Whitepaper

CRC for CAN with flexible data rate (CAN FD)

1. Introduction

In CAN 2.0, the cyclic redundancy check (CRC) method is used for detection of bit errors. The polynomial used in CAN 2.0 is:

$$g_{CAN}(x) = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$$

When coding a data block, 15 zero bits are appended to the block in the first step. The extended block is then interpreted as a polynomial (where the data bits serve as binary coefficients) and is divided by the polynomial $g_{CAN}(x)$. The rest of this division is a polynomial r(x) which has a degree of 14 (or less). The 15 coefficients of polynomial r(x) then replace the 15 zero bits in the extended data block. This ensures that the division of the so formed new data block by $g_{CAN}(x)$ will produce a remainder of zero in the CAN receiver. If the remainder is non-zero, one or more bit errors have occurred. The polynomial divisions are usually realized in hardware using binary shift registers and XOR gates.

The CAN 2.0 CRC produces a Hamming distance of h = 6. This means that two different valid code blocks differ in at least 6 bit positions, assuring the detection of up to h – 1 = 5 bit errors per data block. The polynomial $g_{CAN}(x)$ of degree 15 is able to support a total block length of up to 2^7 - 1 = 127 bits (112 data bits + 15 CRC bits).

In CAN 2.0 (base frame format, CAN 2.0 A), the data block to be CRC-coded consists of 18 bits (arbitration + control field, start of frame=0 not counted) plus a data field of 0…8 bytes (0…64 bits), which is 18…82 bits in sum. In the case of extended frame format (CAN 2.0 B), the data block to be CRC-coded has a length of 38…102 bits.

CAN with flexible data rate (CAN FD) will allow longer data fields than CAN 2.0, i.e. 12, 16, 20, 24, 32, 48 or 64 bytes. The number of bits to be CRC-coded can be as high as 690 bits (header + payload + worst case stuff bits). For this reason, a CRC polynomial of higher degree for CAN FD is necessary.

2. New CRC-polynomial for CAN FD

The Hamming distance generated by a CRC polynomial for CAN FD was chosen as h = 6, which is the same as in the case of CAN 2.0. A new generator polynomial of degree 21 was designed which supports a maximum block length of 1023 bits (1002 data bits + 21 CRC bits). A CRC polynomial of lower degree (e.g. degree 20) would not be able to support a block length of 690+20 = 710 bits.

The new CRC polynomial of degree 21 with Hamming distance h = 6 chosen for CAN FD is:

$$g_{CAN\ FD}(x) = x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1$$

3. Mathematical Background of CAN FD CRC polynomial

Polynomials with h = 6 can be found based on the theory of error-correcting BCH-codes (Bose-Chaudhuri-Hocquenghem-codes). BCH-codes are cyclic codes which are represented by a generator polynomial g(x). If a BCH-code is designed which is able to correct two errors for a total block length of 1023 bits, then this code will have a Hamming distance of h = 5. If then its polynomial g(x) is multiplied by a factor of (1+x), the resulting code will have an additional parity feature, and the Hamming distance will increase to h = 6.

For designing a BCH code with a block length of 1023 bits (next power of 2 above 690+21, minus 1), we have to find a primitive, irreducible polynomial P(x) of degree 10.

If α is a root of P(x), then P(α)=0. If P(x) is primitive and irreducible, then all non-zero elements of the Galois field GF(2^{10}) can be represented as successive powers of α: 1, α, α^2, α^3, ... α^{1022}.

In the next step, two minimal polynomials $g_1(x)$ and $g_2(x)$ have to be found that fulfill the conditions $g_1(\alpha) = 0$ and $g_2(\alpha^3) = 0$. A minimal polynomial is a polynomial with the lowest possible degree that fulfills the condition. The desired BCH generator polynomial is then $g(x) = g_1(x) \cdot g_2(x)$ (more precisely: the least common multiple of $g_1(x)$ and $g_2(x)$).

Exactly 60 different primitive, irreducible polynomials of degree 10 exist.
One of them is

$$P(x) = x^{10}+x^3+1.$$

The corresponding minimal polynomials are

$$g_1(x) = x^{10}+x^3+1 = P(x) \qquad \text{and} \quad g_2(x) = x^{10}+x^3+x^2+x^1+1.$$

Now the CRC polynomial with h = 6 can be calculated:

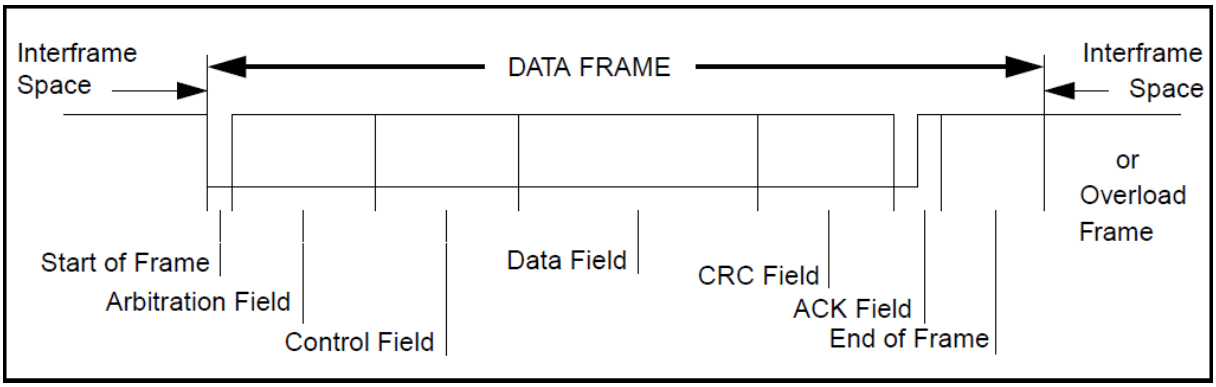
$$\begin{aligned} g_{CAN\ FD}(x) &= (x+1)\cdot g_1(x)\cdot g_2(x) \\ &= (x+1)\cdot(x^{10}+x^3+1)\cdot(x^{10}+x^3+x^2+x^1+1) \\ &= x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1 \end{aligned}$$

Using this polynomial, the correction of two errors and the detection of any odd number of errors would be possible, but the error correction capability is not utilized. Therefore, up to 5 bit errors can be detected using the polynomial as CRC generator.

The primitive, irreducible polynomial P(x) used was chosen among the 60 possible candidates so that the resulting CRC polynomial $g_{CAN\ FD}(x)$ has the smallest possible number of non-zero coefficients.

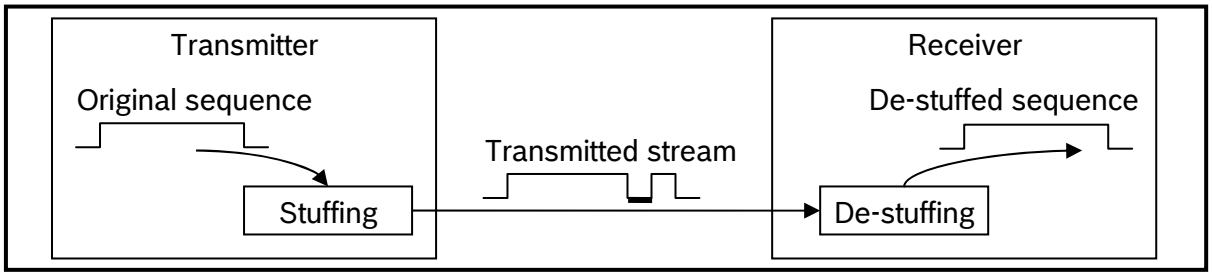
4. Bit stuffing

For the transmission of data frames, the CAN protocol allows a maximum of five consecutive equal-valued bits between start of frame and the end of the CRC field.



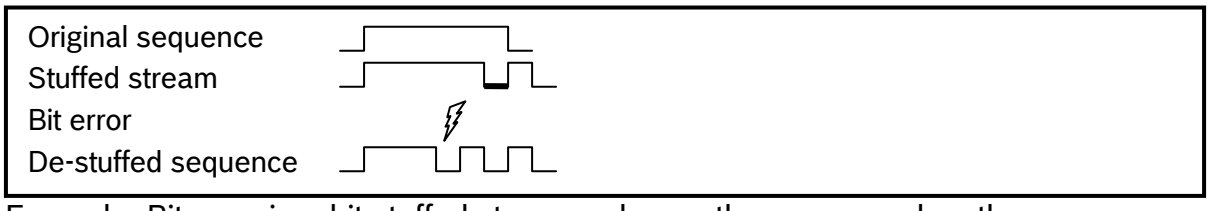
Structure of CAN FD data frames

Bit stuffing is one method to ensure bit changes within the bit stream: whenever five consecutive equal-valued bits have been transmitted, the transmitter inserts an additional inverse bit (the stuff bit) into the transmitted bit stream, whereas the receiver checks and removes the stuff bit again. The number of transmitted stuff bits thus depends on the data content of the sequence; it can increase the sequence length by up to 25% in worst-case.



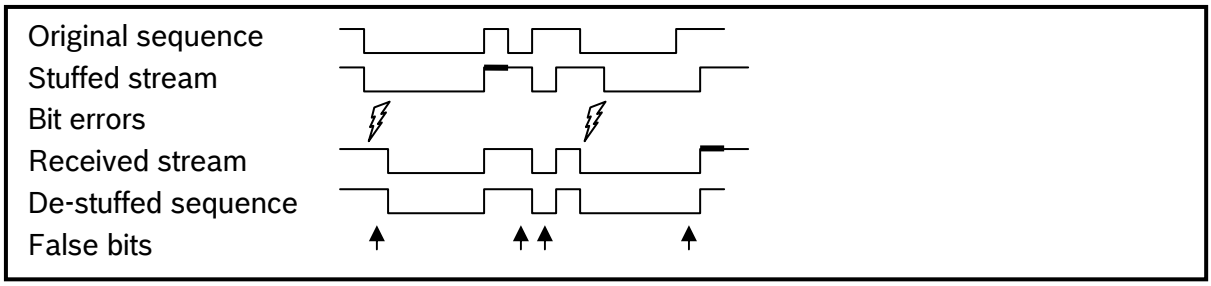
Transmission system using bit stuffing

At the receiver, bit errors in a bit-stuffed stream can cause a misinterpretation of stuff bits as information bits, and vice versa. Typically, this increases or decreases the length of the de-stuffed sequence compared to the original sequence length and thus is detectable in a frame length check.



Example: Bit error in a bit stuffed stream enlarges the sequence length

Nevertheless, combinations of bit errors with balanced length increasing and decreasing can result in de-stuffed sequences of original length, in which data bit patterns are shifted in position. For these critical combinations, the bit-stuffing mechanism potentially increases the effective number of false bits.



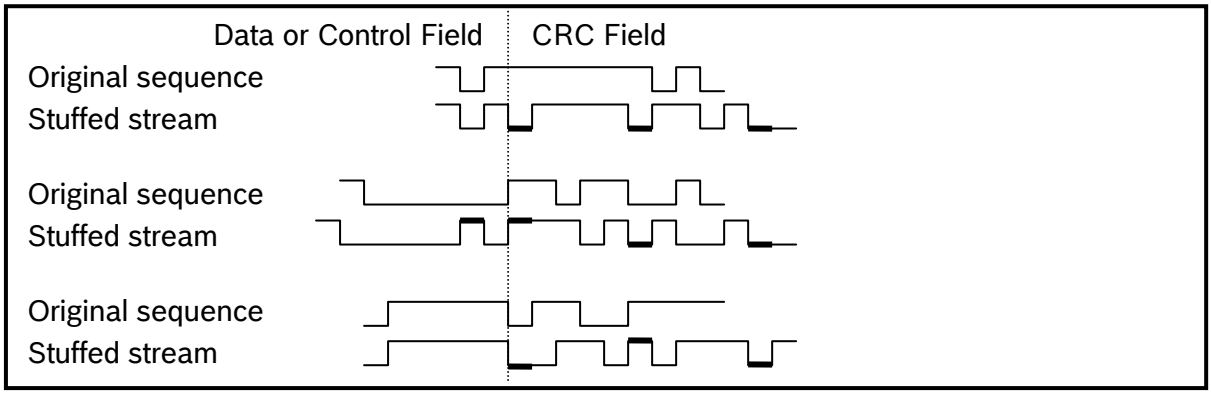
Example: Bit error combination with balanced length increasing and decreasing

In CAN FD data frames, bit stuffing is used for the transmission of all bits occurring before the CRC field; these bits are first stuffed and then CRC-protected. The chosen order of stuffing and CRC-protecting avoids the undermining of the CRC's Hamming distance by a potential increase in false bits due to critical combinations described above.

5. Fixed stuff bit positions for CRC Field transmission

For the transmission of the CRC field, a scheme of fixed stuff bit positions is used: the transmitter inserts a fixed stuff bit prior to the CRC field transmission, even if the preceding bits do not fulfill the CAN stuff condition. If the last bits of the preceding field do fulfill the CAN stuff condition, the stuff bit is interpreted as fixed stuff bit, so it shall not be taken into account for CRC calculation or CRC check and there is no second stuff bit at the start of the CRC sequence.

Further fixed stuff bits are inserted after each fourth bit of the CRC sequence. The value of each fixed stuff bit is the inverse value of the preceding bit. At the receiver, the fixed stuff bits are checked and removed from the CRC bit stream before executing the CRC check.



Examples: Insertion of fixed stuff bits for CRC Field transmission

This scheme also ensures a maximum of five consecutive equal-valued bits in the transmitted stream. Fixed stuff bit positions induce a fixed number of stuff bits, and thus prevent from length increasing or decreasing misinterpretation.