Automotive Connectivity
Module 1: Lesson 3

Carlo Augusto Grazia

Tenure-Track Assistant Professor

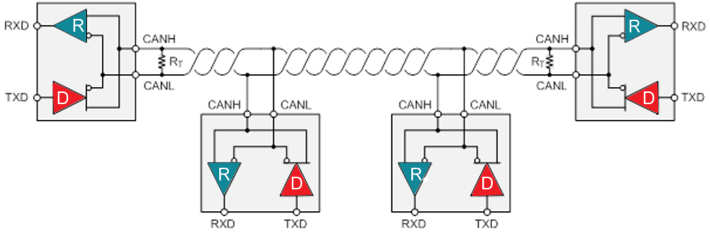
Department of Engineering *Enzo Ferrari*
University of Modena and Reggio Emilia

Modena, 19th September 2024

The **DARK** electronic side of the CAN Bus

CAN Wires

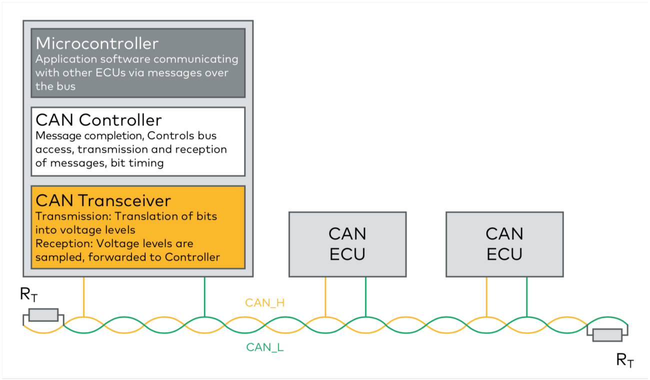
Do you remember this figure?



- Two wires give noise resistance and increase resiliency
- CAN High & CAN Low, twisted (if one brakes, CAN low survives)
- Impedance R_T of 120Ω at the ends

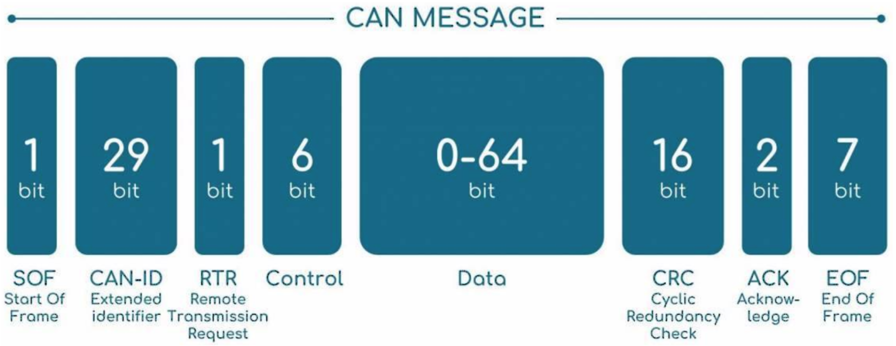
CAN ECUs Structure

Each ECU attached to the CAN must follow the CAN interface


Microcontroller
Application software communicating with other ECUs via messages over the bus
CAN Controller
Message completion, Controls bus access, transmission and reception of messages, bit timing
CAN Transceiver
Transmission: Translation of bits into voltage levels
Reception: Voltage levels are sampled, forwarded to Controller
CAN ECU
CAN ECU
R_T
CAN_H
CAN_L
R_T

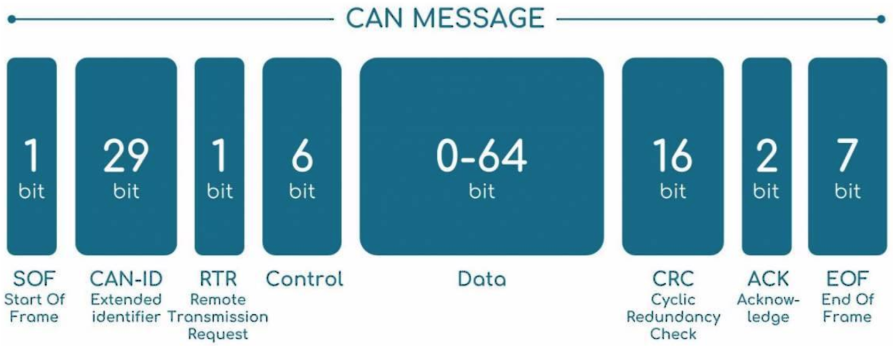
The CAN interface consists of a CAN controller and a CAN transceiver

CAN Message



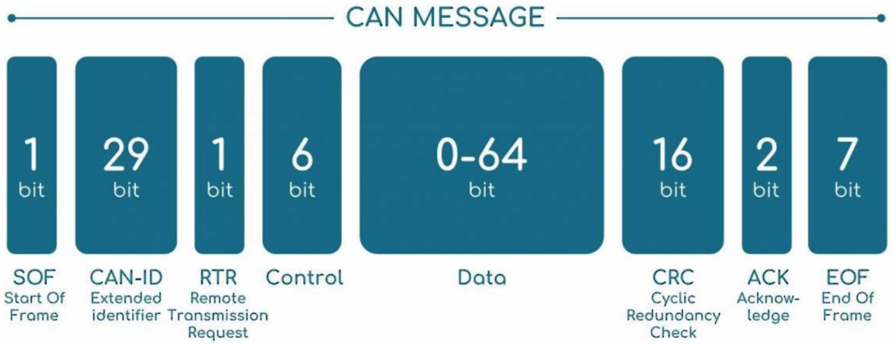
- **SOF**: The Start of Frame is a 'dominant 0' to tell the other ECUs that a message is coming
- **CAN-ID**: Contains the message identifier - lower values have higher priority (e.g. RPM, wheel speed, ...)

CAN Message



- **RTR**: The Remote Transmission Request allows ECUs to "request" messages from other ECUs
- **Control**: Informs the length of the Data in bytes (0 to 8 bytes)
- **Data**: Contains the actual data values, which need to be "scaled" or converted to be readable and ready for analysis

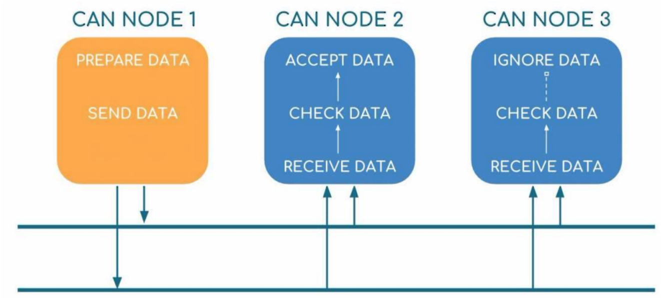
CAN Message



- **CRC**: The Cyclic Redundancy Check is used to ensure data integrity
- **ACK**: The ACK slot indicates if the CRC process is OK
- **EOF**: Marks the end of the CAN message

CAN Message passing

CAN uses the receiver-selective form of addressing



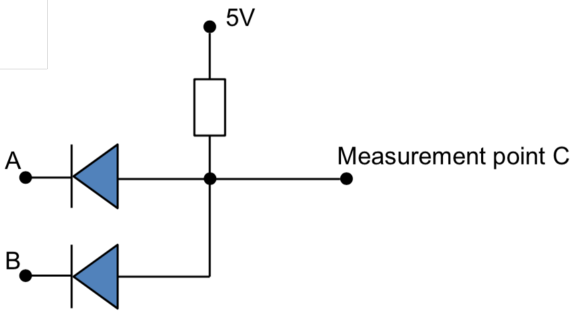
- Cost savings through shared use of sensors by all ECUs on the bus.
- Easy implementation of distributed functions
- It allows different configurations without adaptation of hardware or software

CAN bus bit logic

- Each ECU reads the wire (through a buffer)
- Each ECU **can** write on the line (through a transistor)
- Base state of CAN bus:
  - Transistor in non-conductive state
  - The base state is *up* (+5V, bit logical value of 1)
- One or more ECUs turn transistor conductive (diode)
  - This connects bus to signal ground
  - Bus level is *low* (0V ground, bit logical value of 0) independently form other ECUs
  - The 0 bit is the **dominant** level
- Wired AND (if one ECU writes a 0, the state will be 0)

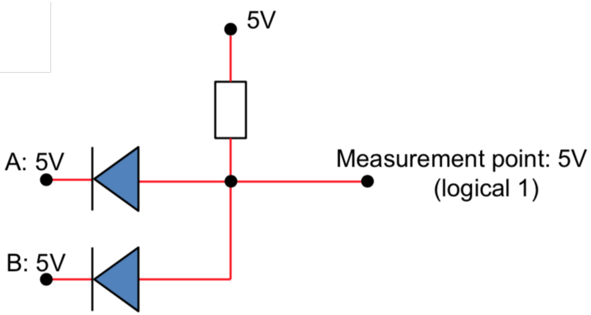
CAN Bus: Wired AND

A and B are the logical value the 2 ECUs want to write on the bus (C)



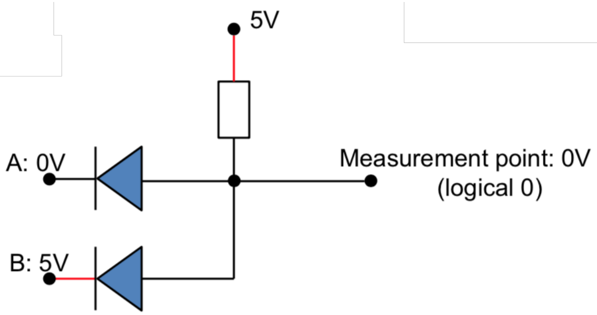
CAN Bus: Wired AND

If A and B are both 1 (+5V), than C will be 1 (+5V)



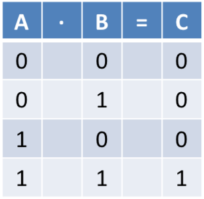
CAN Bus: Wired AND

If A is 0 (ground) and B is 1 (+5V), or the opposite, than C will be 0

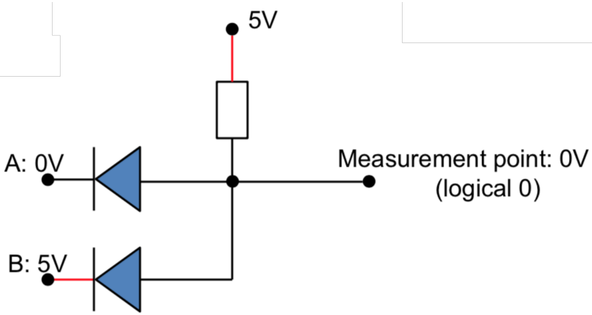


CAN Bus: Wired AND recap

One truth table worth more than thousands words



A	·	B	=	C
0		0		0
0		1		0
1		0		0
1		1		1



Problem

Which ECU can transmit?
When?

CAN Message Tx

- CAN is an event-driven bus system
  - no need to wait for a precise scheduled time-slot
  - possible collisions (do you remember CSMA-CD and CSMA-CA?)
- ECU X registers an event e: it is authorized to access the bus immediately and send data
- Exception: if another ECU Y is already transmitting data, then X waits

CAN Message Tx

How long takes a message to be sent?

- Maximum size M of 130bit
- Average bus speed B is 500kbit/s
- Tx time is $\frac{M}{B} = 0.25ms$

CAN Message Tx

How to avoid collisions?

- What if ECU X and Y are waiting for ECU Z to end the transmission?
- They (probably) start to transmit together once the bus is free -> **collision**
- Solution: **CSMA-CR** and not CSMA-CD as a wired connection would be expected to

CAN Message Tx: CSMA-CR

Priorities Instead of Collisions

- ECU X wants to send: it must check whether the bus is free (Carrier Sense – **CS**)
- If the bus is busy, the ECU must wait.
- When the bus is available again, maybe both ECUs X and Y would start together to transmit (Multiple Access – **MA**)
- How to avoid the impending damage from this collision? (Collision Resolution – **CR**)
- The answer is **bitwise arbitration**

CAN Message Tx: CSMA-CR

Priorities Instead of Collisions: bitwise arbitration

1. All ECUs with a transmission request simultaneously send the identifier of their respective CAN message to be transmitted, bitwise from the most significant to least significant bit.
2. **Remember!** A bit with significance 0 is dominant on the CAN bus.
3. if two ECUs X and Y simultaneously transmit different bit values, the 0 value prevails over the 1 value on the bus.
4. Each ECU compares the value on the bus with the value it sent: bit monitoring.
5. The rules of the arbitration logic determine whether an ECU may continue sending or must stop

CAN Message Tx: CSMA-CR & bitwise arbitration

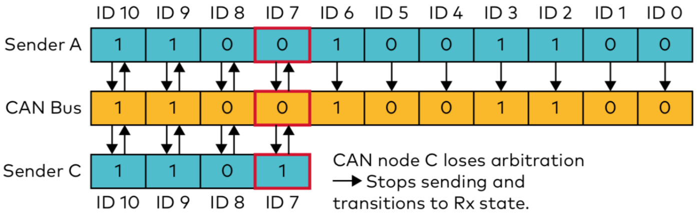
The ECU with lower ID wins

Wired-AND Bus Logic: 0 = dominant

Sender A	Sender B	Bus Level
0	0	0
0	1	0
1	0	0
1	1	1

Arbitration Logic

Sender	Bus	Interpretation
0	0	Next
0	1	Fault
1	0	Stop
1	1	Next

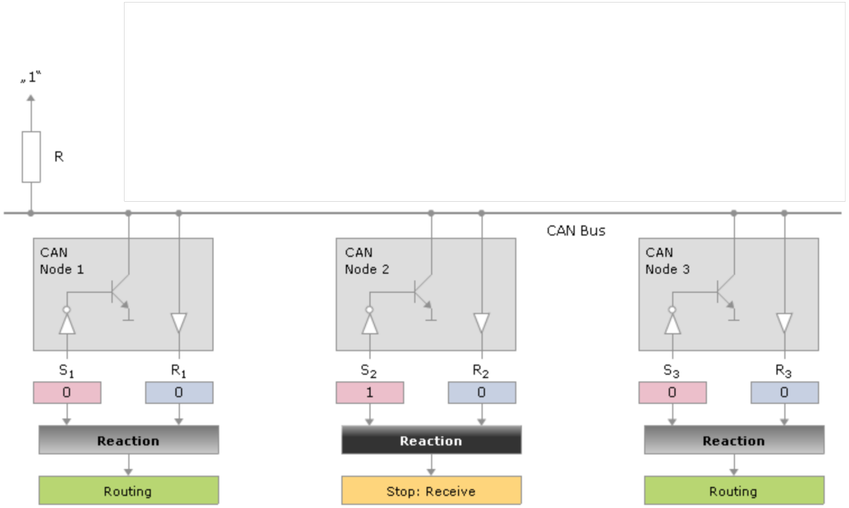


ID 10 ID 9 ID 8 ID 7 ID 6 ID 5 ID 4 ID 3 ID 2 ID 1 ID 0
Sender A 1 1 0 0 1 0 0 1 1 0 0
CAN Bus 1 1 0 0 1 0 0 1 1 0 0
Sender C 1 1 0 1
ID 10 ID 9 ID 8 ID 7

CAN node C loses arbitration
$\longrightarrow$ Stops sending and
transitions to Rx state.

CAN Message Tx: CSMA-CR & bitwise arbitration

Rare picture of a ECU node that dies in a battle



CAN Message Tx: CSMA-CR & bitwise arbitration

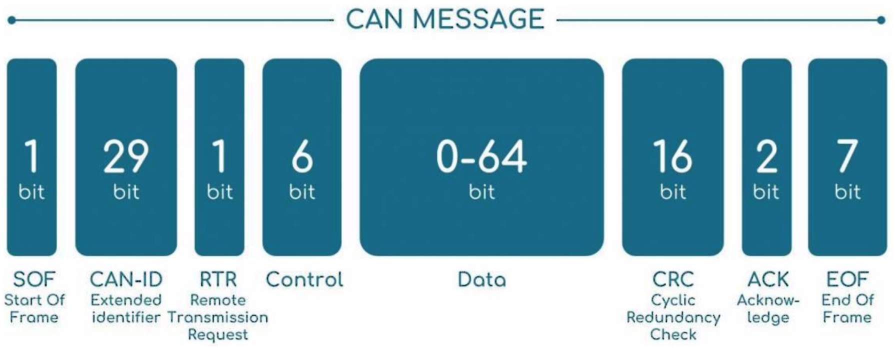
Priorities Instead of Collisions: pros

- The bus logic and arbitration logic not only prevent collisions.
- It ensure a priority-controlled bus access.
- Smaller ECU's ID, higher priority.
- random, nondestructive, and priority-controlled bus access ensures fair and very fast bus access.

Problem

How Sender & Receiver agree on message integrity?

CAN Message error detection



Focus on the Data (up to 64 bits) and on the CRC bits group (16 bits)

Problem

unfortunately....
we need maths in binary

Before to start, do you remember polynomial division?

N.B. Polynomial division **IS NOT** numerical division

We need to divide the polynomial $M(x)$ (the data message) by the polynomial $G(x)$ (the generator)

condition: degree of $M(x)$ greater or equal of the degree of $G(x)$

$$deg\big(M(x)\big) \geq deg\big(G(x)\big)$$

Before to start, do you remember polynomial division?

Polynomial Remainder Theorem

given two polynomials $M(x)$ (the dividend) and $G(x)$ (the divisor), asserts the existence (and the uniqueness) of a quotient $Q(x)$ and a remainder $R(x)$ such that:

$M(x) = Q(x)G(x) + R(x)$

N.B. The degree of $R(x)$ is strictly lower than the degree of $G(x)$

$$deg\big(R(x)\big) < deg\big(G(x)\big)$$

Before to start, do you remember polynomial division?

Example: find the quotient and the remainder of the division of $x^3 - 2x^2 - 4$, the dividend, by $x - 3$, the divisor.

$$
\begin{array}{r|rrrr}
\multicolumn{1}{r}{} & & x^2 & +x & +3 \\
\cline{2-5}
x-3) & x^3 & -2x^2 & & -4 \\
\multicolumn{1}{r}{} & -x^3 & +3x^2 & & \\
\cline{2-3}
\multicolumn{1}{r}{} & & x^2 & & \\
\multicolumn{1}{r}{} & & -x^2 & +3x & \\
\cline{3-4}
\multicolumn{1}{r}{} & & & 3x & -4 \\
\multicolumn{1}{r}{} & & & -3x & +9 \\
\cline{4-5}
\multicolumn{1}{r}{} & & & & 5
\end{array}
$$

$$x^3 - 2x^2 - 4 = (x-3)\underbrace{(x^2 + x + 3)}_{q(x)} + \underbrace{5}_{r(x)}$$

Polynomial Arithmetic and Cyclic Redundancy Checks

The calculation of CRC depends on the arithmetic of modulo 2 polynomial.

A modulo 2 polynomial is a polynomial
$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$

where the coefficients $a_n, \cdots, a_1, a_0$ are integers modulo 2 (i.e. $\mathbf{0}$ or $\mathbf{1}$)

Polynomial Arithmetic and Cyclic Redundancy Checks

The tricky part of polynomial arithmetic modulo 2 is that, obviously, the coefficients obey to modulo 2 arithmetic themselves

Addition

+	0	1
0	0	1
1	1	0

and this is an XOR "$\oplus$" in electronics!

Polynomial Arithmetic and Cyclic Redundancy Checks

Exercise

$(1) + (1) = 0$
$(x) + (x) = 0$
$(x) + (1) = x + 1$
$(x + 1) + (1) = x$
$(x^4 + x^2 + 1) + (x^4 + x + 1) = x^2 + x$

Polynomial modulo 2 can be easily written with bits:

$x^3 + x + 1 = 1011$
$x^4 + x = 10010$

Polynomial Arithmetic and Cyclic Redundancy Checks

Division is done as usual, but the subtraction stage uses the XOR operation. Super fast and easy to be implemented in hardware/software.

Exercise

Divide $x^7 + 1$ by $x^3 + x^2 + 1$
the result is $x^4 + x^3 + x^2 + 1$ with no reminder

```
        11101
1101)10000001
     1101
      1010
      1101
       1110
       1101
        01101
         1101
            0
```

Polynomial Arithmetic and Cyclic Redundancy Checks

Exercise

Divide $x^7 + x^6 + x^3 + x^2$ by $x^3 + 1$
the result is $x^4 + x^3 + x$ with reminder of $x^2 + x$

$$
\begin{array}{r@{\,}l}
 & 11010 \\
1001) & \overline{11001100} \\
 & \underline{1001} \\
 & 1011 \\
 & \underline{1001} \\
 & 01010 \\
 & \underline{1001} \\
 & 110
\end{array}
$$

Polynomial Arithmetic and Cyclic Redundancy Checks

DO NOT TRY TO CONFUSE POLYNOMIAL WITH NUMBERS

polynomial math

$$\begin{array}{r} 101 \\ 111\overline{)11001} \\ \underline{111} \\ 0101 \\ \underline{111} \\ 010 \end{array}$$

meaning: $\frac{x^4+x^3+1}{x^2+x+1} = x^2 + 1$

with reminder x

numerical math $(\oplus \rightarrow -)$

$$\begin{array}{r} 011 \\ 111\overline{)11001} \\ \underline{111} \\ 1011 \\ \underline{111} \\ 100 \end{array}$$

meaning: $\frac{11001_2}{111_2} = \frac{25_{10}}{7_{10}} = 011_2 = 3_{10}$

with reminder $100_2 = 4_{10}$

CRC Encoding

- We have a **message** $M(x)$ of n bits: $deg\big(M(x)\big) = n-1$
- We have a **generator** $G(x)$ of $m+1$ bits: $deg\big(G(x)\big) = m$
  - *consequent:* The **reminder** $R(x)$ of the division $\frac{M(x)}{G(x)}$ will have a strictly lower degree with respect to $G(x)$ and, in the worst case, the maximum value will be $deg\big(R(x)\big) = m-1$
  - in short: $R(x)$ can be always expressed with m bits
- Add m **zeroes** at the end of $M(x)$: this means to do the following $M(x)x^m$
- Divide the **new message** $M(x)x^m$ by $G(x)$ to obtain the *reminder* of m bits called **CRC**.
- Form $B(x)$ the novel and **final message** $M(x)x^m + CRC$: this means to add the CRC bits at the end of the message replacing the m zeroes padded before.

CRC Encoding: Example step-by-step

- $M(x) = 1101011011$
- $G(x) = 10011$ (m = 4)
- Append m zeroes to form $M(x)x^m$: 11010110110000
- Divide the new message 11010110110000 by the generator 10011

```
          1100001010
10011)11010110110000
      10011
       10011
       10011
        000010110
            10011
             010100
              10011
                1110
```

- Final message $B(x)$: $\underbrace{1101011011}_{M(x)}\underbrace{1110}_{CRC}$

CRC Decoding: Example step-by-step

What the receiver does? (we assume no errors now)

- The receiver **receives** $B(x) = M(x)x^m + CRC = 11010110111110$
- The receiver **knows** $G(x) = 10011$ (m = 4)
- The receiver **divides** the whole message $B(x)$ 11010110111110 by the generator 10011

```
            1100001010
      ______________
10011)11010110111110
      10011
      -----
       10011
       10011
       -----
        000010111
            10011
            -----
             010011
              10011
              -----
                0000
```

- If the receiver obtains no reminder -> transmission succeeded (no errors detected)

CRC Error Resistance

- What a kind of errors can we detect?
- Imagine that an **error** $E(x)$ occurs on the transmission channel and the receiver receives $B(x) + E(x)$ instead of simply $B(x)$
- $B(x)$ alone would be recognized as *correct* because divided by $G(x)$ gives no reminder. If $E(x)$ is multiple of $G(x)$ than $B(x) + E(x)$ divided by $G(x)$ gives no reminder -> the receiver *mark* $B(x) + E(x)$ as a correct message. But it is not!

- The key to do a nice job is to have a good $G(x)$: (*this* is the reason why you can't choose $G(x)$, it is standard in the CRC-encoding used by the standard protocol)

CRC Error Resistance: Examples

- $B(x) = 101101$
- $G(x) = 101$
- $\frac{B(x)}{G(x)}$ gives zero reminder
- $E(x)$ is 010000 and $B(x) + E(x)$ will be 111101
- $\frac{B(x)+E(x)}{G(x)}$ has a non zero reminder

$\frac{B(x)}{G(x)}$ gives zero reminder

$$\begin{array}{r} 1001 \\ 101\overline{)101101} \\ \underline{101} \\ 00101 \\ \underline{101} \\ 00 \end{array}$$

$\frac{B(x)+E(x)}{G(x)}$ has a non zero reminder

$$\begin{array}{r} 1100 \\ 101\overline{)111101} \\ \underline{101} \\ 101 \\ \underline{101} \\ 01 \end{array}$$

ERROR DETECTED

CRC Error Resistance: Examples

- $B(x) = 101100$
- $G(x) = 100$
- $\frac{B(x)}{G(x)}$ gives zero reminder
- $E(x)$ is 010000 and $B(x) + E(x)$ will be 111100
- $\frac{B(x)+E(x)}{G(x)}$ still has zero reminder

$\frac{B(x)}{G(x)}$ gives zero reminder

$$\begin{array}{r}
1011 \\
100\overline{)101100} \\
\underline{100} \\
0110 \\
\underline{100} \\
100 \\
\underline{100} \\
00
\end{array}$$

$\frac{B(x)+E(x)}{G(x)}$ has a non zero reminder

$$\begin{array}{r}
1111 \\
100\overline{)111100} \\
\underline{100} \\
111 \\
\underline{100} \\
110 \\
\underline{100} \\
100 \\
\underline{100} \\
00
\end{array}$$

ERROR <u>NOT</u> DETECTED

CRC Design Principles

- $G(x)$ is extremely important
- Select a $G(x)$ so that $E(x)$ cannot easily be multiple of $G(x)$
- **Detect single bit errors:**
  - $E(x) = x^i$ for error at i-th bit (notation with LSB in position 0)
  - if $G(x)$ has more than 1 term (more than 1 bit) it **cannot** divide x^i
- Mathematical theory help us to design powerful $G(x)$ with fancy characteristics
- $G(x)$ used in CAN Bus is $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$

Why to padd

- $M(x) = 11011$
- $G(x) = 1101$
- $M(x) padded = 11011000$

$\frac{M(x)}{G(x)}$ gives reminder of 001

$$\begin{array}{r} 10 \\ 1101\overline{)11011} \\ \underline{1101} \\ 001 \end{array}$$

If we use 001 as CRC we cannot use the receiver logic seen before

$M(x)$ followed by CRC: 11011001 which divided by $G(x)$ has reminder $\neq 0$

$\frac{M(x) stuffed}{G(x)}$ gives reminder of 101

$$\begin{array}{r} 10001 \\ 1101\overline{)11011000} \\ \underline{1101} \\ 0001000 \\ \underline{1101} \\ 101 \end{array}$$

The CRC computed on $M(x) pudded$ is different from the CRC computed directly on $M(x)$

SENDER & RECEIVER MUST TO AGREE ON THIS