Capture, Decode and Debug of Low Speed Serial Buses

TECHNICAL BRIEF

Tony Minchell
February 27, 2012

Summary

Low speed serial bus validation requires more than simple voltage verses time measurements. LeCroy oscilloscopes have the ability to use a wide range of triggering, analysis and decode capabilities to bring increased confidence to the validation process. This article will cover methods to capture, view, decode and debug a variety of low speed serial data protocols including RS232, generic UARTs, I^2C, CAN, Flexray, LIN, ARINC 429, MIL-STD-1553, MIPI D-PHY, DigRF 3G, DigRF v4 and Audio applications (I2S, LJ, RJ and TDM).

Low Speed Serial Buses

In recent years there has been a proliferation of low speed serial buses entering R&D and coming to market in end user products. Some of these are application specific such as FlexRay in the automotive industry, whereas others span many markets and industries and have been around for a long time, such as RS232 and UARTs. New protocols are continually entering the market providing product designers and end users with more features, performance and robustness, but also with new debug challenges to validate the performance and interoperability. Figure 1 is a menu from a LeCroy oscilloscope showing some of the types of lower speed serial data signals which can be captured, decoded and debugged using modern digital oscilloscopes.

Figure 1: *LeCroy oscilloscopes can capture and decode a wide variety of low speed serial data protocols*

Some low speed serial buses use a differential architecture to increase immunity to common mode noise and interference. This is typical practice with high speed serial buses such as USB or PCI Express which have lower voltage swings and therefore can be more easily affected by noise. CAN is an example of a low speed differential bus. It uses differential data lines with the transmitter and receiver operating at the same bit rate - see Figure 2.

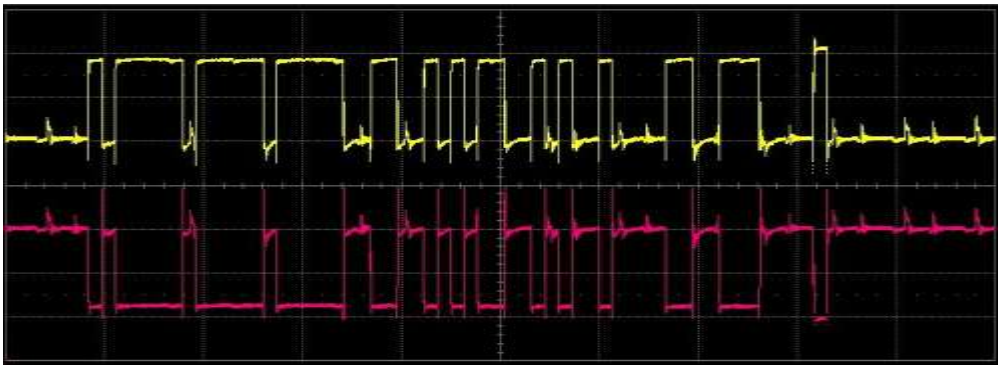

Figure 2: *An example of serial data signal which is sent differentially*

A differentially encoded signal can be extracted by using either the oscilloscope's internal math capability (capturing the two sides of the differential signal into two channels of the scope – then subtracting them), or by using a differential probe, which is the preferable method.

Differential probes are optimized for acquiring differential signals by providing two signal paths that are as identical as possible, matched in overall attenuation, frequency response, and time delay. The two paths are fed to a differential amplifier within the probe thereby maximizing the common mode rejection ratio (CMRR), and extracting the resultant single ended signal for analysis by the oscilloscope.

As a contrast, an example of a low speed serial bus that does not use differential signaling would be I^2C which uses a two wire topology consisting of a data line, SDA, and clock line, SCL.

Initially, capture and debug of a serial bus would start with validating the signal integrity or quality of the physical waveforms before protocol debugging, since a stable and valid physical layer is paramount to the stability of the overall system.

Useful methods for trigger and capture of faults in the clock or serial data signal are the Glitch, Runt, Dropout and Interval trigger capabilities of the modern oscilloscope. One can also use features such as WaveScan to trigger on (or to locate in a previously captured waveform) non-monotonic edges, edges with rise/fall times that are outside the spec of the protocol and or other signal features which violate the specification of the serial data standard.

Most brands of oscilloscopes offer a fast trigger method and the use of persistence display to try to spot signal irregularities. But Smart triggering, as in Figure 3, is far better than using Persistence or fast update mode. This is because these so called high speed acquisition modes still have significant dead time, which can miss the occurrence of an error state especially if this state only happens rarely. Smart trigger keeps the scope active 100% of the time (no deadtime) until the useful data occurs and the scope captures it.

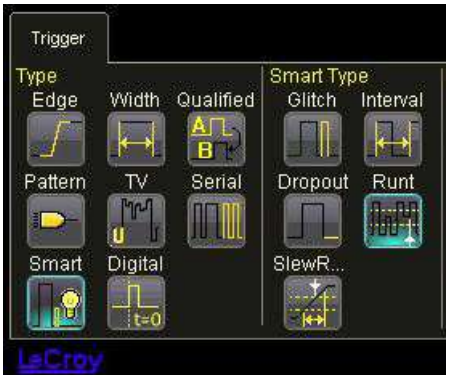

Figure 3: *Some of the choices for Smart Triggering*

Further, many of the fast update modes do not allow combining with a sophisticated triggering mode leaving the user with a simple edge trigger which is virtually useless when debugging burst driven serial buses.

On the other hand a Smart trigger checks every state coming into the scope and has no dead time since the oscilloscope does not trigger until the error condition is met. Often the 'Normal' acquisition mode using a Smart trigger allows greater triggering possibilities, greater acquisition memory depth and better after capture analysis capability. Many modern oscilloscopes will also allow the captured waveform trace to be stored locally in the scope and then quickly re-arm the trigger to carry on monitoring the bus. This waveform trace logging can go on virtually indefinitely if required.

Some oscilloscopes also have Smart acquisition modes which can be used even if the exact trigger condition is unknown. In Figure 4 we can see a Runt condition being searched for on a long CAN bus trace. Any occurrence of a Runt condition will stop the Oscilloscope and allow further analysis. The table on the left in Figure 4 shows 9 Runt occurrences, the 4th occurrence is highlighted in the table and is being shown in detail in the zoom window (lower trace).

Figure 4: *A Runt trigger is used to find a signal integrity problem in a CAN signal. A long set of data is captured in the top trace. Nine faults are found. The fourth one is zoomed for detailed viewing in the lower trace*

The ability to set up a Smart trigger is especially useful when considering that it is not just Runt and Non-Monotonic conditions that can be tested but also duty cycle variations, excessively fast or slow rise and fall times, frequency and period deviation, positive and negative width deviations and time skew variations. The scope can even search for excessive overshoot or undershoot. Any of these trigger types can be coupled with search filtering options, see figure 5, to give extensive debug analysis capabilities.

Figure 5: *Triggering on signal conditions such as runts, pulse widths, frequency, rise/fall time, signal dropout etc. can be numerically filtered by a variety of user choices*

Each serial data protocol has it's own specification giving physical layer and protocol information. The physical layer will typically be specified by the amount of tolerance allowed from an ideal. Such "ideals" and tolerances may be set for bit rate variation, data / clock timing specifics, expected voltage levels and for the shape of the signal.

The oscilloscope mask capability is another tool which can be used for physical layer validation. In Figure 6, a FlexRay example shows an eye diagram with mask violations marked by red circles. The implication is that these signals do not meet the FlexRay specification and are therefore non-compliant. These issues obviously require further investigation.

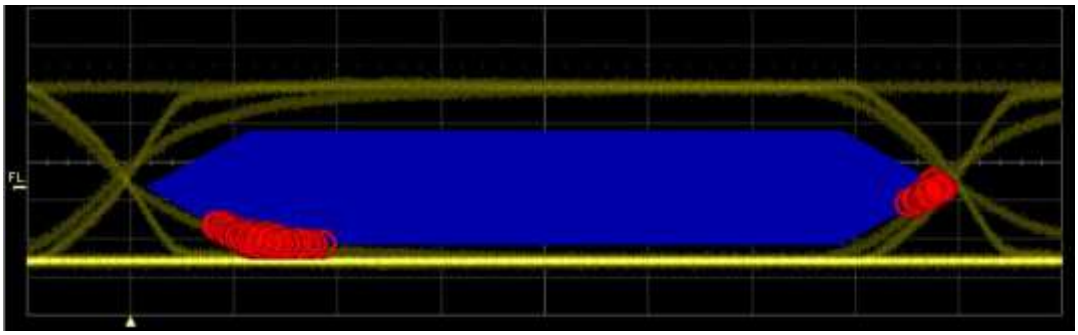

Figure 6: *An eye diagram test of the shape of a serial data signal – in this case Flexray*

Every circuit should be debugged / validated from the hardware stage before moving to protocol debugging / validation stage, i.e., make sure your design is meeting the physical level specifications such as voltage levels, pulse durations / bit rates, clock / data timing, etc, before moving to debug the protocol layer.

Decoding of the voltage vs. time signal to perform protocol debugging is sometimes thought to be unnecessary. An engineer might make the (risky) assumption that the bus and components provided by silicon vendors are all proper since the vendor understands and clearly follows the specification. Unfortunately this is not always the case.

Protocol Trigger and Decode capabilities are usually optional enhancements to an oscilloscope. A single oscilloscope may have several different protocol decode options installed (as shown in Figure 1) allowing it to be used for bus validation in many different parts of the design, or to decode multiple buses simultaneously (for example in traffic latency measurements between buses).

The oscilloscope protocol trigger and decode can take several forms. Some merely use a simple edge trigger and then decode the voltage vs. time signal to show the decoded information to the user. This has distinct disadvantage compared to protocol aware triggering which can let the user trigger on specific address or data information, or on error conditions. The Error condition triggering greatly enhances the whole bus validation process since it can be left monitoring a bus over long periods of time or under differing conditions looking for an error to occur.

Protocol decoding by different oscilloscope manufacturers also varies considerably. In Figure 7 we can see the protocol aware triggering, a decode listing table, and the waveform itself with color coded decode overlay turned on. This simplifies the whole process of understanding the events taking place on the bus.

Figure 7: *An example showing setup of triggering on a certain range of address values. The scope triggers and decoes the signal when an interesting address occurs.*

Figure 8 shows an example of the MIL-STD-1553 Error triggering options and the resultant captured error. This error condition is now ready for further investigation to see what events led up to it.

Figure 8: *Protocol Aware Triggering and Bus Decode. In this case the scope is triggering on Error conditions*

Summary

Serial bus debug and validation requires an understanding of the bus itself and its protocol. There are many tools which can help. The bus signal quality, or signal integrity, should always be the starting point using tools such as an oscilloscope with Smart Trigger and WaveScan, before moving to the protocol validation stage. Other technical notes and LAB briefs are available which give more details on the use of these features.

Many oscilloscopes have bus trigger and decode options available. These debug tools have the advantage of running in real time as part of the oscilloscope, which leads to greater serial bus debug and validation efficiency.