, Data	Computes the total number of messages in the decoding that meet conditions.
Time@Msg	ID, Data	Computes time from trigger to start of each message that meets conditions.

Graphing Measurements

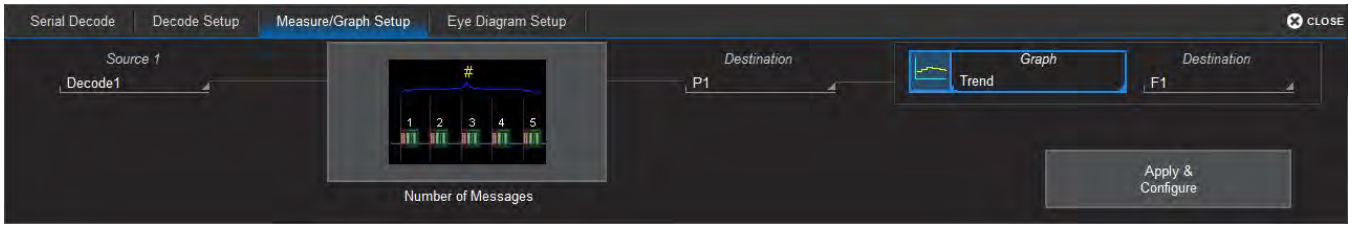
The Measure/Graph package include simplified methods for plotting measurement values as:

- **Histogram** - a bar chart of the number of data points that fall into statistically significant intervals or bins. Bar height relates to the frequency at which data points fall into each interval/bin. Histogram is helpful to understand the modality of a parameter and to debug excessive variation.
- **Trend** - a plot of the evolution of a parameter over time. The graph's vertical axis is the value of the parameter; its horizontal axis is the order in which the values were acquired. Trending data can be accumulated over many acquisitions. It is analogous to a chart recorder.
- **Track** - a time-correlated accumulation of values for a single acquisition. Tracks are time synchronous and clear with each new acquisition. Track can be used to plot data values and compare them to a corresponding analog signal, or to observe changes in timing. A parameter tracked over a long acquisition could provide information about the modulation of the parameter.

To graph a measurement, just select the plot type from the Measure/Graph dialog when setting up the measurement. All plots are Math functions that open along side the deocoding in a separate grid.

Measure/Graph Setup Dialog

Use the Measure/Graph Setup dialog to apply serial data measurement parameters to the decoded waveform and simultaneously graph the results. This dialog appears behind the Decode Setup dialog and is active when measurements are supported.



1. Select the **Measurement** to apply and the **Destination** parameter (P*n*) to which to assign it.
2. The active decoder is preselected in **Source 1**, indicating the measurement will be applied to the decoder results; change it if necessary. If the measurement requires it, also select an appropriate Source 2 (such as an analog waveform for comparison).
3. Optionally:
   - Touch Graph to select a plot type. Also select a **Destination** function (F*n*) for the plot.
   - Touch **Apply & Configure** to set a filter, gate or other qualifiers on the measurement.

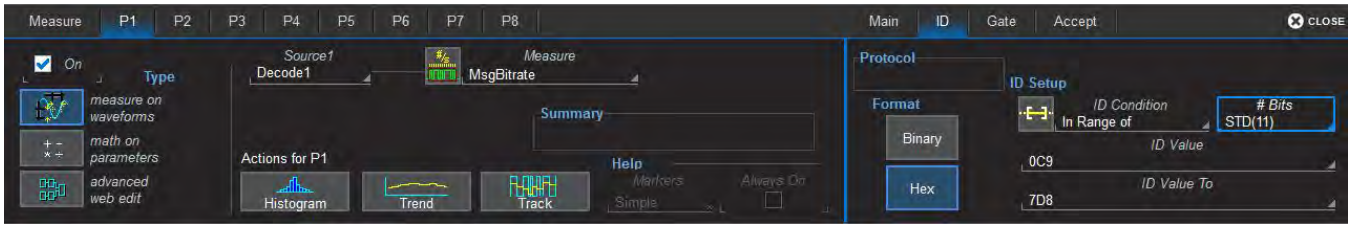
Filtering Measurements

Certain serial decode measurements can be filtered to include only the results from specified IDs or specific data patterns. As with all measurements, you can set a gate to restrict measurements to a horizontal range of the grid corresponding to a specific time segment of the acquisition.

After creating a measurement on the Measure/ Graph Setup dialog, touch **Apply&Configure**. The touch screen display will switch to the standard Measure setup dialogs for the parameter you selected. Set filter conditions on the right-hand subdialogs that appear next to the P*n* dialogs.

ID Filter

This filter restricts the measurement to only frames/ packets with a specific ID value. Settings on this dialog may change depending on the protocol.

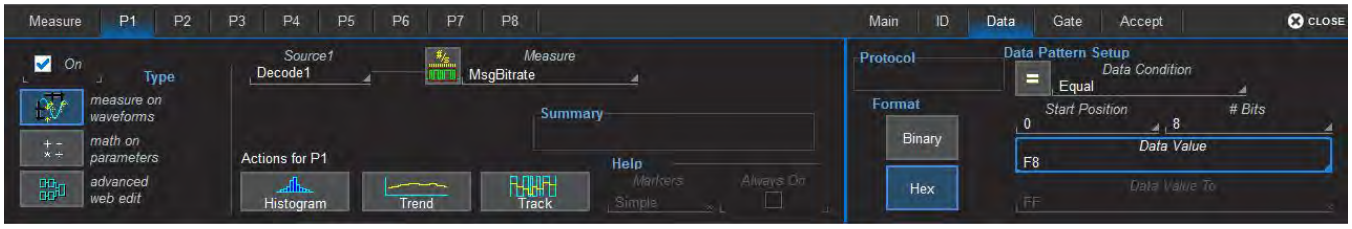


1. On the Main subdialog, choose to **Filter** by **ID** or **ID + Data**.
2. On the **ID subdialog**, choose to enter the ID in **Binary** or **Hex**(adecimal) format.
3. If the field appears, select the # **Bits** used to define the frame ID. (This will change the ID Value field length.) For CAN, choose from standard 11-bit or extended 29-bit.
4. Using the **ID Condition** and **ID Value** controls, create a condition statement that describes the IDs you want included in the measurement. To set a range of values, also enter the **ID Value To**.

Tip: On the value entry pop-up: use the arrow keys to position the cursor; use Back to clear the previous character (like Backspace); use Clear to clear all characters.

Data Filter

This restricts measurements to only frames containing extracted data that matches the filter condition. It can be combined with a Frame ID filter by choosing **ID+Data** on the Main subdialog.



Use the same procedure as above to create a condition describing the **Data Value(s)** to include in the measurement. Use "X" as a wild card ("Don't Care") in any position where the value doesn't matter.

Optionally, enter a **Start Position** within the data field byte to begin seeking the pattern, and the **# Bits** in the data pattern. The remaining data fields positions will autofill with "X".

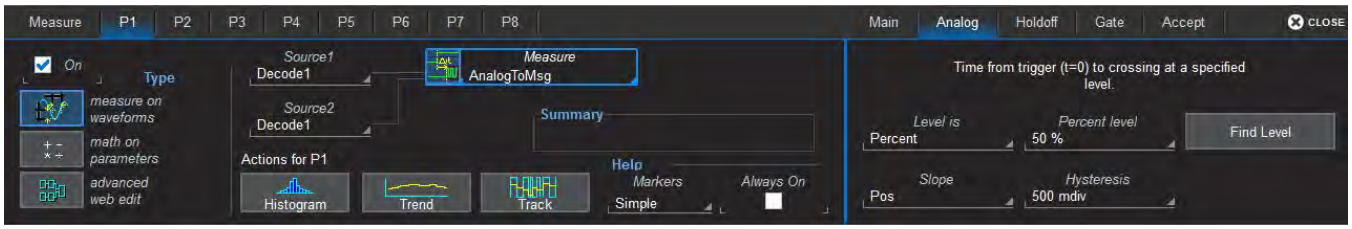
Note: For MsgtoMsg measurements, the data condition is entered twice: first for the Start Message and then for the End Message. The measurement computes the time to find a match to each set of conditions.

Analog Settings

The measurements AnalogToMsg and MsgToAnalog allow you to use crossing level and slope to define the event in the Analog waveform that is to be used as the reference for the measurement.

As with the decoder, Level may be set as a percentage of amplitude (default), or as an absolute voltage level by changing **Level Is** to Absolute. You can also use **Find Level** to allow the oscilloscope to set the level to the mean Top-Base amplitude.

A **Slope** and **Hysteresis** selection is also offered. The width of the Hysteresis band is specified in milli-divisions. See Setting Level and Hysteresis for more information on using these controls.



Eye Diagram Tests

The -DME and -TDME options provide easy eye diagram setup and eye mask testing.

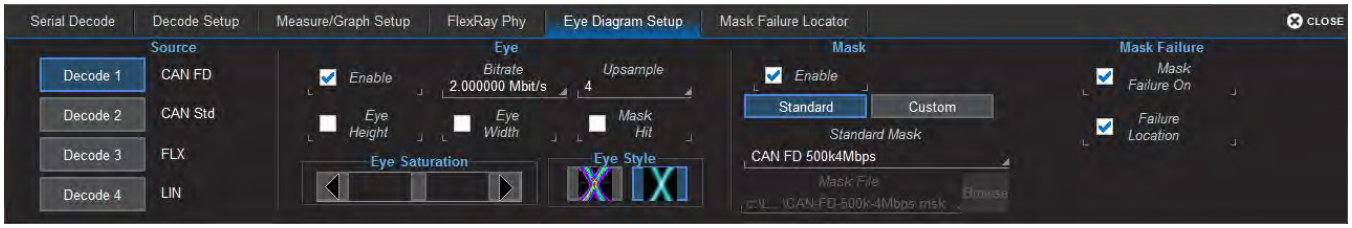
Eye diagrams are a key component of serial data analysis. They are used both quantitatively and qualitatively to understand the quality of the signal communications path. Signal integrity effects such as intersymbol interference, loss, crosstalk and EMI can be identified by viewing eye diagrams, such that the eye is typically viewed prior to performing any further analysis.

Each pixel in the eye takes on a color that indicates how frequently a signal has passed through the time and voltage specified for that pixel. The eye diagram shows all values a digital signal takes on during a bit period. A bit period (also referred to as unit interval, or UI) is defined by the data clock, whether explicit or extrapolated depending on the protocol.

Eye diagrams show the acquired signal that is currently being shown on the decoder result table. They are not persistent, as are eye diagrams generated in some other serial data analysis software; the eye will change from one acquisition to the next and when the result table is filtered. Our recommended approach for using the eye diagrams is to:

- Make single shot acquisitions with decoder and eye diagram enabled to check that both are working correctly.
- Run a normal acquisition with Mask Testing and Stop On Failure enabled in the Mask Failure Locator, or with a Pass/ Fail test set on one of the eye parameters.

Eye Diagram Setup Dialog



Create Eye Diagram

Open the **Eye Diagram Setup** dialog and select the **Decode** for which to create an eye diagram.

Under Eye, check **Enable** to display the eye diagram.

The **Bitrate** is automatically read from the decoder setup. This value is linked to the decoder bit rate setting, and changing it in either place will update both settings.

The **Upsample** factor increases the number of sample points used to compose the eye diagram. Increase from 1 to a higher number (e.g. 5) to fill in gaps. Gaps can occur when the bitrate is extremely close to a submultiple of the sampling rate, such that the sampling of the waveform does not move throughout the entire unit interval. Gaps can also occur when using a record length that does not sample a sufficiently large number of unit intervals.

The **Eye Style** may utilize color-graded or analog persistence:

- With **color-graded** persistence  , pixels are given a color based on the pixel's relative population and the selected Eye Saturation. The color palette ranges from violet to red.
- With **analog** persistence  , the color used mimicks the relative intensity that would be seen on an analog oscilloscope.

Use the **Eye Saturation** slider to adjust the color grading or intensity. Slide to the left to reduce the threshold required to reach saturation.

Choose to display the **Eye Height**, **Eye Width**, or **Mask Hit(s)** measurement parameters. These are added to the Measure table in the first open parameter slots.

Eye Mask Test

Under Mask, check **Enable** to turn on eye mask testing.

Select to use either a **Standard** or **Custom** mask, then either select the **Standard Mask** or **Browse** to and select your custom **Mask File**.

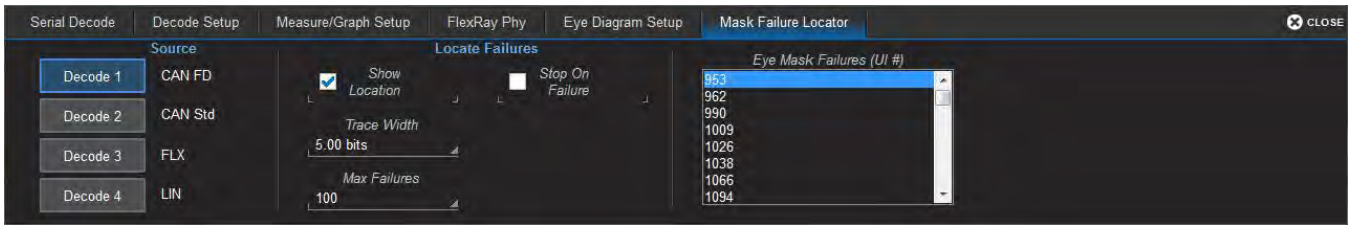
Tip: Masks previously created on the instrument are stored in D:\Masks. For ease of selection, copy other .msk files to this location.

Check **Mask Failure On** to mark the parts of the eye diagram that fail the mask test. Mask violations appear as red failure indicators where the eye diagram intersects the mask.

Check **Failure Location** to display the Mask Failure Locator dialog.

Mask Failure Locator Dialog

Use this dialog to quickly search the acquisition for eye diagram mask test failures.



In **Trace Width**, enter the number of UIs surrounding the mask violation to display as "padding."

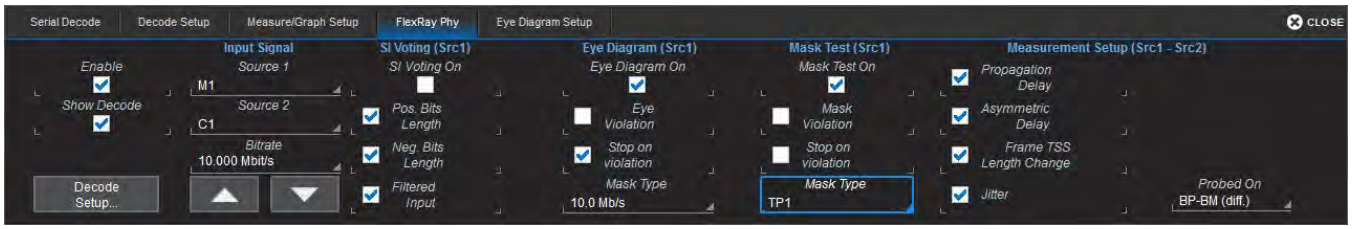
Check **Stop On Failure** to stop acquisition whenever an eye mask failure occurs.

Enter the **Max Failures** to retain in the Eye Mask Failure list.

Select from the **Eye Mask Failure** list to mark and zoom to the location of that failure. Yellow circles appear over the red failure indicators to show the location of the failure.

FlexRay Physical Layer Testing

The FlexRaybus TDMP option includes physial layer tests specified by the FlexRay standard. When this option is activated, an additional FlexRay Phy dialog appears following the Measure/Graph Setup tab.



The dialog contains selections for SI Voting, Eye Diagram, Mask Testing, and Measurement Setup to quickly align signals to the FlexRay specification. These tests can be run simultaneously with the protocol-level decoding to quickly identify where errors and anomalies occur.

General Settings

Mark **Enable** to enable physical layer testing.

Mark **Show Decode** to display the signal decoding during the physical layer test. The current FlexRay decoder settings from the Decode Setup dialogs will apply.

The **Decode Setup** button provides quick access to the Decode Setup dialog.

Input Setup

Touch **Source 1** and choose the transmitter test signal.

Note: The Source 1 signal is used for SI Voting, Eye Diagram, and Mask Test functions. Source 1 and Source 2 are both used for the Measurement function.

Optionally, touch **Source 2** and choose the receiver test signal for physical layer measurements.

Enter the **Bitrate** as close as possible to the the bitrate of the bus to which you are connected. Use the arrows to scroll through the standard bitrates (2.5 Mb/s, 5 Mb/s and 10 Mb/s) and make a selection, or touch the field to enter the value using the Adjust Knob or the Virtual Keypad.

SI Voting Setup

SI Voting was developed as an alternative for users who found eye diagrams and masks to result in failed tests on signals which would actually pass in practice. SI Voting filters every bit pattern of the form 00010 and 11101 (i.e., all single-bit pulses), then measures the bit-length between a point on the first edge and a symmetrical point on the second edge, providing a set of time deltas between the two edges. The result is then tested against the specification constraints on shortest and longest acceptable bit-length, maximum acceptable bit-length variation, longest acceptable edge time, and Top and Base levels.

Note: SI Voting disables the application and display of FlexRay physical layer measurements.

Mark **SI Voting On** to enable voting.

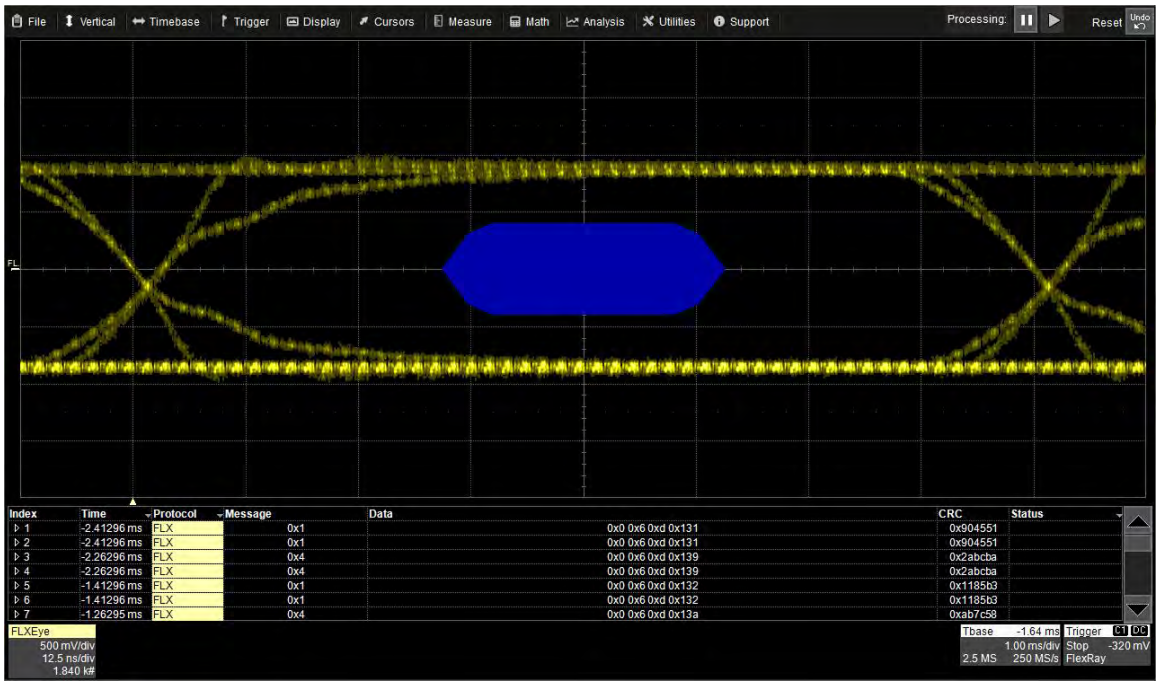
Mark each item on which to vote:

- **Pos. Bits Length**
- **Neg. Bits Length**
- **Filtered Input**

Eye Diagram Setup

Eye diagrams are created by slicing the bits transmitted in the FlexRay signal and superimposing each bit on to an eye diagram. The signal is sliced based on measurements taken at the falling edge of the first Byte Start Sequence (BSS) and the time between consecutive BSS symbols. These measurements allow the algorithm to compute the rate of the embedded clock and slice the FlexRay waveform in to sub-waveforms that are one bit in length. The clock uses a constant bitrate specified by the user and is resynchronized on every BSS. These sub-waveforms are then scaled to fill 8 horizontal divisions on the oscilloscope and represent 1 Unit Interval (UI) in the eye diagram and superimposed on top of each other.

The resulting Eye Diagram behaves as detailed in the FlexRay specification (Electrical Physical Layer Application Notes, Chapter 2.19) and produces an eye of all the bits in a message by synchronizing on the BSS transition. Then, it uses a constant clock to slice bits up to the next BSS. Masks are defined for 2.5, 5.0, and 10 Mb/s speeds.



FlexRay eye diagram.

Mark **Eye Diagram On** to enable eye diagramming.

Note: It will take more time to generate an eye diagram from long acquisitions than from short.

Mark **Eye Violation** to display violations. Mask violations appear as red failure indicators where the eye diagram intersects the mask.

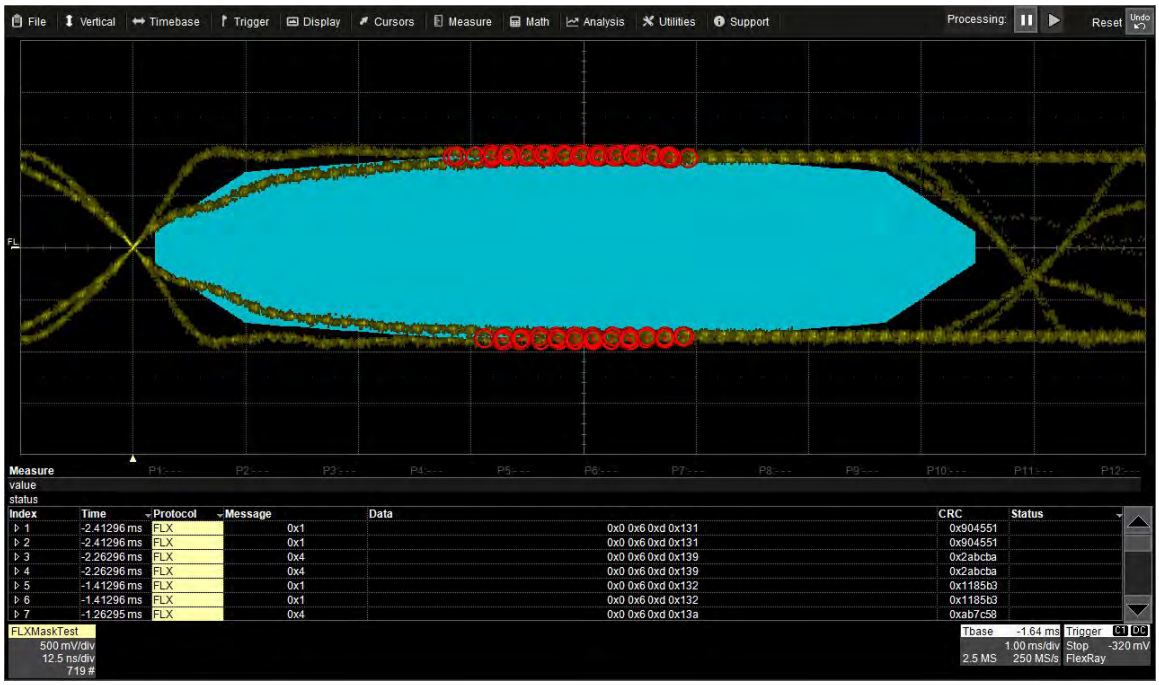
Mark **Stop On Violation** to stop the acquisition whenever an eye violation occurs. This will allow time to examine the decoding compared to the eye diagram violation and understand exactly where the violation occurred.

Mask Test Setup

Mask testing can be performed on an eye diagram with masks defined at TP1 and TP11. The mask is aligned horizontally by computing the time for a single UI and centering it on the display. The mask is centered vertically around 0V.

As explained in the FlexRay Electrical Physical Layer Specification, Chapter 7, the mask test makes an eye of the bits following any transition (rising and falling) with no clock recovery. Only a 10 Mb/s mask is defined in the specification, but assuming a controller passing at 10 Mb/s also passes at other bitrates, other masks are provided for different test points and setup configurations.

Note: This eye is computed differently than the Eye Diagram selection, as it is done without clock recovery, according to the specification. If you have already selected Eye Diagram, selecting Mask Test will create a second eye for the mask test.



FlexRay eye diagram mask test showing red violation indicators.

Mark **Mask Test On** to enable the mask test.

Mark **Mask Violation** to show violations on the display. Mask violations appear as red failure indicators where the eye digram intersects the mask.

Mark **Stop on violation** to stop the acquisition whenever a mask violation occurs. This will allow time to examine the decoding compared to the mask violation and understand exactly where the violation occurred.

Touch **Mask Type** and choose from:

- TP1
- TP1 Bus Driver
- TP11
- TP11 Active Star

Physical Layer Measurement Setup

You can apply four measurements defined in the FlexRay physical layer specification to characterize the timing properties of the signal along the communication channel. Results appear in the measurement readout table below the grid.

Clear **SI Voting On** to enable measurement selection.

Mark each measurement you wish to apply to the test signal:

- **Propagation Delay**, measured on two points along the communication channel from the emitter node module to the receiver node module. Propagation Delay characterizes the propagation time of the signal using the first transition of the Byte Start Sequence (BSS).
- **Assymetric Delay**, measured on two points along the communication channel from the emitter node module to the receiver node module. Assymetric Delay characterizes the difference in delay between rising and falling edges.
- **Frame TSS Length Change**, measured on two points along the communication channel from the emitter node module to the receiver node module. Frame TSS Length Change measures the change in width of the Transition Start Sequence (TSS).
- **Jitter**, measured on a single point, usually the receiving node. Jitter measures the change of length betwee the last BSS and the FSS. This should usually be 1µs.

Touch **Probed On** and choose the type of line you are measuring:

- **BP-BM (diff)** - a differential signal on the communication channel
- **RxD-TxD (dig.)** - a two-level digital signal in the communication controller intrface.