What information can eye diagrams provide for CAN?

Thomas Stüber (Teledyne LeCroy)

Eye diagrams are a popular method for quickly evaluating the signal integrity of serial data systems. In this article, we would like to explain how to generate eye diagrams and how they can be applied to CAN. Additionally, we will discuss how to generate separate eye diagrams for the arbitration and data phase of CAN signals, and what valuable information they provide. Finally, we will discuss how to use the information gained from the eye diagrams to evaluate a CAN network.

What is an eye diagram and how it is generated?

The eye diagram is a very simple method for evaluating the signal quality of serial data. It is a standard analysis tool for high-speed data signals such as PCI Express®, SAS, SATA, etc. It is much less commonly used for serial buses such as CAN bus, but it provides a simple way to characterize the signal quality.

Unlike high-speed bus systems, the CAN bus transmits data in short bursts with blocks of bits. Acquiring multiple waveforms and “stacking” them on a persistence display is how the traditional eye diagram is formed, showing the history of all the acquisitions. Although this method is possible, it is not useful for CAN bus because in this mode there is no way to distinguish whether the data comes from a single packet or even from the same device on the bus.

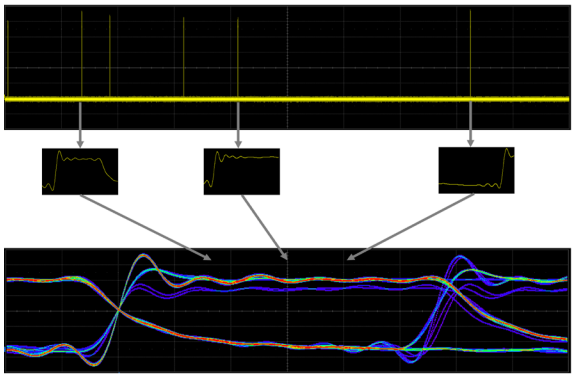


Figure 1: For the classic eye diagram, the oscilloscope is triggered on any edge while running in persistence mode. For CAN bus, this kind of eye diagram will contain bits from different data packages and different nodes.

A better way to form the eye diagram is to acquire a record containing a complete data packet and to overlay all bits from this one data packet to form the eye diagram. This can be done with the use of a special „slicer“ software function in the oscilloscope, which breaks down the waveforms into slices. These “slices” are overlaid on top of each another to generate the eye diagram of the data packet. To cut the slices into the correct length and adjust them on top of each other, a “virtual” clock must be used. In the case of CAN, this clock needs to be extracted from the waveform itself because it is not available as a separate signal.

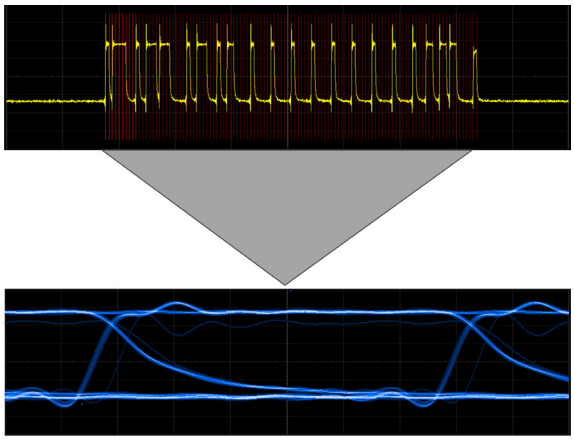


Figure 2: For a “sliced” eye diagram, all bits from a data packet are overlaid using a virtual clock. In the top graph, the red bars show the separation of the bits extracted by the virtual clock.

This method works quite well for classic CAN where we have a constant bit rate in a frame, but what happens if we use this method for CAN-FD or CAN-XL where the bit rate changes?

To get a usable eye diagram for CAN-FD and CAN-XL, where we have different parts in the packet using different clock speeds, we need to refine the generation of the eye diagram by splitting the CAN message into the two parts of the packet, the arbitration phase and the data phase. The easiest way, and this is also the way most oscilloscope vendors have implemented the eye diagram for CAN signals, is to create the eye only from the data part of the CAN message.

A much better way is to generate separate eyes for both the arbitration phase and data phase of each frame. To be able to separate the phases and overlay the correct bits for the different eye diagrams, the oscilloscope must decode and interpret the CAN frame first. For example, in the figure below, only the bits highlighted in blue are relevant to the arbitration phase eye diagram. In the picture it is hard to see, but to get a correct eye diagram containing all parts of the arbitration phase, we have to overlay bits that occur both before and after the data phase. Similarly, the second eye diagram for the data phase uses only the part of the packet that is highlighted in green in Figure 3.

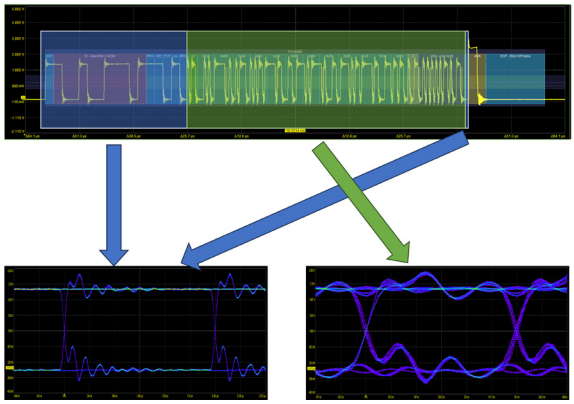


Figure 3: To get a useful eye diagram for a CAN bus network, we have to generate two kinds of eye diagrams, one for the arbitration phase (blue) and a second one for the data phase (green).

Both eye diagrams also need a different virtual clock because we have different data rates in both phases. The eye diagrams do not include the acknowledge bit, since it does not belong to the data packet and other nodes send it. For generating a CAN-XL eye diagram, we have to extend this method further. In CAN-XL there is a third class of bits, the ADS and DAS bit sequences, which are needed by the protocol to give the receiver additional time to synchronize to the higher bit rate before returning to the arbitration phase. Because a real receiver does not care about the signal shape during this “transition” time, these bits have to be ignored when generating the eye diagrams, because including them would most likely give a false Fail indicator when doing mask testing.

How to calculate the virtual clock for the eye diagram?

To get a better understanding of what is needed for the virtual clock, we need to look at how synchronization works in CAN. The bit time is divided into four segments: the synchronization segment, the propagation time segment, the phase buffer Segment 1, and the phase buffer segment 2. Each segment consists of a specific, programmable number of time quanta. The length of the time quantum (tq), which is the basic time unit of the bit time, is defined by the CAN controller’s system clock and the Baud Rate Prescaler. More information than could be discussed in this article can be found in the CAN specification.

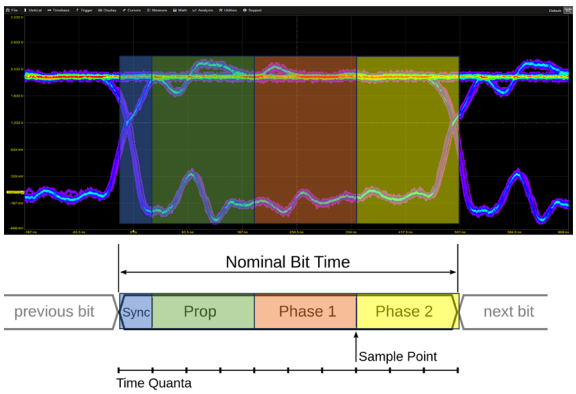


Figure 4: The bit time is divided into four segments: the Synchronization Segment, the Propagation Time Segment, the Phase-Segment 1, and the Phase-Segment 2, and each segment consists of a specific, programmable number of time quanta.

There are two types of synchronization in CAN: the hard synchronization and the resynchronization. Hard synchronization occurs at every Start-Of-Frame when a

recessive to dominant bit edge is detected after the intermission or idle period. Resynchronization occurs throughout the frame whenever a recessive to dominant bit edge is detected outside of the SyncSeg (Synchronization Segment). If an edge is too late, Phase-Segment 1 is lengthened to compensate and to adjust the time between the edge and the sampling point. If it is too early, Phase-Segment 2 is shortened to adjust the time between the edge and the sampling point. To simulate this behavior in the eye diagram on the oscilloscope, the resynchronization of the virtual clock is done also on a recessive-to-dominant edge. In a real receiver, this setting can only be made with the resolution of a time quantum, which differs from how the setting is made with an oscilloscope, which can set the edge with a much higher resolution. We must take this difference in resolution into account if we want to overlay a mask on the oscilloscope eye diagram.

What does this mean for the virtual clock and the generation of the eye diagram in the oscilloscope?

For the data phase, we have to adjust the timing for the eye diagram in a way similar to how it is done in a real receiver during resynchronization, meaning we have to resynchronize the software clock recovery on every recessive-to-dominant edge.

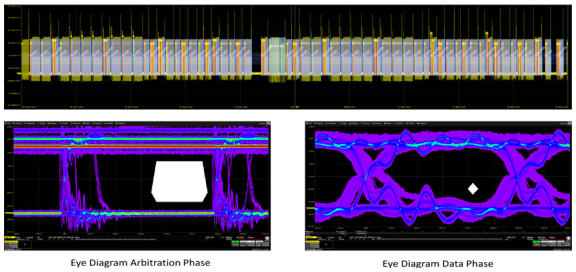


Figure 5: Two eye diagrams should be generated for CAN signals. One for the arbitration phase (lower left) and one for the data phase (lower right).

When looking at the lower right of Figure 5, we can see an eye diagram of the data phase. In this phase, all bits are aligned like a real receiver that uses resynchronization on each recessive-to-dominant data edge. To overlay all data bits, we use the nominal bit rate for all edges between the resynchronizations.

To align the eye diagram of the arbitration phase, which is shown on the lower left of Figure 5, it is necessary to overlay acquisitions in such a way that the transitions do not hit the mask. The solution to the problem of applying a mask is to use the first recessive-to-dominant edge (the leading edge of the start-of-frame) as the reference. According to the ISO definition, all other arbitration transitions must fall (occur) in front of the mask area, so that a violation of this is a real reception problem.

In many oscilloscopes (including the Teledyne LeCroy oscilloscopes), masks for CAN are available where, as is common in other standards, the mask is centered on the middle of the bit. In real CAN systems, this type of mask is not useful, as due to the expected ringing, the sampling point is not set in the middle of the bit length. For testing a real system, the mask needs to reflect this behavior. Figure 5 shows examples of masks that are better suited to the requirements of a real CAN network. The possibility of limiting an eye diagram to a participant or a specific message via a filter function enables a more in-depth analysis and simplifies trouble-shooting.

What are eye diagrams useful for when analyzing the CAN bus?

As we know the sampling point, we can check if there is a stable condition at the time of the sampling point, and we can also quantify the timing of the ringing by simply measuring the ringing in the eye diagram. In Figure 6, we see a very typical reflection, but it is unclear where in the network this refection occurs.

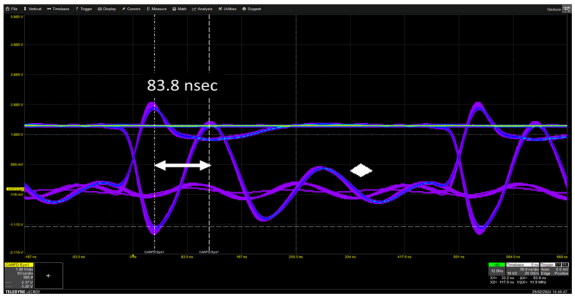
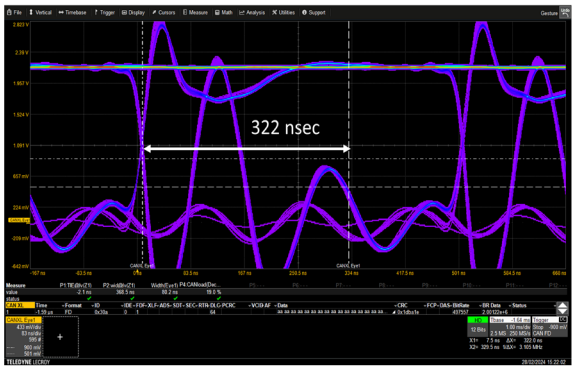


Figure 6: The timing of a reflection can be measured by using oscilloscope cursors to measure the time on the eye diagram between the first maxima and the first reflection maxima.

To get information about the reflection point, we can measure the time between the first maxima and the first reflection maxima on the eye diagram. In Figure 6, the cursor measurement shows a delta of ~84 nsec. We can assume a typical runtime in a CAN network of 5 nsec/m, which corresponds to approximately 16.8 m. This is not the distance, because we have to take into account that the signal travels to the reflection point and that it will also travel back the same distance, so that we have to divide this value by 2. Based on the cursor measurement this reflection happens in a distance of 8.4 m.

However, we can also use this example to estimate up to which data rate we can accept this reflection, or up to which data rate the system will work without errors. Since the time for the reflections is fixed, we can measure on the eye diagram when the earliest sampling point may be in order to reliably detect the signal. The measurement for our example shows that the signal reflection is settled after 320 ns. From this data, we can determine the maximum data rate at which the sampling point occurs after the last reflection crosses the detection level. If we assume a sampling point at 70%, we can see that with 320 ns we are already very close to the maximum data rate, because the sampling point will be at 350 ns. For a system where we use a sampling point at 85%, the 320 ns settling time would allow a theoretical bit rate of 2.66 Mbit/s.



Bit rate	Sample Point (in nsec)					
	60%	66%	70%	75%	80%	85%
1.5 Mbit/s	400.0	440.0	466.7	500.0	533.3	566.7
2 Mbit/s	300.0	330.0	350.0	375.0	400.0	425.0
2.5 Mbits/s	240.0	264.0	280.0	300.0	320.0	340.0

Figure 7: The eye diagram can be used to measure the time until the reflection has leveled off below the CAN signal thresholds.

What other signal integrity parameters can we derive from the eye diagram?

Usually, the eye diagram is also used to evaluate noise and jitter in the system. To see how this can be applied to the CAN bus, first let's have a look at the noise. Due to the reflections occurring in the system, we usually see an overlay of reflection and noise in the eye diagram, but this does not preclude analysis at the sampling point. In the eye diagram of a CAN bus signal, it is very possible to determine whether the levels at the sampling point are below or above the defined thresholds of 0.4 V for recessive and 0.9 V for dominant bit.

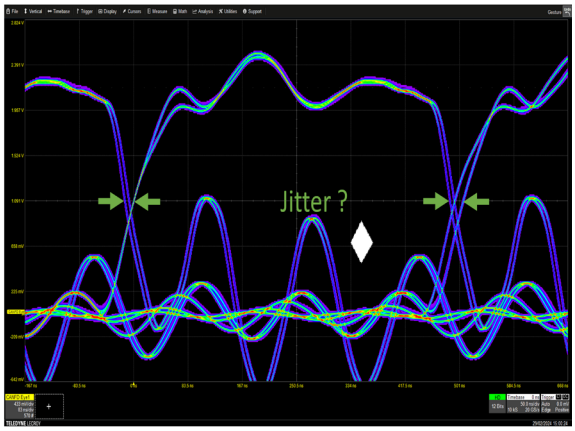


Figure 8: In the eye diagram, it looks as if the signal has different bit lengths, which we could consider as jitter.

But what about analyzing jitter in an eye diagram for a CAN bus system? In Figure 8, we see a CAN signal that has different bit lengths, which we normally would consider as jitter. But is that really the case? Here we need to take a closer look at the process of resynchronization. Whenever a node performs a resynchronization and inserts or removes an additional time tick, the length of the bit is changing. In Figure 8, while this looks like jitter in the eye diagram, it is not. So, we have to be careful about trying to identify jitter using the eye diagram of CAN signals. There is another important point that we need to consider: since we have a resynchronization after 5 bits at the latest due to the insertion of the stuff bits, the analysis of the jitter in the eye diagram is not very meaningful.

Conclusion

Eye diagrams are very useful for analyzing the signal integrity of a CAN network and provide very helpful information about reflections or violations of the levels at the sampling point. However, it is important to create the eye diagram based on the CAN definitions and to carry out a separate analysis for the arbitration and data phase.

Thomas Stüber
Teledyne LeCroy
Im Breitspiel 11c
DE-69126 Heidelberg
www.kvaser.se