

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 \cdot (AND). We denote the set of polynomials of indeterminate x over \mathbb{F}_2 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 $\mathbb{F}_2[x]$.

$$C_k = \{m(x)g(x) : m(x) \in \mathbb{F}_2[x], \deg[m(x)] < k\} \quad (1) \triangleright$$

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

$$c(x) = \left(\frac{m(x) \bmod(g(x))}{=r(x)} \right) + x^M m(x)$$

is performed. Using this form of encoding, the redundancy is the polynomial remainder 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 $v(x)$, $\deg[v(x)] \leq k + M - 1$, 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 $\rho_0 = \rho_1 = \rho_2 = \dots = \rho_{\mu-1} = 0$ 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 ρ_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 $m_i \triangleright$



CAN@net NT
CAN-to-Ethernet Gateway/Bridge
with 4 x CAN and 2 x CAN FD

CAN and CAN FD

Repeater, Bridges and Gateways

- Save costs due to simple wiring
- Increase your system reliability and protect devices by galvanic isolation (up to 4 kV)
- Filter/conversion functionality as well as coupling of CAN and CAN FD
- Bridging of large distances and easy system access via Bluetooth or Ethernet
- **NEW:** Cloud connection via MQTT and easy execution of tasks using "Action Rules" – no programming!



Discover more:
www.all4CAN.com



CANblue II
Bluetooth PC Interface,
Bridge, Gateway



CANbridge NT
(up to 4 x CAN /
2 x CAN-FD)



CAN-CR120/HV
CAN / CAN FD Repeater
(3 kV galv. iso.)



CAN-CR300
CAN / CAN FD Repeater
(4 channels)



CAN-CR 110/FO
CAN / CAN FD
to fiber optic

HMS Industrial Networks GmbH

Emmy-Noether-Str. 17 · 76131 Karlsruhe

+49 721 989777-000 · info@hms-networks.de

www.anybus.com · www.ewon.biz · www.intesis.com · www.ixxat.com



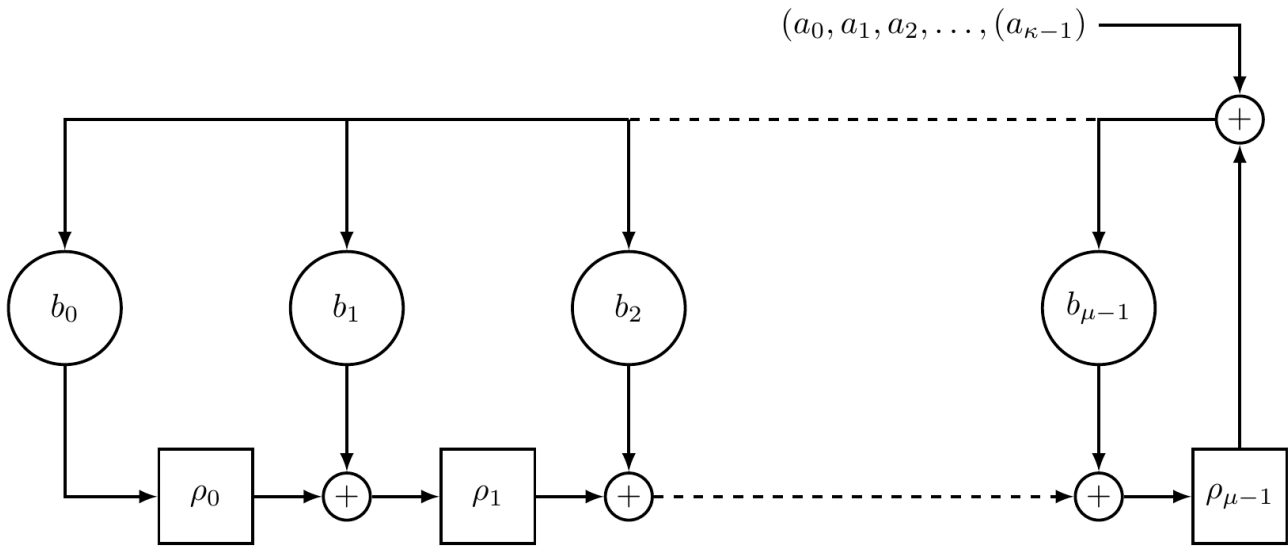


Figure 1: Linear feedback shift register for use with polynomial codes. All operators are from F_2 , i.e., $+$ denotes an XOR operation, b_i surrounded by a circle denotes an AND operation with b_i as one of the operands (Source: Dr. Christian Senger)

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.,

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

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

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 $wt[e(x)] \geq HD_k$ 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 $wt[e(x)] < HD_k$ 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 $(A_k[0], A_k[HD_k], \dots, A_k[n])$ of \mathcal{C}_k is required. Its \triangleright

components $A_k[w], w \in \{0, \text{HD}_k, \dots, n\}$, 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

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

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

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 $A_k[\text{HD}_k]$ 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 $\text{HD}_k = 6$. 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:

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

This shows that, in any case, up to $\text{HD}_k - 1$ 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 $e(x) \neq 0, \text{deg}[e(x)] < k + M$, we define the following two notions: The trailing coefficient $e(x) \neq 0, \text{deg}[e(x)] < k + M$ and the leading coefficient $\ell[e(x)] = \text{deg}[e(x)] < k + M$.

The value $\ell[e(x)] - t[e(x)] + 1 \in \{1, \dots, k + M\}$ 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

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

and thus

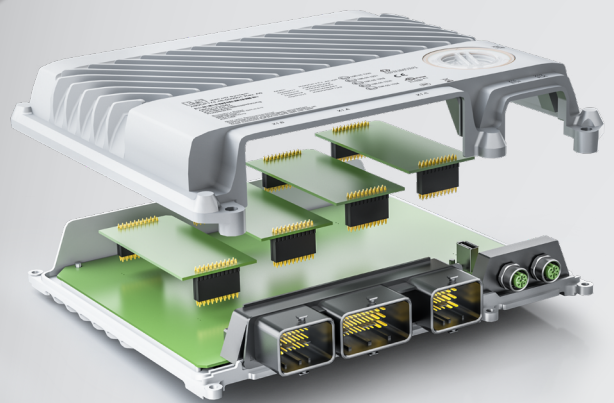
$$\begin{aligned} \ell[e(x)] - t[e(x)] + 1 \\ = \text{deg}[g(x)] + \overbrace{\ell[m(x)] - t[m(x)]}^{\geq 0} - t[g(x)] + 1 \\ > \text{deg}[g(x)] - t[g(x)] = M - t[g(x)]. \triangleright \end{aligned}$$



Completely integrated automation for mobile machinery - X90

Complete portfolio:

www.br-automation.com/mobile-automation



- Easy handling
- Integrated safety
- Faster development



PERFECTION IN AUTOMATION
A MEMBER OF THE ABB GROUP



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 $\text{wt}[e(x)] \leq \text{HD}_k - 1$.
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 $\text{wt}[e(x)]$ 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 $\text{HD}_{34}, \text{HD}_{35}, \text{HD}_{36}, \text{HD}_{37} \geq 6$.

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

$$g_{\text{HCRC}}(x) = x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^5 + x^2 + x + 1 \\ = (x^{12} + x^{10} + x^9 + x^8 + x^6 + x^4 + x^3 + x^2 + 1) \cdot (x + 1)$$

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 $g_{\text{HCRC}}(x)$ (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 $P_{\text{ue},34}, P_{\text{ue},35}, P_{\text{ue},36}, P_{\text{ue},37}$ 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 $g_{\text{HCRC}}(x)$ (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, $s \in \{0, \dots, 3\}$ 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 $(\rho_0, \rho_1, \dots, \rho_{12}) = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

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 $34 + 13 + 1 \cdot 8 = 56$ and $34 + 13 + 2048 \cdot 8 = 16431$ bit. Thus, the task at hand is to find an $M = 32$ generator polynomial that achieves $HD_{56}, \dots, HD_{16431} \geq 6$. 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
FDL	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 ($M = 32$)

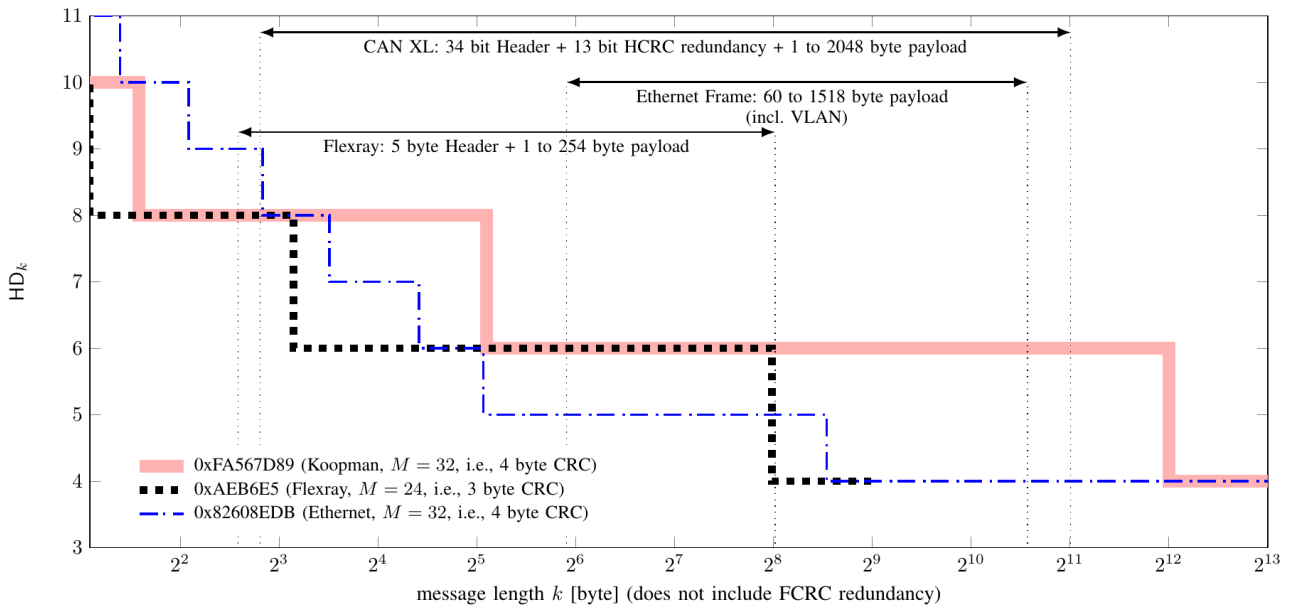


Figure 2: Hamming distance profiles for the Flexray, Ethernet, and proposed CAN XL FCRC generator polynomials. Note that the x-axis is logarithmic and given in byte and thus message length is $k = 8 \cdot x$ (since k is given in bit). All polynomials are given in Koopman notation. (Source: Dr. Christian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

Additionally, due to the fact that 0xFA567D89 has least significant coefficient $g_0 = 1$ and factor $x + 1$ (see definition of $g_{\text{HCRC}}(x)$ 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

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

For the convenience of the reader we state $g_{\text{FCRC}}(x)$ 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 $(\rho_0, \rho_1, \dots, \rho_{31}) = (1, 0, \dots, 0)$.

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

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

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

$$\underbrace{(0, 1, 0, 0)}_4, \underbrace{(1, 0, 1, 1)}_B, \underbrace{(1, 1, 1, 0)}_E, \underbrace{(1, 1, 0, 1)}_D, \underbrace{(0, 1, 0, 1)}_5 \longleftrightarrow 0x4BED5.$$

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

$$\underbrace{(0, 1, 0, 0)}_4, \underbrace{(1, 0, 1, 1)}_B, \underbrace{(1, 1, 1, 0)}_E, \underbrace{(1, 1, 0, 1)}_D, \underbrace{(0, 1, 0, 1)}_5 \longleftrightarrow 0x4BED5. \triangleleft$$

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

$$\underbrace{(0, 0, 1, 0)}_2, \underbrace{(0, 1, 0, 1)}_5, \underbrace{(1, 1, 1, 1)}_F, \underbrace{(0, 1, 1, 0)}_6, \underbrace{(1, 0, 1, 0)}_A, \underbrace{(1, 0, 1, 0)}_A \xleftrightarrow{\quad} 0x25F6A.$$

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.



HIGH-END CONNECTIVITY AND DATA MANAGEMENT

TELEMATICS AND CLOUD SYSTEMS FOR IOT AND SERVICE 4.0

www.s-i-e.de

Sontheim 

Continuous digitization for smart vehicles

Modular on-board units with Linux – ready for condition based monitoring. Including flash-over-the-air and embedded diagnostic functionality.

Sontheim IoT Device Manager and IoT Analytics Manager – for a highly secure, comfortable and individual visualization and management of your data.

Telematic ECU – COMhawk® xt



IoT Device Manager and IoT Analytics Manager



Integrated flash-over-the-air functionality



Modular on-board telematics series



Embedded diagnostics functionality



Multi-protocol support (J1939, J2534, UDS, KWP, ...)



Ready for condition based monitoring