Debugging CAN Buses Using Mixed Signal Oscilloscopes

Andreas Siegert, Agilent Technologies

This presentation will introduce new features of the todays Real Time Oscilloscope generation to debug serial busses, e.g. CAN. New capabilities not just offer decoding functions but allow triggering on the content itself. Specific identifiers, payload, even failure indications like error checksums or error frames can be detected and recorded within seconds.

Physical and Logical Layer Evaluation

In general the performance of serial buses can be evaluated in two different ways: the physical and the logical layer.

Physical layer measurements will comprise standard measurements like voltage levels, rise time / fall time, correct bit rate etc. All these measurements are well-known and will typically be performed on real-time oscilloscopes.

The Logical Layer gives however inside information about the CAN frame content itself. Besides the payload CAN buses contain helpful information like frame type, identifier, size of the payload, checksum and finally the acknowledge bit from other nodes in the network.

In former times such logic and protocol analysis functionalities have only been found in specific protocol analyzers. The today's scope generation does however offer similar functionalities as well. So called Mixed Signal Scopes are in fact three instruments in one: analogue scope, protocol analyzer and logic analyzer. Mixed Signal Scopes have a market share of app. 70% of all scopes up to 1 GHz bandwidth.

But they won't be able to completely replace protocol analyzers, due to a scope-specific disadvantage: scopes will never be able to record and to decode very long traces of more than a few seconds! The technical limits will be explained later.

The real strength of an oscilloscope is however the capability, to decode CAN signals and to show several time-correlated analogue waveform at the same time! This correlated view gives the user the tool not just to see that protocol errors occur, but to find possible reasons for that in the analogue waveform.

This presentation will focus on Protocol Layer measurements.

Typical Embedded Design

In our presentation we show a simplified electrical schematic of a typical embedded system. This could be an industrial application, cell phone, an automotive ECU, a digital control unit for an aerospace/defense system, a handheld GPS, etc. What all these embedded systems have in common is that they are typically based on control by a microcontroller / microprocessor / DSP, they typically interface with the real-world with analog inputs and outputs, and they usually have within them one or more serial buses used for transfer of data between chips or used to control and communication between remote sub-systems.

Why are there so many different types of cereal down the breakfast cereal aisle?
One bus doesn't fit all. If it did, then it would be very costly and complex. Some buses, such as SPI, have been designed with simplicity of implementation and low cost in mind. Others like CAN have been designed for faster communication when data transfer speeds are important, or perhaps robustness of communication. In other words, lots of information embedded within a single frame or packet.
Some applications, such as industrial, cars and aircraft require buses that insure

safety and security with lots of error detection and error recovery.
When you step on the brake pedal and the serial packet gets lost or corrupted, don't you won't the automotive brake system's ECU to send the packet again.

Another example in the industry. Precision cross winders and high speed winders are controlled via CAN. Whenever the yarn or fiber impends to rip you want the controller to detect this critical situation as quick as possible.

And finally, some buses have been optimized for transmitting over long distances and in electrically noisy environments, such as between remote terminals in an aircraft.

The CAN bus may be used for more safety critical systems. It is not designed for highest speeds, but eliminates the disadvantages of other buses:

- Spontaneous transmission to one or to several other node
- Build-in security checks like CRC (Cyclic Redundancy Check), DLC (Digital Length Code) and ACK (Acknowledgement)
- Overload Indicator

Just to mention a few ones.

Why do engineers need serial bus functionality in their scopes?

The key reason why serial bus functionality in a scope is so important is so that engineers can easily correlate between protocol and physical layer signals. A serial bus protocol analyzer or monitor can provide engineers with a very high level of serial protocol information, but they can't tell if a particular bit met its required amplitude or timing specifications.

It is often necessary to measure the physical layer characteristics of serial buses, such as the voltage levels of bits, as well as bit timing; and be able to time-correlate particular bits to protocol transfers. This is especially important for the higher speed and longer distant type buses such as CAN, FlexRay and MIL-STD 1553.

Engineers also need to trace serial communication and message transfers… and if corrupted, be able to correlate corrupted protocol transfers to actual signals. Perhaps the corruption was due to noise or system interference.

And finally, it is often necessary to time-correlate various analog and digital I/O signals in the system to serial communication. For example, if a particular serial bus transmission is suppose to re-position an actuator in the system, did the motor control sub-system actually receive the instruction and generate the required analog signals for repositioning?

CAN Protocol, "brute force" decoding

The CAN protocol foresees both the Standard ID format as well as the Extended ID format.

Believe or not, many engineers are still using this old and very tedious method of manually/visually decoding serial buses on their oscilloscopes. In the presentation we show an example of decoding an CAN remote transfer (RTR) request frame. The engineer first sets up his scope to randomly capture the CAN serial bus signal using one channel of the oscilloscope. He must then divide the waveform into one-bit time slices and identify the stream of "1's" and "0's". And then using knowledge of the CAN protocol, the engineer must re-assemble the stream of data based on known fields in order to decode the data stream. This is a very tedious and time-consuming process.

But what if after manually decoding this particular stream of data, the engineer discovers that this wasn't the particular frame that he was looking for. After all, using the scope's conventional edge triggering, the scope was only able to capture a random stream of data. He could simply try again for another random stream of data… and start all over!

There must be a better way… and there is. Many of today's mid-range and higher performance oscilloscopes have built-in

serial bus triggering and decoding capabilities. No more bit-counting!

Decodable low speed buses are e.g.:
CAN. LIN, Flexray, I2C, SPI, UART / RS232, JTAG

Examples for High Speed buses:
PCIe, SATA, USB, MIPI, SAS

CAN: Controller Area Network

The CAN bus is a 2-wire differential bus with an embedded clock. But what we mean by "2-wire" is actually a single differential signal consisting a "High" CAN signal (CAN_H) and a "Low" CAN signal (CAN_L). Probing this bus requires a differential active probe, which means that a just single oscilloscope channel is required to trigger on, capture, and decode this bus.

Alternatively and just for decoding reasons it may be sufficient just to use a (single ended) standard passive probe with 500 MHz bandwidth.

The CAN bus was originally developed by Bosch for automotive applications, but it is used today in a wide range of industrial automation equipment and medical equipment. Its key attribute is that it can be used for relatively long range communication up to about 10 meters. Since it is a differential bus, it has a lot of noise immunity; and the physical bus itself is typically a twisted wire pair. Although this bus is specified to operate up to 1 Mbps, the highest baud rates used today is 500 kbps.

Hardware- and Software-based Decoding: two different technologies which look identical, at first glance

Today's scopes use different decoding approaches.

The most common one is realized in software. The scope first captures a piece of waveform and decodes afterwards. This technology is called Software Trigger. It's relatively cheap and sufficient whenever post-processing is acceptable and you don't have any time constraints.

Software-based decoding is very similar to the manual/visual bit-counting technique that we showed earlier… but much faster. But it can still be relatively slow, especially when using deep memory on the scope. Software-based serial decoding on some scopes can take as long as 5 seconds if deep memory is turned on.

This technology finds its limits, whenever you want to trigger on specific CAN contents or on CAN errors. You need a technology which does not find such events just by chance. You want to find each individual event with 100% reliability.

Scopes which grant such functionality work with hardware protocol trigger and decoding. Which means, the scope decodes the CAN signal in real-time. Whenever the defined trigger event occurs, it will identify and trigger on it with 100% guarantee.

Hardware trigger functionality is completely realized in FPGA, because it's much quicker that any build-in PC plus software.

Waveform updates can be maintained at rates up to 100,000 waveforms/sec, and decode updates are as high as 60/sec, which is the maximum refresh rate of the scope's display.

Fast decode update rates are important for two key reasons. First of all, faster decode update rates improve the probability of capturing and displaying communication errors. Secondly, fast updates rates improve the usability of the scope. If a scope's display is updating just once every 5 seconds, then the scope can become sluggish and difficult to use.

Automatic "Search & Navigation"

Most scopes offer the capability to search and to identify particular events of interest quickly. After capturing a stream of serial data, you just need to set up the search criteria similar to triggering, and then navigate to each marked event using the specific front panel navigation keys.

Multi-Bus Decoding

A relatively new feature in the market is the capability to decode up to 4 buses in parallel. Imagine you have got several CAN buses or a combination of different bus types like CAN, LIN, SPI, I2C etc. on your board. Probably plus additional analogue signals which you want to evaluate and to display at the same time.

Limits of Real Time Oscilloscopes: no continuous recording without dead times

A very common question: does a scope allow to record minutes or even hours without interrupts? Unfortunately the answer is a clear No.

One point is very often not really known by scope users: oscilloscopes have dead times. They are not able to record very long traces without interruption. Depending on the scope memory depth and the sample rate max recording times will vary between a few 100ms and a few seconds. This is related to the scope architecture: an oscilloscope triggers on an event and stores the sampled waveform in its deep memory. After that the waveform will be reconstructed, visualized, measurements will be performed. This process takes a small amount of time and during that time the scope will be completely blind for anything what happens on its inputs. All manufacturers work hard to reduce dead times, but they will never shrink down to zero.

For long bus recordings without the need to analyze the analogue waveform a CAN protocol analyzer will be the better solution.

Overcoming of Real Time Scope Limits by Focusing on the Real Parts of Interest (Segmented Memory)

There is however a great tool available, which very often comes as standard feature on oscilloscopes. But many scope users are not aware of this feature.

With Segmented Memory (other vendors have different names for the same functionality) you can split your deep memory into small slices.

The idea behind Segmented Memory: a lot of buses like CAN work with data bursts and shorter or longer idle times in between. In standard mode a scope would trigger and capture the whole waveform, including all idle times. This is literally a misuse of precious memory depth.

Segmented Memory captures just the parts of interest, which may either mean all frames or just specific frames. But no idle times!

Imagine you try to find the root cause for error frames, which occur from time to time.

Within standard acquisition mode the chance is extremely low to capture such rare anomalies for further analysis.

Segmented Memory acquisition can effectively extend the scope's total acquisition time by dividing the scope's available acquisition memory into smaller memory segments. The scope then selectively digitizes just the important portions of the waveform under test at a high sample rate as illustrated in the presentation. This enables your scope to capture many successive single-shot waveforms with a very fast re-arm time — without missing important signal information.

Segmented memory option can be used in conjunction with serial bus triggering and decoding. With segmented memory acquisition, you can set up the scope to trigger on and capture selective frames/packets of data within a serial bus stream. This optimizes the scopes acquisition memory in order to capture consecutive occurrences of selective data… without missing important waveform information. And with precise time-tagging, you know the precise time between each captured event.

You just capture what you really need for debugging. All idle times – or frames you are not interested in – won't be recorded.

After a segmented memory acquisition is performed, you can then easily view all captured waveforms overlaid in an infinite-persistence display, as well as quickly scroll through each individual waveform segment. And in the case of serial bus applications, the scope also automatically provides protocol decode of each captured packet/segment. Although most of the signal dead/idle-time between each segment is not captured, the scope provides a time-tag for each segment so that you know the precise time between each pulse, each burst, or each serial packet captured.

In the example shown in the presentation, we have captured 2000 consecutive occurrences of CAN error frames. The total acquisition time as indicated by the time-tag of segment #2000 was over 250 seconds. Capturing this much data using conventional acquisition memory would be impossible.

Measurement Example #1: Capturing 1000 Consecutive CAN Frames

This slide shows a CAN bus measurement with the scope setup to trigger on every start-of-frame (SOF) condition. Using this triggering condition with the segmented memory acquisition mode turned on, the scope easily captures 1000 consecutive CAN frames for a total acquisition time of 2.385 seconds. After acquiring the 1000 segments/CAN frames, we can then easily scroll through all frames individually to look for any anomalies or errors. In addition, we can easily make latency timing measurements between frames using the segmented memory's time-tagging.

Measurement Example #2: Capturing 1000 Consecutive CAN Data Frames wit ID: 07F

After capturing 1000 consecutive CAN frames based on a start-of-frame (SOF) triggering condition, perhaps we notice something peculiar about one specific frame ID, such as data frame 07F, and we now want to further analyze our CAN serial bus activity, but only when data frame 07F is generated. We can change the scope's trigger condition from triggering on SOF (all frames) to trigger on ID: 07F in order to selectively capture 1000 consecutive occurrences of just frame ID 07F as shown in this slide. In this example, the scope captured nearly a 20 seconds time span. Notice in the scope's protocol lister/table display, each captured frame has the same frame ID (07F). Also notice that the scope captured an error frame (highlighted in red) during segment #986, which occurred 18.83 seconds after the first captured frame/segment.

Measurement Example #3: Capturing 100 Consecutive CAN Error Frames

The next step in this CAN serial bus debug process while using segmented memory acquisition might be to then setup the scope to selectively capture consecutive occurrences of all flagged error frames, regardless of the frame ID. To do this we would change the scope's trigger condition from triggering on data frame ID: 07F to trigger on Error Frames. But since error frames occur very infrequently, we will change the number of segments to capture from 1000 segments to just 100 segments.
In this slide we can see that the scope captured 100 consecutive CAN error frames over an approximate 12.5 second time-span. We can see from the protocol lister that error frames appear to be occurring during just frame IDs 07F, 0BD, 000. Also notice that segment #98, which is frame ID: 07F and is shown in the zoomed waveform display, contains a narrow glitch near the end of the frame. Perhaps this glitch is the culprit that is causing error frames to sometimes occur during frame ID: 07F.

Segmented Memory Acquisition Specifications

If you believe that a scope with a Segmented Memory acquisition mode may help you debug and characterize your designs, then these are the characteristics/specifications of this

oscilloscope acquisition mode that you should consider and compare:

- Minimum re-arm time (time between segments)
- Minimum time-tag resolution
- Maximum acquisition time
- Maximum number of segments

Segment sources including:

- Analog channels
- Analog and digital channels and serial bus decode

Difference between Analogue and Digital Inputs

Mixed Signal Oscilloscopes offer both analogue and digital inputs which you can use for decoding purposes. Digital channels – as the name already says – work in principle like a logic analyzer. The scope asynchronously samples the analogue signal and takes the decision: all levels above the threshold are translated into logical Ones, all samples below as logical Zerow. Any signal anomaly won't be detected. Glitches, too slow rise times, wrong voltage levels, all these information will be lost.

Furthermore the speed of the digital sampler will limit the best possible time resolution. For a digital sample rate of e.g. 1 GSa/s the best possible time resolution is 1 ns. As described above, the vertical resolution is one bit (0 and 1).

A logic analyzer allows saving in a so called "state qualified mode" and to sample synchronously to a reference clock signal. This method saves a lot of memory because for each individual logical state just one sample needs to be stored.

Scopes don't offer synchronous sampling. Per logical state they will capture and save much more sample information. The number of samples depends on the analogue sample rate (for analogue channels) or the digital sample rate (for digital inputs).

Digital inputs are ideally suited for signals with well-known and qualified analogue quality. Such signals can be decoded and visualized perfectly.

You can use the analogue inputs to evaluate other buses or analogue signals on your design in parallel. This gives you a maximum of up to 20 or 36 channels (depending on the scope platform).

Whenever the quality of the physical layer is unclear to you, the input of choice should be analogue. The decoding engine behind is exactly the same.

Transmitter and Receiver Test

In this presentation we've so far focused on the evaluation of the transmitter part. It can however be very useful, to test the receiver part of a CAN node as well. Such tests are called Receiver Stress Test. The sensitivity of a CAN receiver can be tested by applying a signal, which is knowingly distorted. For instance the amplitude can be tuned down to a critical level, from which on the receiver starts to detect bit errors.

Further parameters for Receiver Stress Tests may be additional noise insertion (random or with a specific crest factor) and / or jitter (typically sinusoidal jitter). Stressed signals can be generated in different ways. A typical solution would be a pattern generator plus an additional jitter or noise source. CAN signal and distortion signal will be combined, either within the pattern generator or externally. Software tools exist, with which all instruments can be controlled automatically. The final end result is a curve which gives the operator a clear understanding, within which conditions his receiver will work error-free – and when he should expect bit errors.

A second way to generate stressed signals is extremely easy. You can record live data streams with an oscilloscope and reproduce the signal with an Arbitrary Waveform Generator (ARB).

Minimum Sample Rate for Accurate Waveform Reconstruction and Decoding

Whenever you want to record as much waveform as possible, the scope memory depth typically gives you the upper limit. The only option to squeeze a longer trace

into the deep memory is by reducing the sample rate to a reasonable minimum. A rule of thumb for serial data transmission is: 50% of the bit rate (which gives you the fundamental frequency) multiplied by 5 is the maximum relevant frequency content of your digital signal. Apply at minimum factor 2 (Nyquist theorem) – or better 3 – to this frequency. The result will be your minimum sample rate!

Example:
CAN with 125 kbit/s
Fundamental frequency:
125 kbit/s / 2 = app. 60 kHz
Relevant frequency content:
60 kHz * 5 = 300 kHz
Minimum Sample Rate:
300 kHz * 3 = 900 kSa/s

This calculation is of course only valid, as long as no other signal contents (like glitches) with higher frequency content overlay your CAN signal.

Summary

Serial buses are pervasive in today's digital designs, and you will find serial buses in general and CAN in particular in all industries. Automatically triggering on and decoding of serial buses will save you debug time and money. No more bit counting!
State-of-the-art architectures should offer features including:

- Hardware-based decoding (finds communication errors quickly)
- Segmented memory acquisition with automatic decoding
- Easy Search & Navigation functions on captured serial data

Andreas Siegert
Sales Specialist
Agilent Technologies Sales and Services GmbH & Co KG
Herrenberger Straße 130
71034 Böblingen
Phone 05066 914134
Fax 05066 914135
andreas_siegert@agilent.com
http://www.agilent.com

Author 2
Johnnie Hancock
Agilent Technologies
Colorado Springs, CO
USA