CANtegrity background Part 1

How the receiver finds bits in the CAN-frame

This article describes how a CAN-controller synchronizes the bits contained in a CAN-frame. From this explanation, it should be understood that a receiver can only synchronize within one Time Quanta and that a CAN transmission's precision and robustness increases with the number of TQ in the CAN-bit. It should also be understood that even with a perfect oscillator, a receiver can add or remove a TQ at every resynchronization edge. This knowledge will help you to better use the information provided by Kvaser's CANtegrity tools.

Processing the received signal

The CAN-controller is a state machine that processes information at a fixed rate defined by the clock updating the logic state. However, the logic only takes a snapshot of the world at each clock cycle and has no knowledge of what happens between those snapshots.

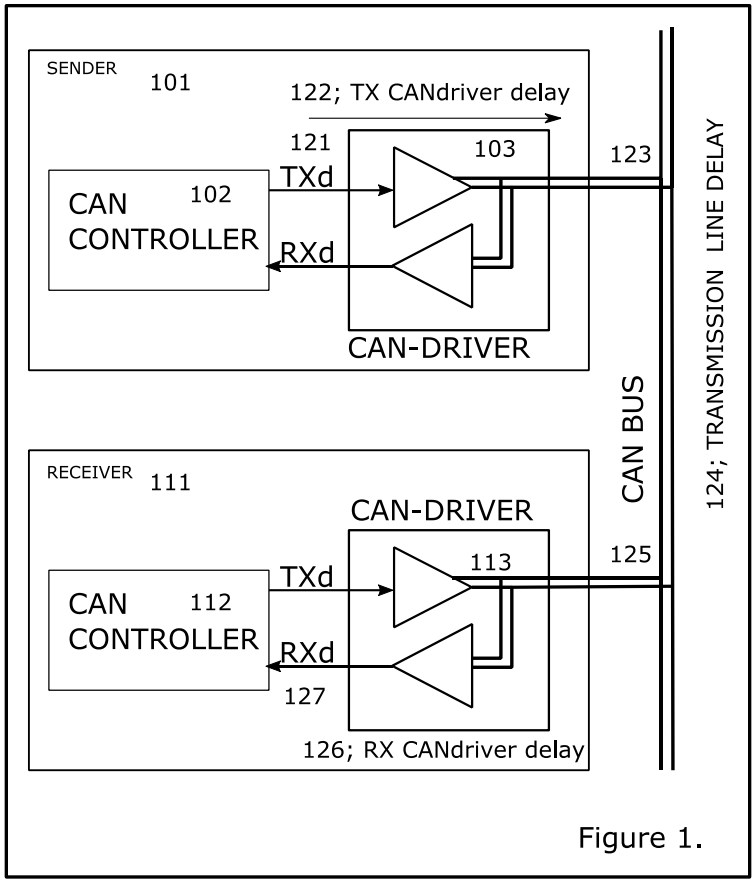

Figure 1.

Figure 1 shows a CAN system in the process of transmitting a signal, with the CAN-frame travelling from the sender to the receiver. The CAN sender logic transmits the bit at the output TXd, indicated by 121. Signal 121 from sending CAN-controller 102 passes through CAN-driver 103 out on the CAN-bus on drop line 123, then along the CANbus to drop line 125 and to receiving CAN-driver 113, over signal line 127 to the receiving CAN-controller. This path results in a signal delay between 121 to 127.

The receiver logic 112 continuously samples the physical layer signal that is converted to a standard digital signal 127 by the physical layer transceiver circuit 113.

The lower signal 2A in Figure 2 shows the communication logic signal 121 at the send point. The upper signal 2K represents the signal, 127, to the receiver communication logic. The signal from the sender must pass through the physical layer driver, the cable to the physical layer receiver and from there to the communication logic at the receiver. This delays the signal, which is indicated by arrow 2B.

The drifting locations of TQ samples

The receiver logic samples the digital signal continuously and an edge is detected by sampling a level different from the previous sample. Sampling by the receiver is indicated in figure 2 with arrows 2L, 2C, 2D, 2E, 2F, 2G and 2H around the first received bit. Sampling is performed continuously after each Time Quanta, TQ. Signal 2K is high at sample 2C, and the previous sample 2L was also high, so no edge is detected.

At sample 2D, the

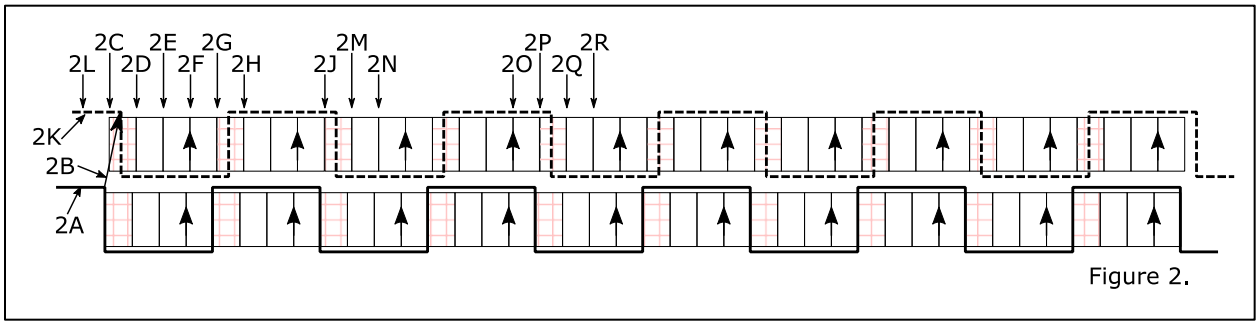

Figure 2.

signal sampled is low, differing from sample 2C. It can therefore be assumed that there is an edge somewhere between sample 2C and 2D. The TQ between sample 2C and 2D where the edge is detected is called the sync segment in the CAN standard. In this case, the bit has 4 TQ and the bit value is that given by sample 2F between TQ3 and TQ4. The TQ that is a sync segment is indicated with a check pattern and the sample point for the bit value is indicated by an up-arrow. The sample at 2G is low and the next sample 2H high, which the logic detects as an edge. This edge is in the expected sync segment in the following bit, so everything is in perfect sync. In CAN, where low to high edges are less precise than high to low edges, those edges are ignored even if they occur outside the sync segment. In other protocols where the low and high bits are symmetric, both edges can be used to adjust the location of the sync segment. The next edge will be detected by the low sample at 2M compared to the high sample at 2J. References 2N to 2R are in figure 2 for comparison with the same references in figure 3 and 4 under different signal conditions.

The transmitter and the receiver samples are not synchronized, so the location of TQ samples at the sender will drift over time relative to the TQ at the receiver. Figure 3 show the case where the sender TQ samples are delayed by almost half a TQ. With the same delay 3B==2B will cause the signal edge to move from the center of the receiver sync segment to a location close to, but before, sample 3D.

In figure 3, it is assumed that the sample at 3D is low, even if the edge is very close to the sampling point. The logic has no means of knowing the exact location of the edge in the sync segment. The only certainty is that the edge must be somewhere between 3C and 3D and the uncertainty is as large as the TQ is long. In figure 3, there is a new edge close to 3M and in this case the sample at 3M is low compared to sample 3J's high. This condition is identical to that at 3D, so the sync segment is placed at the same location between 3J and 3M. The following high to low edge occurs close to 3Q. It is assumed that this edge is delayed and, in that case, that the 3Q sample is high, which has the same value as sample 3P. To the logic, this indicates that there is no edge between 3P and 3Q and in that the TQ between 3P and 3Q is not a sync segment. The next sample, 3R, is sampled low, differing from the

previous sample, 3Q, which is high. This condition indicates that there is an edge between 3Q and 3R and this condition places the sync segment between 3Q and 3R.

All about timing

There are two reasons that the edge is delayed relative to the TQ sample in this example. If the sender has a clock that is somewhat slower than the receiver, the edge will be received later in the following bit, and sooner or later the edge will pass by a receiver TQ sample. The other reason is that the signal is not 100% perfect due to imperfections in the

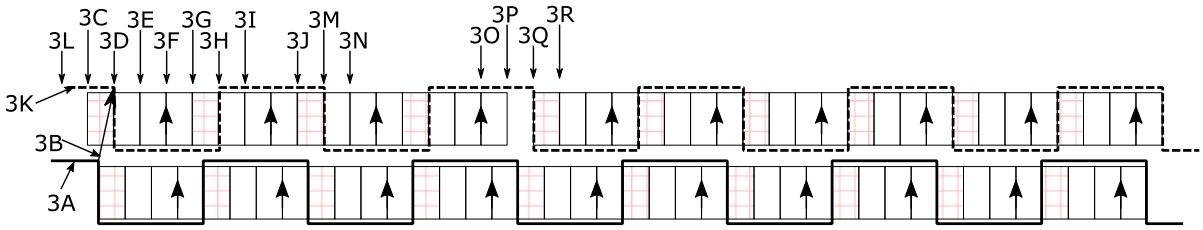

Figure 3.

CAN-drivers and electro-magnetic disturbance from the surrounding environment. This causes phase jitter in the edge location and if the edge is close to the TQ sample, phase jitter will cause the edge to randomly occur before or after the TQ sample. Random phase jitter will cause the logic to move the sync segment back and forth depending on which side of the TQ sample the edge is located in every following bit.

Figure 4 shows the other extreme location of the edge, close to but after the sample at 4C. The sync segment TQ remains between sample 4C and 4D, but the signal edge is now located just after 4C, which compares with figure 3 where the signal edge was located close to 4D. In figure 4, a new edge appears close to 4J and in this case, the sample at 4M is low compared to high at sample 4J. This condition is identical to that at 4D, so the sync segment is placed at the same location between 4J and 4M. The following high to low edge occurs close to 4P. In this case it is assumed that this edge comes early and therefore, sample 4P is low, which is different to the previous sample at 4O. To the logic, this indicates that there is an edge between 4O and 4P and that the TQ between 4O and 4P is a sync segment. It may look odd to have the sync segment located at the bit sample point at 4O but as seen in the following bit, the sample point 4R is now located exactly in the center of the bit. Phase jitter may also cause the sync segment to be placed before or after the received bit edge; less problematic as the bit edge will always be at least one TQ away from the bit sample point.

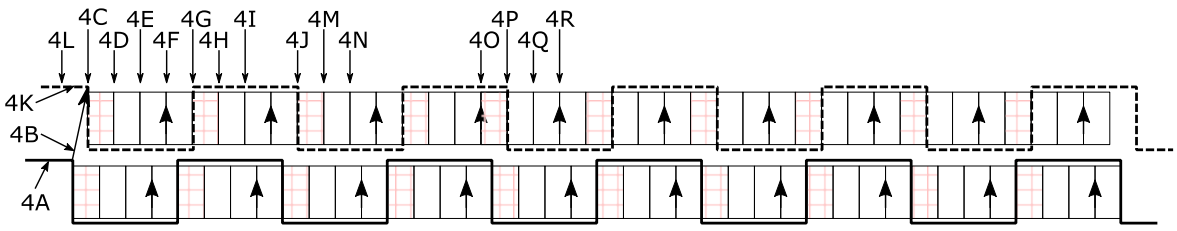

Figure 4.

The receiver has no means of knowing precise edge location, in time, as the TQ sample rate and the edge could be anywhere in the sync segment between 4C and 4D. The obvious solution to increase precision is to increase the sample rate and therefore get more TQ in each bit, thus decreasing the uncertainty of the edge location relative to the length of the bit. In this example with 4 TQ in the bit, the uncertainty is 25% of the bit length. In a UART communication, where 16 TQ are typically used in each bit, the edge is within 6.25% of the bitlength.

BACKGROUND

What is CANtegrity?

CANtegrity is an abbreviation of 'Signal Integrity for CAN'. In this context, signal integrity refers to how to secure signal quality and by secure, we mean 'will the receiver receive the same data that as the sender sent?' CANtegrity is a combination of hardware and software that helps you to understand the weak spots in your CAN communication. It should be noted that CAN is very robust and, in many cases, problems in the CAN communication originate in the software sending or receiving the CAN-frames.

CANtegrity has two goals; to measure the robustness of a CAN physical layer and secondary, to provide clues on how to improve the CAN physical layer and solve problems that become apparent. As it is today, it is more like an advanced oscilloscope that provides detailed information to the user, from which the user can identify the cause of the problem.

Delays in a CAN system

The delay between CAN-controller 102 and CAN-driver 103 is only a few nanoseconds, but the transmitting delay through the CAN-driver is from 30 to 300 nanoseconds and this delay changes over temperature, supply voltage and by ageing. The delay over the dropline 123, CANbus 124 and drop-line 125 is typically 5 nanoseconds per meter. Also, the receiving delay in CAN-driver 113 is in a range from 30 to 300 nanoseconds and this delay also changes due to temperature, supply voltage and ageing. The delay between the CAN-driver 113 and the CAN-controller 112 over 127 is also 5 nanoseconds per meter but this distance is typically only a fraction of a meter. If there is galvanic isolation between CAN-driver 113 and CAN-controller 112, this delay must also be considered. This consideration for 127 is also valid for all TXd and RXd signals between CAN-controller and CAN-driver.

Signal locations in figure 1 displayed in figure 2,3 and 4

To illustrate what's actually going on at the receiving CAN controller, 2K in figure 2 shows the bit representation of receiver 127 in figure 1. In this example, there are four Time Quanta, TQ, in each bit. The lower part, 2A, shows the bit representation in sender 121, which in this case has an identical oscillator to the receiver, identifiable by its identical bit-length to the receiver. The sender sets the bit value at the start of the sync segment, indicated by a checker pattern, and this sync segment is one TQ long. In this example, the signal is represented with a square pattern. The square signal results in a level shift at the start of every sync segment, indicated by signal A. The signal produced by the sender logic is delayed throughout the signal path from the sender to the receiver, and this delay is indicated by arrow

B. This signal delay depends on the signal path and this path differs for all receivers. In a multidrop data communication such as CAN and protocols using RS-485, the signal path indicated by B differs from receiver to receiver. Other physical layers that use multidrop – some physical layers for Ethernet, for example – also experience different signal delays at different receivers.

The sender logic sends a stream of edges at some standard logic level 121. This signal is normally converted into a physical level signal that is better suited to the cable and the electrical environment around the cable. The specification for the physical layer drivers, 103, is described in different standards like ISO 11898-2, RS-485, 802.3da, RS-232 and other physical layers for different types of data communication. A matching driver, 113, is used to convert the physical line signal back to a standard logic level, 127, that can be processed by the receiving communication logic 112. Both the sender and the receiver physical interface are normally described in the same standard and are in most cases combined in one package, a transceiver.