# CAN XL Error Detection Capabilities

Dr. Arthur Mutter, Robert Bosch

**CAN XL offers data rates and payload sizes that are many times higher than in Classical CAN and CAN FD [1], [2]. Besides this, CAN XL also provides improved error detection capabilities.**
**Error detection is a crucial functionality provided by communication protocols. A receiving node has to be able to judge if a frame was received with or without errors. Autonomous driving and other safety relevant applications require that frame errors are detected with a very high probability. The acceptance of an erroneous frame should be practically impossible.**
**This paper first introduces the three CAN Error Types known in literature that might occur in a frame in harsh environments: (1) bit error, (2) bit drop and bit insertion, (3) burst errors. The two main pillars of the CAN error detection mechanism are: (A) the cyclic redundancy code (CRC) check and (B) the format checks. Both pillars are strengthened during the currently ongoing specification of CAN XL, to fit to tomorrow's applications.**
**This paper explains how these pillars were improved. Therefor it shows the reasons for the chosen CRC concept of having both a header CRC and a frame CRC in a CAN XL frame. Further, it introduces the available format checks in CAN XL. Finally, the paper shows systematically how the CAN XL error detection mechanisms master to detect the three error types. A deep dive into the properties and strengths of the used CRC polynomials is given in [9].**

## 1. Introduction

CAN XL is the next step in the evolution of CAN. It is currently being specified inside the CiA's CAN XL Special Interest Group. The first specification meeting took place in Nuremberg (Germany) on December 17th 2018. The CiA610 1 specification docu¬ment, which focuses on OSI layer 2 (known as CAN XL protocol), was not yet finished at the time of writing this paper. Con¬se¬quently, the final CiA601 1 specifi-cation may show differen¬ces compared to the content presented in this paper. [3] gives an overview about the current CAN XL status.

Some of the main features of CAN XL are:
- data field size up to 2048 byte
- gross bit rate of 10Mbit/s and more
- strong error detection capabilities

With this set of features CAN enables the usage of higher layer protocols like IP (Internet Protocol). At the same time, it eases the implementation of safety critical applications with its excellent error detec-tion capabilities and its well-known robust-ness.

Two very essential functions in a communi¬cation protocol are the Error Detection and the Error Handling. They have a large impact on the reliability of the communi¬cation system.

The focus of this paper focus are the **Error Detection Mechanisms** in CAN XL.

This paper consist of three parts. Part 1 introduces the CAN XL error detection mechanisms and explains how these were improved compared to CAN FD. In this part, the reasons are given for the chosen CRC concept of having a header CRC and a frame CRC in a CAN XL frame. Part 2 introduces the error types known in literature, along with their properties. Part 3 performs a systematic evaluation to show how the CAN XL error detection mecha¬nisms master to detect all known error types up to a given extent.

## 2. CAN XL Error Detection Mechanisms

In CAN communication, all nodes in a etwork check the validity of each frame, including the transmitter of the current frame.

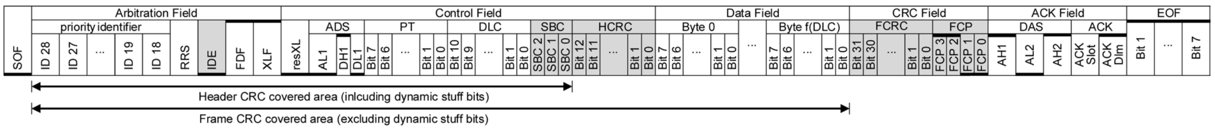

*Figure 1: CAN XL Frame Format*

The checks are based on a combination of several protocol mechanisms for error detection. They are described in the following.

Figure 1 shows the current version of the CAN XL frame format. The bits used to implement additional or updated error detection mechanisms (compared to CAN FD) are shaded.

## *2.1 Bit Monitoring*

Bit monitoring means that a node that transmits a bit also monitors the bit values on the CAN bus. If the transmitted and received (monitored) bit values differ, the reaction of the node depends on the bit position in the frame. As example, if the transmitting node transmitted a 1 and received a 0 in the data field, it regards this as a bit error. However, if the same happens in the arbitration field, it regards this as arbitration lost.

A detailed explanation of the bit monitoring in CAN FD can be found in [10]. If error signaling (via Error Frames) is enabled in CAN XL, bit monitoring is nearly equal to that in CAN FD. For the case that error signaling is disabled, bit monitoring is not yet fully specified in the current CiA610-1 draft.

## *2.2 Frame Format Check*

Most parts of a CAN frame (identifier, control, or data bits) are variable or are calculated from the variable bits (CRC sequence), but some bits (delimiters, end of frame) have a fixed format (see Figure 1). The bit values of these bits are marked in the figure with a bold line. A receiver detects a form error when it samples a fixed format bit with the wrong value.

A special case is the reserved bit following the XLF bit in CAN XL frames. The reserved bit is expected to be dominant. In current applications, a form error is detected when this bit is sampled as recessive. For future applications, this bit may be used to distinguish between the CAN XL frame format and another – not yet defined – new frame format. When this alternative is selected (by software configu-ration) and if then this bit is sampled as recessive, the receiver enters a protocol exception state until the bus is idle again. This allows the introduction of future new frame formats that are tolerated by existing CAN XL implementations.

A node transmitting a CAN XL frame sends the FDF and XLF bits as recessive (logical ‘1’). These bits are part of the Arbitration field, which is different compared to CAN FD. This means, if the transmitting node samples one of these bits as dominant, it loses arbitration and becomes a receiver.

In CAN XL, beside the bit rate, also the mode of the transceiver can be switched. In the error free case, the CAN XL protocol controller signals the mode switch to the transceiver during the bits AL1 and AH1. The signaling of the mode switch to the transceiver, as well as the mode switch of the transceiver may have side effects on the RXD input signal of the protocol controller. Due to this, a CAN XL node does not perform a format check at the fixed format bits (bold lines mark bit value) AL1 and AH1.

## *2.3 Format Check Pattern (FCP)*

The FCP field contains only fixed format bits and is used by a receiver for two purposes. The first purpose is that it provides a synchronization edge before the receiver switches from the data phase to the arbitration phase.

The second purpose is that a receiver can check with help of the FPC field if its frame decoding is aligned with the actual transmitted bit position. Disturbed synchronization edges may lead to so called bit insertions and bit drops in the receiver (see chapter 4). A receiver can detect, with help of the FPC field, a misalignment of 3 bit in both directions.

## *2.4 Cyclic Redundancy Check*

In general, the transmitter and the receivers of a frame calculate the CRC sequence. After reception of the CRC sequence, each receiver performs a CRC check, to judge if it received the frame correctly or not.

**Requirements & CRC Concept**

For the CRC's error detection capability to succeed with a very high probability the following two requirements have to be fulfilled:

- RQ1: Transmitter and receiver of the frame calculate the CRC sequence based on the equal number bits.
- RQ2: The receiver checks the CRC sequence at the right position inside the transmitted frame.

To fulfill RQ1, the CAN XL frame format uses fixed stuff bits (see chapter 1.6) in nearly the whole frame. Dynamic stuff bits are only used in the first bits of the header, to be compatible to CAN FD. A bit insertion or drop error at a dynamic stuff condition changes the number of bits fed into the CRC. As the error just adds or removes a dynamic stuff bit, the format checks described up to now cannot detect that error. With fixed stuff bits, the frame has a defined length in bits and the receiver can feed the exact number of bits into the CRC calculation.

To fulfill RQ2, we need to make sure that a transmission error cannot change easily the position, where the receiver expects the CRC. For example, if the DLC (Data Length Code) is falsified, the receiver checks the CRC at a wrong position. To solve this, CAN XL uses, like Flexray, a header CRC and a frame CRC. The header CRC safeguards a header of well-known length. If a receiver saw a valid header CRC, it is very likely that the DLC is correct. With the correct DLC, the data field length is also well known.

**Scope of the Frame CRC**

The frame CRC is calculated over the header and the data field (see Figure 1), which is similarly done in Flexray. The author in [9] describes in detail, which bits are included and which are excluded from CRC calculation. This "double checking" of the header is done, because on the one side the frame CRC performance is practically not weakened by safeguarding these few additional header bits. On the other side, "double checking" increases the probability to detect transmission errors in the header, which were not detected by the header CRC.

**Dynamic Stuff Bits**

If the dynamic stuff bits are not included into a CRC calculation (like in Classical CAN), an undetectable error can be caused by two bit flips, if one bit flip adds and the other removes a dynamic stuff condition. This case is described in [4]. If the dynamic stuff bits are included into the CRC calculation (like in CAN FD), the CRC calculation may be vulnerable to bit insertions and bit drops at dynamic stuff conditions [10]. CAN XL includes the dynamic stuff bits into the header CRC calculation, but excludes them from the frame CRC calculation. This enables detection of both aforementioned error cases.

In [9] the author assesses the performance of the CAN XL CRC polynomials and compares the results with the CRC polynomials used in Flexray and Ethernet. Both CAN XL CRC polynomials guarantee at least a Hamming distance (HD) of 6, up to the largest CAN XL frame length. This means that at least 5 bit errors can be detected. Beside this, both CRCs are able to detect any odd number of bit errors. Regarding burst errors, the header CRC can detect one burst error of up to 13 bit length, and the frame CRC of up to 32 bit length.

## *2.5 Acknowledgement*

Transmitters expect to get an active acknowledgement for their frames, which is a dominant bit in the ACK slot. When a transmitter does not sample a dominant bit during ACK slot, it regards this as an ACK error. The transmitter considers a frame that does not get an acknowledgement as invalid and retransmits it (if retransmission is not intentionally disabled).

*2.6 Stuff Rule Check*

The bits of a CAN frame are coded by the method of bit stuffing. CAN uses as line coding Non-Return-to-Zero (NZR) which has no guaranteed edges. The purpose of stuff bits is to ensure that there are enough edges in the bit stream for resynchro-nization of the receivers.

Receivers check the stuff rule and detect a stuff error if the stuff bit has not the expected value.

**Dynamic Bit Stuffing**
Before the FDF bit, a dynamic stuffing rule is applied. That means, the transmitter inserts, after each sequence of five consecutive equal bits, one bit of inverse value, called a dynamic stuff bit.

**Fixed Bit Stuffing**
In the data phase, starting at DL1 bit up to the last bit of FCRC, a fixed stuffing rule is applied. That means, the transmitter inserts, after S-1 consecutive bits a fixed stuff bit. The fixed stuff bit has the inverse value of its preceding bit. This means every $S^{th}$ bit is a fixed stuff bit. Currently S=15, but this value may be decreased in the final specification, depending on the results of the phase margin calculations.

*2.7 Dynamic Stuff Count Check*

For compatibility reasons with CAN FD, the CAN XL frame header uses dynamic bit stuffing in the header before the FDF bit. To satisfy requirement RQ1 from chapter 2.4, we need to make sure that transmitter and receiver of a frame see the same amount of dynamic stuff bits.

CAN FD solved this requirement by adding the field SBC (Stuff Bit Count) which contains the number of dynamic stuff bits in the frame modulo 8.

CAN XL also uses this this solution and has therefore an SBC field in the header of the frame. It is located before the header CRC, because it is used to check the validity of the header. The number of dynamic stuff bits in a CAN XL frame is in the range 0 to 3. Therefore, the SBC field in the CAN XL frame has 3 bits, the first 2 bits contain information on the number of dynamic stuff bits in the Arbitration Field and the 3rd bit is a parity bit.

The receiver detects a header CRC error if the SBC does not match to the number of received dynamic stuff bits, or if the SBC parity does not match.

*2.8 Interaction between Error Detection and Error Signaling*

CAN XL allows to enable or to disable error signaling. The software can enable and disable error signaling with a configuration bit in the CAN XL implementation.

In case the user disables error signaling, the respective CAN XL node does not transmit error frames.

In case the user enables error signaling, the error signaling is done with help of error frames, which is identical to the error signaling in CAN FD, which is described in [10]. Error signaling with error frames disturbs the current frame and thereby converts local errors into global errors in order to ensure data consistency in the network.

## 3. Improved Error Detection in CAN XL

This chapter highlights the five improve-ments in the CAN XL error detection compared to CAN FD.

**(1) Header CRC:** The newly introduced header CRC allows checking the validity of the header, which includes the DLC value. This allows fulfilling RQ2 from chapter 2.4 and by this strengthens the CRC check.

**(2) Frame CRC:** CAN XL uses a 32 bit frame CRC with a respective CRC gene-rator polynomial to keep the Hamming Distance at 6 (HD6) despite the long data field. The frame CRC polynomial was chosen carefully and it outperforms the polynomials of Ethernet and Flexray according to [9].

**(3) Fixed Stuff Bits:** CAN XL uses fixed stuff bits in the data phase of the frame (short bits). This allows fulfilling RQ1 from chapter 2.4 and by this strengthens the CRC check.

**(4) Frame CRC safeguards the header:** The frame CRC also safeguards the header, which means a "double checking" for the header. To do this effectively, it excludes the dynamic stuff bits. The reason for that is given in chapter 2.4 and can be summarized as follows: If the CRC calculation does not include dynamic stuff bits, it is vulnerable to a special error case known from Classical CAN [4]. If it includes dynamic stuff bits, it is vulnerable to another error case [10]. The header CRC safeguards the header including dynamic stuff bits and the frame CRC safeguards the header excluding dynamic stuff bits. This enables detection of both special error cases.

**(5) FCP (Format Check Pattern):** The format check pattern is a new field (see chapter 2.3). The receiver checks via FCP if it is aligned to the transmitted bit position. A receiver can detect, with help of the FPC, a misalignment of 3 bit in both directions.

## 4. Error Types

This chapter gives an overview of the existing error types. Details to these error types are described in [10].

### *4.1 Bit Error (Bit Flip)*

Bit Error or Bit Flip means that a CAN node samples a bit with the inverse (flipped) value compared to the transmitted bit value. Figure 2 shows an example for such a bit error at bit 3.

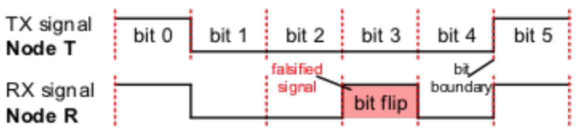

*Figure 2: Bit Error example*

### *4.2 Bit Drop or Bit Insertion*

Bit drop or bit insertion means that a receiving node drops a bit from or inserts a bit into the bit sequence. This is caused by a disturbed RXD signal and can occur only in receiving nodes.

In order to cause a bit drop or insertion, the following needs to happen: A disturbance (e.g. EM radiation) influences the CAN physical layer. As consequence, additional or shifted falling edges appear in the RXD signal. The receiving node resynchronizes, based on these faulty edges. This resynchronization may increase the phase error ([6], [2]) between transmitting and receiving node. When the absolute value of the phase error is above a critical level, the receiving node drops a bit from or inserts a bit into the bit sequence.

Figure 3 shows an example for a bit drop. Here a resynchronization on a falsified edge causes the receiver to drop one bit. The receiver samples the transmitted bit sequence "100000i" as "100001" ('i' stands for a dynamic stuff bit).

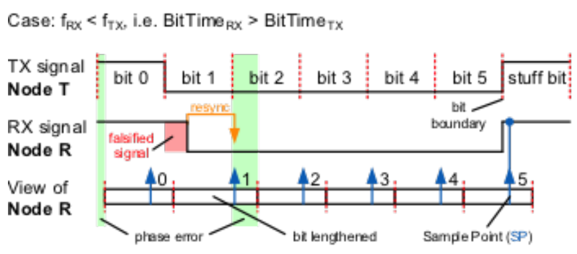

*Figure 3: Bit drop example*

Important properties of bit drops and bit insertions are [10]:

- They can theoretically happen at any position in the frame. It is not limited to dynamic stuff conditions.
- This error type requires many pre-requirements: e.g. large clock tolerance between sender and receiver, disturbance needs to hit one or more dedicated edges, etc.
- Drop and Insertion can practically not happen in the same frame. However, several bit drops or several bit insertions may happen in the same frame.
- Since many factors have to come together, a bit drop or insertion is much more difficult to cause, compared to a bit error. Therefore, one bit drop or insertion should be considered from the likelihood point of view as a "multi bit error".

### *4.3 Burst Error*

Several bit errors that are locally close to each other are called a burst error. The burst length (in bit) is the distance from the first to the last bit error.

We distinguish here two types of burst errors. Type 1 is where all bits in the burst are forced to the same value, e.g. by a glitch. Figure 4 shows an example. We consider this a realistic type of burst error on the CAN physical layer.

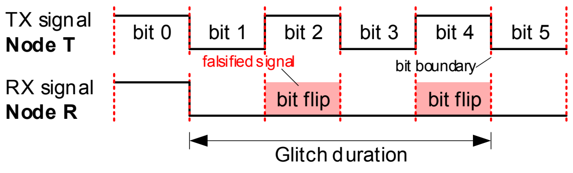

*Figure 4: Burst Error – all bits forced to one value*

The second type of burst error is type 2, where several bits are flipped, but not necessarily all. Figure 5 shows an example.
We assume this type of burst error is very unlikely to be caused by glitches.

However, this type of burst error can be caused by two errors, where the first error leads to a misalignment of the receiver and the second error reverts the misalignment. As long as the receiver is misaligned, it sees all transmitted bits shifted by e.g. 1 bit. This can be achieved by two bit errors, where the one adds a dynamic stuff condition and the other bit error removes a dynamic stuff condition [4]. Consider that the CAN XL frame uses dynamic bit stuffing only at the beginning of the frame. The header CRC can detect this error easily, as it does include dynamic stuff into the CRC calculation – this means from header CRC point of view, there is no misalignment and consequently the two bit errors cause no burst error.

Another way to cause such a temporal misalignment of the receiver is a bit drop and a bit insertion in the same frame [10], which could theoretically occur also in the CAN XL data field [10]. However, one bit insertion and one bit drop, both in the same frame, are assumed practically impossible to occur [10].

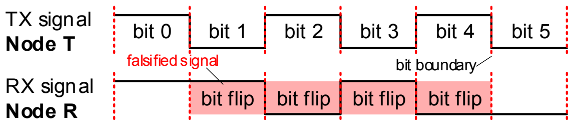

*Figure 5: Burst Error – due to several bit errors*

### 4.4 *Overview of Error Types*

Table 1 gives an overview to the error types known in CAN. The table also shows how an external cause (like a glitch on the bus lines) or an internal cause (like wrong system design) can create these errors. Further, it shows which error detection mechanism from chapter 2 can detect the error.

## 5. Evaluation: Burst Error Detection

Chapter 4.3 introduced two burst error types. As described, burst errors of type 2 (several bits are flipped, but no necessarily all) can be caused by several circum-stances. Based on the arguments from chapter 4.3 we conclude that this type 2 burst error is practically extremely unlikely to occur and therefore can be neglected.

*Table 1: Overview of Error Types in CAN*

| Error Type | External Cause: EMI | Internal Cause | How to detect the Error? | Literature |
|---|---|---|---|---|
| **Bit Error (Bit Flip)** | Glitch length: ≈ one bit length | Bit asymmetry is too large | ○ CRC Check<br>○ Format Checks (limited) | Different bit error rates (BER) mentioned in: [4], [5], [7], [8] |
| **Bit insertion or Bit drop** | Glitch length: < one bit length | CAN clock tolerance is too large | ○ Format Checks, FCP<br>○ Dynamic Stuff Bit Count (SBC) | First described in [10] |
| **Burst Error** | Glitch length: > one bit length | temporary misalign-ment of receiver to transmitted bit stream | ○ Format Checks<br>○ CRC Check (up to a limited burst length) | [7] mentions an average burst error length of 5 us;<br>[8] is less explicit |

Burst error type 1 (all bits in the burst are forced to the same value, e.g. by a glitch) is considered as very realistic. The remainder of this chapter evaluates if and how the error detection mechanisms can detect such a burst error.

Since CAN XL can be used at different bit rates, the same glitch on the bus lines can cause very different error scenarios for a receiver. Figure 6 visualizes the impact of a 2 us glitch. At 500 kbit/s this leads to one bit error, while at 2 Mbit/s it leads to a burst error of 4 bit and at 8 Mbit/s it leads already to a burst error of 16 bit.

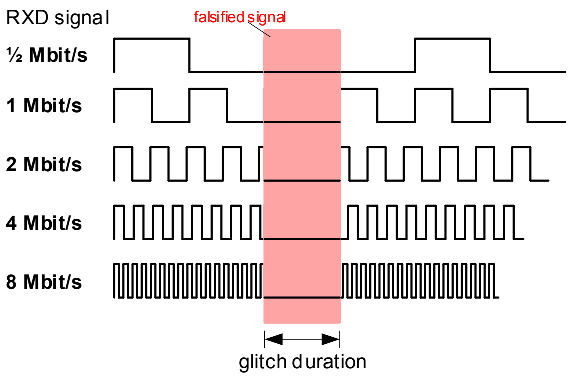

*Figure 6: Errors caused by a 2 us glitch at different bit rates*

The diagram in Figure 7 shows the relation between glitch length and burst length in bits. Two glitch lengths are shown: 2 us and 5 us. These glitches translate to a burst duration of the same value. The actual glitch length that may occur on a specific CAN bus depends on the environment around the CAN bus. The authors in [7] observed in a very aggressive environment an average burst duration of 5 us. For example, at 5 Mbit/s a 5 us glitch causes a burst length of 25 bit.

Figure 7 also shows the main CAN XL error detection mechanisms that are capable detecting burst errors.

- **Stuff Rule Check** (see chapter 2.6): The focus is here on the data phase where each S$^{th}$ bit is a fixed stuff bit. Figure 7 assumes S=10. The arrow in the figure shows, that this check can detect any burst error with a length larger than S bit. Shorter burst errors may also be detected, but only if they hit a stuff bit. With S=15 the effectiveness decreases slightly for short burst lengths.
- **Frame CRC** (FCRC, see chapter 2.4): The frame can detect one burst error with a length of up to 32 bit.
- **Header CRC** (HCRC, see chapter 2.4): The header can detect one burst error with a length of up to 13 bit.

We conclude that both, the header and frame CRC, can detect one short burst error and the stuff rule check can detect long burst errors. In sum, these mechanisms can detect all burst errors.

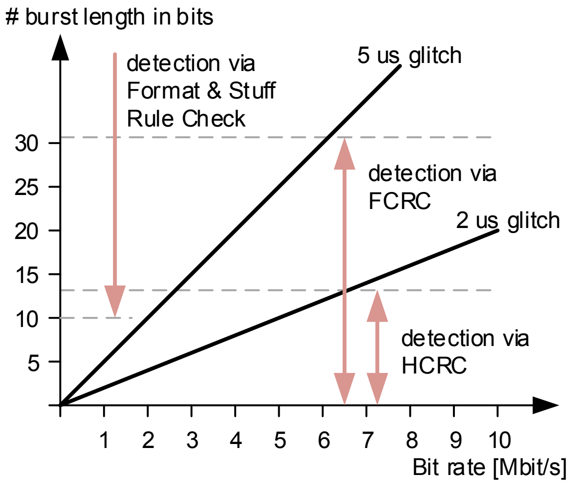

*Figure 7: Error detection mechanisms versus burst errors*

## 6. Evaluation: Detection of Bit Errors and Bit Drops/Insertions

This chapter focuses on the two remaining error types: "bit errors" and "bit drops/ insertions". It evaluates systematically whether they can be detected by CAN XL. The evaluation is limited to 5 bit errors (corresponds HD6) and 2 bit insertions/drops (corresponds to an equivalent of roughly > 4 bit errors).

To simplify the description, the CAN XL frame is virtually partitioned into four parts. The evaluation in Table 2 is partitioned accordingly. In each part, both error types are listed. For each error type, the relevant number of occurrences of this error type are listed. Additionally, special error cases generated by these two error types at dynamic stuff bits are also listed. Consequently, each row of the table corresponds to one error case. For each error case, the table contains information about the misalignment of the receiver and the way in which the receiver detects the error.

*Table 2: Systematic Overview of Error Cases*

| **Case** | **Receiver misalignment** | **Error detected mainly by** |
|---|---|---|
| Frame Part 1 [SOF to IDE] | | |
| **Bit Drop or Insertion** @ dynamic stuff condition | | |
| 1 bit drop or insertion | no | SBC (dynamic stuff bit count changes) |
| 2 bit drops or insertions | no | SBC (dynamic stuff bit count changes) |
| 1 bit drop + 1 bit insertion | 1 bit temporary for CRC | Practically not possible; header CRC |
| | | |
| **Bit Drop or Insertion** @ **no** dynamic stuff condition | | |
| 1 bit drops or insertion | 1 bit | Format Check: IDE = ‘1’ or FDF = ‘0’ |
| 2 bit drops or insertions | 2 bit | Format Check: IDE = ‘1’ or XLF = ‘0’ |
| 1 bit drop + 1 bit insertion | 1 bit temporary | Practically not possible; header CRC |
| | | |
| **Bit Error** @ dynamic stuff condition ➔ adds or removes stuff condition | | |
| 1 bit error (add/remove) | 1 bit | Format Check: FDF = ‘0’ or IDE = ‘1’ |
| 2 bit errors (add/remove) | 2 bit | Format Check: XLF = ‘0’ or IDE = ‘1’ |
| 3 bit errors (add/remove) | 3 bit | AL1 = ‘1’ ➔ Transceiver will be not switched; or Format Check: FDF = ‘0’ |
| 1 bit error (add) + 1 bit error (remove) | not for CRC | Header CRC |
| | | |
| **Bit Error** @ **no** dynamic stuff condition | | |
| 1 to 5 bit errors | no | Header CRC |
| | | |
| Frame Part 2 [FDF to AL1] | | |
| **Bit Drop or Insertion** | yes | Format Check |
| **Bit Error** | no | Format Check |
| | | |
| Frame Part 3 [DH1 to HCRC] | | |
| **Bit Drop or Insertion** | | |
| 1 bit drop or insertion | 1 bit | If DLC wrong ➔ all together, else FCP |
| 2 bit drops or insertions | 2 bit | If DLC wrong ➔ all together, else FCP |
| 1 bit drop + 1 bit insertion | 1 bit temporary | Practically not possible; header CRC |
| | | |
| **Bit Error** | | |
| 1 to 5 bit errors | no | Header CRC |
| | | |
| Frame Part 4 [Data field and CRC field] | | |
| **Bit Drop or Insertion** | | |
| 1 bit drop or insertion | 1 bit | FCP |
| 2 bit drops or insertions | 2 bit | FCP |
| 3 bit drops or insertions | 3 bit | FCP |
| 1 bit drop + 1 bit insertion | 1 bit temporary | Practically not possible |
| | | |
| **Bit Error** | | |
| 1 to 5 bit errors | no | Frame CRC |

## 7. Summary and Conclusion

CAN XL has 5 major improvements regarding Error Detection, compared to CAN FD. These are (1) header CRC, (2) 32 bit frame CRC, (3) fixed stuff bits in the data phase, (4) frame CRC additionally safeguards header, (5) Format Check Pattern.

Three major error types are known in CAN: (1) Bit error, (2) bit drops/insertions, and (3) burst errors. These error types are introduced in detail.

The paper shows how the error detection mechanisms can detect a burst error, where all bits in the burst are forced to the same value, independent of its length. Further, it shows systematically how bit errors and bit drops/insertions can be detected up to a given extent. We conclude that the error detection mechanisms in CAN XL can detect all known error types to a sufficient extent. This work can serve as basis for a review of the CAN XL error detection capabilities, which is planned by the SIG CAN XL.

## 8. Acknowledgment

I want to thank Florian Hartwich and Christian Senger for the valuable discussions.

Author Dr. Arthur Mutter
Robert Bosch GmbH
Postfach 13 42
DE-72703 Reutlingen
www.can.bosch.com

## References

[1] F. Hartwich, „CAN with Flexible Data-Rate,“ in Proceedings of the 13th international CAN Conference, Hambach Castle, Germany, 2012.

[2] ISO 011898-1:2015, Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signaling, 2015.

[3] F. Hartwich, "Introducing CAN XL into CAN Networks" in Proceedings of the 17th international CAN Conference, Baden Baden, Germany, 2020.

[4] J. Unruh, H. J. Mathony und K. H. Kaiser, „Error Detection Analysis of Automotive Communication Protocols,“ in SAE Int. Congress, No. 900699, Detroit, 1990.

[5] J. Charzinski, "Performance of the Error Detection Mechanisms in CAN," in Proceedings of the 1st International CAN Conference, 1994.

[6] A. Mutter, „Robustness of a CAN FD Bus System – About Oscillator Tolerance and Edge Deviations,“ in Proceedings of the 14th international CAN Conference, Paris, France, 2013.

[7] J. Ferreira, et al., "An Experiment to Assess Bit Error Rate in CAN", in Proceedings of 3rd International Workshop of Real-Time Networks, 2004

[8] N. Navet and Y.-Q. Song, "Performance and Fault tolerance of Real-Time Applications Distributed over CAN", in Proceedings of the International CAN Conference, 1997.

[9] C. Senger, "CRC Error Detection for CAN XL" in Proceedings of the 17th international CAN Conference, Baden Baden, Germany, 2020.

[10] A. Mutter and F. Hartwich, "Advantages of CAN FD error detection mechanisms compared to classical CAN," in Proceed-ings of The international CAN Conference, Vienna, Austria, 2015.