June 2020

CAN Newsletter

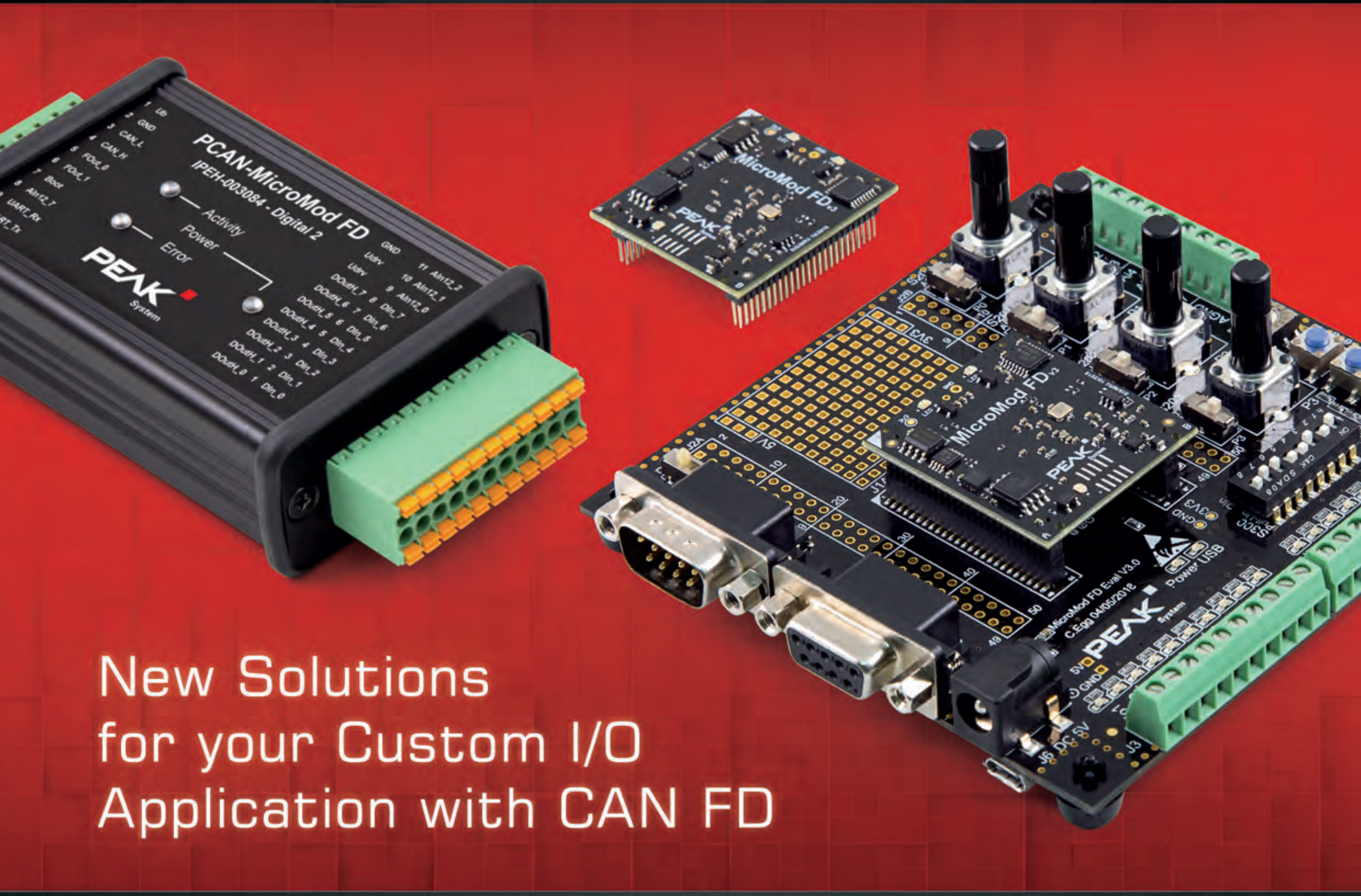
Hardware + Software + Tools + Engineering

Coronavirus

Covid-19 and CAN business

Covid-19: More pallet stackers and forklifts are needed

www.can-newsletter.org



New Solutions for your Custom I/O Application with CAN FD

PCAN-MicroMod FD

Universal I/O module with CAN FD interface

The PCAN-MicroMod FD is a small plug-in board which provides a CAN FD connection and enhanced I/O functionality for the integration into your hardware. An evaluation board facilitates the development of your custom solution. The module is configured with a Windows software via the CAN bus and then operates independently.

Features:

- NXP LPC54618 microcontroller
- 1 High-speed CAN connection
  - Complies with CAN specifications 2.0 A/B and FD
  - CAN bit rates from 20 kbit/s up to 1 Mbit/s
  - CAN FD bit rates from 20 kbit/s up to 10 Mbit/s
  - Microchip MCP2558FD CAN transceiver
- 8 digital inputs and 8 digital outputs
- 2 frequency outputs
- 8 analog inputs
  - Measuring range unipolar 0 to 3 V
  - Resolution 12 bit, sample rate 1 kHz
- Configuration via the CAN bus with a Windows software
- Selective configuration of up to 16 devices in a CAN bus
- Extended operating temperature range from -40 to 85 °C
- Dimensions: 33 x 36 mm
- Voltage supply 3.3 V

Ready-to-use motherboards

The PCAN-MicroMod FD is available with motherboards that provide peripherals for specific applications.

Common Features:

- Board with plugged on PCAN-MicroMod FD
- CAN connection with switchable CAN termination
- 2 frequency outputs (Low-side switches, adjustable range)
- Analog input for voltage monitoring up to 30 V (12 bit)
- Aluminum casing with spring terminal connectors
- Extended operating temperature range from -40 to 85 °C
- Operating voltage 8 to 30 V

PCAN-MicroMod FD Analog 1:

- 8 analog inputs (16 bit, adjustable range)
- 4 analog inputs (12 bit, 0 - 10 V)
- 4 analog outputs (12 bit, adjustable range)
- 4 digital inputs (pull-up or pull-down)

PCAN-MicroMod FD Digital 1 / Digital 2:

- 8 digital inputs (pull-up or pull-down)
- 3 analog inputs (12 bit, 0 - 10 V)
- Digital 1: 8 digital outputs with Low-side switches
- Digital 2: 8 digital outputs with High-side switches

www.peak-system.com

Take a look at our website for the international sales partners. Scan the QR code on the left to open that page.

PEAK-System Technik GmbH

Otto-Roehm-Str. 69, 64293 Darmstadt, Germany
Phone: +49 6151 8173-20 - Fax: +49 6151 8173-29
E-mail: info@peak-system.com

Coronavirus

Protocol

CANopen FD

Imprint

Publisher
CAN in Automation GmbH
Kontumazgarten 3
DE-90429 Nuremberg

publications@can-cia.org
www.can-cia.org

Tel.: +49-911-928819-0
Fax: +49-911-928819-79

CEO
Reiner Zitzmann

AG Nürnberg 24338

Downloads March issue:
(retrieved May 28, 2020)
4198 full magazine

Editors
Olga Fischer (of)
Cindy Weissmueller (cw)
Holger Zeltwanger (hz)
(responsible according
to the press law)
pr@can-cia.org

Layout
Nickel Plankermann

Media consultant
Meng Xie-Buchert
(responsible according
to the press law)

Distribution manager
Rosanna Rybin

CAN XL and CAN FD light

The Covid-19 disease does not stop innovations in CAN technology. CAN in Automation (CiA) members are developing the third generation of the CAN data link layer protocol – also known as CAN XL. It provides more than 2 048 byte of payload including a 1-byte protocol type field indicating the content of the 4-byte address field and the 2 048 byte data field. One of the new features is the separation of frame priority and address information. New is also a cascaded CRC (cyclic redundancy check) featuring a Hamming distance of 6 meaning five randomly distributed bit-errors are detected under all conditions.

Additionally, an embedded data link layer security protocol is under development. It features a node-to-node protection comprising a 4-byte header with cipher control information, the secure channel ID and the freshness value as well as the 128-bit authentication tag.

CAN XL has as CAN FD two bit-rate phases. In the arbitration phase the bit-rate is limited to 1 Mbit/s as in Classical CAN. In the data-phase, the bit-rate can be increased, because just one node is transmitting. CiA members also develop a new CAN physical layer, which supports bit-rates up to 10 Mbit/s and above. This dual-mode approach has two modes: slow and fast. In both modes, bus biasing is active. In the slow mode, there are dominant and recessive bits (as known from Classical CAN). In the fast mode, there are level-1 and level-0 signals. Dual-mode transceivers are connected to the CAN XL protocol controller by means of Mici, the medium-independent CAN interface.

CiA members have also started to develop a master/slave protocol based on CAN FD. It is intended for simple sensor and actuator communication. A typical example is modern LED lamps in passenger cars, which are somehow price sensitive. The CAN FD master node synchronizes the slave nodes with hundreds of LEDs. This simplifies the implementation of the CAN FD slave nodes, because there is no arbitration needed, since they just react on the master requests. It is intended to run such networks with one bit-rate and is therefore limited to 1 Mbit/s, which is fast enough for such applications.

Of course, the novel coronavirus challenges CiA. But CiA made good experiences with online meetings, when discussing CAN XL and CAN FD light.

hz

CAN XL error detection capabilities

With its higher data-rates and payload sizes, CAN XL is the next step in the evolution of CAN. Besides this, CAN XL also provides improved error detection capabilities.

CAN XL offers data-rates and payload sizes that are many times higher than in Classical CAN and CAN FD [1], [2]. 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 article 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.

We explain how these pillars were improved. Therefor we show the reasons for the chosen CRC concept of having both a header CRC and a frame CRC in a CAN XL frame. Further, we introduce the available format checks in CAN XL. Finally, we show 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].

Introduction

CAN XL is currently being specified inside the CiA's (CAN in Automation) CAN XL Special Interest Group. The first specification meeting took place in Nuremberg (Germany) on December 17th 2018. The CiA 610-1 specification document, which focuses on OSI layer 2 (known as CAN XL protocol), was not yet finished at the time of writing this article. Consequently, the final CiA 601-1 specification may show differences compared to the content presented in here. [3] gives an overview about the current CAN XL status. Some of the main features of CAN XL are:

- data field size up to 2 048 byte
- gross bit-rate of 10 Mbit/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 detection capabilities and its well-known robustness. Two very essential functions in a communication protocol are the error detection and the error handling. They have a large impact on the reliability of the communication system. The focus of this article is the error detection mechanisms in CAN XL.

This article 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 mechanisms master to detect all known error types up to a given extent.

CAN XL error detection mechanisms

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

Bit monitoring

Bit monitoring means that a node that transmits a bit also monitors the bit values on the CAN network. 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.

▷

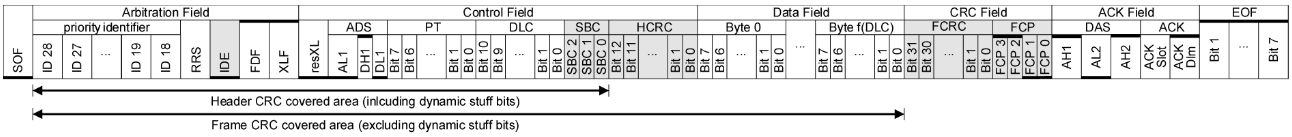


Figure 1: CAN XL frame format (Source: Bosch)

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 CiA 610-1 draft.

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 configuration) and if then this bit is sampled as recessive, the receiver enters a protocol exception state until the network is idle again. This allows the introduction of future new frame formats that are tolerated by existing CAN XL implementations.

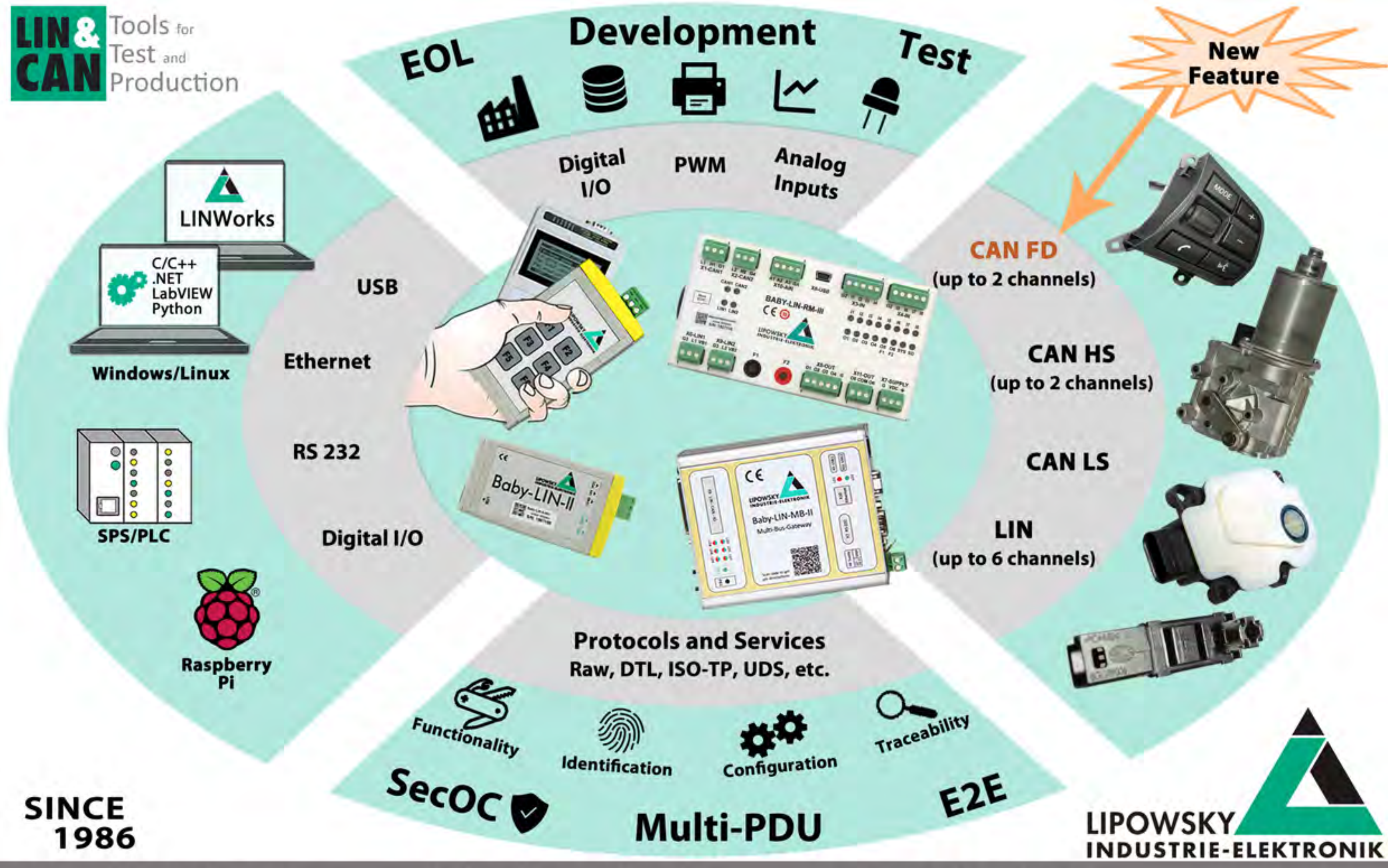
A node transmitting a CAN XL frame sends the FDF and XLF bits as recessive (logical ‘1’). These bits are part of the arbitration field, which is different compared to CAN FD. This means, if the transmitting node samples one of these bits as dominant, it loses arbitration and becomes a receiver.

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.

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. A receiver



can detect, with help of the FPC field, a misalignment of 3 bit in both directions.

CRC concept

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

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 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.

Acknowledgement

Transmitters expect to get an active acknowledgement for their frames, which is a dominant bit in the ACK (acknowledgement) 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).

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.

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.

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 Sth 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.

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.

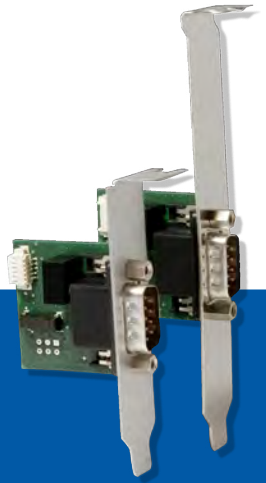
Error signaling

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.

Improved error detection in CAN XL

This chapter highlights the five improvements 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 and by this strengthens the CRC check.
2. Frame CRC: CAN XL uses a 32-bit frame CRC with a respective CRC generator 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 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 further in the article 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.

Error types

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

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.

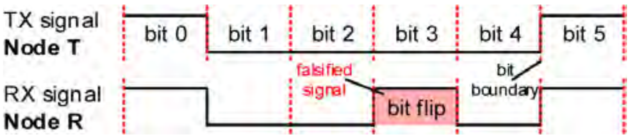


Figure 2: Bit error example (Source: Bosch)

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.

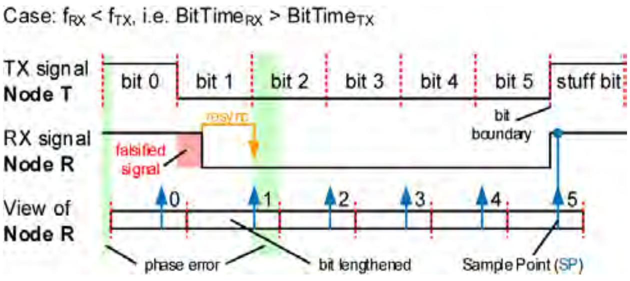


Figure 3: Bit drop example (Source: Bosch)

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).

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”.

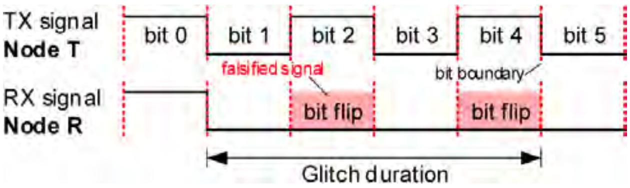


Figure 4: Burst error – all bits forced to one value (Source: Bosch)

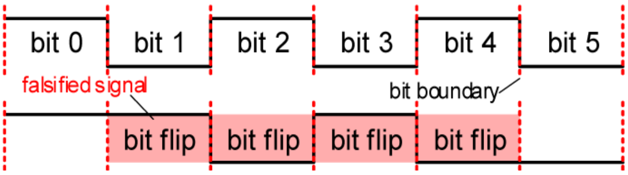


Figure 5: Burst error – due to several bit errors (Source: Bosch)

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. 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].

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 can detect the error.

▷

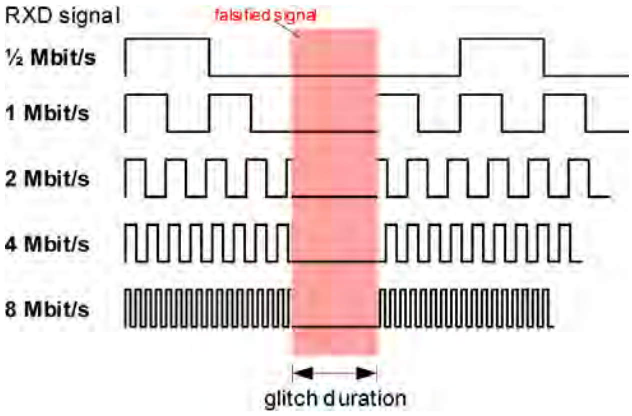


Figure 6: Errors caused by a 2 us glitch at different bit rates (Source: Bosch)

Evaluation: burst error detection

We 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 circumstances. Based on the arguments we mentioned, we conclude that this type 2 burst error is practically extremely unlikely to occur and therefore can be neglected.

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.

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 network depends on the environment around the CAN network. 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.

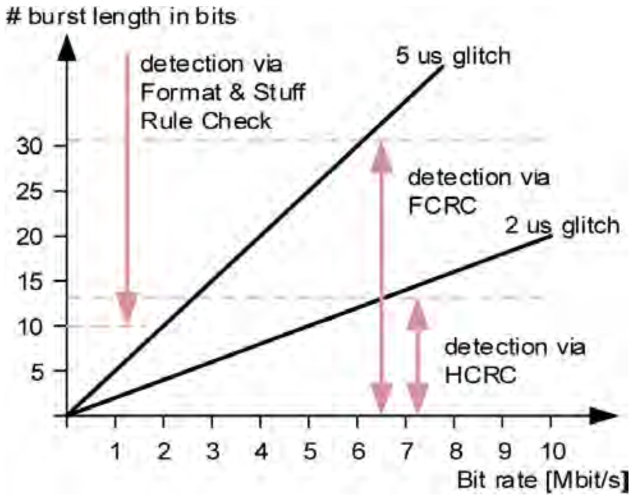


Figure 7 : Error detection mechanisms versus burst errors (Source: Bosch)

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	o CRC check o 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	o Format checks, FCP o Dynamic stuff bit count (SBC)	First described in [10]
Burst error	Glitch length: > one bit length	temporary misalignment of receiver to transmitted bit stream	o Format checks o CRC check (up to a limited burst length	[7] mentions an average burst error length of 5 us; [8] is less explicit

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

- *Stuff rule check:* 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:* The frame can detect one burst error with a length of up to 32 bit.
- *Header CRC:* 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.

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.

▷

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.

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

Summary and conclusion

CAN XL has five 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 article 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. ◀

Author

Dr. Arthur Mutter
Robert Bosch
info@de.bosch.com
www.bosch.com

CRC error detection for CAN XL

CRC generator polynomials for detection of transmission errors in headers and frames of the upcoming CAN XL standard are proposed. Properties, which are chosen to provide error detection performance (compared to competing standards) in the CAN XL scenario, are described.

These properties include achieving Hamming distance 6 for the full range of possible message lengths. At the beginning of the article, a self-contained recap of CRC codes is given.

A new version of the CAN protocol is currently under development: CAN XL. With net data rates up to 10 Mbit/s and beyond, it is designed to bridge the gap between CAN FD and 100Base-T1 Ethernet [1]. Among the design goals for CAN XL are full interoperability with CAN FD as well as large payload length (up to 2 048 byte) in order to enable the use of higher layer protocols such as IP (Internet Protocol) and even encapsulation of complete Ethernet frames [2].

As in any communications system, data transmission in CAN XL is not perfect and transmission errors are inevitable. That is, a transmitted logical zero is detected at the receiver as a logical one or vice versa — a so-called bit error or bit flip. Due to certain physical perturbances in an actual system, bit errors tend to occur in temporally confined groups: so-called burst errors.

Based on elaborate mechanisms that exploit the CAN FD/CAN XL frame structure, certain transmission errors can be detected [3], [4] and corresponding measures can be taken. Frame structure-based error detection alone is not able to provide the required state of the art error detection performance for today's applications, namely probability of undetected bit error below 10^{-20} and guarantee to detect burst errors of a certain length. Thus, in order to provide the required error detection performance, CRC (cyclic redundancy check) codes are employed (Note: that the term "cyclic" at this point is misleading, as many CRC codes used these days do not actually fulfill the definition of a cyclic code (cf. textbooks on error control coding such as [5]). Today, this naming is mainly used for historical reasons).

Competing standards such as Flexray and Ethernet also use CRC codes for error detection and it is our goal to provide at least the same or better error detection performance for CAN XL. This can be accomplished by choosing particular CRC codes, which is the main contribution of this article.

Choosing a particular CRC code is based on certain performance criteria such as the probability of undetected error and the maximal length of a burst error that can be detected with certainty. These in turn depend on the messages that need to be protected and thus on the CAN XL frame structure. The choice is particularly challenging in cases where the messages have variable lengths. For that reason, it was decided early on in the design process of CAN XL to protect the comparatively short and fixed-length header by a so-called header CRC and the whole frame (whose length may vary from several to more than 2 000 byte) by a separate CRC, the so-called frame CRC.

CRC codes

We restrict ourselves to codes over the binary field $\mathbb{F}_2$, i.e., codes over the set {0,1} with operators + (XOR) and · (AND). We denote the set of polynomials of indeterminate x over as $\mathbb{F}_2[x]$. For some $p(x) = \sum_{i=0}^{\infty} p_i x^i$ from $\mathbb{F}_2[x]$ we denote the largest i where $p_i \neq 0$ as $\deg[p(x)]$, the degree of $p(x)$.

In general, the purpose of codes is to cope with transmission errors. The main idea is to add redundancy to a message and transmit the resulting codeword. At the receiver, the redundancy can then be used to recover the transmitted codeword, even if it got corrupted during transmission. This is called error correction. A much simpler task is to use the redundancy in order to determine whether the transmission was error-free or not. This is called error detection.

The message could, for example, be a polynomial $m(x)$ of degree at most $k-1$ (having at most k nonzero coefficients) from $\mathbb{F}_2[x]$. Such a message of message length k can be augmented by M redundant coefficients that are calculated as a function of the message. The process of augmenting message by redundancy is called encoding, M is called the CRC length, $n = k + M$ the code length. The result of encoding is referred to as a codeword. Encoding is called systematic if in any codeword, message and redundancy can be clearly separated (such as in the codeword

$$c(x) = \sum_{i=0}^{k+M-1} c_i x^i = \underbrace{\sum_{i=0}^{M-1} r_i x^i}_{=r(x)} + x^M \underbrace{\sum_{i=0}^{k-1} m_i x^i}_{=m(x)}$$

consisting of message $m(x)$ in the most significant coefficients and redundancy $r(x)$ in the least significant coefficients). Systematic encoders are preferred in practice due to their obvious implementation advantages.

One way of encoding messages $m(x)$, $\deg[m(x)] < k$, into codewords $c(x)$, $\deg[c(x)] < k + M - 1$, is to multiply them with a fixed generator polynomial

$$g(x) = \sum_{i=0}^{M} g_i x^i$$

of degree M from $\mathbb{F}_2[x]$.

$$\mathcal{C}_k = \{m(x)g(x) \; : \; m(x) \in \mathbb{F}_2[x], \deg[m(x)] < k\} \qquad (1)$$

▷

The set of all possible codewords obtainable in this way is called the CRC code $\mathcal{C}_k$, where we maintain the message length k as an index for purposes. This canonical way of encoding (multiplication of messages with the generator polynomial) is not systematic, message and redundancy are intertwined in the resulting codewords and cannot be clearly separated. Due to its definition in (1) one could also refer to $\mathcal{C}_k$ as a polynomial code.

Systematic encoding can be achieved as follows. Instead of multiplying messages with the generator polynomial, the mapping

$$c(x) = \left(\underbrace{m(x)\mathrm{mod}(g(x))}_{=r(x)}\right) + x^M m(x)$$

is performed. Using this form of encoding, the redundancy is the polynomial remain-der of the division $m(x)/g(x)$, i.e., the remainder of polynomial long division (over $\mathbb{F}_2$) applied to message and generator polynomial. It is easy to see that the codewords obtained this way can be written as $c(x) = m'(x)g(x),\ \deg[m'(x)] < k$, and thus $c(x) \in \mathcal{C}_k$. That is, systematic encoding leads to the same code $\mathcal{C}_k$ as canonical encoding, only the mapping from messages to codewords is different.

The effect of systematic encoding as presented before can be described in words: codewords are polynomials of degree at most $k + M - 1$, where the message is shifted into the k most significant coefficients $c_M, \dots, c_{k+M-1}$ and the redundancy is written into the M least significant coefficients $c_0, \dots, c_{M-1}$.

The main reason for the popularity of polynomial codes as described above is the fact that the polynomial remainder of $m(x)/g(x)$ can be calculated using a simple linear feedback shift register. In general, the register in Figure 1 calculates the polynomial remainder of $a(x)/(x^\mu + b(x)),\ \deg[b(x] < \mu$, and stores it (after k clock cycles) in the memory elements $\rho_0, \rho_1, \rho_2, \dots, \rho_{\mu-1}$.

It is clear that the register can be used to calculate $r(x)$ as in (2) by setting $a(x) = m(x)$ $(\kappa = k)$ and $x^\mu + b(x) = g(x)$. Note that $m(x)$ is fed into the register starting with its most significant coefficient m_{k-1} and that its memory elements must be reset to some fixed binary vector (called the initialization vector) beforehand. After m_0 is fed into the register it holds $r(x) = \sum_{i=0}^{M-1} \rho_i x^i$.

Besides calculating $r(x)$ as required for systematic encoding, the same register can also be used to determine whether a given polynomial $v(x), \deg[v(x)] \le k+M-1$, is a codeword. In case it is a codeword, it has to be a polynomial multiple of the generator polynomial $g(x)$ as stated in (1). But this implies that $g(x)$ divides $v(x)$ and thus $\rho_0 = \rho_1 = \rho_2 = \cdots \rho_{\mu-1} = 0$ has to hold if the register is fed with $a(x) = v(x)$ $(\kappa = k+M)$ and $x^\mu + b(x) = g(x)$ $(\mu = M)$. Otherwise (if at least one out of the p_i is nonzero after $k+M$ clock cycles), $v(x)$ cannot be a codeword. It is important to note that the memory elements must be reset to the same initialization vector as used for encoding in the previous paragraph before the $v_{M+k-1}, \dots, v_0$ are fed into the register. We stress that in case $v(x)$ is indeed a codeword, we have $v_{M+k-1} = m_{k-1}$, $\dots, v_M = m_0, v_{M-1} = r_{M-1}, \dots, v_0 = r_0$, where m_i ▷

NEW!
MQTT support for cloud connection and easy-&-fast programming using „Action Rules“

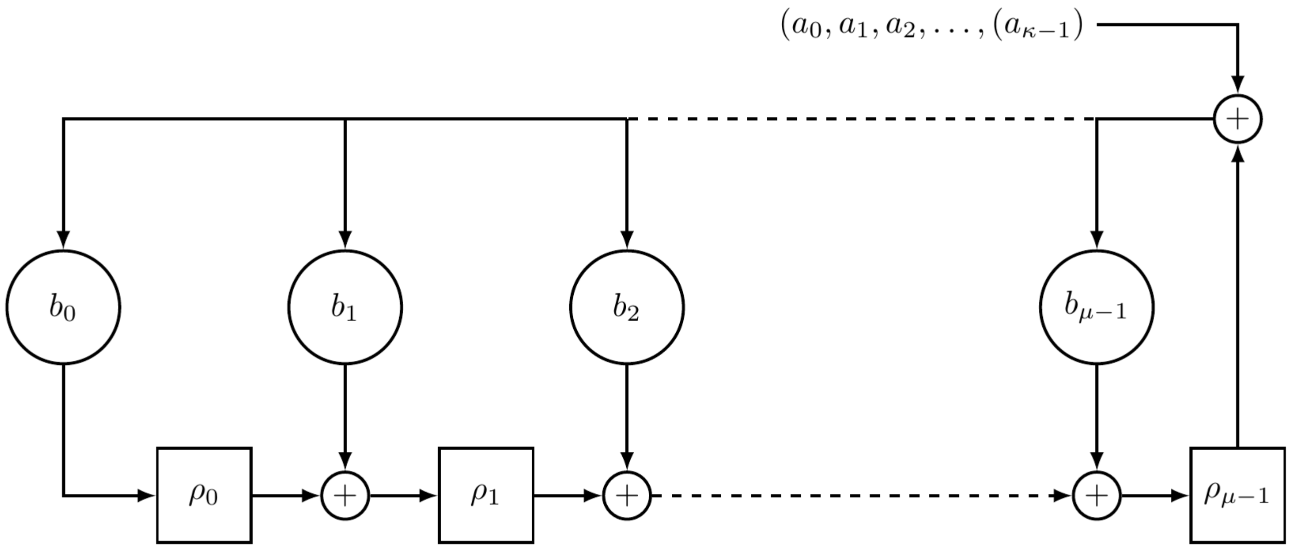


Figure 1: Linear feedback shift register for use with polynomial codes. All operators are from F_2, i.e., + denotes an XOR operation, b_i surrounded by a circle denotes an AND operation with b_i as one of the operands (Source: Dr. Christian Senger)

and r_i are the coefficients of message $m(x)$ and redundancy r(x), respectively.

In practice, CRC codes are used as follows: First, generator polynomial $g(x)$ and initialization vector are chosen as system parameters and made known to both transmitter and receiver. Each message $m(x)$ is encoded into a codeword $c(x)$ at the transmitter (systematically as in (2) using the linear feedback shift register from figure 1 in order to calculate the redundancy $r(x)$.

The codeword is transmitted over a communications channel where it may be exposed to bit and burst errors. As a result, the received word at the receiver $v(x)$ may not be identical to $c(x)$. The receiver now uses the register (configured with $g(x)$ and the initialization vector) in order to check whether $v(x)$ is a codeword or not. If it is not a codeword then a transmission error is detected and appropriate measures are taken.

If it actually is a codeword then two cases are possible: Either $v(x)$ coincides with $c(x)$, which means errorfree transmission. Otherwise, if it does not coincide with $c(x)$, the channel transformed $c(x)$ into another codeword from $\mathcal{C}_k$. The receiver has no way of distinguishing between the two cases and thus the latter case corresponds to an undetected error. Since the probability of having undetected errors depends on the actual generator polynomial, choosing generator polynomials that result in low undetected error rate is of utmost importance.

Properties of CRC codes

As we will see in the following, the undetected error rate is mainly determined by a code parameter referred to as minimum Hamming distance or, in the context of CRC codes, simply Hamming distance HD_k. It states the minimum number of coefficients, in which any two codewords $c(x), c'(x) \in \mathcal{C}_k$, $c(x) \neq c'(x)$ differ.

In our setting (since the considered polynomial codes are linear), HD_k is defined by the minimum Hamming weight of the codewords from $\mathcal{C}_k$, i.e.,

$$\mathrm{HD}_k = \min_{c(x) \in \mathcal{C}_k} \{\mathrm{wt}[c(x)]\}.$$

The Hamming weight $\mathrm{wt}[\cdot]$ of a polynomial $p(x) \in \mathbb{F}_2[x]$ is in turn defined as the number of its nonzero coefficients, i.e., $\mathrm{wt}[p(x)] = \left|\{i \in \{0, \ldots, k+M-1\} : p_i \neq 0\}\right|$.

Since the CRC length M is fixed (by the choice of generator polynomial) the code rate $R = {}^{k}/_{(k+M)}$ approaches one as the message length k grows. Consequently, larger k results in a denser packing of the linear code space and thus (in general) in smaller Hamming distance. Since CAN XL (both header and frame) generates a range of message lengths we have to carry k along as an index for both $\mathcal{C}_k$ and HD_k. Transmission errors can be represented by nontrivial error polynomials

$$e(x) = \sum_{i=0}^{k+M-1} e_i x^i$$

with $\deg[e(x)] \leq \deg[c(x)]$ that distort transmitted codewords $c(x) \in \mathcal{C}_k$ into received polynomials

$$v(x) = c(x) + e(x) = \sum_{i=0}^{k+M-1} v_i x^i.$$

In order to cause an undetected error, the channel has to cause at least HD_k nonzero coefficients in $e(x)$, i.e., it has to cause $\mathrm{wt}[e(x)] \geq \mathrm{HD}_k$ bit errors. It is not possible to take the transmitted $c(x)$ to a different codeword with a smaller number of bit errors and thus transmission errors with $\mathrm{wt}[e(x)] < \mathrm{HD}_k$ bit errors can always be detected. Consequently, larger Hamming distances result in smaller undetected error rates, which is why we always aim for large Hamming distance in the rest of the paper.

Undetected error rate

The undetected error rate states the probability that transmission of a codeword $c(x) \in \mathcal{C}_k$ results in received word $v(x) \in \mathcal{C}_k$ and $v(x) \neq c(x)$. It can be calculated explicitly under the assumption (suggested in [6]) that the transmission channel is a binary symmetric channel (BSC) that flips each transmitted bit with cross-over probability p. Besides this assumption, the weight distribution $(A_k[0], A_k[\mathrm{HD}_k], \ldots, A_k[n])$ of $\mathcal{C}_k$ is required. Its

▷

components $A_k[w], w \in \{0, \mathsf{HD}_k, \ldots, n\}$, give the number of codewords in $\mathcal{C}_k$ having Hamming weight w. Despite being computationally not trivial, it is still possible to calculate weight distributions for moderately sized polynomial codes.

Under the given assumptions, the undetected error rate of a code can be calculated as

$$P_{\mathrm{ue},k} = \sum_{w=\mathsf{HD}_k}^{k+M} A_k[w] p^w (1-p)^{k+M-w}.$$

Since we assume a BSC it is ${}^{(1-p)}\!/_p$ times less likely to have $\mathrm{wt}[e(x)] = t+1$ compared to having $\mathrm{wt}[e(x)] = t$. This fraction goes to infinity as $p \longrightarrow 0$ and thus $P_{\mathrm{ue},k}$ is dominated by its first term, that is, $A_k[\mathsf{HD}_k] p^{\mathsf{HD}_k} (1-p)^{k+M-\mathsf{HD}_k}$.

As a consequence, our criterion for picking generator polynomials for the header CRC in Section V from multiple candidate polynomials with the same HD_k is going to be small $A_k[\mathsf{HD}_k]$ for the full range of relevant message lengths k.

Guaranteed-detectable errors

Some transmission errors can be detected with guarantee. Take for example a code $\mathcal{C}_k$ with $\mathsf{HD}_k = 6$. Any two distinct codewords $c(x), c'(x) \in \mathcal{C}_k$ differ in at least 6 coefficients. That is, taking $c(x)$ and flipping at most 5 arbitrary coefficients cannot result in some $c'(x) \in \mathcal{C}_k$. Or, in other words:

$$\begin{aligned} &\mathrm{wt}[e(x)] < \mathsf{HD}_k \wedge c(x) \in \mathcal{C}_k \\ &\Longrightarrow c(x) + e(x) = v(x) \notin \mathcal{C}_k. \end{aligned}$$

This shows that, in any case, up to $\mathsf{HD}_k - 1$ bit errors can be detected with guarantee. Many transmission errors with much larger Hamming weight can be detected as well but this can in general not be guaranteed. An exception (where there actually are guarantees) are burst errors of a certain maximal length as we will see in the following.

For any transmission error $e(x) \neq 0$, $\deg[e(x)] < k+M$, we define the following two notions: The trailing coefficient $e(x) \neq 0$, $\deg[e(x)] < k+M$ and the leading coefficient $\ell[e(x)] = \deg[e(x)] < k+M$.

The value $\ell[e(x)] - t[e(x)] + 1 \in \{1, \ldots, k+M\}$ is referred to as the burst-length of the error. In general, detecting errors is easier if their Hamming weight and their burst-length are small.

If any $e(x) \neq 0$ is a codeword then (by definition) it has to be a polynomial multiple of $g(x)$. That is, $e(x) = m(x)g(x)$ for some $m(x) \in \mathbb{F}_2[x]$. But this implies

$$\begin{aligned} \ell[e(x)] &= \overbrace{\ell[g(x)]}^{\deg[g(x)]} + \ell[m(x)] \\ t[e(x)] &= t[g(x)] + t[m(x)] \end{aligned}$$

and thus

$$\begin{aligned} &\ell[e(x] - t[e(x)] + 1 \\ &= \deg[g(x)] + \overbrace{\ell[m(x)] - t[m(x)]}^{\geq 0} - t[g(x)] + 1 \\ &\qquad > \deg[g(x)] - t[g(x)] = M - t[g(x)]. \end{aligned}$$

▷

As a result, $e(x)$ cannot be a codeword if $\ell[e(x)]-t[e(x)]+1 \leq M-t[g(x)]$ and consequently any transmission error can be detected as long as its burst-length is at most $M-t[g(x)]$.

In order to guarantee detection of preferably long burst errors it is instrumental to choose $g(x)$ with $g_0=1$ resulting in $t[g(x)]=0$, which (with the above) guarantees detection of error bursts up to burst-length M.

Generator polynomials from $\mathbb{F}_2[x]$ having the special form $g(x)=(x+1)a(x)$, where $\deg[a(x)]=\deg[g(x)]-1$, impose the factor $x+1$ on any codeword $c(x)=m(x)g(x)$, i.e., any codeword can be written as $c(x)=(x+1)b(x)$ with $\deg[b(x)]=\deg[c(x)]-1$. For any such codeword holds $c(1)=0$ because in $\mathbb{F}_2$ we have 1 + 1 = 0 (XOR operation). But the evaluation at 1 of any polynomial from $\mathbb{F}_2[x]$ results in 1 if its Hamming weight is odd and in 0 if the Hamming weight is even. This lets us conclude that all codewords from the resulting CRC code have even Hamming weight and consequently a received word of odd Hamming can never be a codeword. In other words: if the generator polynomial $g(x)$ has x + 1 as a factor then all transmission errors $e(x)$ affected by an odd number of bit flips are detected with guarantee.

In summary we can list types of non-trivial transmission errors $e(x) \neq 0$, $\deg[e(x)] < k+M$, that are guaranteed-detectable by CRC codes with certain generator polynomials g(x):

1. In any case: $e(x)$ is guaranteed-detectable as long as $\mathrm{wt}[e(x)] \leq \mathrm{HD}_k - 1$.
2. If $g_0=1$: $e(x)$ is guaranteed-detectable as long as it contains a single burst error of burst-length at most M.
3. If $g(x)$ has $x+1$ as a factor: $e(x)$ is guaranteed-detectable as long as $\mathrm{wt}[e(x)]$ is odd.

CAN XL frame structure

CAN XL frames consist of a multitude of fields, out of which some are protected by the header CRC (HCRC), some by the frame CRC (FCRC), and some by both. Table I provides an overview. Here, being protected by a CRC means being included in its message polynomials.

It can be seen in the table that besides the obvious data field also the header fields ID, RRS, PT, DLC, SBC as well as the HCRC redundancy are part of the FCRC messages. This approach, which provides extra protection to the header fields at negligible cost, was decided as a result of discussions with Dr. Arthur Mutter and Florian Hartwich, Robert Bosch. The same approach is taken in the Flexray standard.

Some of the fields are affected by dynamic bit stuffing after each run of five identical bits, namely SOF, ID, RRS, and IDE. The number of dynamic stuff bits is stored in the SBC field. Note that the last dynamic stuff bit may be added after the IDE field. Fixed stuff bits as well as any fixed-value fields are not included in any CRC calculation.

We emphasize that the dynamic stuff bits are protected by the HCRC but not by the FCRC. The explanation is given in the following.

Excluding dynamic stuff bits from CRC messages (as in Classical CAN) can result in an undetectable error caused by two bit flips if one bit flip adds and the other removes a dynamic stuff condition. This case is described in [7]. However, including dynamic stuff bits (as in CAN FD) makes the CRC code vulnerable to bit insertions and bit drops at dynamic stuff conditions as described in [3].

Therefore, it was decided to include the dynamic stuff bits in the HCRC calculation but to exclude them from the FCRC calculation. This enables detection of both aforementioned types of errors.

Header CRC (HCRC)

As mentioned before, a dedicated header CRC is proposed for CAN XL. The same approach is followed by the Flexray standard, where fixed-length headers are protected by an M = 11 bit CRC code that achieves Hamming distance 6. The Ethernet standard does not stipulate a dedicated header CRC.

The achievable undetected error rates of codes with Hamming distance 6 are well below 10^{-20}. For relevant CAN XL scenarios with data rates around 10 Mbit/s this means that less than one undetected header error per year per billion devices can be expected. Thus, going to larger Hamming distance seems to be over the top. Consequently, the proposed generator polynomial for protecting the CAN XL header provides Hamming distance 6.

Due to dynamic bit stuffing, HCRC message polynomials consist of at least 34 and at most 37 coefficients (Table I). Thus, any HCRC candidate has to fulfill $\mathrm{HD}_{34}, \mathrm{HD}_{35}, \mathrm{HD}_{36}, \mathrm{HD}_{37} \geq 6$.

It can be verified by exhaustive search that the smallest CRC length M for which candidates fulfilling the HD requirement can be found is M. Out of all the candidates, we propose the generator polynomial

$$\begin{aligned} g_{\mathrm{HCRC}}(x) &= x^{13}+x^{12}+x^{11}+x^{8}+x^{7}+x^{6}+x^{5}+x^{2}+x+1 \\ &= (x^{12}+x^{10}+x^{9}+x^{8}+x^{6}+x^{4}+x^{3}+x^{2}+1) \\ &\quad \cdot (x+1) \end{aligned}$$

for use in the HCRC. Our arguments are described in the following. Note that when we talk about header in this context we mean HCRC message as given by table 1 plus HCRC parity. First, we have (as for any CRC code with Hamming distance 6):

1. Any erroneous header that is affected by no more than 5 bit errors can be detected with guarantee.

Additionally, due to our special choice of $g_{\mathrm{HCRC}}(x)$ (least significant coefficient $g_0=1$ and factor $x+1$) we have:

2. Any erroneous header that is affected a single burst error of burst-length no more than 13 can be detected with guarantee. In other words, any received header where the bit flips are constrained to a set of 13 consecutive bits is guaranteed-detectable.
3. Any erroneous header that is affected by an odd number of bit errors can be detected with guarantee.
4. The undetected error rates $P_{\mathrm{ue},34}$, $P_{\mathrm{ue},35}$, $P_{\mathrm{ue},36}$, $P_{\mathrm{ue},37}$ are minimal among all possible candidate generator polynomials with properties (1) to (3).

We stress that many error patterns that do not fall into cases (1) to (3) can also be detected, but without guarantee.

For the convenience of the reader we state $g_{\mathrm{HCRC}}(x)$ (x) in three commonly used notations:

▷

M	ISO	Normal	Koopman
13	0x39E7	0x19E7	0x1CF3

In order to cope with the aforementioned vulnerability to bit insertions and bit drops related to dynamic bit stuffing the linear feedback shift register must never assume the all-zero state in the first $12+s$ clock cycles when it is fed with the message (cf. [3]). Here, $s \in \{0,\ldots,3\}$ denotes the number of dynamic stuff bits that occur in, in between or after the SOF, ID, RRS, and IDE fields. Note that 12 bits protected by the HCRC (ID and RRS fields) are affected by dynamic bit stuffing. The all-zero state can be avoided by choosing a particular initialization vector such as the proposed initialization vector $(\rho_0,\rho_1,\ldots,\rho_{12}) = (1,0,0,0,0,0,0,0,0,0,0,0,0)$.

Frame CRC (FCRC)

Standards competing with CAN XL such as Flexray and Ethernet utilize CRC codes that achieve Hamming distance HD = 4 for maximum-length frames. For minimum-length frames, Hamming distance HD = 8 (Flexray) and HD = 6 (Ethernet) is achieved. The M = 24 generator polynomial 0xAEB6E5 (Koopman notation) used in Flexray achieves HD = 8 only for ultra short payload sizes (up to 8 byte) and goes down to HD = 6 already at a payload size of 9 byte. On the other hand, it maintains HD = 6 almost up to the maximal payload size of 259 byte. It is thus fair to say that the Flexray FCRC provides HD = 6 for almost all practical payload sizes.

The M = 32 generator polynomial 0x82608EDB (Koopman notation) used in Ethernet performs comparatively bad (despite having 8 bit more redundancy): it achieves only HD =5 for very small payload sizes and this deteriorates to HD = 4 already at a payload size of 372 byte.

In order to achieve comparable CRC error detection performance as the Flexray and Ethernet polynomials, we propose to use a generator polynomial that achieves HD = 6 throughout the full range of possible CAN XL payload sizes, i.e., from 1 byte to 2 048 byte. This is not possible with the M= 24 Flexray polynomial and, in fact, it is not possible with any generator polynomial with M< 31. Thus, in order to provide some safety margin, we propose to use a generator polynomial with M= 32 for the CAN XL FCRC (same CRC length as Ethernet).

It follows that the FCRC message length varies between 34 + 13 + 1 · 8 = 56 and 34 + 13 + 2048 · 8 = 16431 bit. Thus, the task at hand is to find an M = 32 generator polynomial that achieves $\mathrm{HD}_{56},\ldots,\mathrm{HD}_{16431} \geq 6$. Ideally (in order to lower the undetected error rate by guaranteed detection of long burst errors and any odd number of bit errors), the polynomial should have $g_0 = 1$ and it should be divisible by $x+1$.

Finding such polynomials is a computationally very demanding task. For the case M = 32, it has already been tackled in literature. Reference [8] lists the M = 32 generator polynomial 0xFA567D89 (Koopman notation). The same polynomial was already found in [9], but wrongly listed as 0x1F6ACFB13 (normal notation), while it should have been 0x1F4ACFB13 as pointed out by [8].

The Hamming distance profile of the code generated by 0xFA567D89 is shown (among the Flexray and Ethernet polynomials) in Figure 2, solid curve. It can be clearly seen that the polynomial achieves HD = 8 for small payload sizes.

The Hamming distance goes down to HD = 6 at message length k = 275 bit, which is maintained until the maximal payload size. As stated, this includes most of the header fields as well as the HCRC redundancy and thus double-protecting these fields by the FCRC causes no degradation in terms of Hamming distance.

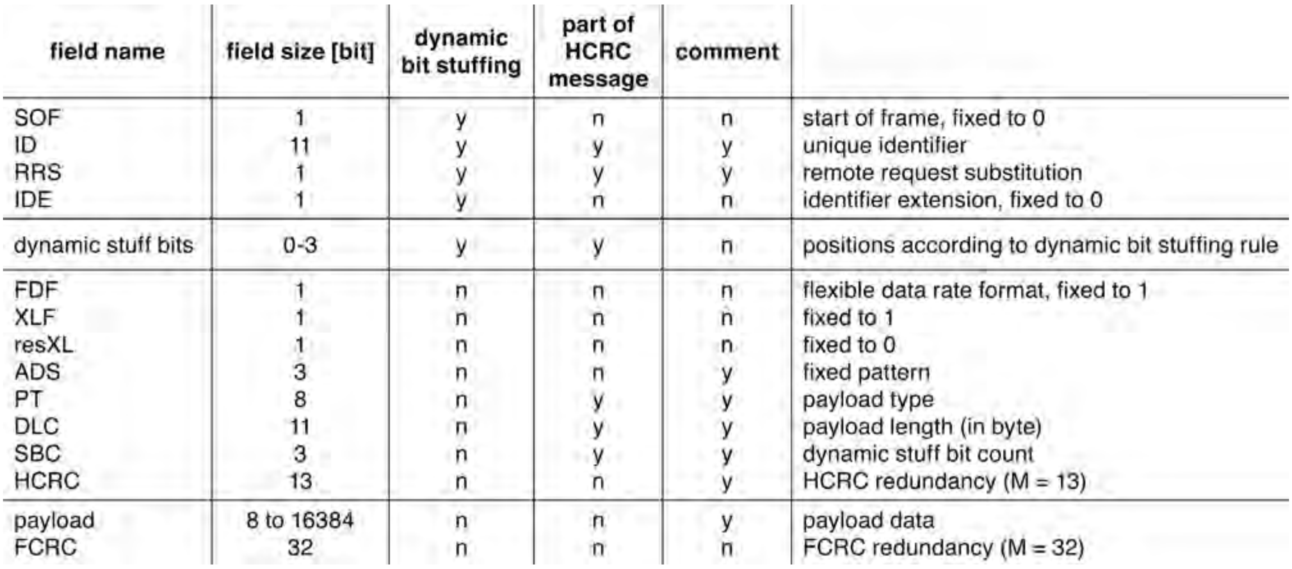
We stress that 0xFA567D89 never falls below one of the Flexray and Ethernet polynomials in the full range of possible CAN XL payload sizes (it actually also outperforms the Ethernet polynomial over the full range of possible Ethernet payload sizes and also the Flexray polynomial over almost the full range of Flexray payload sizes).

Using the code generated by 0xFA567D89 results in the following properties of the FCRC:

1. Any erroneous frame (including all fields marked as "part of FCRC message" in table 1) that is affected by no more than 5 bit errors can be detected with guarantee.

▷

Table 1: Fields of the CAN XL frame that are protected by either of the two CRCs



field name	field size [bit]	dynamic bit stuffing	part of HCRC message	comment	
SOF	1	y	n	n	start of frame, fixed to 0
ID	11	y	y	y	unique identifier
RRS	1	y	y	y	remote request substitution
IDE	1	y	n	n	identifier extension, fixed to 0
dynamic stuff bits	0-3	y	y	n	positions according to dynamic bit stuffing rule
FDF	1	n	n	n	flexible data rate format, fixed to 1
XLF	1	n	n	n	fixed to 1
resXL	1	n	n	n	fixed to 0
ADS	3	n	n	y	fixed pattern
PT	8	n	y	y	payload type
DLC	11	n	y	y	payload length (in byte)
SBC	3	n	y	y	dynamic stuff bit count
HCRC	13	n	n	y	HCRC redundancy (M = 13)
payload	8 to 16384	n	n	y	payload data
FCRC	32	n	n	n	FCRC redundancy (M = 32)

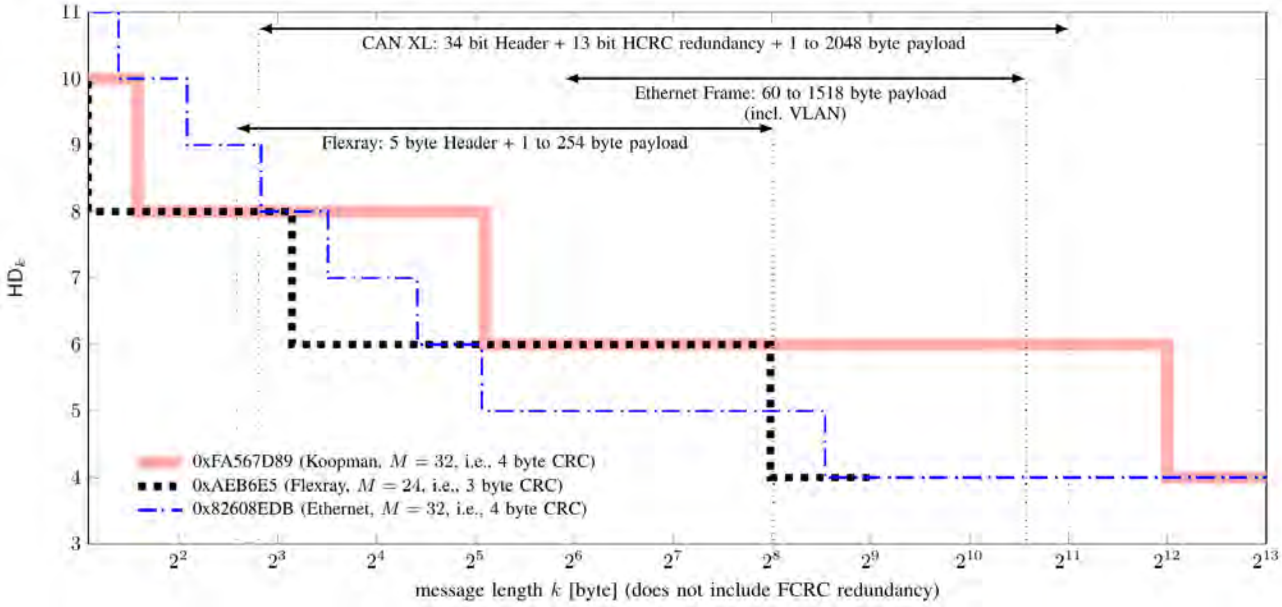


Figure 2: Hamming distance profiles for the Flexray, Ethernet, and proposed CAN XL FCRC generator polynomials. Note that the x-axis is logarithmic and given in byte and thus message length is k = 8 · x (since k is given in bit). All polynomials are given in Koopman notation. (Source: Dr. Christian Senger)

Additionally, due to the fact that 0xFA567D89 has least significant coefficient g_0 = 1 and factor x + 1 (see definition of g_HCRC (x) below) we have:

2. Any erroneous frame that is affected by a single burst error of burst-length no more than 32 can be detected with guarantee. In other words, any received header where the bit flips are constrained to a set of 32 consecutive bit is guaranteed-detectable.
3. Any erroneous header that is affected by an odd number of bit errors can be detected with guarantee.

We stress once again that many error patterns that do not fall into cases (1) to (3) can also be detected, but without guarantee.

Due to its aforementioned properties we propose to use 0xFA567D89 as the FCRC generator polynomial, that is, we propose

$$\begin{aligned} g_{\mathrm{FCRC}}(x) = {} & x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{26}+x^{23} \\ & +x^{21}+x^{19}+x^{18}+x^{15}+x^{14}+x^{13} \\ & +x^{12}+x^{11}+x^{9}+x^{8}+x^{4}+x+1 \\ = {} & (x^{15}+x^{14}+x^{11}+x^{6}+x^{4}+x^{3}+x^{2}+x+1) \\ & \cdot (x^{15}+x^{4}+1) \\ & \cdot (x+1)^2. \end{aligned}$$

For the convenience of the reader we state $g_{FCRC}(x)$ in the three commonly used notations:

M	ISO	Normal	Koopman
13	0x1F4ACFB13	0xF4ACFB13	0xF4ACFB13

The initialization vector plays only a minor role since dynamic stuff bits are excluded from the FCRC. However, defining an initialization vector is inevitable and we propose to use $(\rho_0, \rho_1, \ldots, \rho_{31}) = (1, 0, \ldots, 0)$.

Conclusion and outlook

We presented generator polynomials for use in the header and frame CRCs of the current CAN XL draft and showed that their error correction performance matches or outperforms the CRC codes in competing standards. Further improvements in the undetected error rate could be achieved by taking the actual error patterns that occur in CAN XL systems into consideration, which would require a detailed characterization of those patterns for different real-world scenarios. So far, our proposal is based on the simplifying assumption that the CAN XL bus behaves like a binary symmetric channel with occasional error bursts. In order to improve the detection capabilities for burst errors, CRC codes over larger alphabets could be taken into consideration.

Appendix

Generator polynomials are frequently represented as hexadecimal numbers in order to save space. One way to do that is used in ISO 11898 [10] and works as follows: write the coefficient vector of the polynomial with most significant bit (MSB) first, pad it on the left with zeros to length 4s, where $s = \lceil (M+1)/4 \rceil)$, and then interpret each block of four bits by the corresponding hexadecimal number (again MSB left). This is called the ISO notation. in which, for example, the generator polynomial

$$\begin{aligned} g(x) = {} & x^{18}+x^{15}+x^{13}+x^{12}+x^{11}+x^{10}+x^{9} \\ & +x^{7}+x^{6}+x^{4}+x^{2}+1, \end{aligned}$$

having coefficient vector $(1,0,0,1,0,1,1,1,1,1,0,1,1,0,1,0,1,0,1)$ and $s = 5$, is represented by

$$(\underbrace{0,1,0,0}_{4},\underbrace{1,0,1,1}_{\mathrm{B}},\underbrace{1,1,1,0}_{\mathrm{E}},\underbrace{1,1,0,1}_{\mathrm{D}},\underbrace{0,1,0,1}_{5}) \longleftrightarrow \mathrm{0x4BED5}.$$

For the first alternative notation, write the coefficient vector with MSB first, pad it on the left with zeros to length 4s, replace the leftmost nonzero bit (i.e., g_M = 1) by a zero and then interpret each block of four by the corresponding hexadecimal number. This is called the normal notation in which, for example, g(x) as above is represented by

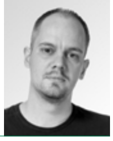
$$(\underbrace{0,1,0,0}_{4},\underbrace{1,0,1,1}_{\mathrm{B}},\underbrace{1,1,1,0}_{\mathrm{E}},\underbrace{1,1,0,1}_{\mathrm{D}},\underbrace{0,1,0,1}_{5}) \longleftrightarrow \mathrm{0x4BED5}.$$

▷

Another alternative representation (popularized by Koopman [8]) can be obtained for $g(x)$ that fulfill the $g_0 = 1$ property (such as the polynomials proposed): write the coefficient vector with most significant bit (MSB) first, pad it on the left with zeros to length $4s+1$, delete the rightmost bit (i.e., $g_0 = 1$) and then interpret each block of four by the corresponding hexadecimal number. This is called the Koopman notation. In Koopman notation, g(x) as above is represented by

$$(\underbrace{0,0,1,0}_{2},\underbrace{0,1,0,1}_{5},\underbrace{1,1,1,1}_{F},\underbrace{0,1,1,0}_{6},\underbrace{1,0,1,0}_{A},\cancel{1}) \longleftrightarrow 0x25F6A.$$

It is straightforward to recover g(x) from any of the hexadecimal notations by simply reversing the respective process. ◀



Author

Dr. Christian Senger
Institute of Telecommunications, University of Stuttgart
senger@inue.uni-stuttgart.de
www.inue.uni-stuttgart.de

References

[1] "Standard for Ethernet - Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)," ISO/IEC/IEEE 8802- 3:2017/Amd 1:2017(E), pp. 1–92, March 2018.

[2] Robert Bosch GmbH. (2019) CAN XL, Next step in CAN evolution. [Online]. Available: https://www.bosch-semiconductors.com/ news/ t-newsdetailpage-4.html

[3] A. Mutter and F. Hartwich, "Advantages of CAN FD error detection mechanisms compared to classical CAN," in In Proceedings of The international CAN Conference 2015 (iCC 2015), 2015.

[4] A. Mutter, "CAN XL error detection capabilities," in In Proceedings of The international CAN Conference 2020 (iCC 2020), 2020.

[5] F. MacWilliams and N. Sloane, The Theory of Error-Correcting Codes, 2nd ed. North-Holland Publishing Company, 1978.

[6] J. Charzinski, "Performance of the Error Detection Mechanisms in CAN," in Proceedings of the 1st International CAN Conference, September 1994, pp. 1/20–1/29.

[7] J. Unruh, H.-J. Mathony, and K.-H. Kaiser, "Error detection analysis of automotive communication protocols," SAE Transactions, vol. 99, pp. 976–985, 1990.

[8] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in Proceedings International Conference on Dependable Systems and Networks, 6 2002, pp. 459–468, doi: 10.1109/DSN.2002.1028931.

[9] G. Castagnoli, S. Brauer, and M. Herrmann, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits," IEEE Transactions on Communications, vol. 41, no. 6, pp. 883–892, June 1993, doi: 10.1109/26.231911.

[10] "Data link layer and physical signalling," ISO 11898-1:2015, pp. 1–65, December 2015.

Facts & Figures

ISO 15765-5

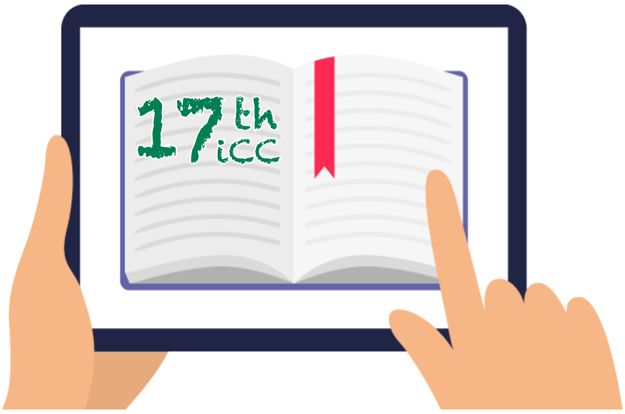
This document specifies the CAN lower-layer requirements for Classical CAN and CAN FD including the bit-timing and the connector. Additionally, it references the network and transport layer as standardized in ISO 15765-2. ISO 15765-5 has passed successfully the DIS (draft international standard) ballot. It will be published soon as international standard (IS).

CiA 406-J + CiA 410-J

These profiles specify the mapping of parameters to J1939 parameter groups. They are based on the CiA 406 and CiA 410 encoder respectively inclinometer profiles. The safety functions are not mapped to J1939.

Home office with kids is challenging.
But we already sold the office facilities.

The 17th international CAN conference has been postponed, because of the new coronavirus. Nevertheless, the iCC proceedings can be purchased in electronic format from CiA office. They contain all 22 papers and the keynote.



More than 2 million forklifts in 2026

Fortune Business Insights predicts an Cagr (compound annual growth rate) of 5,1 percent for forklift trucks. The related study, titled “Forklift trucks Market Size, Share and Industry Analysis, By Type, Application and Regional Forecast, 2019-2026”, counted some 1,5 million units in 2018. Most forklifts are using embedded CAN networks.

CiA 309 series updated

Part 1 and part 3 of the CANopen remote access specification series have been improved. The services and ASCII protocol provide now also TCP/CANopen gateway management functionality. Additionally, part 5 has been introduced specifying Restful HTTP and Websocket protocols to access CANopen networks remotely.

CiA 312-7

This conformance test specification describes how to check CANopen interfaces compliant with the CiA 454 profile series for light electrical vehicles, especially battery-powered bicycles. This includes tests for boot-up behavior, profile-specific SDO communication, and application-specific finite state automaton transitions. ◀

Covid-19 and CAN business

The Sars-CoV-19 virus pandemic has impact on all industries: decreasing business with just a few exceptions on increasing sales for medical equipment, for example. A third of the world population is on lockdown and sometimes supply chains are partly broken.

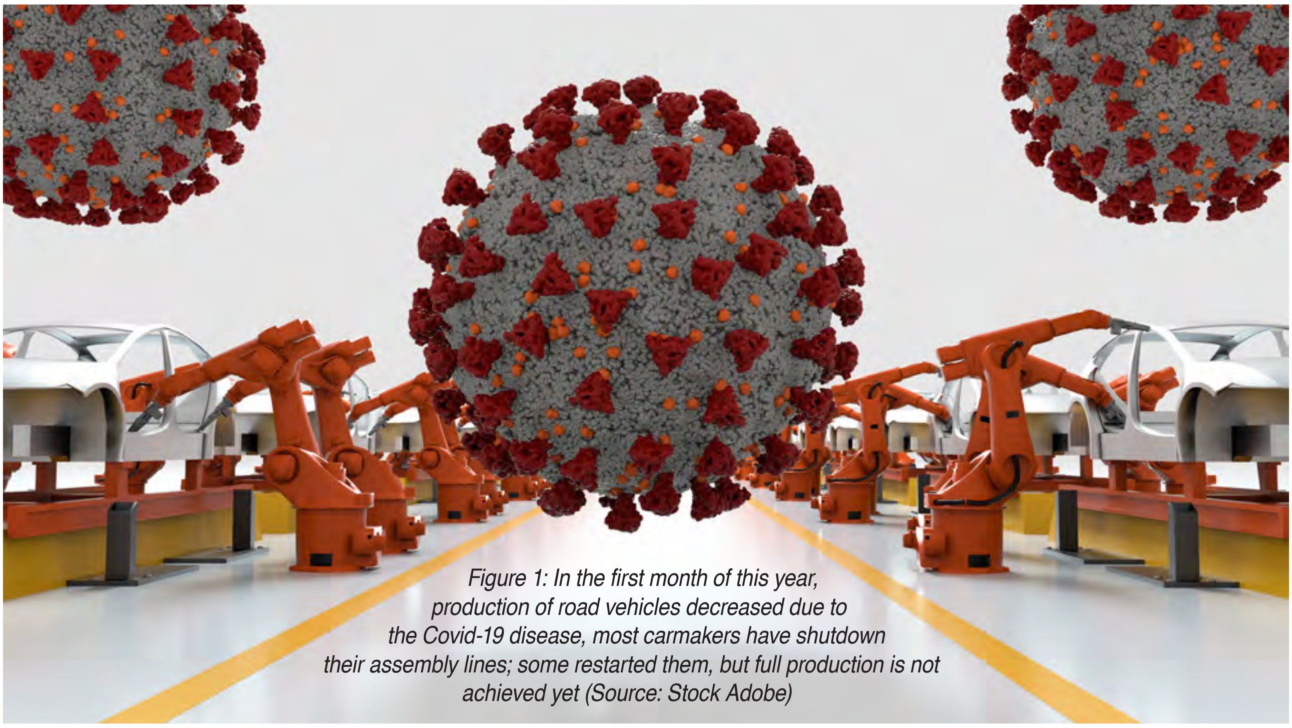


Figure 1: In the first month of this year, production of road vehicles decreased due to the Covid-19 disease, most carmakers have shutdown their assembly lines; some restarted them, but full production is not achieved yet (Source: Stock Adobe)

CAN chip business is down, because the automotive industry has closed many of its factories. Some of them started production again, but carefully and slowly. ACEA, the European association of road-vehicle manufacturers reported already in April a lost production of 1,5 million motor vehicles. This figure includes passenger cars, trucks, vans, buses, and coaches. The number will climb further, if shutdowns are prolonged or more plants are closed. In May, first European carmakers ramped-up production.

CAN chip sales decreases

The automotive industry is by far the largest CAN application domain. Conservative market figures for installed CAN nodes in 2019 are two billions. This year, there will be a significant and dramatic slump in CAN sales figures. On the other hand, for low-volume markets the availability of CAN chips, especially CAN transceivers, should be not a problem.

NXP, the market leading CAN transceiver supplier, explained in an April press release: "While the supply chain disruption experienced post Lunar New Year in China appears to be subsiding, the end market demand trends in the rest of the world have started to significantly deteriorate. Throughout March, the demand headwinds accelerated in the automotive market where many global auto OEMs (original equipment manufacturers) outside of China have shut production lines, and within the industrial and mobile markets where customer demand trends have resulted in the push-out of orders."

Infineon, another chipmaker focused on the automotive markets, has withdrawn its forecast for this year. "The more and more pronounced coronavirus pandemic worldwide is causing severe disruptions to global supply chains, end-markets, and economies. Developments around the coronavirus are very dynamic and result in low visibility." Originally the German company had anticipated to grow revenues by five percent year-over-year (plus or minus two percentage points). "The impact of the coronavirus pandemic can result in a deviation from this expectation and can lead to a noticeable decline in revenue compared to the last fiscal year," stated the company in a press release. The anticipated reduction in revenue will weigh on Infineon's profitability in the 2020 fiscal year, as underutilization charges will go up further compared to the original assessment. Nevertheless, Infineon finalized the integration of Cypress. In June 2019, the two companies had signed an agreement under which ▷

Infineon would acquire Cypress. Both companies produce micro-controllers with CAN controllers.

Microchip reported that their production facilities in Philippines and Malaysia are working with just 10 percent to 30 percent of employees, due to the governmental restrictions in these countries. The supply chain partners, have not had any major disruptions, informed the U.S. chipmaker. “While a few shipments have missed our original committed shipment dates, by and large more than 95 percent of our shipments have met our original commitment dates,” stated Microchip. “We are engaging with clients and continue to accept orders.”

Other CAN semiconductor manufacturers face similar problems due to the Covid-19 pandemic. Renesas (Japan) runs its Malaysian facilities at a limited capacity, because of the governmental restrictions. The Chinese production is since end of March under normal operation.

ST Microelectronics, another market-leading supplier for MCUs (micro-control units) with CAN connectivity, reported that the revenues came in about five percent below the mid-point of our outlook when entering the quarter. CEO Jean-Marc Chery said: “The Covid-19 outbreak and subsequent containment measures by governments around the world brought challenges in our manufacturing operations and, especially in the last few days of the quarter, logistics.” He added: “Our second quarter outlook is taking into account the declining demand environment, especially in Automotive, as well as the ongoing operational and logistics challenges due to current governmental regulations. We anticipate that all of our manufacturing sites will be operational. Some of them will run at reduced capacity, with unsaturation charges currently estimated to be about 400 basis points.”

Development engineers are in home office

Most development engineers are working from home. This is also true for CiA (CAN in Automation) member companies. CiA has re-scheduled all meetings as online events. This means, we are communicating from home office to home office with all the challenges including kids, limited space, and sometimes not optimal technical equipment. Additionally, some companies have put employees in short-time work or have sent them in forced vacations. Short-time work in Germany means reduced salary.

Most of the CiA member companies are operating as usual. Renesas explained in a statement: “We are following guidelines from the government and local authorities and implementing best practices to keep our operations running effectively. There are no plans to shut down headquarters and offices including design centers located within Japan. Employees based in the affected prefectures will continue to work from home.”

Smaller CiA members such as Kvaser (Sweden) informed their customers, that the development team is working from home. “Conscious that more of our customers will be tackling CAN development remotely, our support and field application engineers have put together a few suggestions to help keeping your CAN projects moving ahead.” The Swedish company provides an online guide to use the

CAN Newsletter Online: Coronavirus

Coronavirus

Utilizing service robots to prevent the spread of Covid-19

The Boxer-8110AI fanless embedded box PC from Aaeon is deployed to power automated service robots, helping to reduce person-to-person contact during the corona pandemic. It uses two embedded CAN networks.

Read on

Disinfection robot

Fighting against the coronavirus using CAN

The robots by UVD robots (Denmark) are deployed in hospitals to disinfect rooms and equipment such as patient beds. They use embedded CAN networks.

Read on

Coronavirus

Online in-house seminars by CiA

In the times of the novel coronavirus, face-to-face meetings are not recommended. This is why CAN in Automation (CiA) has withdrawn its scheduled seminars and other training events. As an alternative option, online in-house seminars are offered.

Read on

Disinfection robot

Fighting against the Sars-CoV-2 virus

Within one week engineers by Siemens and Aucma (China) made a normal mobile robot able to disinfect rooms and equipment against the novel coronavirus.

Read on

Covid-19

Corona pandemic and CAN

The new virus goes around the world. CAN is not the right medicine against it, but CAN networks are used in medical equipment helping indirectly in the fight against the Covid-19 disease.

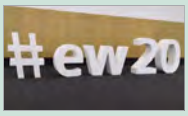
Read on

CAN in Automation

Conference delayed and general assembly postponed

The CAN in Automation (CiA) management has decided to postpone the iCC 2020. No new date has been scheduled yet. Additionally, the CiA general assembly is delayed.

Read on



Embedded World 2020

Covid-19 influenced trade fair

The trade show in Nuremberg (Germany) is an early indicator of the annual trends in embedded electronics. This year, the coronavirus, also known as Sars-CoV-2/Covid-19 influenced the Embedded World 2020.

Read on

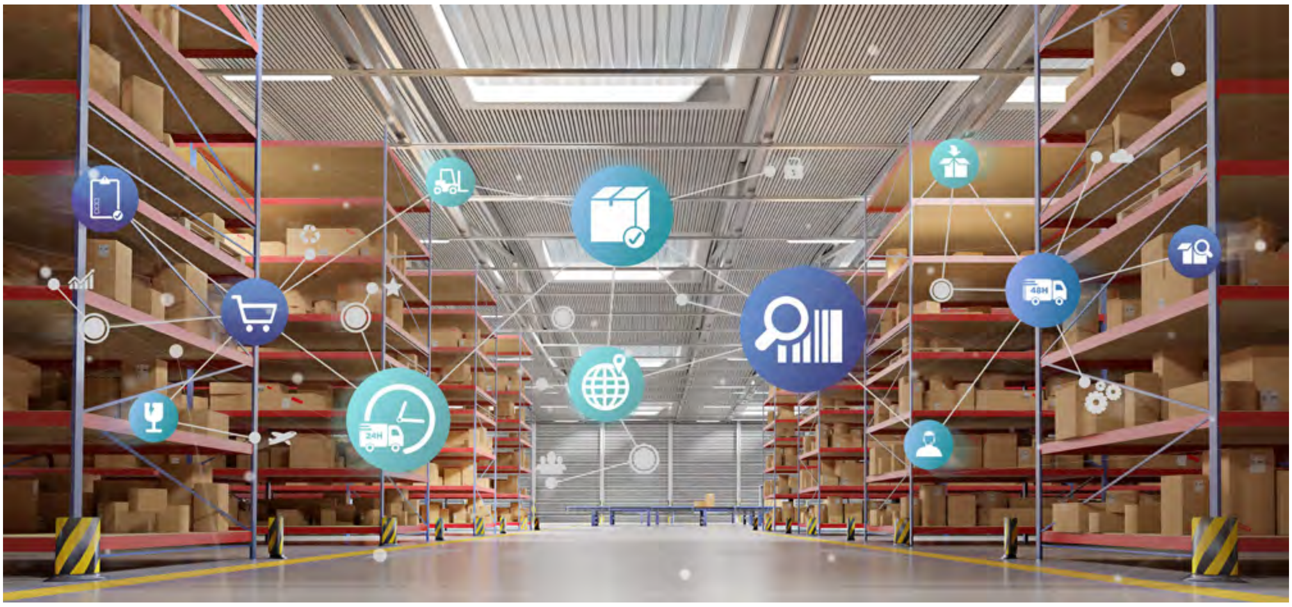


Figure 2: In most cases, the supply-chains for CAN-based devices are not broken, however sometimes single components can cause headaches in the purchase departments (Source: Stock Adobe)

Virtual CAN Driver software coming with the CAN interface boards.

Another topic is licensing: "Aside from ensuring a secure, stable, and fast home connection, a challenge to remote working is the software license you use. Products with multi-user access or maintenance plans often require a license to use software at home. Whilst some companies provide low-cost or free home-access if you make a direct request, others have 30-day free software trials that could get you over the initial 'hump'," explained Kvaser on its website.

Elmo (Israel), a motion control supplier supporting CANopen and CiA 402, continues to operate under the constraints of the new coronavirus. "We are very strict about authority's rules and instructions of hygiene, keeping distance between one employee to another, limiting employees' presence in the working areas, and ramping up remote work from home." The production is working around the clock with less employees in each shift. "It is not efficient production-wise, but it allows us to keep manufacturing in a safe environment," stated the company. "We have experienced a few 'minor' obstacles in our supply chain, but they have been resolved by our purchasing department."

Dunkermotoren, a German drive supplier, prioritizes deliveries related to Covid-19 treatment and analysis. Some of their customers provide intensive care unit beds as well as other medical and laboratory equipment. By default, most of their drives are equipped with CANopen interfaces implementing the CiA 402 profile.

Business trips have been cancelled as far as possible. Maxon (Switzerland), another CiA member providing compact CANopen drives, has decided to cancel or to postpone all trips that are not absolutely necessary. This also includes planned trips to and from customers as well as trips to and from suppliers. The biggest challenge for the company is the supply of materials. "The gaps in the supply chain are increasing and the situation is expected to deteriorate further. Through a short time working program at our headquarters in Switzerland and a global cost savings program we will ensure operational reliability over the coming months. "Until further notice we will continue production five days a week," promised the Swiss company.

Faulhaber (Germany) also manufacturing CANopen drives stated that currently the material supply is guaranteed: "At the production sites, it is a daily challenge for employees to organize themselves, especially due to the restrictions in public life. We also notice this in the availability of production capacities." Nevertheless, the company managed to maintain the delivery situation so far. The sales representatives worldwide are still available to answer questions about order processing and status. Most of them work from home offices and can be reached by e-mail and telephone. "We are currently experiencing strong growth in incoming orders for medical technology products, especially drive systems for respirators, automatic sample analysis and laboratory systems, and infrared cameras for temperature control," reported the CiA member company. "Of course we would like to make our contribution to the medical care to limit and resolve the coronavirus pandemic. For this reason, our production sites continue to work hard to fulfill all customer orders, while maintaining the maximum possible protective measures."

Online event stage

Most of the fairs and exhibitions have been cancelled or postponed. Some companies offer virtual stands and conferences on their websites on dedicated dates. In order not to loose the overview, CAN in Automation will provide an online event stage. On this "stage", which is just a simple list of online events with CAN topics, you can guide yourself to relevant topics appropriate for your project timing. CiA will also go online with its CiA technology days, updating engineers on current CAN developments. The first two events took already place. The list of online events is available on CAN Newsletter Online. ◀

Holger Zeltwanger

Covid-19: More pallet stackers and forklifts are needed

Figure 1: iF design awarded stacker truck (Source: Jungheinrich)

In times of locked down economies with a lot of exit restrictions, online shopping increases. This causes more business for logistics companies requiring increasingly pallet stackers, forklifts, and other transportation equipment.

Besides Amazon, Alibaba, and other e-commerce companies, forklift makers are benefitting from the Covid-19 disease. The logistics centers of online providers need more pallet stackers and forklift trucks to manage the increased goods flow. Most of these vehicles are using embedded CAN networks. The market-leading manufacturers such as Toyota Industries, the Kion Group, and Hyster-Yale Materials Handling have equipped their products with CAN networks since more than two decades.

Already in the 90ties, the U.S.-based Industrial Truck Association (ITA) developed in cooperation with CAN in Automation (CiA) members some recommended practices for CANopen profiles to be used in forklifts. In Europe, the forklift suppliers also installed embedded CAN networks. Some companies developed their own higher-layer protocols, while others such as Jungheinrich implemented CANopen. First the big forklifts were equipped with CAN networks, but today even the smaller pedestrian pallet stackers make use of CAN.

iF-awarded

This year, the Jungheinrich ERC 216zi stacker truck has been awarded with the iF Design Award. Internally, it uses embedded CANopen networks. Due to its integrated lithium-ion battery, the vehicle is compact and agile. The German manufacturer has shortened the truck's length by eliminating the battery trough between the operator's platform and the mast, which was previously common in such trucks. It is at least 170 mm shorter than comparable trucks. This impressed the jury, who honored the ERC 216zi with the iF Design Award in the category Automobiles/Vehicles. The iF Design Award was first presented in 1953 and is considered the oldest independent design award.

In the development of the ERC 216zi, Jungheinrich paid special attention to the ergonomics of the truck. A fixed stand-on platform offers the driver support and comfort. This is an important advantage for the driver, especially during long periods of operation. The operating elements are arranged in such a way that they allow intuitive control of the truck. The vehicle also sets standards in terms of safety. The overhead guard according to ISO 6055 protects against falling objects. The fixed stand-on platform with its fixed side walls additionally offers the operator perfect all-round protection from three sides.

Configuration and diagnostics

Linde, another German supplier, uses in its electric-powered pallet trucks 1152 series and stacker range series 1172 also embedded CAN networks. Besides adjustments to operating parameters, CAN communication is applied for truck diagnosis and maintenance functionality. It monitors all key functions for diagnosis by a service technician. To minimize downtime and to increase productivity, the company also implements maintenance-free AC motors ▷

Figure 2: Pedestrian stackers use CAN networks to improve diagnostics and to simplify maintenance; they also reduce the wiring effort (Source: Still)

Supporting production of disinfectants at a German brewery: In the global effort to fight the coronavirus, basic materials for medical care are urgently needed. The Grooenwohlder brewery (Germany) is therefore producing disinfectants. In support of this commitment, Still (Germany) is providing an electric forklift truck free-of-charge for the company logistics. But do not drink or inject the disinfectants as the U.S. President suggested it "sarcastically" (Source: Still)

with CAN connectivity. The CAN communication enables diagnosis of the service data by laptops. Using this, the service technician can also adjust the performance parameters to the relevant use. In addition, they can reach all of the relevant forklift components behind the motor cover. This also shortens servicing time.

Pedestrian stacker and pallet truck

Yale Europe and its daughter company Hyster have introduced CAN-based pedestrian stackers and pallet trucks. The products use the same platforms and control architectures. They are intended for general-purpose warehouse operations, but are also suitable for low-intensity applications.

Both product families benefit from AC traction motors, regenerative brake systems, emergency reversing device, emergency power disconnect, stepless speed control, and CAN communication that reduces wiring complexity and increases reliability. The pedestrian stacker has a lifting capacity of 1,5 t, while the platform pallet truck has a capacity of 2 t and benefits from the option of electric steering.

The waterproof and dust-proof vehicles are equipped with maintenance-free AC motors. They are connected to the embedded CAN networks as the on-board charger and the lateral battery extraction. ◀

Holger Zeltwanger

Starter kit: Hardware and software

Emotas' CANopen FD starter kit provides a CAN FD micro-controller board, an extension board with CAN FD transceiver, and a CAN FD USB interface to start with CANopen FD immediately.

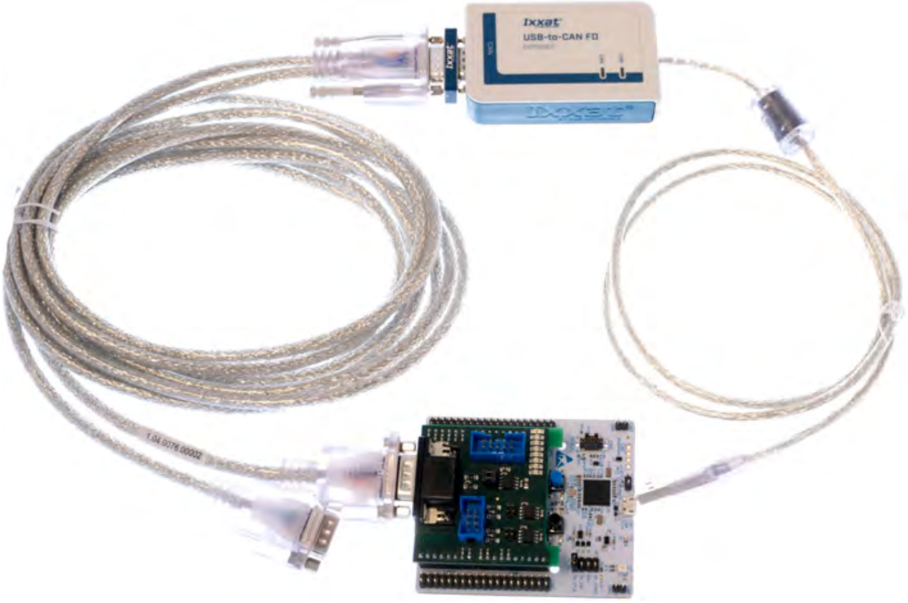


Figure 1: CANopen FD starter kit hardware (Source: Emotas)

CANopen FD as specified in the CAN in Automation (CiA) specification 1301 uses the new features of CAN FD such as a higher data bit-rate and longer frames up to 64 bytes. Most principles of classic CANopen are reused, but some are extended or modified. The most notable improvement is the new USDO service that provides arbitrary access to CANopen FD objects. Compared to the SDO service of classic CANopen it is not only faster but also provides a broad cast mechanism. The other major improvement is the extended PDO length supporting 64 bytes instead of only 8 bytes in one PDO. Last but not least the Emergency messages have been extended as well to provide more detailed information about errors detected by the device. Unfortunately, the number of CANopen FD devices available on the market is currently limited. This is also a problem for developers who would like to evaluate the new features of the improved protocol. In order to provide a cost-effective solution to get started with CANopen FD, Emotas embedded communication, a German company well-known for its CAN and CANopen FD expertise, offers a CANopen FD starter kit.

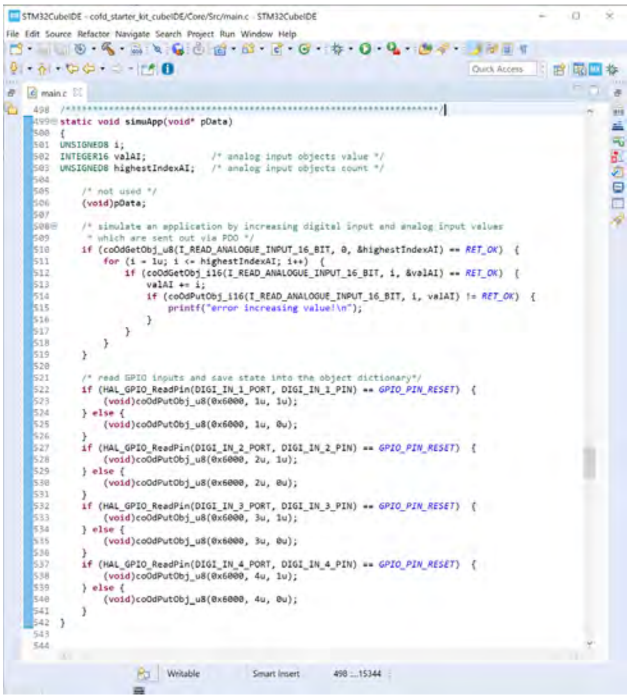
The starter kit unboxed

The CANopen FD starter kit is based on an STM32 Nucleo-64 board with a powerful STM32G4 micro-controller that internally uses an CAN FD controller that supports both Classical CAN and CAN FD. To connect the CAN controller to a CAN FD network a CAN FD transceiver is required. Emotas has developed a specific expansion board with a CAN FD transceiver and DSUB-9 connector to connect the CAN FD network conveniently. The CAN FD transceiver TJA1051 supports up 5 Mbit/s in data phase. In addition to CAN Y-cables to connect also additional devices and two termination resistors the starter kit comes with an Ixxat USB-to-CAN FD interface that supports these bit-rates as well. In addition to the hardware components, the CANopen FD starter kit also includes software: A CANopen FD slave stack to run on the STM32G4 and a CANopen FD tool with CANopen FD master and CAN FD analyzer capabilities.

CANopen FD slave stack

As reported on the CAN Newsletter Online, the Emotas CANopen FD stack was already published in 2017. Based on already three years of CANopen FD experience and recognizing the demand for smaller micro-controllers,

▷



STM32CubeIDE - cofd_starter_kit_cubeIDE/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
main.c
Writable
Smart Insert
498 : 15344

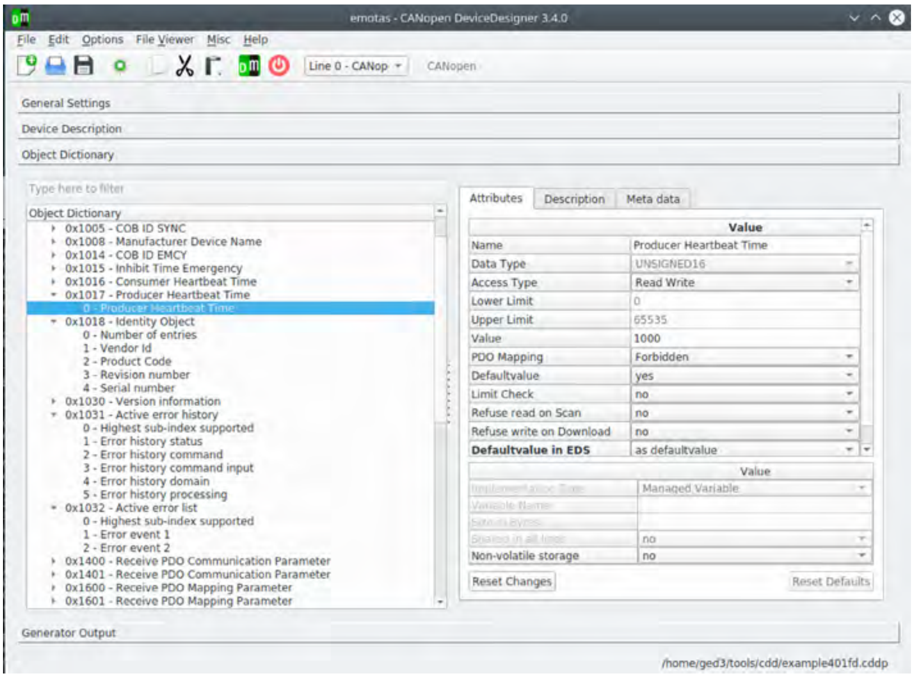
```
/***********************************************************************/
static void simuApp(void* pData)
{
UNSIGNED8 i;
INTEGER16 valAI;             /* analog input objects value */
UNSIGNED8 highestIndexAI;    /* analog input objects count */

    /* not used */
    (void)pData;

    /* simulate an application by increasing digital input and analog input values
     * which are sent out via PDO */
    if (coOdGetObj_u8(I_READ_ANALOGUE_INPUT_16_BIT, 0, &highestIndexAI) == RET_OK)  {
        for (i = 1u; i <= highestIndexAI; i++)  {
            if (coOdGetObj_i16(I_READ_ANALOGUE_INPUT_16_BIT, i, &valAI) == RET_OK)  {
                valAI += i;
                if (coOdPutObj_i16(I_READ_ANALOGUE_INPUT_16_BIT, i, valAI) != RET_OK)  {
                    printf("error increasing value!\n");
                }
            }
        }
    }

    /* read GPIO inputs and save state into the object dictionary*/
    if (HAL_GPIO_ReadPin(DIGI_IN_1_PORT, DIGI_IN_1_PIN) == GPIO_PIN_RESET)  {
        (void)coOdPutObj_u8(0x6000, 1u, 1u);
    } else {
        (void)coOdPutObj_u8(0x6000, 1u, 0u);
    }
    if (HAL_GPIO_ReadPin(DIGI_IN_2_PORT, DIGI_IN_2_PIN) == GPIO_PIN_RESET)  {
        (void)coOdPutObj_u8(0x6000, 2u, 1u);
    } else {
        (void)coOdPutObj_u8(0x6000, 2u, 0u);
    }
    if (HAL_GPIO_ReadPin(DIGI_IN_3_PORT, DIGI_IN_3_PIN) == GPIO_PIN_RESET)  {
        (void)coOdPutObj_u8(0x6000, 3u, 1u);
    } else {
        (void)coOdPutObj_u8(0x6000, 3u, 0u);
    }
    if (HAL_GPIO_ReadPin(DIGI_IN_4_PORT, DIGI_IN_4_PIN) == GPIO_PIN_RESET)  {
        (void)coOdPutObj_u8(0x6000, 4u, 1u);
    } else {
        (void)coOdPutObj_u8(0x6000, 4u, 0u);
    }
}
```

Figure 2: Screenshot application code in STM32CubeIDE (Source: Emotas)



Object Dictionary
- 0x1005 - COB ID SYNC
- 0x1008 - Manufacturer Device Name
- 0x1014 - COB ID EMCY
- 0x1015 - Inhibit Time Emergency
- 0x1016 - Consumer Heartbeat Time
- 0x1017 - Producer Heartbeat Time
  - 0 - Producer Heartbeat Time
- 0x1018 - Identity Object
  - 0 - Number of entries
  - 1 - Vendor Id
  - 2 - Product Code
  - 3 - Revision number
  - 4 - Serial number
- 0x1030 - Version information
- 0x1031 - Active error history
  - 0 - Highest sub-index supported
  - 1 - Error history status
  - 2 - Error history command
  - 3 - Error history command input
  - 4 - Error history domain
  - 5 - Error history processing
- 0x1032 - Active error list
  - 0 - Highest sub-index supported
  - 1 - Error event 1
  - 2 - Error event 2
- 0x1400 - Receive PDO Communication Parameter
- 0x1401 - Receive PDO Communication Parameter
- 0x1600 - Receive PDO Mapping Parameter
- 0x1601 - Receive PDO Mapping Parameter

	Value
Name	Producer Heartbeat Time
Data Type	UNSIGNED16
Access Type	Read Write
Lower Limit	0
Upper Limit	65535
Value	1000
PDO Mapping	Forbidden
Defaultvalue	yes
Limit Check	no
Refuse read on Scan	no
Refuse write on Download	no
Defaultvalue in EDS	as defaultvalue

	Value
Implementation Type	Managed Variable
Variable Name	
Size in Bytes	
Shared in all lines	no
Non-volatile storage	no

/home/ged3/tools/cdd/example401fd.cddp

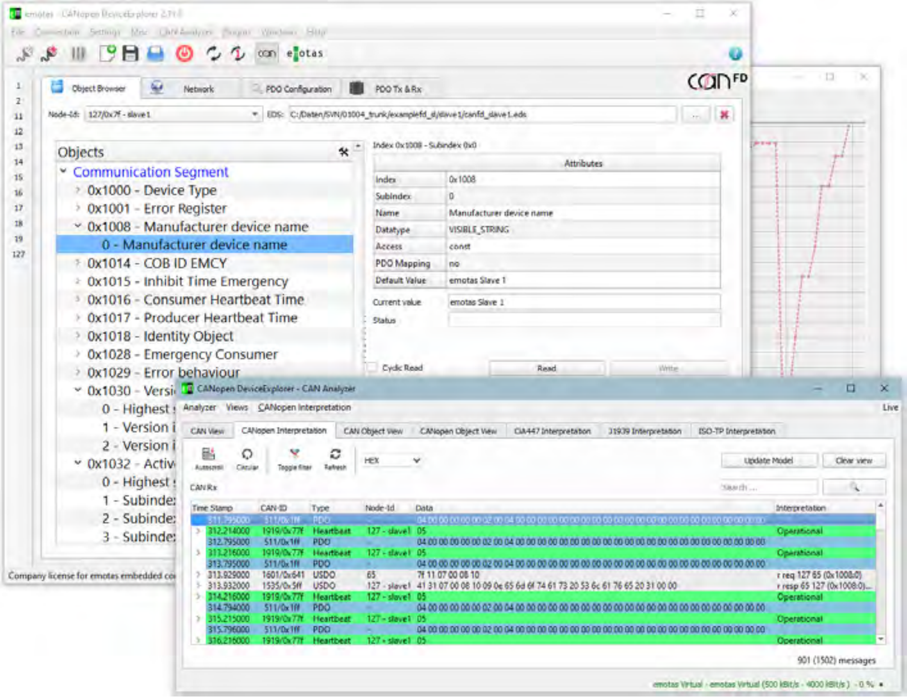
Figure 3: CANopen Devicedesigner (Source: Emotas)

the CANopen FD slave stack has been ported to the STM32G4 recently. An evaluation version of Emotas' CANopen FD slave stack runs inside the STM32G4. It is a binary library of the companies source code stack and to limit the use case to evaluation purposes, the run-time is limited to one hour after reset.

Nevertheless, it comes with the following CANopen FD features:

- NMT Slave
- USDO server with simultaneous connections (expedited unicast and broadcast, segmented unicast and bulk transfer)
- multiple PDO producers and PDO consumers
- Sync consumer
- Heartbeat producer
- 1 Heartbeat consumer
- Emergency producer

An example application is included as STM32CubeIDE project and it simulates a digital/analog I/O device with real and simulated values mapped into longer PDO . Several data objects exceeding the length of 54 bytes are included to show the USDO segmented transfer or USDO bulk transfer as well. By default the bit-rate pair of this example is set to 500 kbit/s nominal bit-rate and 2 Mbit/s data bit-rate. But both the bit-rate pair and the node-ID can be configured from the application. So if one has multiple boards, a network with multiple nodes may be set up. Based on this example, ▷



Node-Id: 127/0x7f - slave1

EDS: C:/Daten/SVN/01004_trunk/examplefd_d/slave1/canfd_slave1.eds

Communication Segment
- 0x1000 - Device Type
- 0x1001 - Error Register
- 0x1008 - Manufacturer device name
  - 0 - Manufacturer device name
- 0x1014 - COB ID EMCY
- 0x1015 - Inhibit Time Emergency
- 0x1016 - Consumer Heartbeat Time
- 0x1017 - Producer Heartbeat Time
- 0x1018 - Identity Object
- 0x1028 - Emergency Consumer
- 0x1029 - Error behaviour
- 0x1030 - Versi
  - 0 - Highest
  - 1 - Version i
  - 2 - Version i
- 0x1032 - Activ
  - 0 - Highest
  - 1 - Subinde
  - 2 - Subinde
  - 3 - Subinde

Index 0x1008 - Subindex 0x0

	Attributes
Index	0x1008
SubIndex	0
Name	Manufacturer device name
Datatype	VISIBLE_STRING
Access	const
PDO Mapping	no
Default Value	emotas Slave 1

Current value: emotas Slave 1

Status:

Time Stamp	CAN-ID	Type	Node-Id	Data	Interpretation
311.795000	511/0x1ff	PDO	-	04 00 00 00 00 00 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
312.214000	1919/0x77f	Heartbeat	127 - slave1	05	Operational
312.795000	511/0x1ff	PDO	-	04 00 00 00 00 00 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
313.216000	1919/0x77f	Heartbeat	127 - slave1	05	Operational
313.795000	511/0x1ff	PDO	-	04 00 00 00 00 00 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
313.929000	1601/0x641	USDO	65	7f 11 07 00 08 10	r req 127 65 (0x1008:0)
313.932000	1535/0x5ff	USDO	127 - slave1	41 31 07 00 08 10 09 0e 65 6d 6f 74 61 73 20 53 6c 61 76 65 20 31 00 00	r resp 65 127 (0x1008:0)...
314.216000	1919/0x77f	Heartbeat	127 - slave1	05	Operational
314.794000	511/0x1ff	PDO	-	04 00 00 00 00 00 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
315.215000	1919/0x77f	Heartbeat	127 - slave1	05	Operational
315.796000	511/0x1ff	PDO	-	04 00 00 00 00 00 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
316.216000	1919/0x77f	Heartbeat	127 - slave1	05	Operational

901 (1502) messages

emotas Virtual - emotas Virtual (500 kBit/s - 4000 kBit/s) - 0 %

Figure 4: Screenshot CANopen Deviceexplorer (Source: Emotas)

the user can extend or modify the object dictionary of the CANopen FD device to meet his requirements. This can be realized by Emotas' CANopen Devicedesigner.

Tool based object dictionary design

The CANopen Devicedesigner allows the definition of CANopen FD object dictionaries based on pre-defined databases e.g. for all CiA 1301 objects. Additionally the user may add own manufacturer-specific and device-profile-specific objects and assign them to PDOs and configure other CANopen FD services. Based on these configurations the tool generates .c and .h files that include the object dictionary implementation and configure the stack. Additionally, the tool generates from the same source a documentation of the object dictionary in CSV, TXT, and HTML files and both a classic CANopen EDS file and a new CANopen FD XDD file according to the latest state of the specification. This XDD file can be imported into CANopen FD tools or CANopen FD masters so that these tools and masters are aware of the slaves' object dictionaries.

CANopen FD master tool with CAN FD analyzer

In order to control a CANopen FD slave device a CANopen FD master is required. Instead of a master device, a PC-based master tool is used to provide a high flexibility especially for development or evaluation. The CANopen Deviceexplorer is such a CANopen FD master tool and additionally includes a sophisticated CANopen FD analyzer that interprets the raw CAN FD messages according to the CANopen FD specification. By using the tool one can access the device' object dictionary by USDO, configure, send and receive PDO and send NMT commands and monitor the raw CAN FD messages and their CANopen FD interpretation simultaneously.

The delivery includes the version for Windows, but the versions for Linux and macOS are available on request. The provided evaluation version of the CANopen Deviceexplorer can only be used with a fixed bit-rate pair of 500 kbit/s and 2 Mbit/s. Object dictionary access, PDO configuration, and NMT commands are limited to the node-IDs 1, 2, 32, and 64. In addition to that, all CANopen FD master features of the tool are enabled and the integrated CAN analyzer interprets all CAN FD messages from all nodes.

Conclusion

The CANopen FD starter kit provides all necessary components to start with a CANopen FD slave development or to evaluate the possibilities of the CANopen FD protocol. In order to spread the usage of CANopen FD, Emotas Embedded Communications not just offers the starter kit for sale, but also provides all software components for download for free. ◀

Author

Torsten Gedenk
Emotas Embedded Communication
ged@emotas.de
www.emotas.de

Vision 3

Enhance human machine interaction

TTControl’s rugged operator interface family Vision 3 was specially designed to meet the requirements of the harsh and more and more digitized off-highway environment.

Features like four simultaneous video streams, support of hardware-accelerated 3D animations, user inputs through a multi-touch capable screen and acoustic feedback via an integrated loudspeaker ensure a perfect interaction of the operator with the vehicle for safe and efficient machine operation.

more information
www.ttcontrol.com/vision3
products@ttcontrol.com

Node-ID assignment using LSS

Since several years the technical interest group developing the CiA 305 specification at CiA discusses which CANopen layer setting services (LSS) should be adapted to CANopen FD. This article summarizes the current work status.

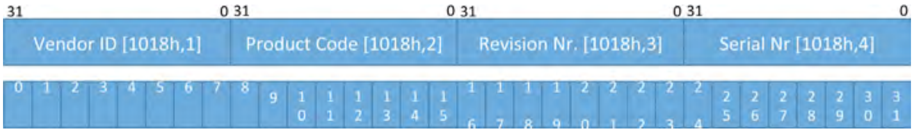


Figure 1: Splitting the 128-bit LSS ID into an array of 32 values of four bit (Source: Embedded Systems Academy)

An updated version of the CiA 305 specification should be published in 2020. The LSS allow low-level configurations of the used CAN (FD) bit-rate(s) and assignment of the node-IDs and network-IDs of the connected CANopen (FD) devices. Enhancing the existing services to configure switching of the bit-rate is relatively simple. Parameters can be added to support the multiple bit-rate combinations of CANopen FD. When it comes to the node-ID and network-ID assignment, a basic requirement for participating devices is the availability of the object 1018_h (identity object) with all four 32-bit sub-indexes: vendor-ID, product code, revision number, and serial number. Together these make a unique 128-bit value (further called 128-bit LSS ID) by which devices can be identified. If the LSS master knows the entire identity object, then node-ID assignment is simple. In a nutshell, the LSS master asks: “Is there anybody here with this 128-bit LSS ID?”. In classic CANopen, this question is fragmented into four CAN frames, in CANopen FD the request goes into a single frame. The LSS device with a matching 128-bit LSS ID replies that it is available. From now on, this device is the only one accepting LSS master configuration commands (all others ignore the commands, as they have not been selected).

Over the last years, there have been several solutions published to the more challenging question: how should the LSS master proceed, if the identity objects of the connected devices are not known. The last standardized approach to this challenge was the LSS Fastscan method using a binary or bit-by-bit search. The LSS master would ask such questions as “Is there anybody here whose highest bit in the 128-bit LSS ID is set?” or “Is this particular bit of your 128-bit LSS ID set or not?”. All LSS devices that want to answer “yes” to such a question reply with a single “ping” message. The “ping” message is a CAN frame with a fixed CAN-ID and the length of zero. If multiple devices transmit the frame at the same time, then there are no errors on the network, as these frames “overlay” and can be transmitted in parallel by multiple devices at the exact same time. On the bottom line, such a method requires the LSS master to typically send 128 requests narrowing the requests down to finally have an exact 128-bit LSS ID selected. Even with a challenging timeout of 30 ms for each request, this takes about 4 s per device to execute.

In one of the CiA (CAN in Automation) application profiles, CiA 447 (CANopen application profile for special-purpose car add-on devices), an additional manufacturer-specific feedback method can be used to significantly shorten the detection time. Here, LSS devices use feedback messages to the LSS master, to inform it about bits in their 128-bit LSS ID. However, this method uses CAN frames with 29-bit CAN-IDs and is limited to 16 devices.

Re-thinking LSS for CANopen FD

As CAN FD is not backward compatible to Classical CAN, LSS for CANopen FD would also not need to be backward compatible, allowing the experts to re-think if other solutions would make sense. One limiting factor is, that any “ping” (“yes” answer to a request) message that multiple devices could send at the same time would still need to be transmitted in Classical CAN format with data length of zero. In CAN FD, even the messages with zero data length may have a difference in at least one bit (error state). Therefore, the messages cannot reliably overlay when they are transmitted at the same time.

For the CiA group, the open demands for a future CANopen FD LSS method are:

- The LSS device side must be simple to implement
- The method must be reliable
- If feedback messages are used, it must be ensured that these also work for a larger number of participants
- Results should be predictable and repeatable

▷

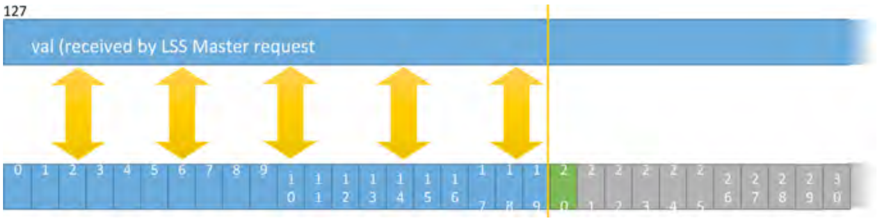
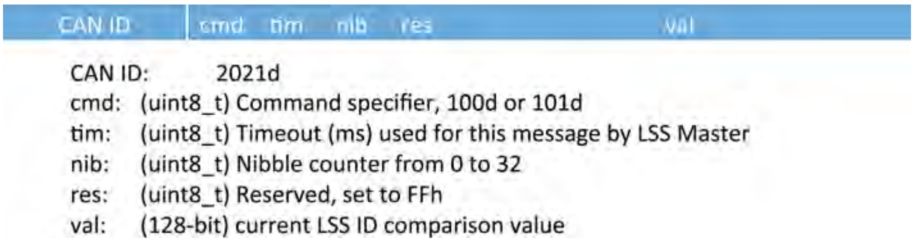


Figure 2: LSS slave handling a request with the nib = 19 (Source: Embedded Systems Academy)


CAN ID
cmd
tim
nib
res
val

CAN ID: 2021d
cmd: (uint8_t) Command specifier, 100d or 101d
tim: (uint8_t) Timeout (ms) used for this message by LSS Master
nib: (uint8_t) Nibble counter from 0 to 32
res: (uint8_t) Reserved, set to FFh
val: (128-bit) current LSS ID comparison value

Figure 3: CAN FD LSS master switch state selective FD request (Source: Embedded Systems Academy)

- The method should be flexible, allowing shortcuts when parts of the 128-bit LSS ID are known

Regarding the predictability (if the same combination of devices is used, the node-ID assignment will be always the same for each node) it must be pointed out, that this was not supported by previous LSS methods either. Any delay in power-up time or internal processing of LSS devices can delay their answers and, thus, participation in LSS assignments. However, once a device has a node-ID permanently assigned (stored in the non-volatile memory) or is known to an LSS master (i.e. stored in a node list on the LSS master side) it can have the same node-ID on every power-up.

To keep a method flexible for faster identification of a partially known 128-bit LSS ID, the LSS device side should be simple. There is already a fast node-ID assignment method when the entire 128-bit LSS ID is known. Optimizations are still required for the few cases where only parts of the 128-bit LSS ID are known.

The new switch state selective FD service

To fulfill the switch state selective FD service, the 128-bit LSS ID has to be divided into 32 pieces (nibbles) of four bit each. The numbers of these nibbles (0 to 31) are shown in Figure 1.

In classic CANopen, LSS devices use a single CAN frame as the “ping” (“yes”) answer to an LSS master identification request. For the new service, 16 feedback messages with the CAN-IDs from $07D0_h$ to $07DF_h$ are used. When the LSS master asks, “What is your first nibble?”, then devices reply with $07D0_h$ if their first nibble is zero, $07D1_h$ if it is one and so forth until $07DF_h$ if their nibble is $0F_h$. The LSS master now picks the first response and packs the nibble value into the next request. Along with the question: “Those of you with the previous nibble matching the one I am sending now, what is your next nibble?” this cycle is repeated until all nibbles have been processed.

Figure 2 shows what happens if an LSS slave receives an LSS master request with a nibble value of 19. If the first 20 nibbles of the LSS slave’s own 128-bit LSS ID matches the one in the request, then it transmits the appropriate response containing the value of its own nibble 20.

Figure 3 shows the contents of the LSS master request. The first byte is the command specifier (cmd). A cmd value of 100_d indicates that only unconfigured nodes should participate in the response. A cmd value of 101_d indicates that all nodes should participate. This allows the LSS master to address nodes that already have a node-ID but should possibly get another one. The second value is the current timeout value (in ms) used by the LSS master (see explanation further in the article). The third is the number of the current requested nibble (0 to 32, last is needed for final confirmation) and the fourth is the current comparison value of the 128-bit LSS ID.

The tests have shown that even with 29 non-configured devices total start-up time is less than 25 s, if devices can initiate feedback transmission within a few single milliseconds. This is not a problem for embedded devices in configuration mode, as during this mode typically no or only a minimal part of the application code is executed.

The code for the LSS device does not require a complex state machine. Each LSS master request is handled by the LSS slaves individually, independent of previous requests.

Timeout handling

LSS devices with limited real-time capability shall only transmit their feedback message, if their internal processing/delay time (i.e. worst delay from receive of a CAN frame to transmitting the response) is 10 ms below the demanded value. For example, devices running a non-real-time operating system (OS) should only provide feedback, when the LSS master timeout is greater than 60 ms. This ensures that devices with slow responses do not delay the entire cycle. Slaves that do not know their response times have to use the largest specified value (100 ms).

The code for the LSS master loops through the 33 (0 to 32) stages of a complete identification cycle. With each transmission of the next LSS master request, the LSS master starts a timeout for collecting feedbacks for the last request. Only the very first feedback received is used, all others are ignored.

Conclusion

The efficiency of the scan cycle using the new switch state selective FD service depends on the LSS master timeout. The tests have shown, that a timeout of 25 ms is realistic and reliable (results in per-node-ID assignment of below 1 s), if all participating LSS devices generate their responses within 10 ms after reception of the LSS master message. For reliability it is essential, that devices not capable of such fast responses only send their response, if the LSS master message signals that a higher timeout is currently in use. ◀

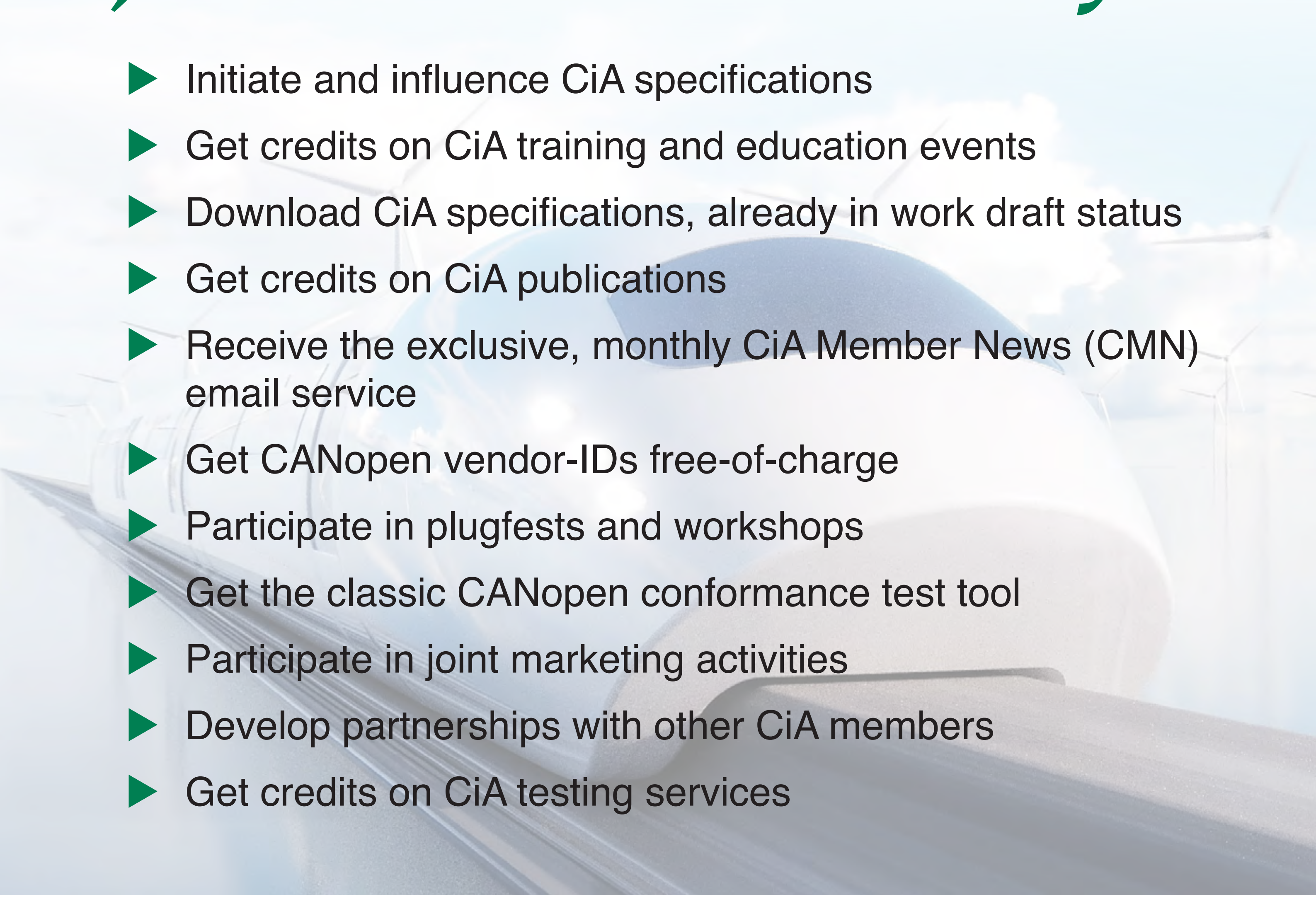
Author

Olaf Pfeiffer
Emsa (Embedded Systems Academy)
info@esacademy.com
www.esacademy.de

CAN in Automation

The non-profit CiA organization promotes CAN and CAN FD, develops CAN FD recommendations and CANopen specifications, and supports other CAN-based higher-layer protocols such as J1939-based approaches.

Join the community!



- Initiate and influence CiA specifications
- Get credits on CiA training and education events
- Download CiA specifications, already in work draft status
- Get credits on CiA publications
- Receive the exclusive, monthly CiA Member News (CMN) email service
- Get CANopen vendor-IDs free-of-charge
- Participate in plugfests and workshops
- Get the classic CANopen conformance test tool
- Participate in joint marketing activities
- Develop partnerships with other CiA members
- Get credits on CiA testing services

For more details please contact CiA office
at headquarters@can-cia.org

www.can-cia.org