Comparing CAN FD with Classical CAN

Kent Lennartsson

Table of Contents

Comparing CAN FD with Classical CAN

If you already use CAN, there are three reasons to switch from Classical CAN to CAN FD:

» CAN FD provides shorter CAN-frames, whilst increasing the bit-rate.
  - Lower latency.
  - Better real-time performance.
  - Higher bandwidth.

» CAN FD can hold more data in the CAN-frame, from 8 up to 64 byte.
  - Less relative overhead = better data throughput.
  - More simple and efficient software when sending larger data objects.

» CAN FD has a higher performance CRC-algorithm.
  - Lowers the risk of undetected errors.

If you can't justify a switch based on the above points, there is no real reason to move from Classical CAN to CAN FD. For an overview of how different stakeholders are impacted by a switch from Classical CAN to CAN FD, a white paper is available here.

If the reasons listed above whet your appetite for more, then please read on. This text is designed as an introduction to the CAN FD protocol though to make you own CAN FD design in VHDL or Verilog, it will be necessary to read and understand both ISO 11898-1 and ISO 16845-1. For readers who plan to make their own CAN FD controller, a good start would be to contact Magnus Persson at Synective Labs, which provides the Kvaser CAN FD IP as a product for your FPGA or ASIC implementations. Alternatively, if you would simply like to test the CAN FD IP from Kvaser, please consider buying our CAN FD interfaces, or our CAN FD IP as implemented in the **MCP2517FD** from MicroChip.

The CAN FD basics

Figure 1 compares a Classical CAN-frame (above) with a CAN FD frame (below). Both frames show a single byte of data and, in this case, the CAN FD frame is without the increased bit-rate. As can be seen, both frames are identical from the SOF throughout the 11 arbitration bits. After arbitration, there is a RTR-bit in Classical CAN (labelled A) and RRS-bit in the CAN FD frame. For data-frames, these bits are always dominant (0) in both frame formats. A dominant bit, normally defined as logic 0 and a signal at 0 Volt, is indicated by a thicker black line at the bottom (labelled B).

The bit after the RTR-bit is the dominant IDE-bit, indicating that this frame is in the base frame format that uses 11-bits for arbitration. Notably, this paper will not cover the EF extended frame format that uses 29-bits for arbitration.

After the IDE-bit comes the r0-bit (the reserved bit), which is always dominant in the Classical CAN frame format. In the CAN FD frame format, this bit is recessive (see C), indicating that this frame is not a Classical CAN frame but a CAN frame in the reserved format now known as CAN FD (CAN Flexible Data-rate). In other words, this bit indicates if the CAN-frame is either a Classical CAN frame or a CAN FD frame. From the release of the ISO 11898-1 standard onwards, this bit is known as FDF-bit (Flexible Data Format) in place of r0-bit, the name given it in previous versions of the ISO 11898-1 standard. If in any old documents or data-sheets you see a reference to the r0-bit, it is the same as the FDF-bit in ISO 11898-1 release 2015.

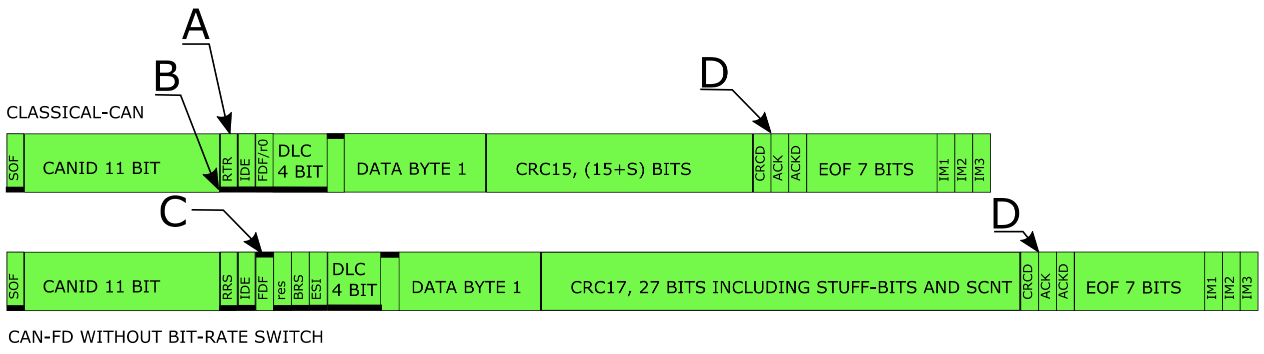

Figure 1 Comparison Classical CAN and CAN FD frame

Additional bits for CAN FD

Following on from the FDF-bit/r0-bit (we'll refer to it as FDF-bit from now on) is the res-bit in the FD-format and the DLC in the Classical CAN-format. In other words, all Classical CAN controllers produced according to the previous ISO 11898-1 standard will misinterpret the CAN FD frame, resulting in an Error-Frame from Classical CAN controllers. After the CRC-delimiter (labelled D in figure 1), Classical CAN and CAN FD are identical in terms of their bit-pattern. In other words, an identical ending pattern is used in both Classical and FD formats, before the next frame starts.

All CAN FD controllers can handle a mix of Classical CAN frames and CAN FD frames. This means that it is feasible to start using CAN FD controllers in your existing system in conjunction with Classical CAN controllers that only use the Classical CAN format. When all old Classical CAN controllers are replaced with CAN FD controllers, it will be possible to mix Classical CAN frames with CAN FD frames or use only one of the two types.

After the FDF-bit in the CAN FD-frame comes the res-bit. Setting this bit to recessive will indicate a future protocol, in the same way as the FDF-bit indicates a switch from Classical CAN to CAN FD format. This future protocol is not yet defined. Notably, it took 25 years before the r0/FDF-bit in the Classical CAN format was used to indicate the CAN FD format.

After the res-bit is the BRS-bit (Bit Rate Switch). This extra bit allows CAN FD frames to be sent in two different formats. If the BRS-bit is sent dominant, all bits will be sent with the same bit-rate as used during the arbitration shown in figure 1. If the BRS-bit is recessive, the frame format will use a higher bit-rate after this bit, up to and including the CRC-delimiter.

the ESI-bit (Error Status Indicator) follows the BRS-bit, which is normally sent dominant. If the CAN FD frame sender become error-passive, this bit will be sent recessive, indicating that the sender has a major communication problem. It is unclear how this bit should be used in wider applications, but it has been included based on requests from automotive manufacturers.

After the three new bits (res, BRS and ESI) follow the four DLC-bits, indicating the number of data bytes in the CAN frame. Table 1 shows how those four bits are used to indicate the number of data-bytes in the CAN frame. Classical CAN frames hold a maximum of 8 byte of data. As you can see from the table, it is possible to send DLC codes above 8, but only 8 byte of data will be placed in the sent CAN-frame. A close look at the table highlights that DLC from 9 to 15 are used differently in the CAN FD format. To have any number of bytes from 9 up to 63 would demand a DLC with 6 bits, and to go to 64 bytes would require a DLC with 7 bits. The compromise was to keep the 4 bit DLC and restrict the number of byte lengths in the CAN FD frames (12, 16, 20, 24, 32, 48 and 64).

Improving data transfer efficiency

Data follows the DLC (Figure 1 shows a CAN-frame with one data byte). The bits before and after the data are a fixed length for any number of data bytes. In this case, to transfer one byte of data requires 55 bits in the Classical format and 70 bits in the CAN FD format. In the worst case, a number of stuff-bits could also be included in the frame. If the frame has more than 5 bits in a row at the same level, the protocol will add an extra bit in the frame with inversed polarity to ensure that level changes can be used to resynchronise the sample-point. This process of adding and removing extra bits for resynchronization is called stuffing and the bits are labelled stuff-bits in the CAN protocol. Data transfer efficiency is improved by packing more data in each CAN frame, as can be seen in the last two columns in table 1. The efficiency equation assumes the worst case number of stuff-bits in the overhead. Classical CAN has slightly better efficiency compared to CAN FD due to its lower overhead. By increasing the number of bytes in the CAN FD frames from 8 byte to 64 byte, it is possible to increase efficiency from 50 to 88%.

DLC code	CRC-code in Classical CAN frame	Bytes in Classical CAN frame	Bytes in CAN FD frame	CRC-code in CAN FD frame	Efficiency % Classical CAN Nx8/(56+Nx8)	Efficiency % CAN FD Nx8/(67+Nx8)
0	CRC-15	0	0	CRC-17	0	0
1	CRC-15	1	1	CRC-17	13	11
2	CRC-15	2	2	CRC-17	22	19
3	CRC-15	3	3	CRC-17	30	26
4	CRC-15	4	4	CRC-17	36	32
5	CRC-15	5	5	CRC-17	42	37
6	CRC-15	6	6	CRC-17	46	42

7	CRC-15	7	7	CRC-17	50	46
8	CRC-15	8	8	CRC-17	53	49
9	CRC-15	8	12	CRC-17		59
10	CRC-15	8	16	CRC-17		66
11	CRC-15	8	20	CRC-21		70
12	CRC-15	8	24	CRC-21		74
13	CRC-15	8	32	CRC-21		79
14	CRC-15	8	48	CRC-21		85
15	CRC-15	8	64	CRC-21		88

In the table, the CRC-coding used in the different frame formats is also included. The Classical CAN format uses 15-bit CRC coding for all frame types, because all frames have a similar length. CAN FD frames are a little more complicated, because a 64 byte frame is almost 8 times longer than an 8 byte frame. To resolve this, two different CRC lengths are used in CAN FD frames: If the frame holds 16 bytes or less, a CRC-17 with 17 bits is used; and if the CAN frame holds 20 bytes or more, a CRC-21 with 21 bits is used.

It is the CRC with 2 extra bits plus the 4 bits in the stuff counter and the fixed stuff-bits that makes the CAN FD frame longer than a Classical CAN frame. Saying this, the comparison is not entirely fair because Classical CAN frames can have up to 3 stuff-bits in the CRC section and three more bits in the control section.

The extra bits in the CRC section of the CAN FD frames provide better protection to the data content. For a safety critical system, this could be adequate justification for switching from Classical CAN to CAN FD alone.

With more than 8 byte of data in the CAN frames, data throughput will increase thanks to improved efficiency. This is the other reason to switch from Classical CAN to CAN FD.

Balancing the requirement for real-time performance

Something that is important to remember is that whilst efficiency does increase from the use of longer CAN-frames, the price is longer CAN-frames and less frames per second, which increases the delay time in the communication and decreases real-time performance. To reduce this problem and to increase the data throughput, it is possible to increase the bit-rate in the CAN FD frames above what is possible in Classical CAN.

The description so far has been for CAN FD with the same bit-rate throughout the complete CAN frame. As described above, a recessive BRS-bit will demand a switch to a higher bit-rate in the data-part of the frame.

In figure 2, a third CAN frame has been added. This third frame is a CAN FD frame with the same content as the middle CAN FD frame, but in this case, the frame is sent at twice the data rate of the CAN FD frame in the middle.

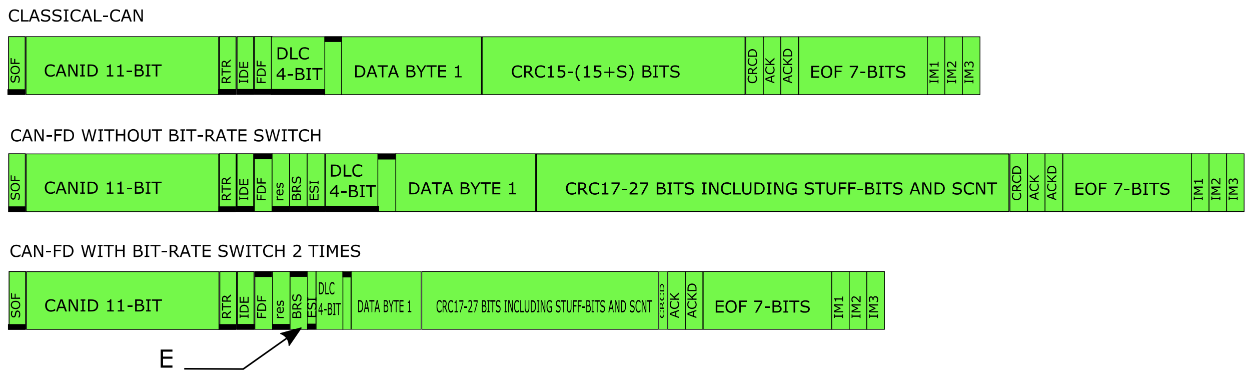

Figure 2 CAN FD frame without and with increased data-rate 2 times.

Because it has the same content, you will get the same DLC, DATA but the BRS-bit will be sent recessive (see E) when the CAN FD is sent with a higher bit-rate. The BRS-bit is included in the CRC-calculation, resulting in two different CRC contents, even if the CAN-ID, DLC and data are identical.

As can be seen in figure 2, the first bit sent at the higher bit-rate is the ESI-bit followed by the DLC, data-bytes and CRC-bits. The last bit sent at the higher bit-rate is the CRC-delimiter. From this it should be understood that the higher bit rate doesn't just apply just to the data section of the CAN FD frame, but to surrounding bits too.

Figure 3 is identical to figure 2, except for one new frame below the previously described frames. This new frame has the same content as all other frames but with a bit-rate of eight times the arbitration bit-rate. The change is relatively large compared with the CAN FD frames with either unchanged bit-rate or with double bit-rate. As can be seen, it is not only the single byte of data that gets a higher bit-rate but also the DLC and the CRC-part of the frame which in total represents about 40 bits.

Figure 4 shows three CAN-frames, with a Classical CAN frame with 8 bytes at the top. In the middle is a CAN FD frame with 64 bytes and the CAN frame at the bottom is the same CAN FD frame content, but with an increased bit-rate (eight times faster).

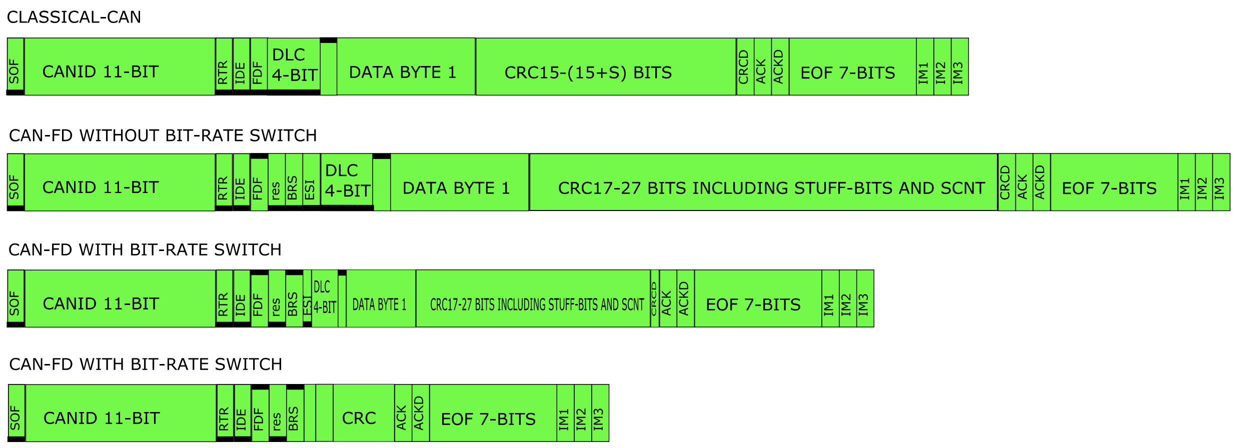

Figure 3, same as figure 2, but also with a CAN FD frame with bit-rate increased 8 times.

From figure 4, it can be see that more data will make CAN-frames longer in time, which will prevent other high priority CAN-frames from starting to send. To keep the real-time performance it is necessary to increase bit-rate in order to reduce the length of the CAN-frame, as well as reduce the time the CAN frame occupies the communication line and prevents other high priority frames from accessing the communication.

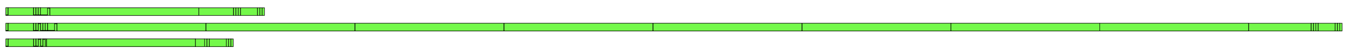

Figure 4, on top a Classical CAN frame with 8 bytes and in the middle a CAN FD frame with 64 byte with the same bit-rate. In bottom a CAN FD frame with 64 byte, but with bit-rate increased 8 times.

In conclusion, CAN FD with high bit-rate will increase real-time performance, because the higher bit-rate makes the CAN-frames shorter in time, reducing latency in the communication. By sending more data in each frame it is possible to increase the data throughput, but this will lower the real-time performance if not combined with the use of higher bit-rate. In many cases, long CAN frames with 64 bytes will be used during programming, which is normally done when a system is on pause and there are no real time controls running. Even without real-time demand, it is still of interest to use the higher bit-rate to increase data throughput and by that, reduce the download time.