CAN FD filter for Classical CAN controllers

Kent Lennartsson, Kvaser AB

Even if all new CAN-controllers will support both CAN FD and Classical-CAN (referred as C-CAN in the rest of the text) it is impossible to use CAN FD frames on a CAN-bus as long as there are old C-CAN controllers connected to the CAN-bus. This paper will describe how it is possible to add some logic to the legacy C-CAN controllers to make it possible to connect them to a CAN-bus with CAN FD frames. If this logic is combined with a CAN-driver in a standard SO8 package, it is possible to make all old C-CAN units CAN FD tolerant just by replacing the CAN-driver circuit. This will be as simple as swapping two ISO 11898-2 driver circuits and should demand very limited testing and validation. This logic complies with all rules in the CAN-protocol as described in ISO 11898-1.

To protect the legacy C-CAN controller it will be necessary to add filter-logic between the C-CAN controller and the CAN-driver as shown in figure 1. This logic is relative complex, similar to logic for devices with partial networking, and could consist of a FPGA or be integrated in the CAN-driver logic. As seen in figure 1 there are two sides with the TX and RX signals, one between the CAN-controller and the logic and one between the logic and the CAN-driver.

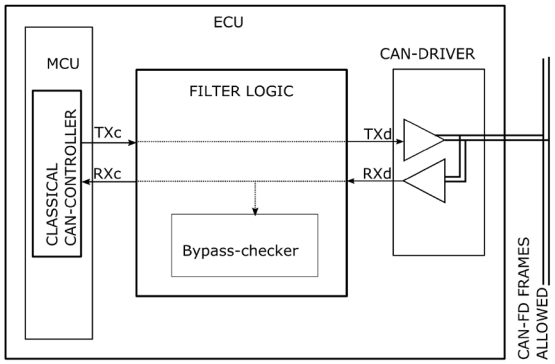

FIGURE 1

To distinguish the two sides the signals are identified as TXc and RXc on the CAN-controller side and TXd and RXd on the CAN-driver side. At start when the CAN-bus is idle the RXd signal is copied unchanged to the RXc signal and the TXc signal is copied unchanged to the TXd. This is visualized in figure 1 by the two dashed lines passing through the logic connecting RXd to RXc and TXc to TXd. In a system where only Classical CAN-frames are used the logic will just add some nanoseconds delay in passing the signals unchanged through the logic. The function of the logic has four different states, as listed in table 1.

Table 1

STATE	Description
Bypass	Logic is transparent
FDF_check	Search for CAN-FD frames
Mimic_frame	Replace FD with legacy frame
Syncing	Synchronize EoF by using Overload-frames

The bypass state

This is the basic state where the logic is transparent and the TXc- and RXd-signals pass through the logic without any modification. Even if the logic does not modify the TXc and RXd signals, the logic will process the RXd-signal with logic very similar to a CAN-FD controller. When the logic is receiving a CAN-frame from the CAN-bus it is processed in the same way as any CAN-FD controller, and just before the FDF-bit is reached the logic will switch to the FDF-check state. This logic, Bypass-checker in figure 1, will also calculate all three CRC-values from the SOF up to FDF-bit. To process the four connected signals the FPGA device will also need: a power supply, an oscillator to clock the logic, the configuration of the CAN/CAN-FD bit-rate and the bit-parameters for the CAN-bits.

The FDF_Check state

In this state the logic will be separated in two parts, as seen in figure 2, one CAN-FD part processing the CAN-FD frame received on the TXd line. The other part is the logic controlling the RXc signal to the Classical CAN-controller during the time when there are CAN-FD frames on the CAN-bus.
The first logic part interfacing TXd and RXd will keep TXd recessive and will receive the CAN-FD frame from the RXd signal. This CAN-FD frame will not be stored in the logic, but the frame will be processed according to all rules for a CAN-FD frame and CRC-17 and CRC-21 used by CAN-FD will be inherited by this logic to complete the error-checking of the CAN-FD frame received from CAN-bus.

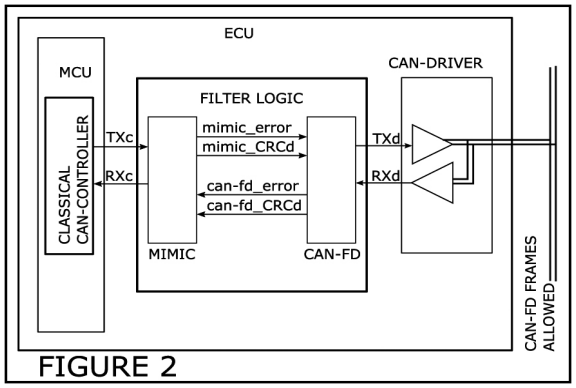

FIGURE 2

The second logic, the Mimic-part, interfacing RXc and TXc will inherit the CRC-15 and take control of the RXc signal.
The FDF-bit will always be dominant for Classical frames and recessive if the CAN-frame is in the FD format. The Classical CAN controller cannot decode a CAN-FD frame where the FDF-bit is recessive. Due to this the logic will force this FDF-bit dominant on the RXc line independent of the RXd signal level. If the FDF-bit is detected dominant on the RXd signal from the bus will it assume that the CAN-frame is a Classical CAN frame and the logic will return to the Bypass State. If the FDF-bit on the RXd line is recessive the logic will switch into the Mimic_frame State.

The Mimic_frame state

The first logic will continue receiving the CAN-FD as described in the FDF_check state.

The task for the logic interfacing the Classical CAN-controller is to make up the end of the CAN-frame that was started on the CAN-bus. To do this the logic will use the CRC-15 updated with dominant FDF-bit and will continue to send the four DLC bits set to zero. After the DLC the logic will send the calculated CRC-15, ACK-bits and the End of Frame.

There is very little the logic can do during the time the CAN-FD frame or the mimic frame are in progress, because they have to follow all CAN-rules for a CAN-frame.

When either CAN-frame reaches the CRC-delimiter it is necessary to take some action. If the dominant edge at the ACK-bit following the CRC-delimiter is not in sync between the CAN-FD frame and the mimic frame there will be a violation of the CAN-rules. As soon as the mimic or the CAN-FD frames reach the CRC-delimiter, the logic will switch to Synch-State.

As seen in figure 2 an Error-condition will change the behavior of the logic. How to handle Errors is described later in the text.

The Syncing state

According to the CAN-protocol any CAN-controller can start sending a new CAN-frame after the three intermissions-bits in the previous CAN-frame. In almost all cases the mimic-frame will have a length different from the CAN-FD frame on the bus. To have a solution that matches the CAN-protocol it is necessary to force the Intermission bits in sync between the CAN-FD frame on the bus and the mimic CAN-frame.

At start this seems impossible but Uwe Kiencke, the father of CAN, included something called Overload frame 1983, which can be used to solve this problem. The overload frame has a format identical to an Error-frame but the Overload frame must only be started in the IM1- or the IM2-bit. The intention with the Overload frame is to make it possible for any receiver on the CAN-communication to delay the start of the next CAN-frame.

This basic Overload function will delay the critical section, but not automatically synchronize the IM1-bit on the CAN-FD frame with the mimic CAN frame. To solve the synchronization problem we have to scrutinize the freedom Uwe Kiencke provides in how we can utilize the Overload frame.
This list below defines the different rules, defined by Uwe Kiencke, that apply to the use of Overload frames in the CAN protocol as included in ISO 11898-1.

- An overload frame is transmitted following end of frame, error delimiter or overload delimiter instead of intermission bit 1 or 2. This gives the filter some room to adjust some phase-error between the two sides by placing the first dominant bit in the Overload-flag little late in the IM1-bit to be better synchronized with the phase in the bits on the other side of the filter.
- An overload frame consists of one overload flag and one overload delimiter.
- The overload flag is transmitted as 6 dominant bits and the overload delimiter is 8 recessive bits.
- The Overload-frame is defined to follow the same rules as the active Error-frame.
- Fault confinement rule 6 in the standard allows 7 consecutive dominant bits after sending an overload flag. In this case the filter will start the Overload-frame and all other CAN-controllers will respond with their own Overload-flag of 6 dominant bits. From this it is obvious that there always will be at least 7 dominant bits as the first part of an Overload-frame, the Overload-flag. After this first part all CAN-controllers will accept 7 additional dominant bits.
- A unit receiving an overload flag in intermission bit 1 will start transmitting its own Overload-flag with 6 bits, then tolerates 7 more dominant bits.
- Error counters are incremented after detecting the 14 consecutive dominant bits.
- The 14 bits do not include the received dominant bit starting the overload frame from the beginning. This is verified by the Bosch VHDL Reference Test.
- The filter will have to use the potential provided by the standard as it will be required to send an Overload flag with 7 to 14 bits followed by the Overload delimiter. The effect of this can be kept hidden inside the filter.
- Every overload frame starts with a falling edge providing synchronization of the sampling point to the following bits in the Overload-frame.
- A maximum of two overload frames is allowed by the CAN specification. There is no error handling action specified when this requirement is not followed. All Classical-CAN controllers tested so far accept hundreds of Overload-frames without causing any change in the Error-counters.

By using the rules above, it is possible to get the CAN-FD frame from the CAN-bus in sync with the internal Classical-CAN controller by using only the two allowed Overload-frames after the CAN-FD frames on the CAN-bus. If the CAN-FD frame is very long in time compared to a C-CAN frame it could be necessary to place several Overload frames on the C-CAN side to delay this part until the CAN-FD frame has ended. The worst case would be a CAN-FD frame without a bit-rate switch holding 64-bytes of data with a pattern that will give a maximum number of stuff-bits. Such frames could be 625-bits long from the FDF-bit up to the IM1-bit, demanding about 28 Overload-frames on the C-CAN side of the filter to get the IM1-bit in sync.

Syncing: detailed description

The basic function of the synchronization is visualized in figure 3. The rough adjustment starts at the edge of the ACK-bit, 827, in the mimic frame and 828, in the CAN-FD frame. By measuring the time between these edges it is possible to get the phase error between the two CAN-frames. In this example the difference is 2.5 bits.

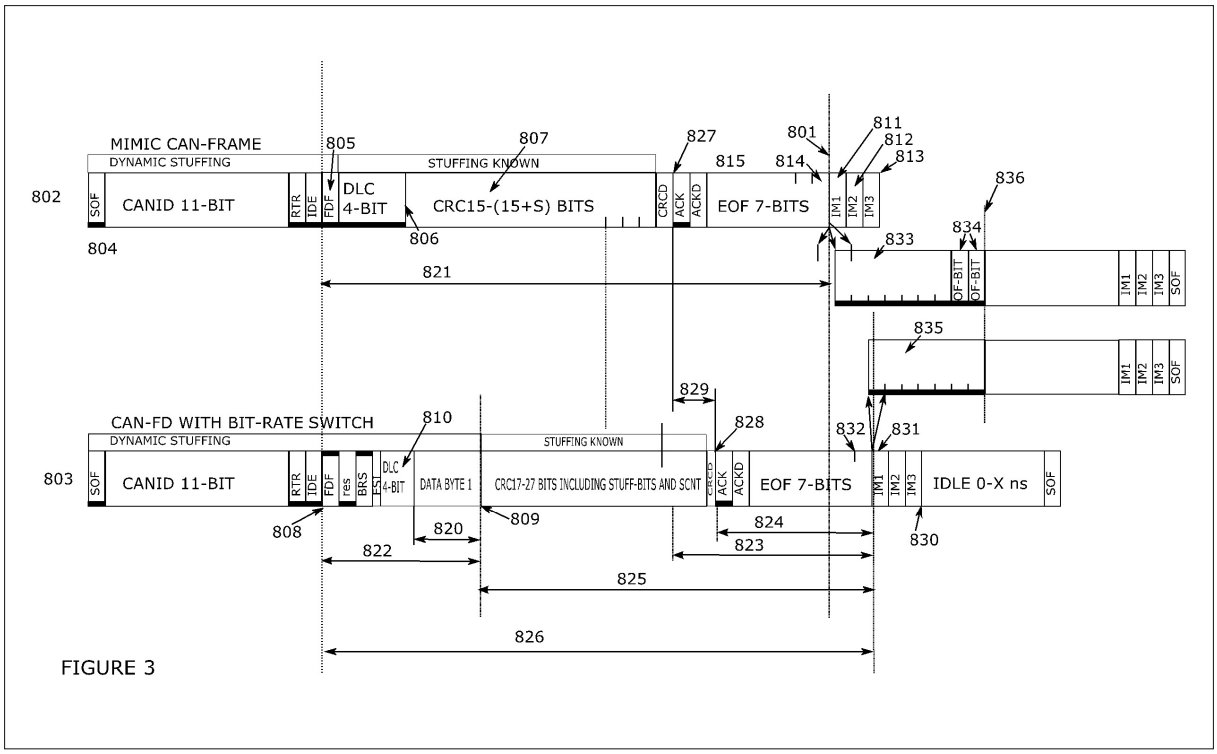

FIGURE 3

There is not a strict correlation between the arbitration and the data-rate so the phase difference can be a fraction of a bit and in this case the data-rate was twice the arbitration rate and the number of data-bits was not even, which resulted in a half bit difference. In this case the mimic frame is 2,5 bits ahead of the CAN-FD frame and it will be necessary to delay the IM-bits at least two bits. There is no available mechanism in CAN to delay the IM-bits because they will always be located after the seven EOF bits. The CAN-protocol does support the possibility to replace the IM-bits with an Overload frame.

When the mimic frame reach the IM1-bit is the CAN-FD frame still in the EoF and it is necessary to delay the IM-bits to get them in sync with the CAN-FD frame. The only possible solution provided by the CAN-rules is to start an Overload-frame indicated by 833.

Normally the Overload flag will end after seven bits but in this case the filter will extend the Overload flag by sending two more dominant bits 834. The result of this is that the expected bits 811-813 will be replaced by the Overload flag 833 plus the two dominant bits 834 making up an Overload-flag with 9 dominant bits. When the CAN-FD frame has reached the first IM1-bit it is necessary to take some action. It is not possible to allow the three Intermission IM1 to IM3 yet, because after the IM3-bit it is possible for any unit on the CAN-bus to start sending a CAN-frame. This is not acceptable because that will collide with the Overload-flag on the other side of the filter with the mimic frame. The only tool available to prevent this is to start sending an Overload-frame 835. In this case it is not necessary to extend the Overload-flag.

When 836 is reached there is nothing more to do, because all units will start counting recessive bits after the Overload-flag has stopped and when they have counted the 8 bits in the Overload delimiter and the three Inter-Mission bits any module will start sending concurrently in the SOF-bit if they have a CAN-frame to send, and the Arbitration will be done according to the CAN-protocol.

Syncing: Variants of overload frames

Figure 3 showed the solution for a particular CAN-FD frame. Even if CAN-FD has a relative fixed format of bytes there is a great variation in length depending on how many stuff-bits there are and which bit-rate is used in the data-part. CAN-FD can be used with an oscillator with relative low tolerance and

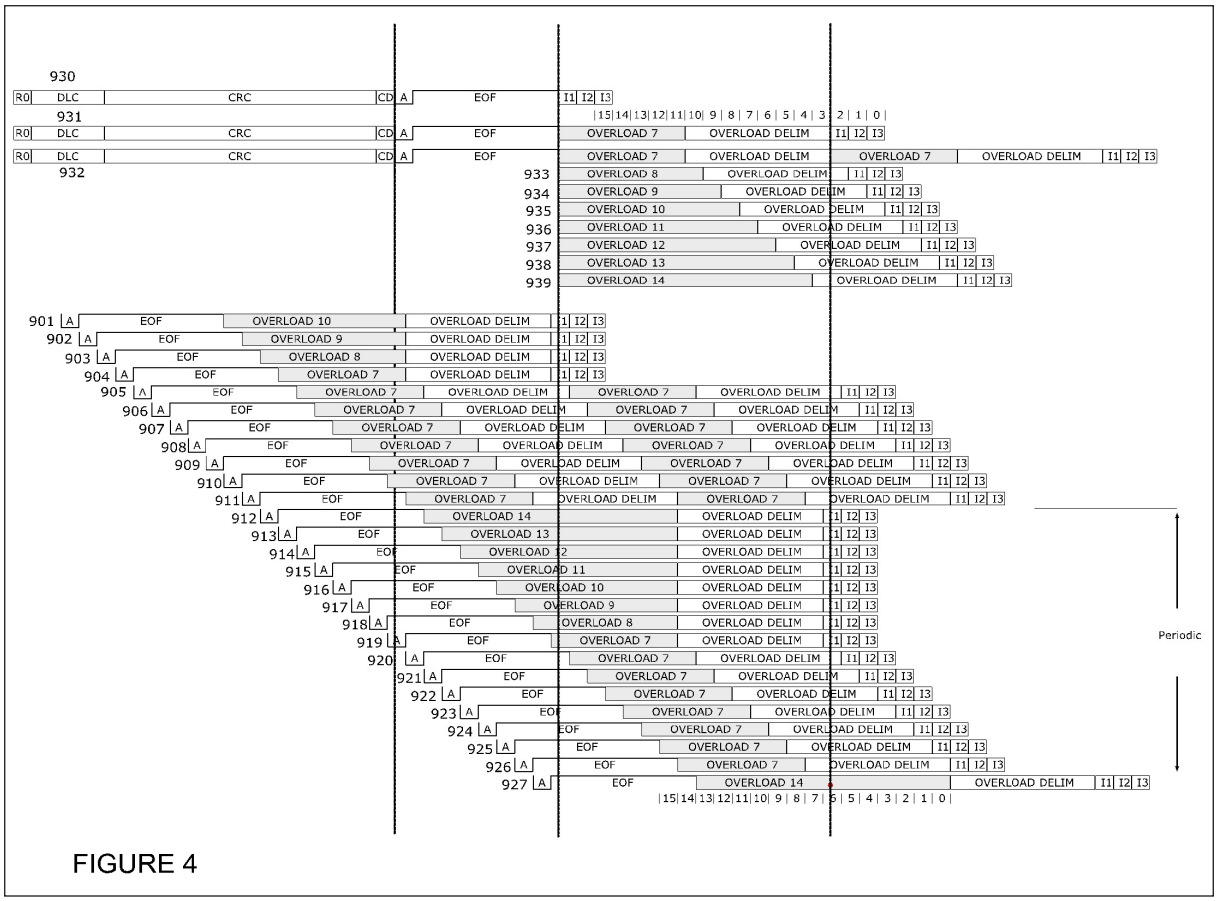

FIGURE 4

as a result of this you will get any number of clock-cycles between the edge in the ACK-bit, 827, in the mimic frame and the ACK-bit, 828 in the CAN-FD frame. For this reason it is necessary to find a generic solution that is not too complex, which can handle any deviation in time between those two edges. The variants shown in figure 4 are all basic variants to be used by the filter.

Frame 930 in figure 4 is the shortest mimic frame with a CRC without stuff-bits. Frame 901 is the shortest CAN-FD possible and to get it requires a very high bit-rate and no data. As can be seen the CAN-FD frame 901 will have the ACK-bit already in the DLC of the mimic frame 930. The IM1-bit in the CAN-FD frame is located half-way into the CRC-part of the mimic CAN-frame. To delay the next message on the CAN-bus, the filter will place an Overload-frame on the CAN-bus after the CAN-FD frame. This will result in an Overload-flag from all units connected to the CAN-bus, resulting in 7 dominant bits. The filter will add three more dominant bits to ensure that the Overload flag ends at the same time as the ACK-bit in the mimic-frame ends.

The ACK-delimiter together with the seven bits in the EOF will match the length of the Overload-Delimiter and this way, all CAN-controllers are in sync at the IM1-bit and the CAN-bus is prepared to handle the next CAN-frame according to the rules in the CAN-protocol.

The next CAN-FD frame, 902 is a little longer and the ACK-bit starts one bit later. This case is almost identical to the process for handling 901. The only difference is that the Overload-flag will be one dominant bit shorter. The CAN-FD frame 903 and 904 is almost the same as 902 with the only difference being that the Overload-flag is 8 respective 7 dominant bits long.

Frame 905 is a little more complicated. The previous method can´t be used because it is impossible to make the Overload-flag shorter than 7 dominant bits. Still it is necessary to place an Overload-flag on the CAN-bus because the mimic frame has not ended yet. The best we can do is to place an Overload-flag on the CAN-bus that will result in 7 dominant bits followed by the Overload delimiter with 8 bits.

This Overload delimiter will end somewhere in the IM1-bit. At the moment all CAN units are in the receiving state because the pervious CAN-FD has finished and the mimic CAN-frame has been sent from the filter. A receiving CAN-controller will accept the Overload frame to start in either the IM1-bit or in the IM2-bit or even in the last bit in the last EOF bit as well as in the last bit in the Overload delimiter. By starting the Overload flag simultaneously on both sides of the filter, both sides will be in sync and all CAN-controller will run parallel and will start the next CAN-frame correctly according to the CAN-rules. The resulting CAN-frames in this case will be 905 in combination with 931 or possible 933 depending where it is necessary to put the edges to adjust the phase error less than one CAN-bit.

The next CAN-FD frame, 906, is very similar to 905. Again the CAN-FD frame ends too early and it is necessary to place an Overload-flag without any extension similar to 905. Even if this Overload frame is as short as possible will it end too late to start at the same time as the Overload-flag starting after the mimic CAN-frame. Again it is the location of this edge caused by the first dominant bit that will define the location in time for the IM1-bits. The filter has full knowledge of where the Overload-flag, after the mimic frame, does start, so it will be possible to start this Overload-flag synchronized at the bit-edges of the bits in the Overload-flag on the CAN-bus. In this case the Overload flag after the mimic frame, 934, will be extended and will end exactly end where the Seventh Overload-flag frame, after the CAN-FD frame 906, ends. This will put the two Overload delimiters in sync even if the Overload-flags contain different amounts of dominant bits.

The following frames 907 to 911 are processed by the same rules as 906, but because the CAN-FD frame ends even later it is necessary to increase the number of dominant bits in the Overload-flag on the mimic side to make it possible to get the Overload-delimiter in phase. In the figure 907 matches 935, 908 matches 936, 909 matches 937, 910 matches 938 and 911 matches 939.

When the CAN-FD frame ends later than the ending in the frame 911 it is not possible to extend the Overload-flag at the C-CAN side because it has already reached 14 dominant bits and that is the maximum number of dominant bits allowed in a CAN-communication. As shown in figure 4 it is now possible to extend the Overload-flag after the CAN-FD frame with seven extra dominant bits, in total 14 bits, to push the Overload-delimiter to be located in the same instance in time as you will get if you have an ordinary Overload-flag with 7 dominant bits, as in the mimic frame, 931. In figure 4 you will see the process of CAN-FD frame 912 to 919 is very similar to the process of the CAN-FD frames 901 to 904. In this case the task is to align the Overload-delimiter after the CAN-FD frame with an Overload-delimiter after the mimic-frame where in the CAN-FD frames 901 to 904 were an alignment to an End Of Frame EOF.

The only difference in the CAN-FD frames 912 to 919 is that the Overload flag must be made shorter and shorter to ensure that the Overload delimiter will be aligned with the Overload frame after the mimic frame, 931.

In the same way the CAN-FD frames 905 to 911 was aligned with mimic frames as shown by 933 to 939 respectively, CAN-FD frame 920 to 926 will be aligned with 933 to 939. A skilled person will see that all possible solutions are covered by the solution used for any of the CAN-FD frames 912 to 926 and all other solutions are a variation of any of those 15 solutions. This is of course given by the fact that an Overload-frame with 15 bits can never be out of phase more than 15 bits, because it is repeated every 15 bits.

Oscillator tolerance

The filter is just a CAN-controller and the demands on this will be the same as on any other CAN-controller connected to the CAN-bus. This will demand that this filter do have the same bit-time configuration and oscillator tolerance as all other units connected to the CAN-bus. When used as a filter in a unit the best will be to use the same oscillator as the C-CAN controller.

As for any CAN-controller will the behavior improve if the oscillator tolerance is better than most other CAN-controllers connected to the CAN-bus.

There are some more calculation and test to ensure the necessary oscillator tolerance when this filter is used. The only difference with this solution is the longer Overload flag that could demand a little better oscillator tolerance.

The necessary oscillator tolerance is given by some equations:

Rule II:

$$df < \frac{\min(PS1_N, PS2_N)}{2 \cdot (13 \cdot BT_N - PS2_N)}$$

Rule II will change to:

$$df < \frac{\min(PS1_N, PS2_N)}{2 \cdot (15 \cdot BT_N - PS2_N)}$$

Even if there is problem it should be hidden by the ruling for the SOF-bit that can start anywhere after the sample-point in the IM2-bit up to the sample-point in the SOF-bit and still be a valid start of the next CAN-frame.

As described in my paper from ICC 2012 in Paris is it possible to make dominant edges also in the recessive bits. In the EoF or in the Overload Delimiter is it possible to place an dominant edges in the Sync-Segment in every bit as long as this dominant level returns to recessive before the bit is sampled by the receiver. This dominant edge will result in a Soft-Synch in the receiving C-CAN controller and by this rule in the CAN-protocol is it possible to move the sample-point and the bit-edge the length of the Synch-Jump-Width (SJW) in every of the 7 bits in EoF or Overload delimiter.

This function is very secure to implement in a filter because the connection between the filter and the C-CAN controller is a local point to point communication. If the filter is connected to C-CAN bus will it also work as long as there is only one filter-bridge between a CAN-FD bus and a C-CAN bus.

To have several filter-bridges units sending dominant edges in the EoF will result in several edges, one for each filter-bridge unit, that will result in a pattern of edges that will cause different Synchronization in different units depending on the relative distance along the CAN-bus.

Error-Frames

The filter solution itself will not cause any Error-Frames, but as in any CAN-communication there are possibilities for faults that will violate the CAN-rules. The probability for such Error on the CAN-bus will be the same whether a unit has a filter or not, because the filter is just a CAN-controller connected to the CAN-bus. If the filter detects an Error-condition on either side, the filter will place an Error-Frame on both sides. This will ensure that all units' Error-counters will update in the same way as they would do if they were directly connected to the CAN-bus. The only difference will be if the Error-condition is detected during sending of an Overload Flag on the other side. In such a case the filter will extend the Overload flag to more than 14-bits which will increase the Error-counter but will not cause an Error-Frame.

Even if the probability for an Error between the filter and the C-CAN controller is very low, an Error-Frame sent from the C-CAN controller will also be sent on the CAN-bus, to ensure that all units keep the Error-counters as identical as possible to a case where there are no filters used.

Performance

There is a penalty in performance; because as soon as you introduce a CAN-FD frame it will have a worst case length of the CAN-FD frame plus two ordinary Overload-frames with 15-bits. This is roughly the same penalty you will get if you switch from standard to extended CAN-ID. This should be a low cost when you add a few CAN-FD frames or if you use CAN-FD frames under special condition, like programming units.

Kent Lennartsson
Kvaser AB
Aminogatan 25A
SE-43153 Mölndal

Tel. +46 31 886344
Fax +46 31 886343
kent@kvaser.com
www.kvaser.com

References

[1] ISO 11898-1 in all variants
[2] ISO 11898-2 in all variants