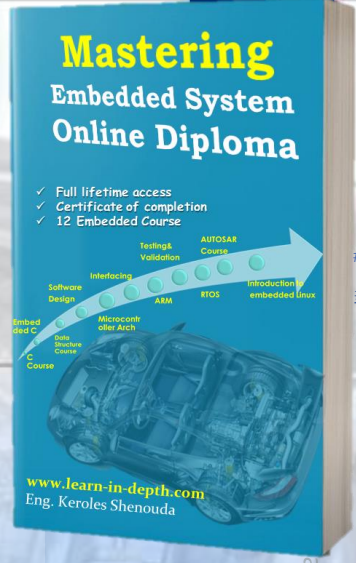
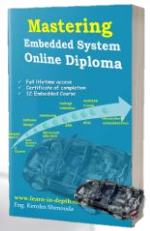
Unit(Mastering CAN Network)

Session 1 : CAN Protocol (Theoretical Part)

- In-vehicle networks (IVN)
- Evolution of in-vehicle networks
  - classic flat wiring harness architecture
  - domain architecture
  - zonal architecture
- A typical wiring harness for a car.
- central compute Based Zonal Architecture
- CANBUS Introduction
  - What is CANBUS?
  - Who uses CANBUS?
  - CANBUS history
  - CANBUS timeline
  - CANBUS Characteristics
  - OSI Model
  - Physical Layer
- CAN Protocol Specification
- General Characteristics
- Types of CAN Messages
  - Data frame
  - Remote frame
  - Error frame
  - Overload frame
- CAN bus Characteristics
  - CAN Node
  - Single Ended Vs Differential
  - Recessive And Dominant Signals
  - Bit Rate / Bus Length
  - CAN transceiver
- CAN Bus Errors
  - Bit error
  - Stuff error
  - CRC error
  - Acknowledgment error
  - Form Error
- Error Confinement Mechanism
- CAN Bit Monitoring and Stuffing

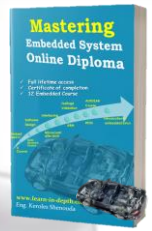




#LEARN_IN DEPTH

#Be_professional_in_embedded_system

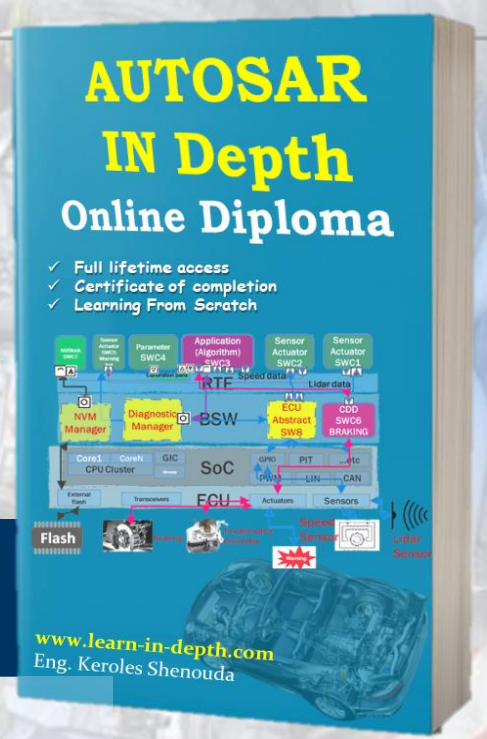
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

In-vehicle networks (IVN)

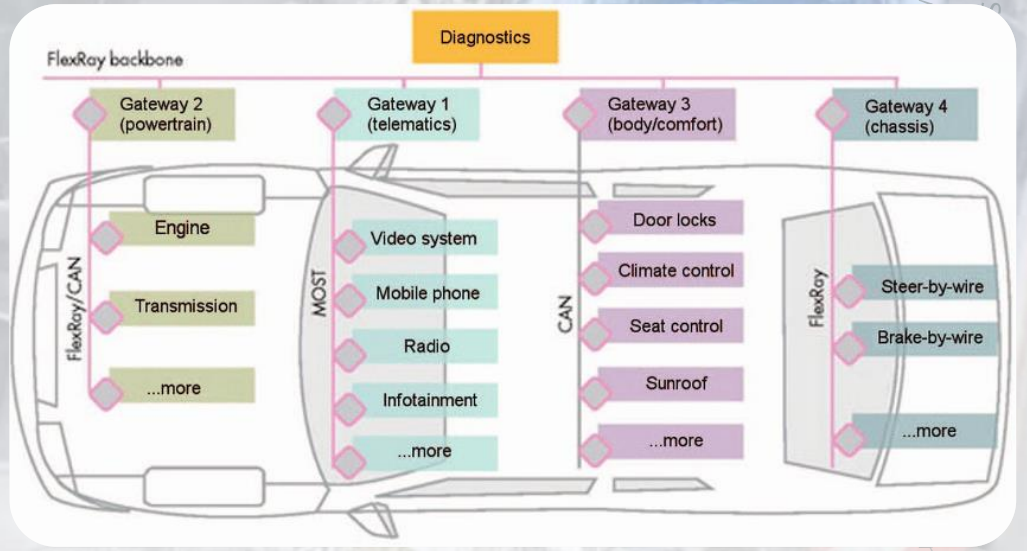


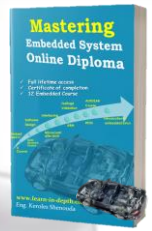
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

In-Vehicle Networks (IVN)

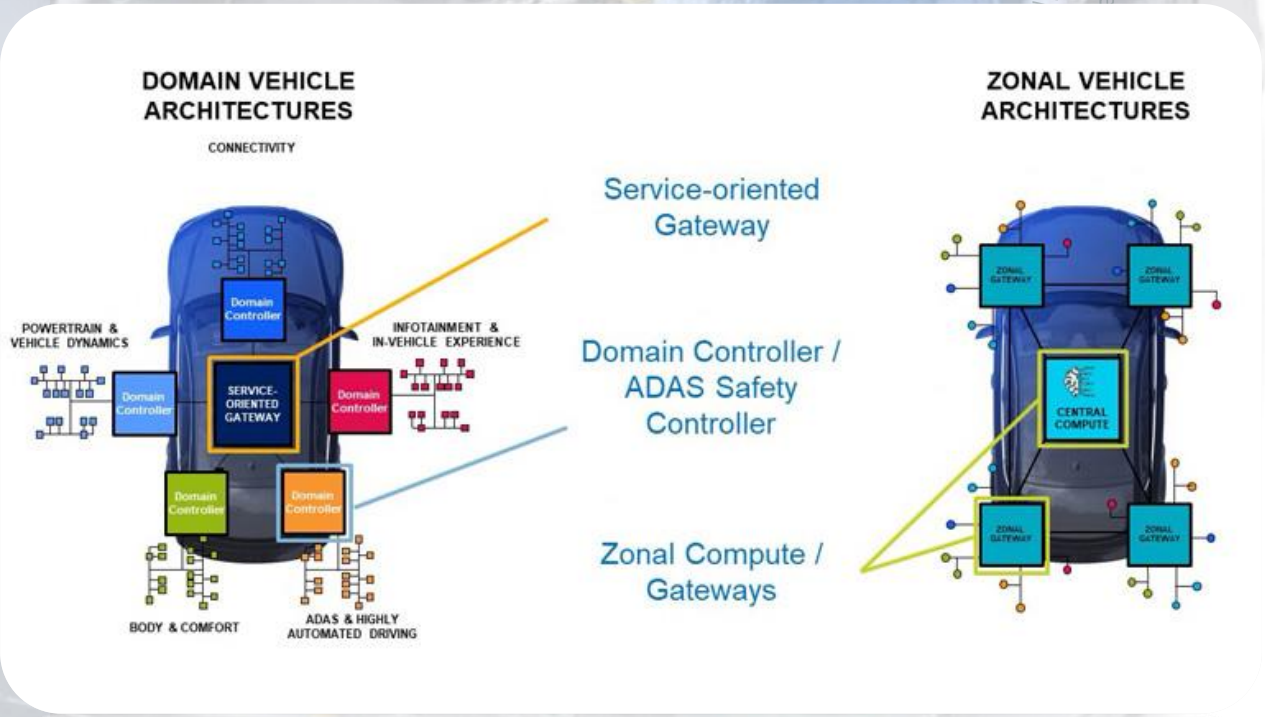
- A modern vehicle, even if not fully featured, already has 70 to 100 ECUs, with over 2500 signals to transmit internally.
- To coordinate communication among ECUs, In-Vehicle Networks (IVNs) are composed of several kinds of bus protocols.
- To manage increased complexity and higher data rates as new versions of existing protocols find their way into vehicle networks (CAN FD, Ethernet).





Evolution of in-vehicle networks

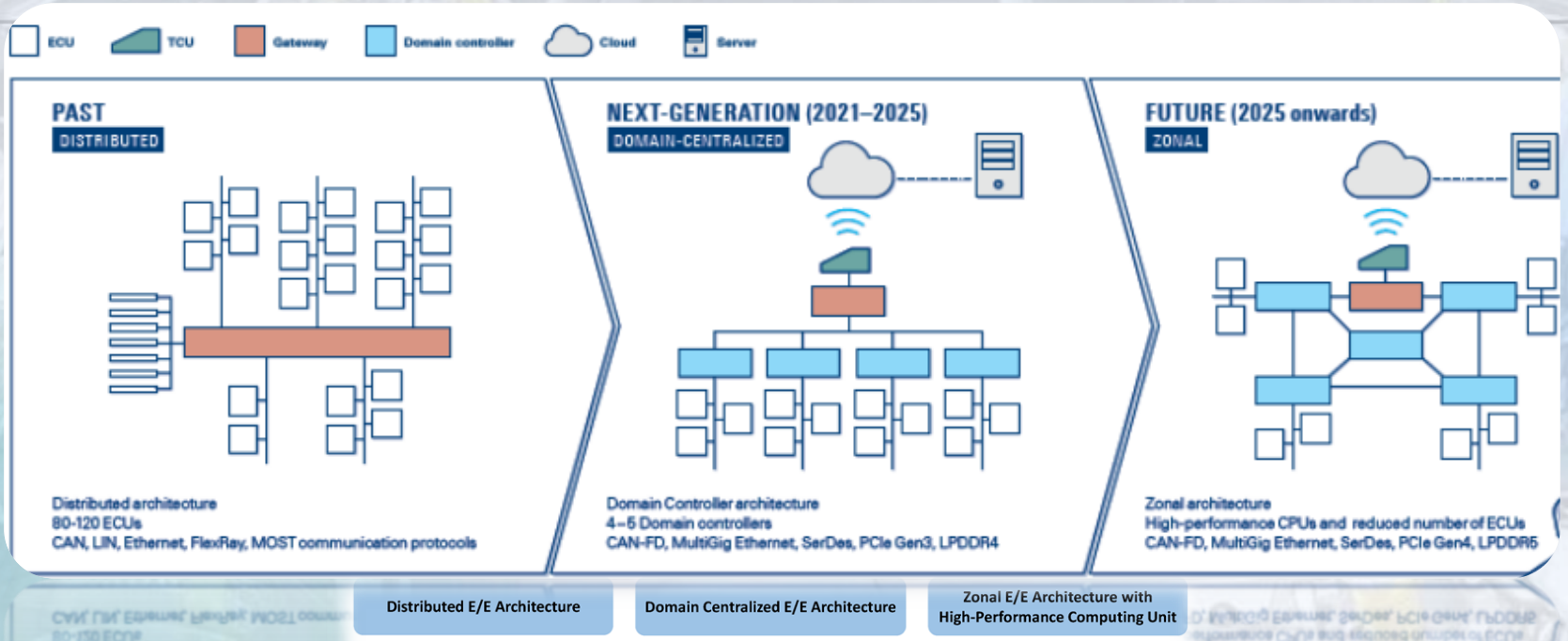
- **New communication protocols**, **higher bandwidth demands**, **new applications**, **more complex communication matrices**: all of this **impacts the network architecture requirements.**
- Historically, in-vehicle networks were organized into logical domains, such as **"Body", "Chassis", and "Powertrain".**
- These domains were interconnected through **a central gateway**.
- **In the future**, the concept **of specialized ECUs for domain-specific functions will continue**, but **the general trend** is moving toward **separation according to physical location (Zones) rather than by logical function**.
- **Zone ECUs** connect via **high-speed networks** to **a central ECU** where much of the processing is done.

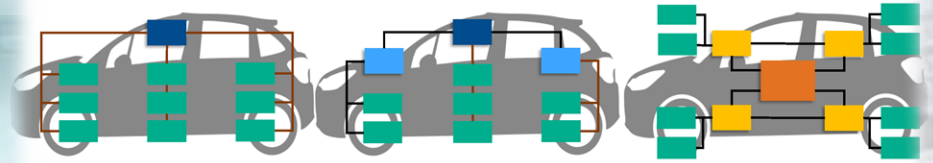


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Evolution of in-vehicle networks

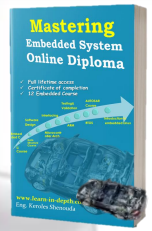
- the **classic flat wiring harness architecture** is changing to **a domain and zonal architecture** with **Automotive Ethernet as the backbone**.



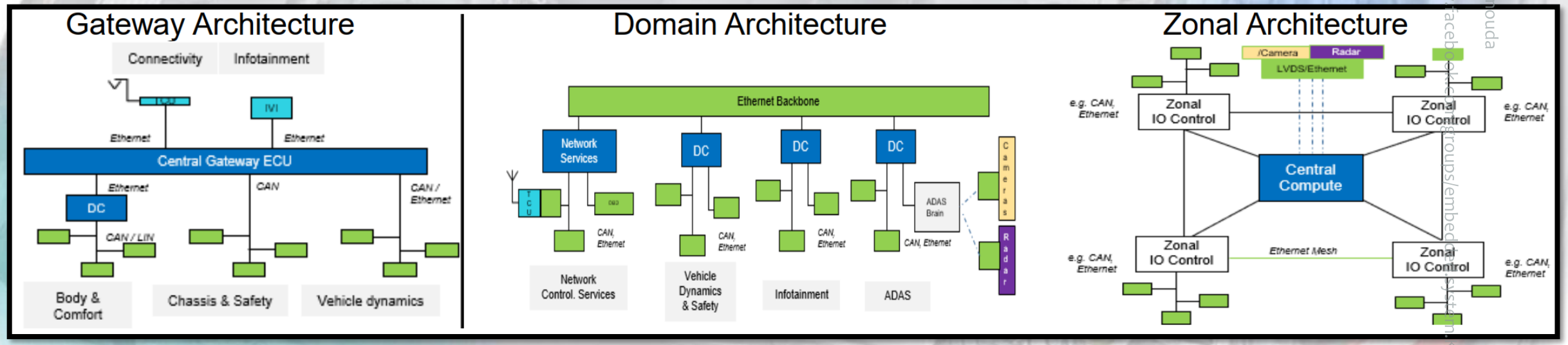


Optional ECUs e.g., central gateway
Automotive ECUs (function-specific)
Domain-specific ECUs
Central high-performance computing unit
Zonal ECUs
CAN BUS connection
Automotive Ethernet connection

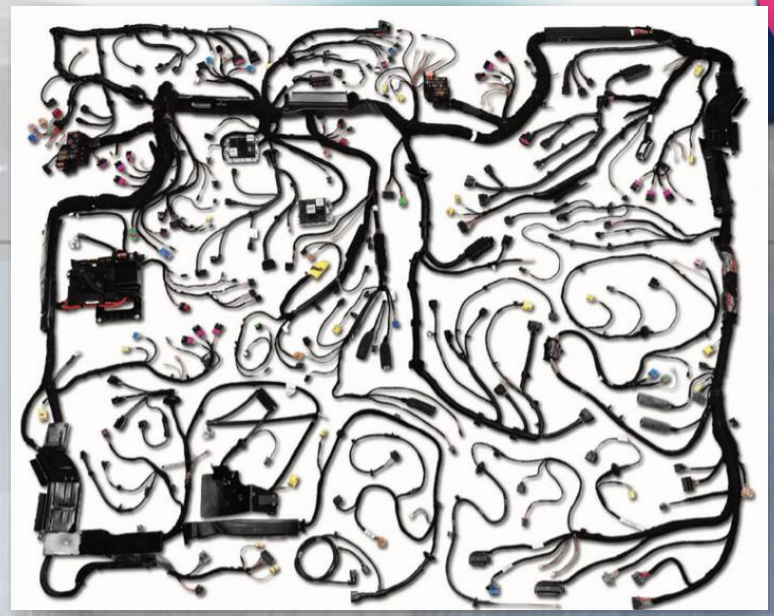
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Evolution of in-vehicle networks Cont.

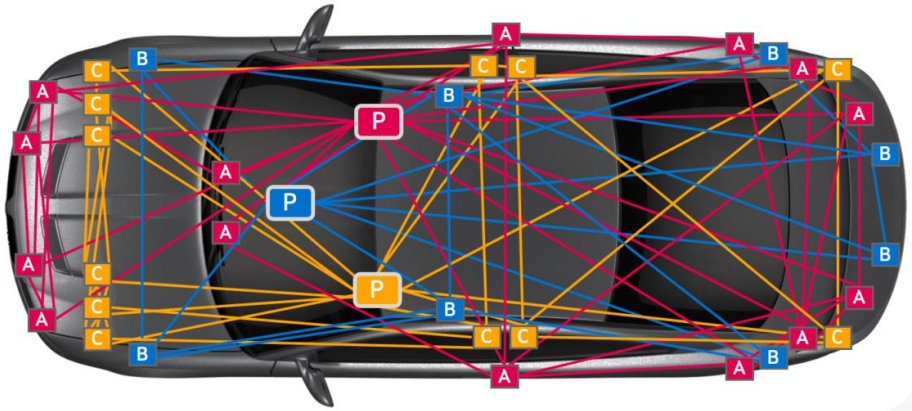


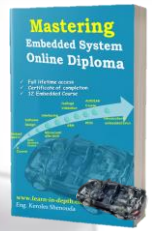
A typical wiring harness for a car.



- In many vehicles, the wiring that connects the electronic and electric components is so complex and extensive that, were it laid end to end, all those cables would **stretch over a mile**. This cable Obviously, complexity and weight both add significant **costs**, as well as **energy consumption while driving**.
- That's because many of today's new vehicles are still built around the concept of **Domain Architecture**
- where extensive cable harnesses connect independent domains such as **Body**, **Telematics**, **ADAS**, **Infotainment** and **Powertrain** *directly to components that are spread all over the car*

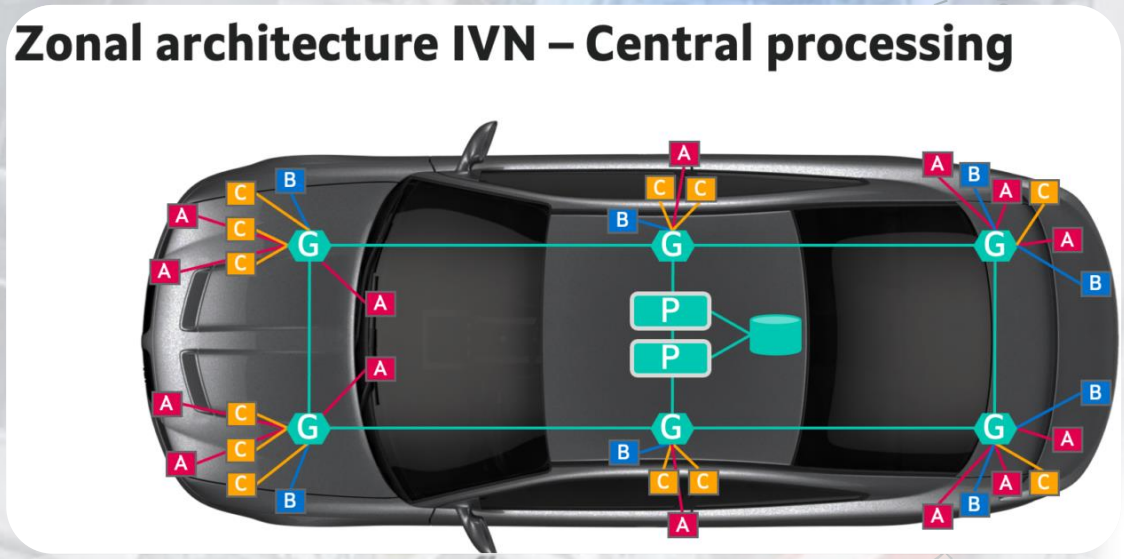
Domain architecture IVN





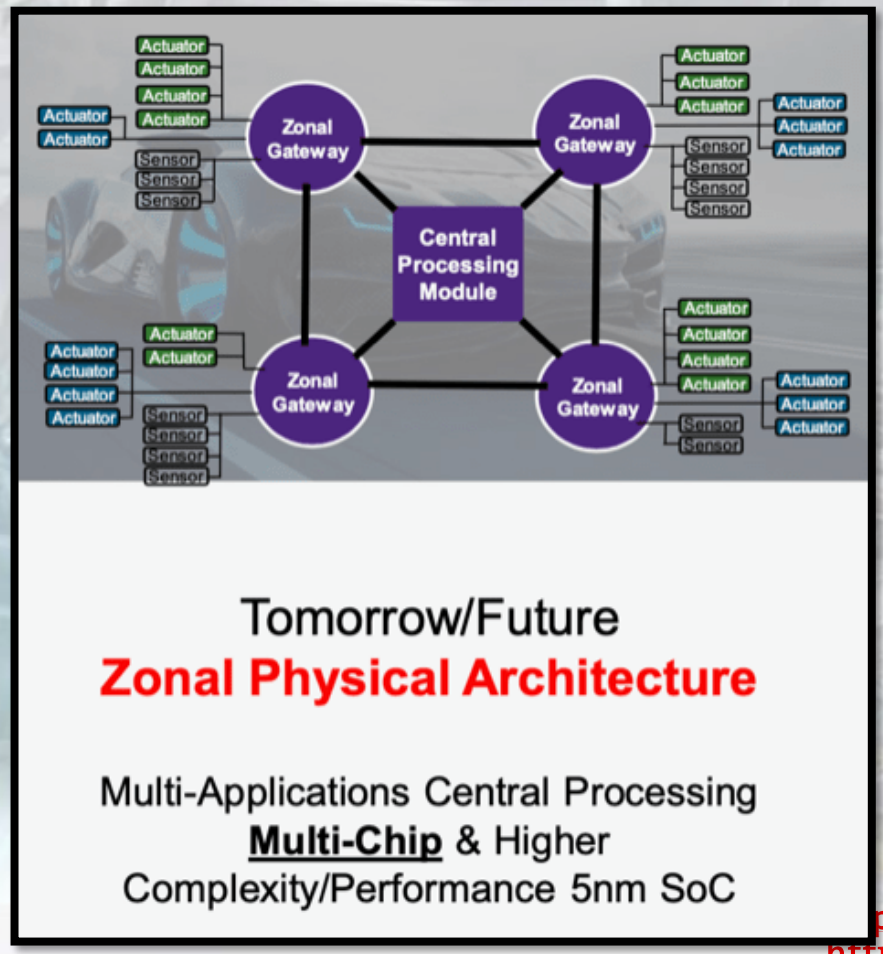
A typical wiring harness for a car Cont.

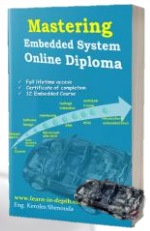
- Shedding the constraints of **traditional wired harnesses**, the most advanced vehicle designers are **shifting toward Zonal Architecture**, which leverages **Ethernet as the backbone protocol** between the car's different zones.
- _Zonal Architecture dramatically streamlines ***automotive wiring requirements, reducing cable, labor and energy costs.***



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

central compute Based Zonal Architecture

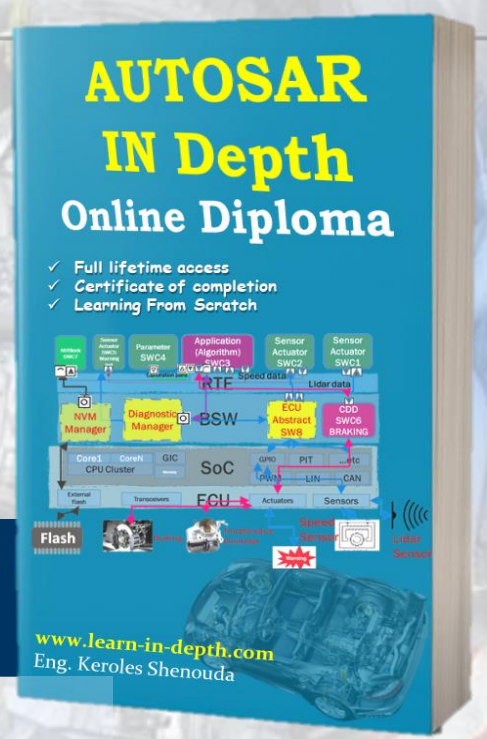




#LEARN_IN DEPTH

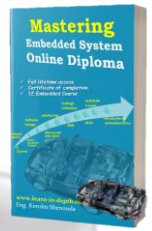
#Be_professional_in_embedded_system

Why CAN bus ?



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

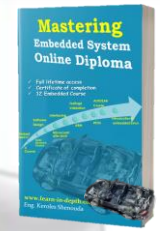


CANBUS or CAN bus – Controller Area Network bus

An **automotive serial** bus system developed to satisfy the following requirements:

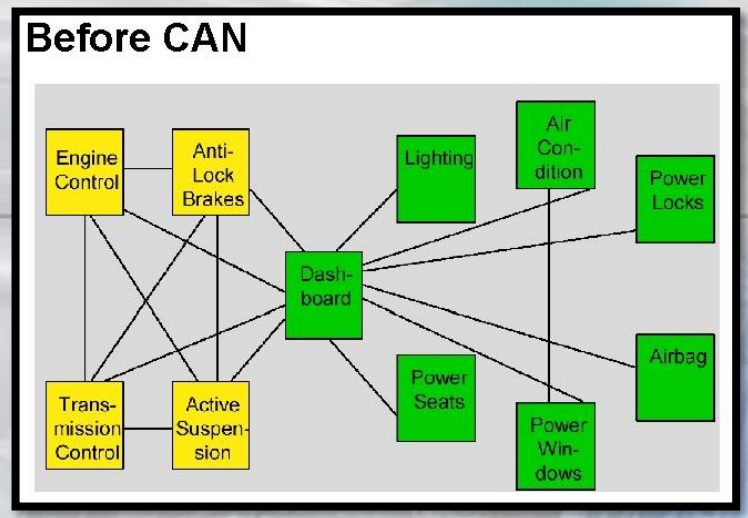
- Network multiple microcontrollers with 1 pair of wires.
- Allow microcontrollers communicate with each other.
- High speed, real-time communication. (BaudRate up to 1 MbPS)
- Provide noise immunity in an electrically noisy environment.
- Low cost
- prioritization of messages

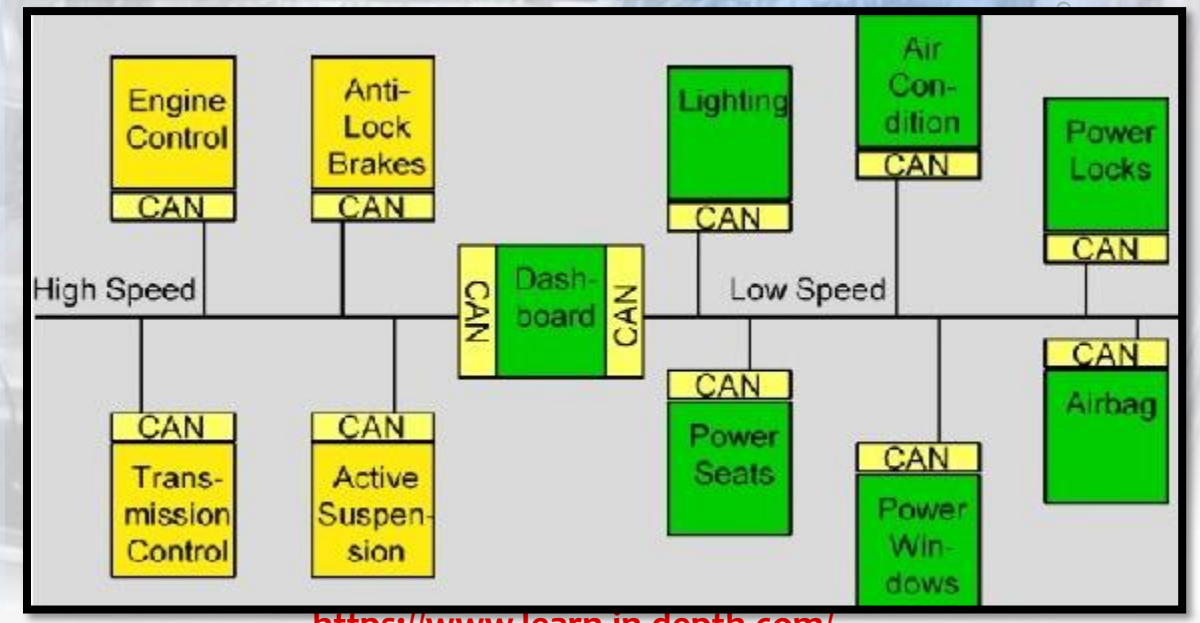
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

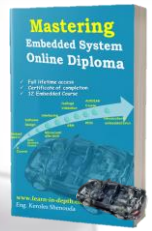


Why CAN bus ?

- Low cost
- Network multiple microcontrollers with 1 pair of wires.

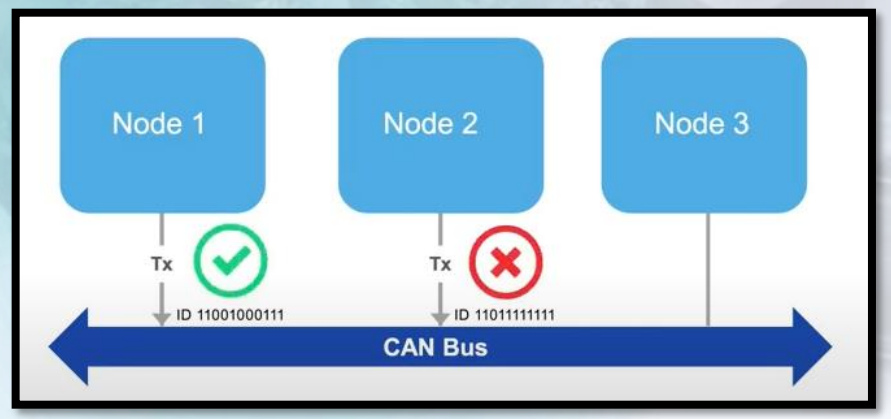


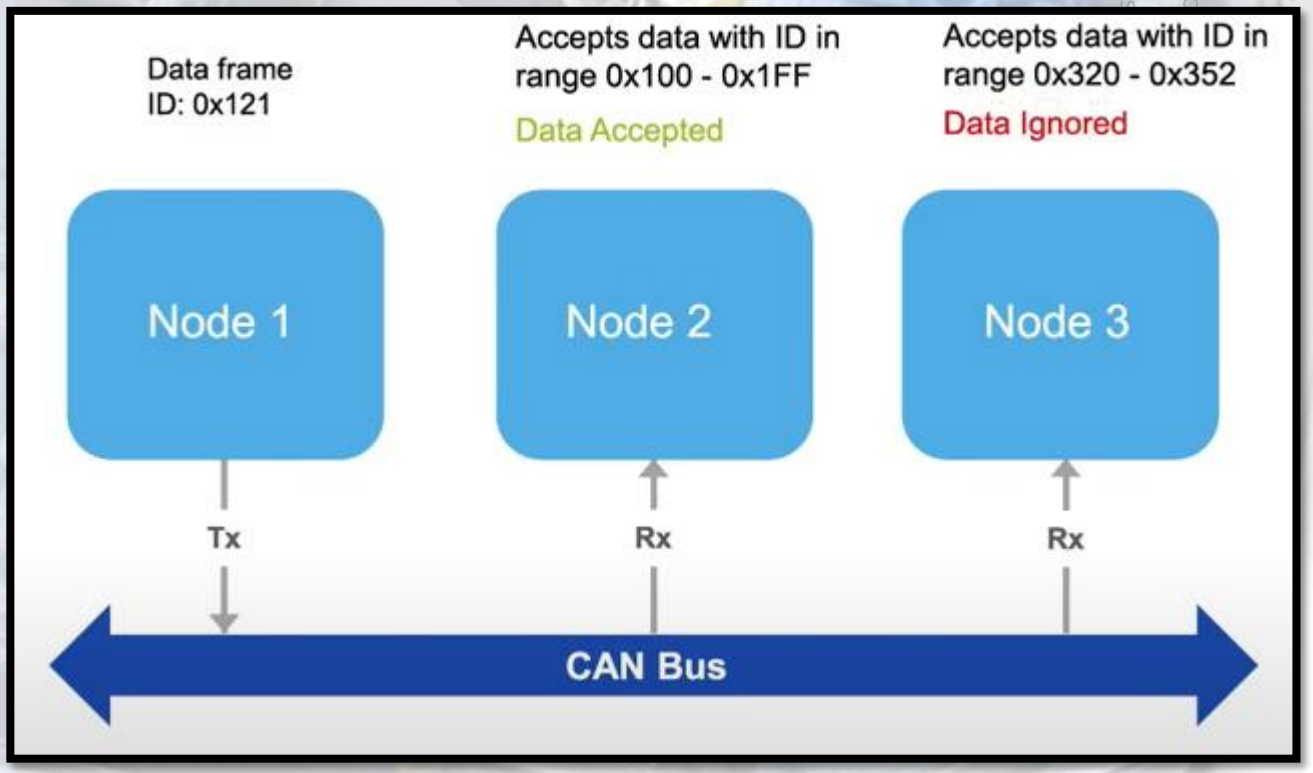




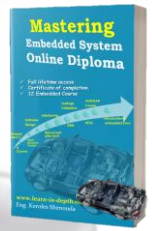
Why CAN bus Cont.

- Low cost
- Network multiple microcontrollers with 1 pair of wires.
- **Efficient**



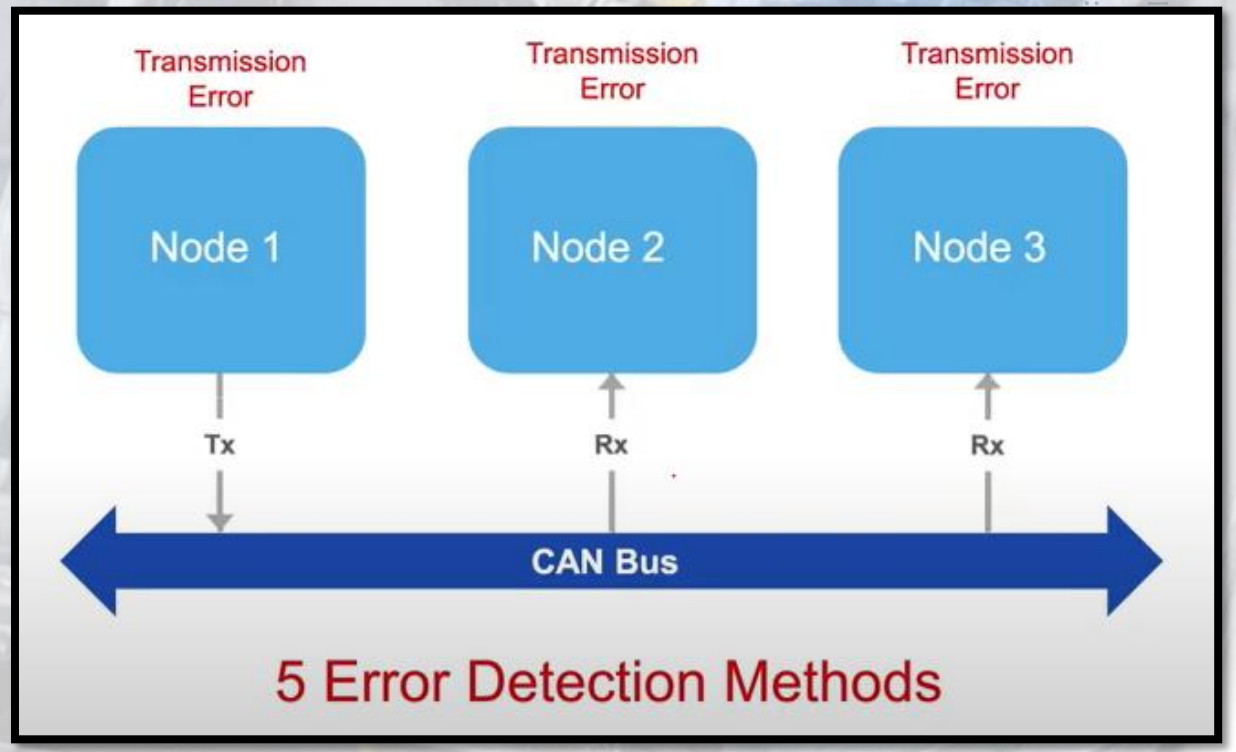


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

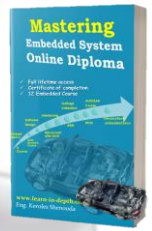


Why CAN bus Cont.

- Low cost
- Network multiple microcontrollers with 1 pair of wires.
- Efficient
- **Reliable**
  - There are five mechanisms for detecting errors in the CAN protocol:
    1. Bit monitoring
    2. Bit stuffing
    3. Frame check
    4. Acknowledgment check
    5. Cyclic redundancy check

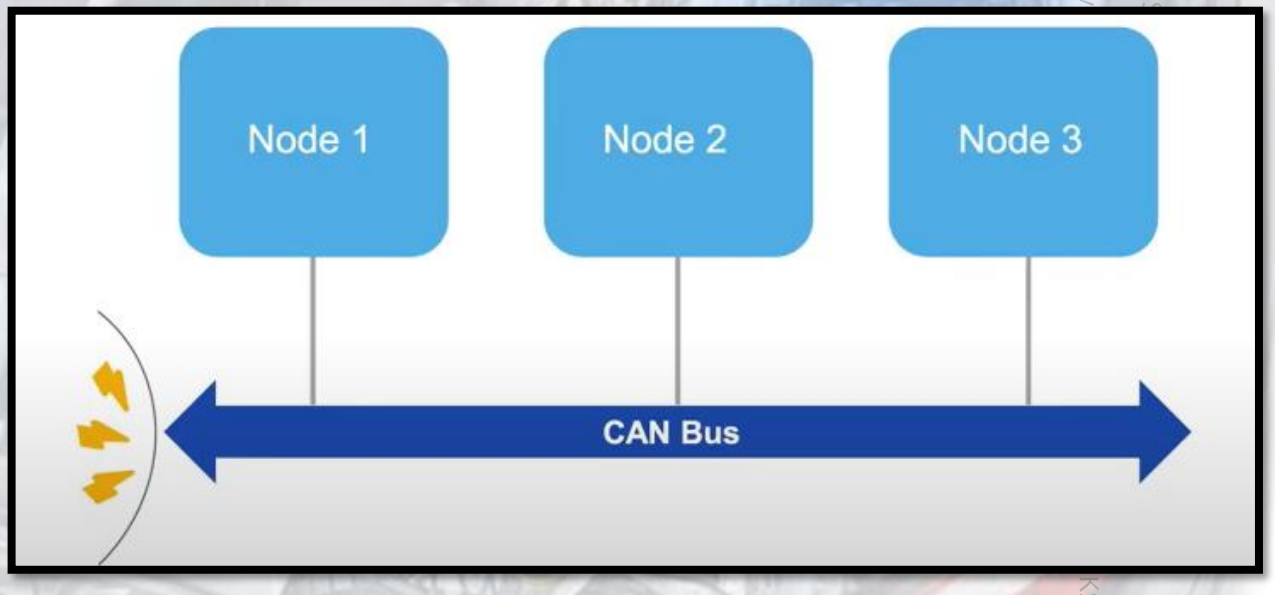


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

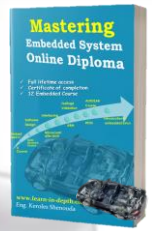


Why CAN bus Cont.

- Low cost
- Network multiple microcontrollers with 1 pair of wires.
- Efficient
- Reliable
- **ROBUST**
  - High speed data lines are resistant to electrical disturbances

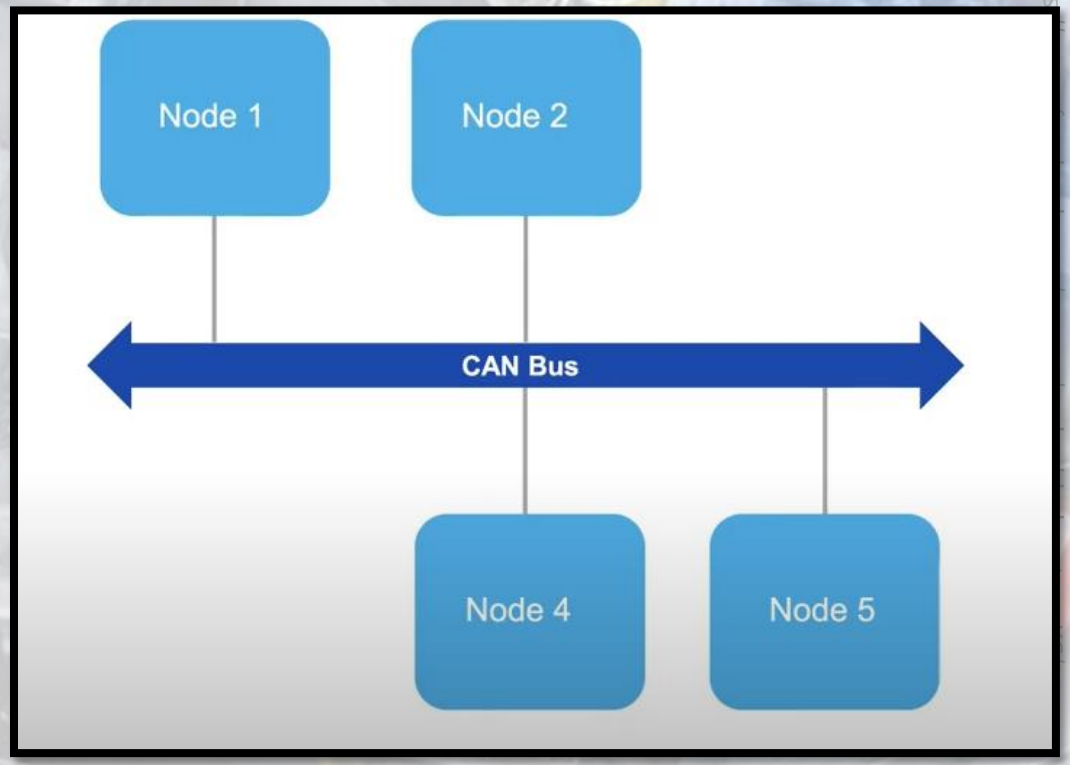


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

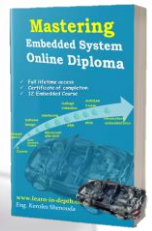


Why CAN bus Cont.

- Low cost
- Network multiple microcontrollers with 1 pair of wires.
- Efficient
- Reliable
- ROBUST
  - High speed data lines are resistant to electrical disturbances
- **Flexible**
  - Message-based protocol, This allows nodes to be added or removed from the system without hardware or software modifications to be done



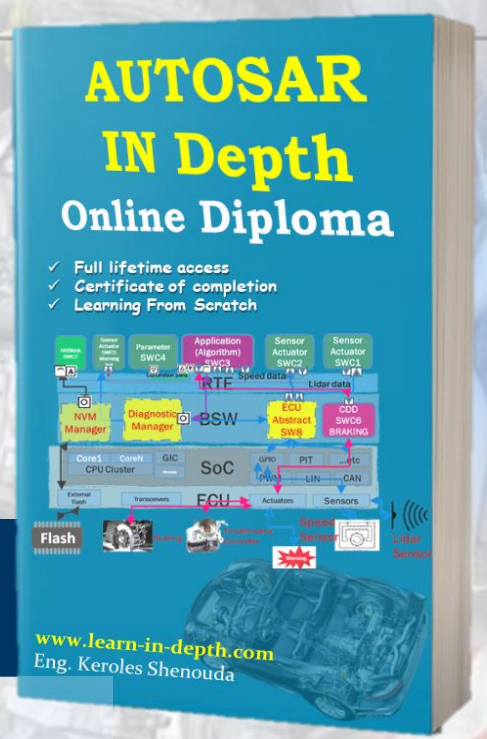
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

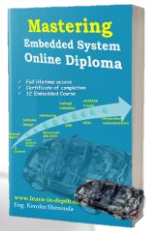
#Be_professional_in_embedded_system

Who uses CANBUS?



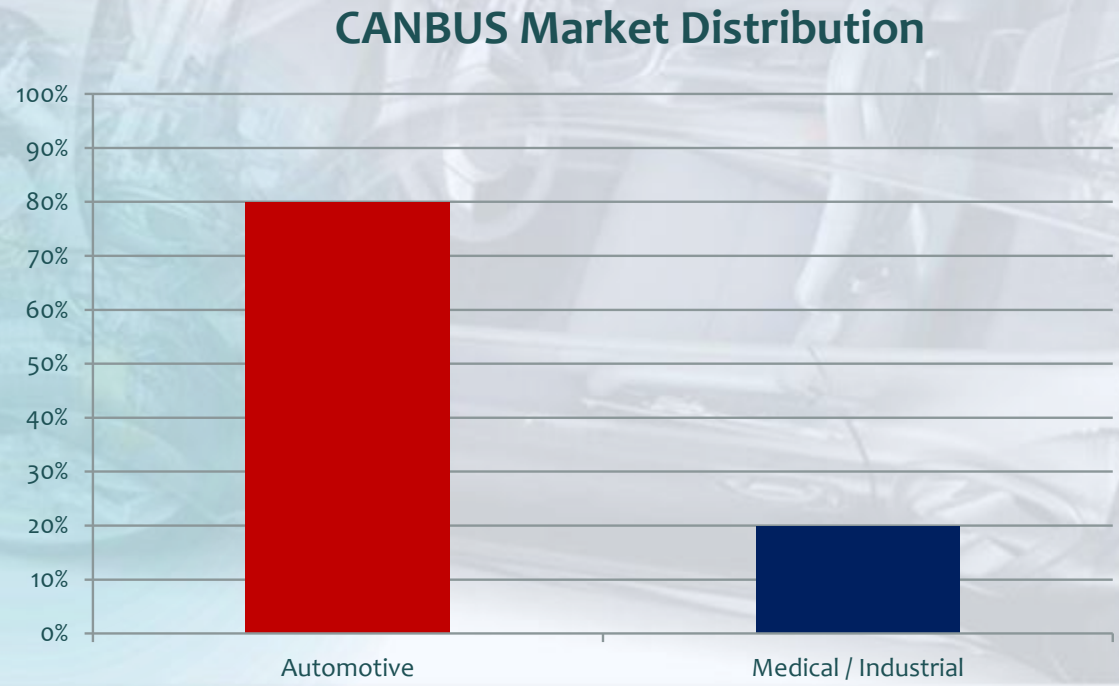
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

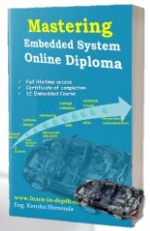


Who uses CANBUS?

- Designed specifically for automotive applications
- Today - industrial automation / medical equipment



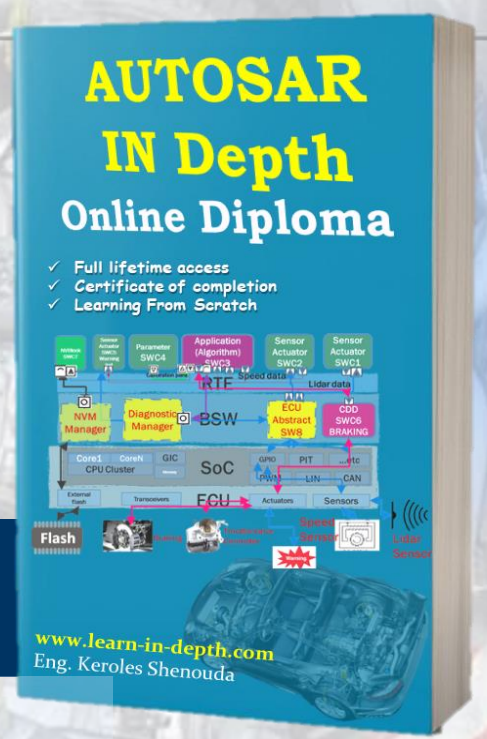
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

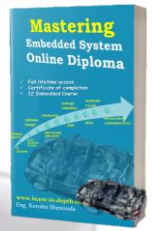
#Be_professional_in_embedded_system

ISO 11898 (CAN Specification)



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



https://www.iso.org/standard/63648.html

Standards About us News Taking part Store EN

ICS › 43 › 43.040 › 43.040.15

ISO 11898-1:2015

Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling

ABSTRACT PREVIEW

ISO 11898-1:2015 specifies the characteristics of setting up an interchange of digital information between modules implementing the CAN data link layer. Controller area network is a serial communication protocol, which supports distributed real-time control and multiplexing for use within road vehicles and other control applications.

ISO 11898-1:2015 specifies the Classical CAN frame format and the newly introduced CAN Flexible Data Rate Frame format. The Classical CAN frame format allows bit rates up to 1 Mbit/s and payloads up to 8 byte per frame. The Flexible Data Rate frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 byte per frame.

ISO 11898-1:2015 describes the general architecture of CAN in terms of hierarchical layers according to the ISO reference model for open systems interconnection (OSI) according to ISO/IEC 7498-1. The CAN data link layer is specified according to ISO/IEC 8802-2 and ISO/IEC 8802-3.

ISO 11898-1:2015 contains detailed specifications of the following: logical link control sub-layer; medium access control sub-layer; and physical coding sub-layer.

There are three implementation options. They are the following: support of the Classical CAN frame format only, not tolerating the Flexible Data Rate frame format; support of the Classical CAN frame format and tolerating the Flexible Data Rate frame format; and support of the Classical CAN frame format and the Flexible Data Rate frame format.

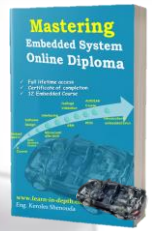
The last option is recommended to be implemented for new designs.

BUY THIS STANDARD

FORMAT	LANGUAGE
✔ PDF	English
PAPER	English

CHF 178 BUY

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

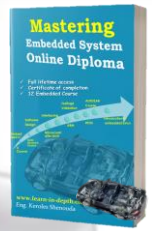


CANBUS Timeline

NOMENCLATURE	STANDARD	MAX. SIGNALING RATE	IDENTIFIER
Low–Speed CAN	ISO 11519	125 kbps	11-bit
CAN 2.0A	ISO 11898:1993	1 Mbps	11-bit
CAN 2.0B	ISO 11898:1995	1 Mbps	29-bit

- 1983 : First CANBUS project at Bosch
- 1986 : CAN protocol introduced
- 1987 : First CAN controller chips sold
- 1991 : CAN 2.0A specification published
- 1992 : Mercedes-Benz used CAN network
- 1993 : ISO 11898 standard
- 1995 : ISO 11898 amendment
- Present : The majority of vehicles use CAN bus.

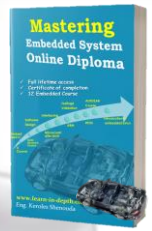
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Standard CAN or Extended CAN

- **The first version** of the CAN standards listed in Table 1, **ISO 11519 (Low-Speed CAN)** is for applications up to **125** kbps with a *standard* **11-bit** identifier.
- **The second version**, **ISO 11898 (1993),** also with **11-bit identifiers** provides for signaling rates from **125 kbps to 1 Mbps** (Standard CAN Version 2.0A,)
- The more recent **ISO 11898** (1995) introduces the *extended* 29-bit identifier (CAN Version 2.0B)
- **Note:** The Standard CAN 11-bit identifier field provides for 2^11, or 2048 different message identifiers
- while the Extended CAN 29-bit identifier provides for 229, or 537 million identifiers.

NOMENCLATURE	STANDARD	MAX. SIGNALING RATE	IDENTIFIER
Low–Speed CAN	ISO 11519	125 kbps	11-bit
CAN 2.0A	ISO 11898:1993	1 Mbps	11-bit
CAN 2.0B	ISO 11898:1995	1 Mbps	29-bit



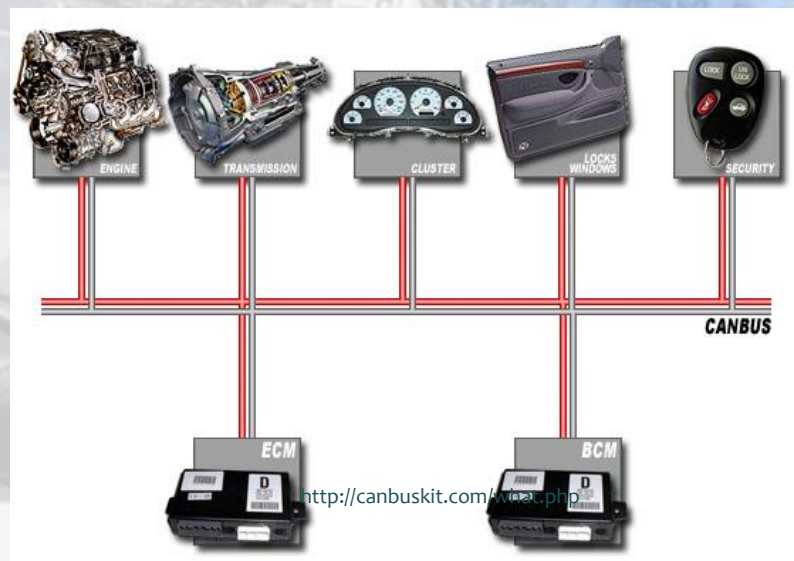
CANBUS Physical Layer

- Physical medium – two wires terminated at both ends by resistors.
- Differential signal - better noise immunity.
- Benefits:
  - Reduced weight, Reduced cost
  - Fewer wires = Increased reliability

Conventional multi-wire looms

vs.

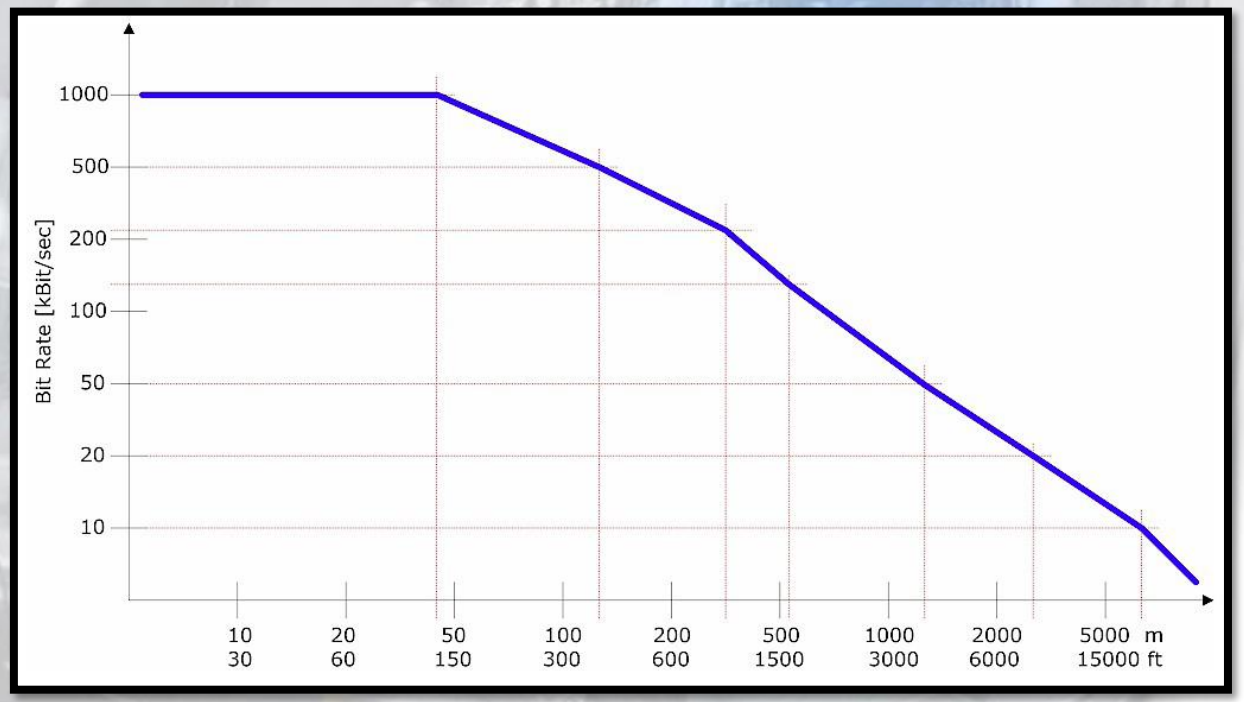
CAN bus network



http://canbuskit.com/what.php

Transmission Characteristics

- Up to 1 Mbit/sec.
- Common baud rates: 1 MHz, 500 KHz and 125 KHz
- All nodes – same baud rate
- Max length:120’ to 15000’ (rate dependent)

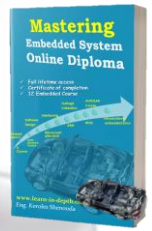


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Example of Message Transaction

	ID	Data
▪ Instrument panel ECU says "can anyone tell me what the block temperature is?	400	
▪ Block ECU sees this message and issues a message "block temperature is 76 Celsius"	400	076
▪ Instrument panel ECU sees block temperature message and displays it on console		

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

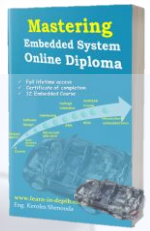


CAN "Controller Area Network"

- Overview
- General Characteristics
- Types of CAN Messages
- CAN bus Characteristics
- CAN Bus Errors

CAN
Controller Area Network

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



CAN Overview

- Standard CAN

NOMENCLATURE	STANDARD	MAX. SIGNALING RATE	IDENTIFIER
Low–Speed CAN	ISO 11519	125 kbps	11-bit
CAN 2.0A	ISO 11898:1993	1 Mbps	11-bit
CAN 2.0B	ISO 11898:1995	1 Mbps	29-bit

CAN
Controller Area Network

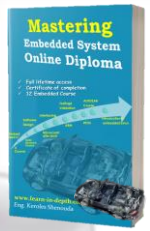
- Overview
- General Characteristics
- Types of CAN Messages
- CAN bus And CAN Node
- CAN transceiver

- Multi-master protocol
- Broadcasting
- Event-Driven
- Asynchronous communication (Event Triggered)
- Serial communication technology
- Priority-based bit-wise arbitration
- Variable message priority based on 11-bit (or extended 29 bit) packet identifier
- Originally developed by Robert Bosch for automobile in-vehicle network in the 1980s
- Differential, two wire with speed 1 Mbps
- CSMA-CA= CSMA with Collision Avoidance

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

General Characteristics

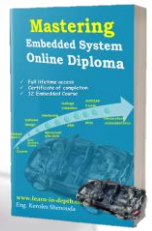
- Overview
- General Characteristics
  - Event-Driven
- CAN
  - Controller Area Network
- Types of CAN Messages
- CAN bus
- CAN transceiver



#LEARN_IN DEPTH

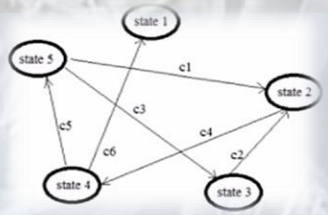
#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

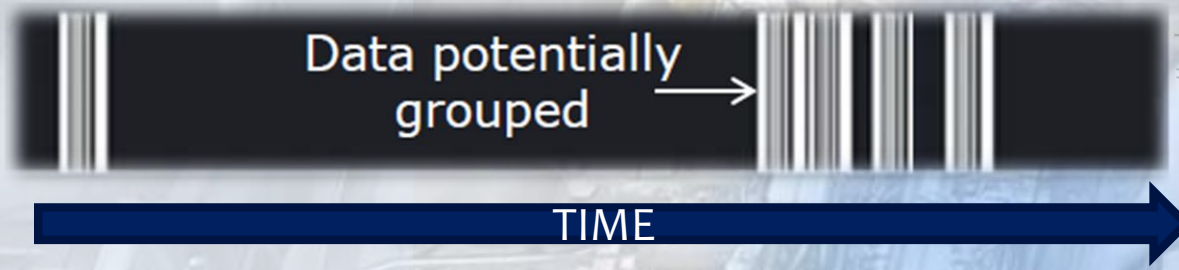


Event Driven vs. Time Driven

- Event Driven



  - Medium used only when necessary
  - Time of message arrival is unknown
  - Medium might be overloaded



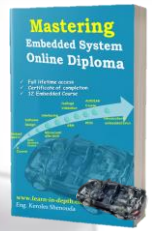
- Time Driven

  - Bandwidth utilization is known (duration of how long the medium is used)
  - Time of arrival is defined / guaranteed
  - Time Driven = Deterministic

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

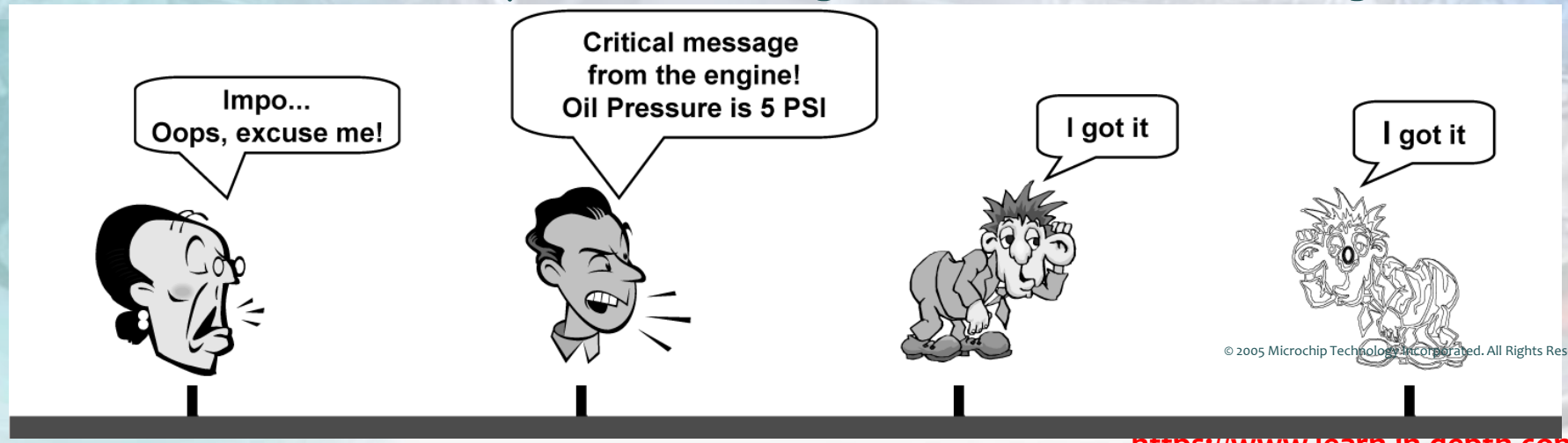
General Characteristics

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



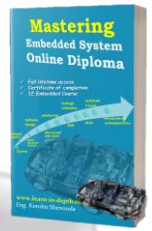
Bus Arbitration

- Arbitration – needed when multiple nodes try to transmit at the same time
- Only one transmitter is allowed to transmit at a time.
- A node waits for bus to become idle
- Nodes with more important messages continue transmitting



CAN bus

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

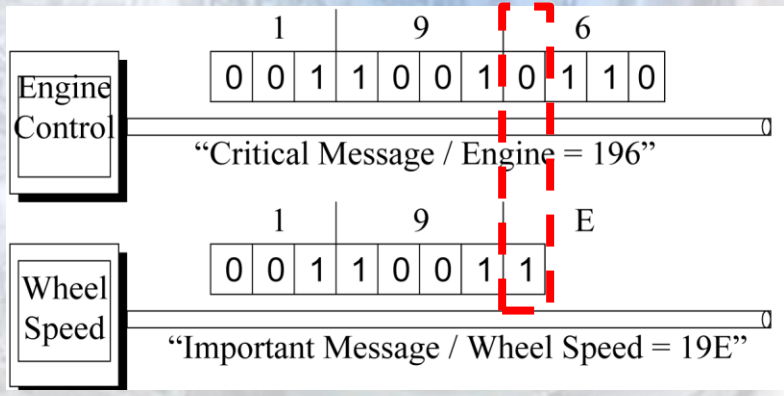


Bus Arbitration

- Message importance is encoded in message ID.

  Lower value = More important

- As a node transmits each bit, it verifies that it sees the same bit value on the bus that it transmitted.
- A "0" on the bus wins over a "1" on the bus.
- Losing node stops transmitting, winner continues.

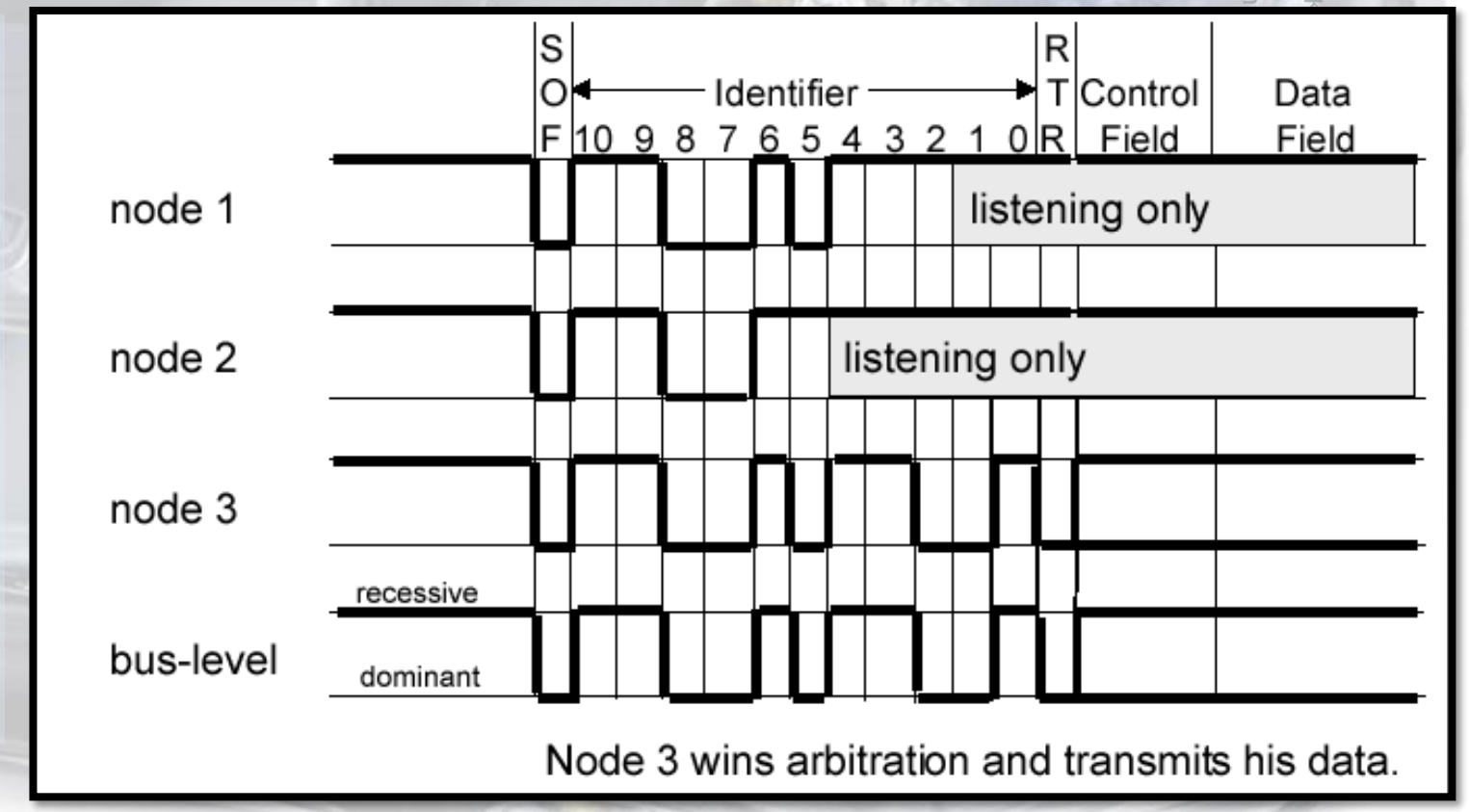


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

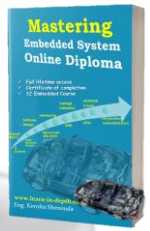
Arbitration type
ND + CSMA/CA + AMP

Non-Destructive + Carrier Sense Multiple Access / Collision Avoidance + Arbitration on Message Priority

- If two messages are simultaneously sent over the CAN bus, the bus takes the "**logical AND**" of the signal
  - Hence, the messages identifiers with the **lowest binary number gets the highest priority**
  - Every device listens on the channel and backs off as and when it notices a mismatch between the bus's bit and its identifier's bit



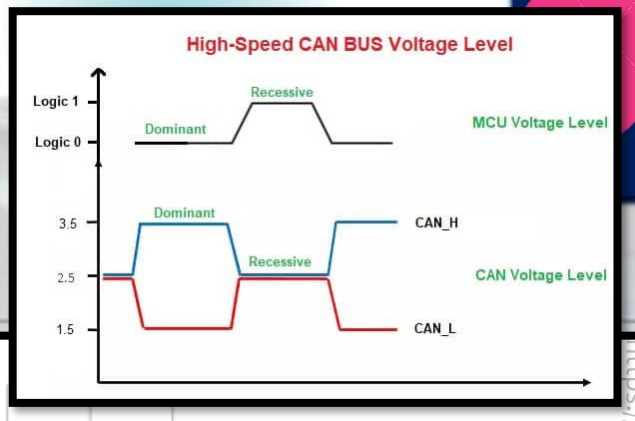
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

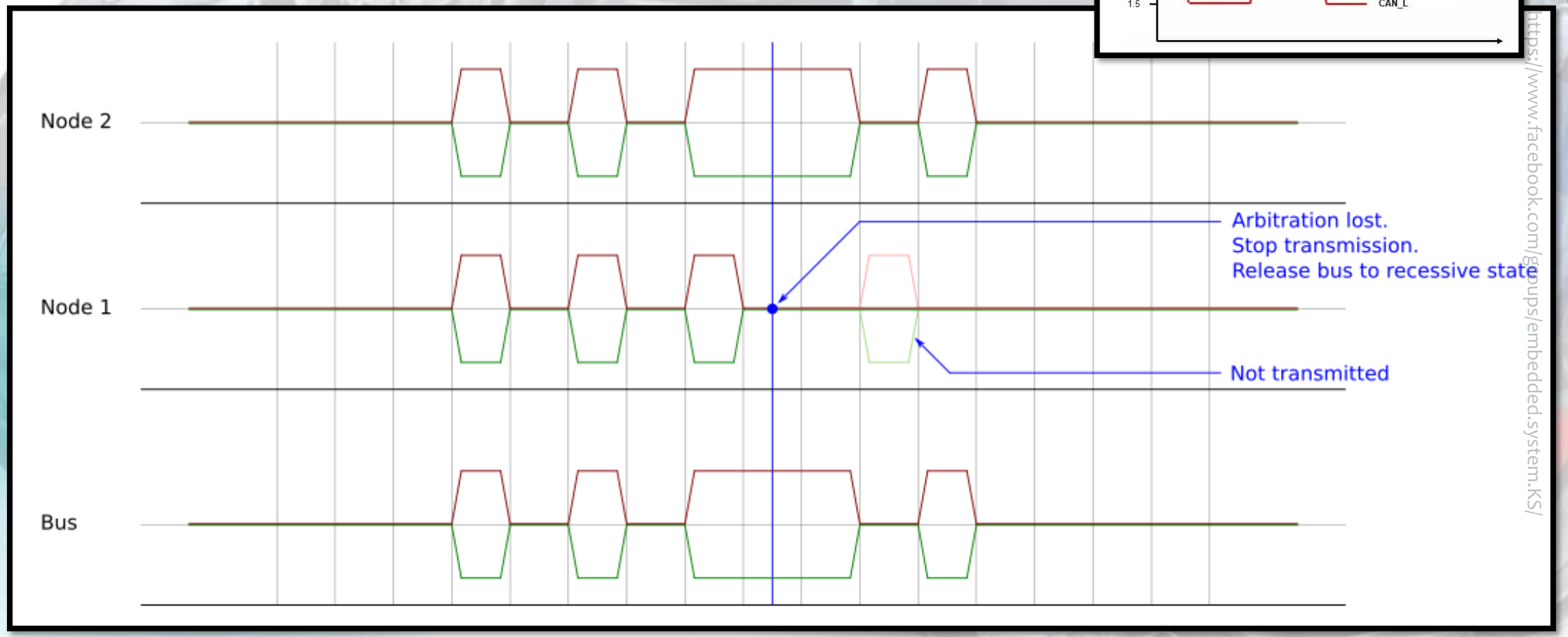


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

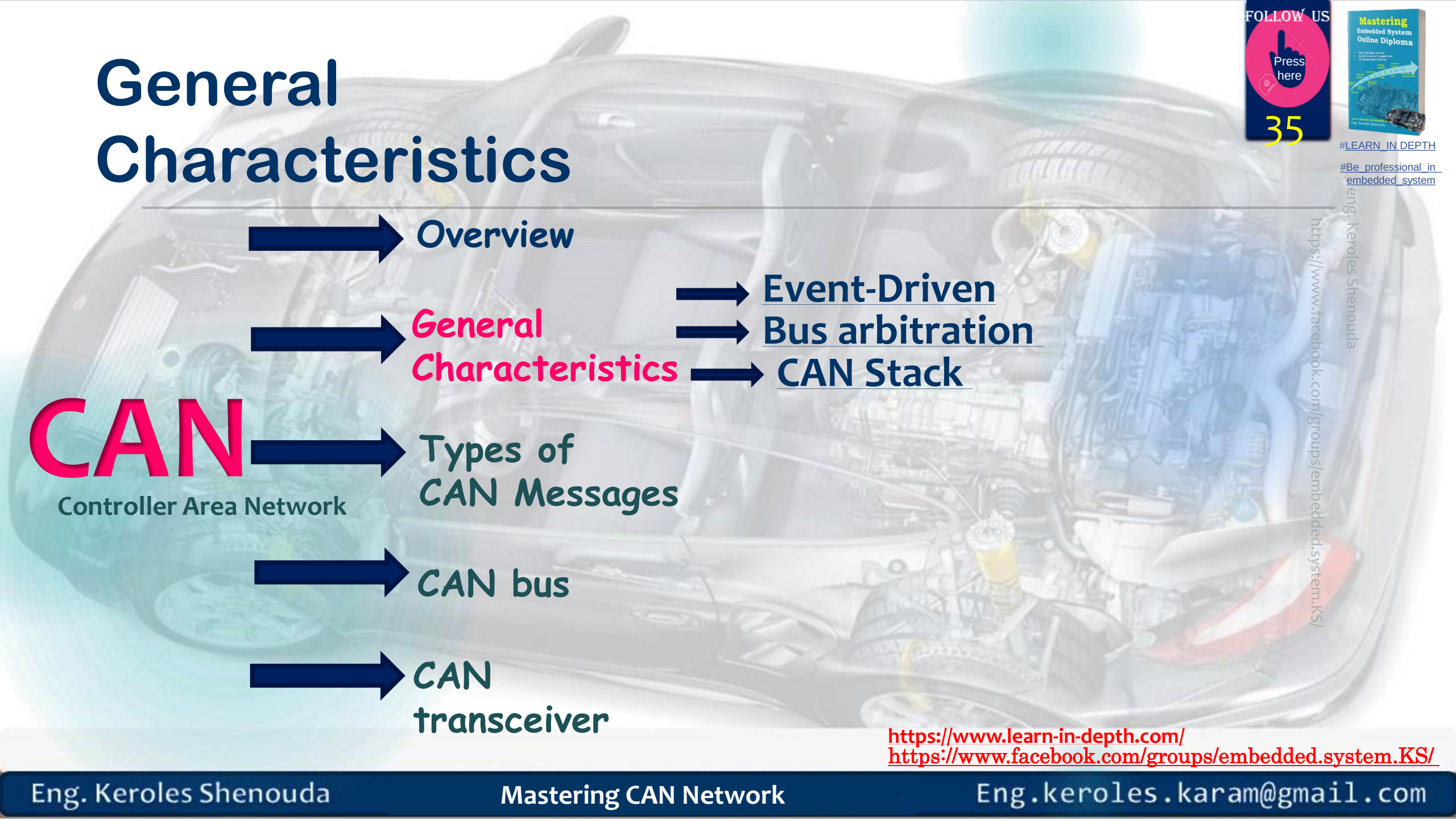
Arbitration (CAN H/L)



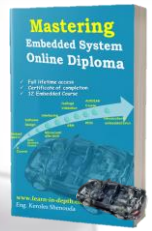


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

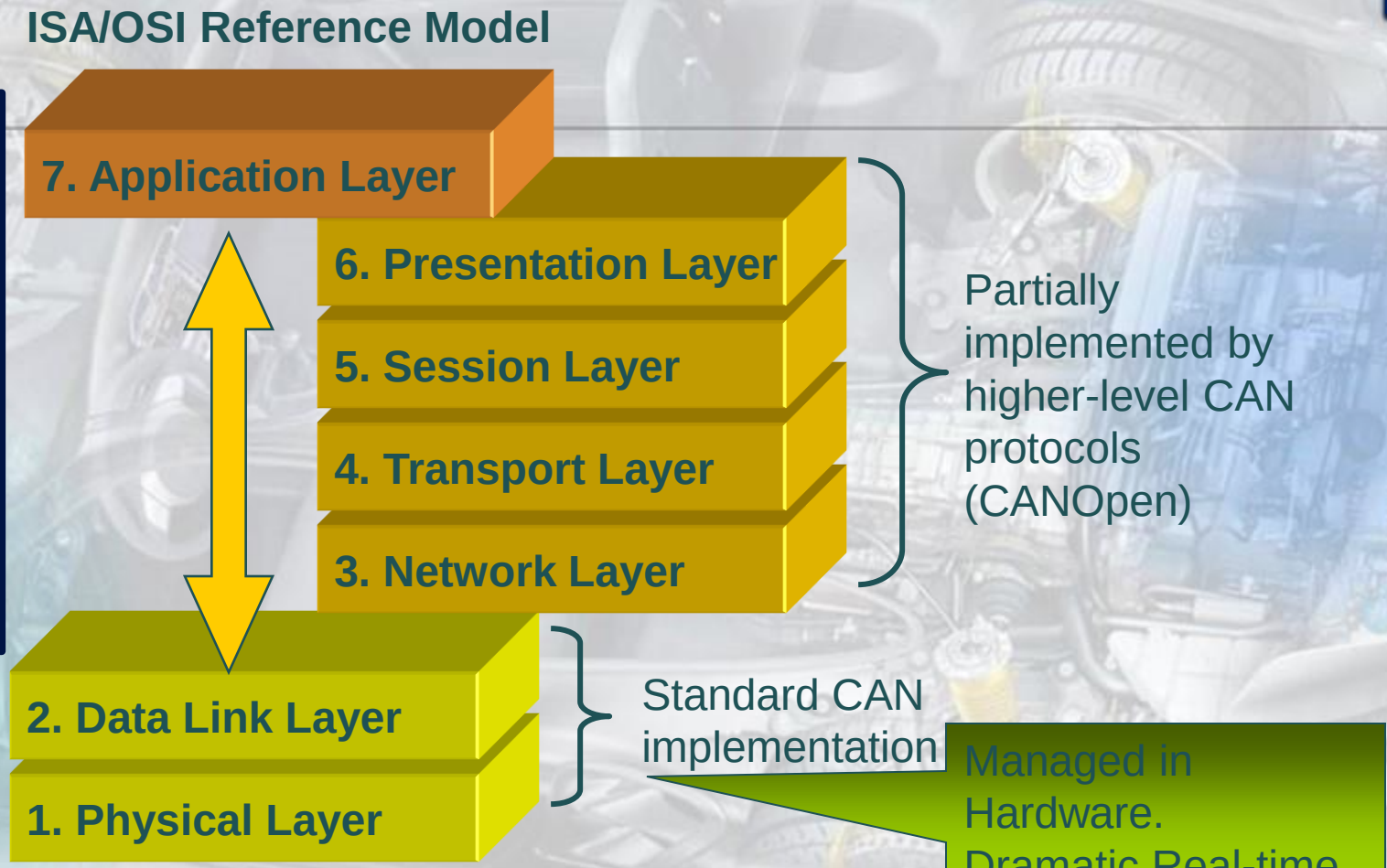
General Characteristics

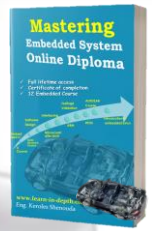


CAN (link layer and Physical layer)



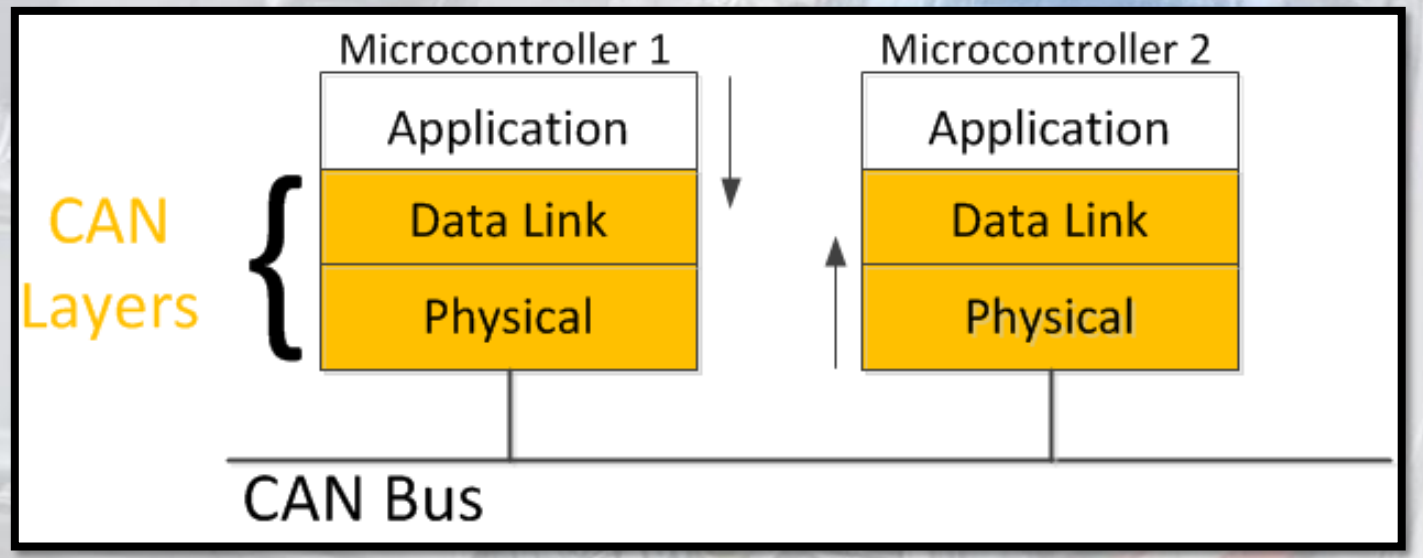
The standard CAN implementation bypasses the connection between the Data **Link layer** and the **Application layer**. The layers above the Data Link Layer are implemented in software which as per definition are called the Higher Layer Protocol





CANBUS and the OSI Model

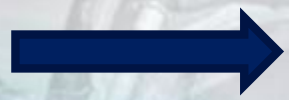
- CAN is a closed network
  - - no need for security, sessions or logins.
  - - no user interface requirements.
- Physical and Data Link layers in silicon.



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

General Characteristics

Overview

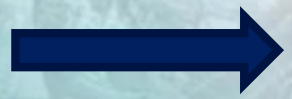


General Characteristics

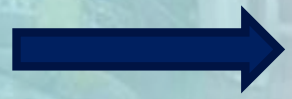
-  Event-Driven
-  Bus arbitration
-  CAN Stack
-  Message Filtering

CAN

Controller Area Network



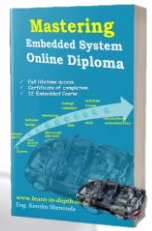
Types of CAN Messages



CAN bus

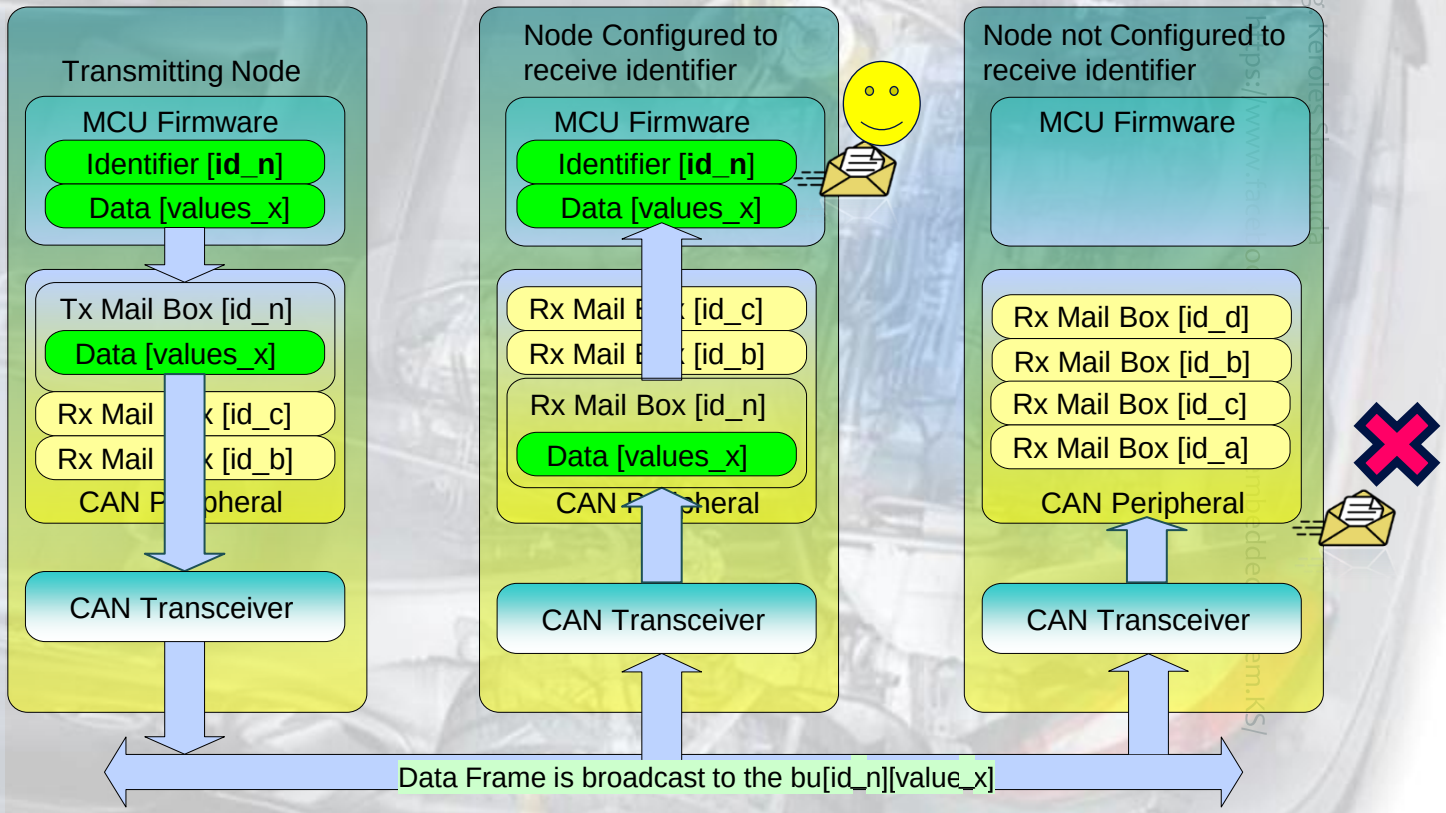
CAN transceiver

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



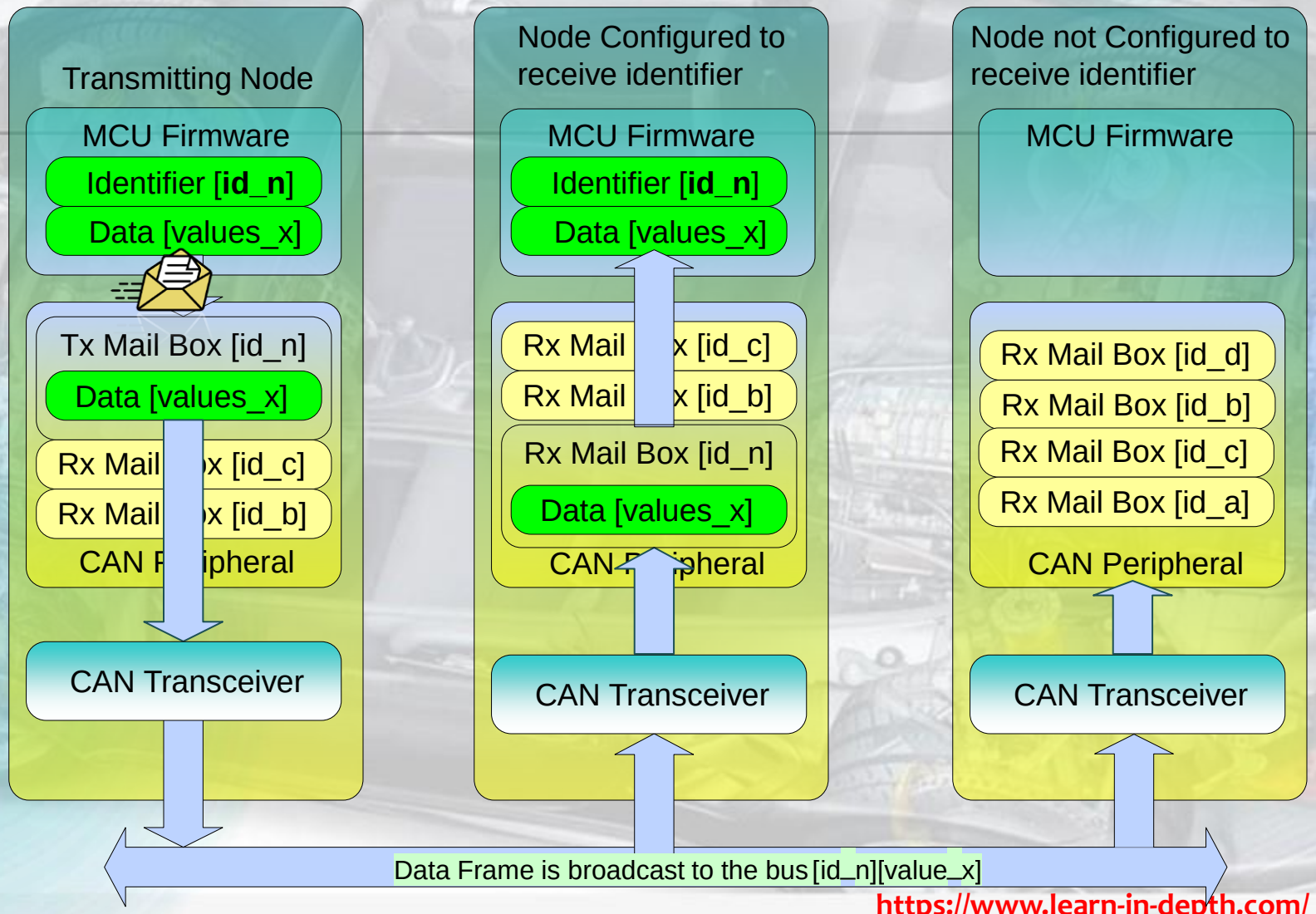
Message Filtering

- A node uses filter (s) to decide whether to work on a specific message.
- Message filtering is applied to the whole identifier.
- A node can optionally implement mask registers that specify which bits in the identifier are examined with the filter.
- If mask registers are implemented, every bit of the mask registers must be programmable.



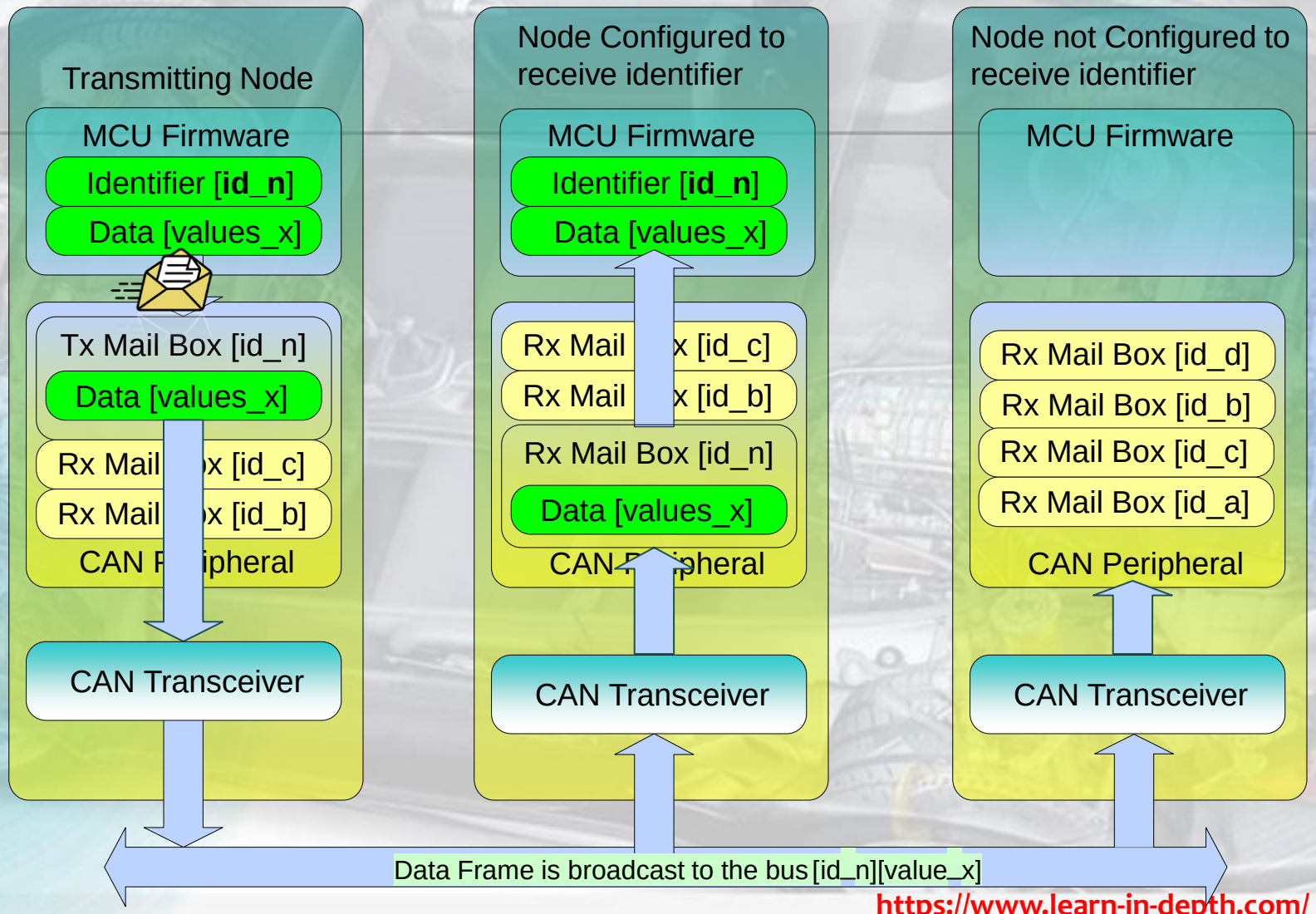
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Data Flow in CAN

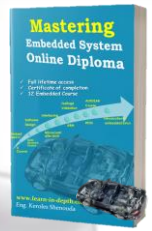


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Data Flow in CAN



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



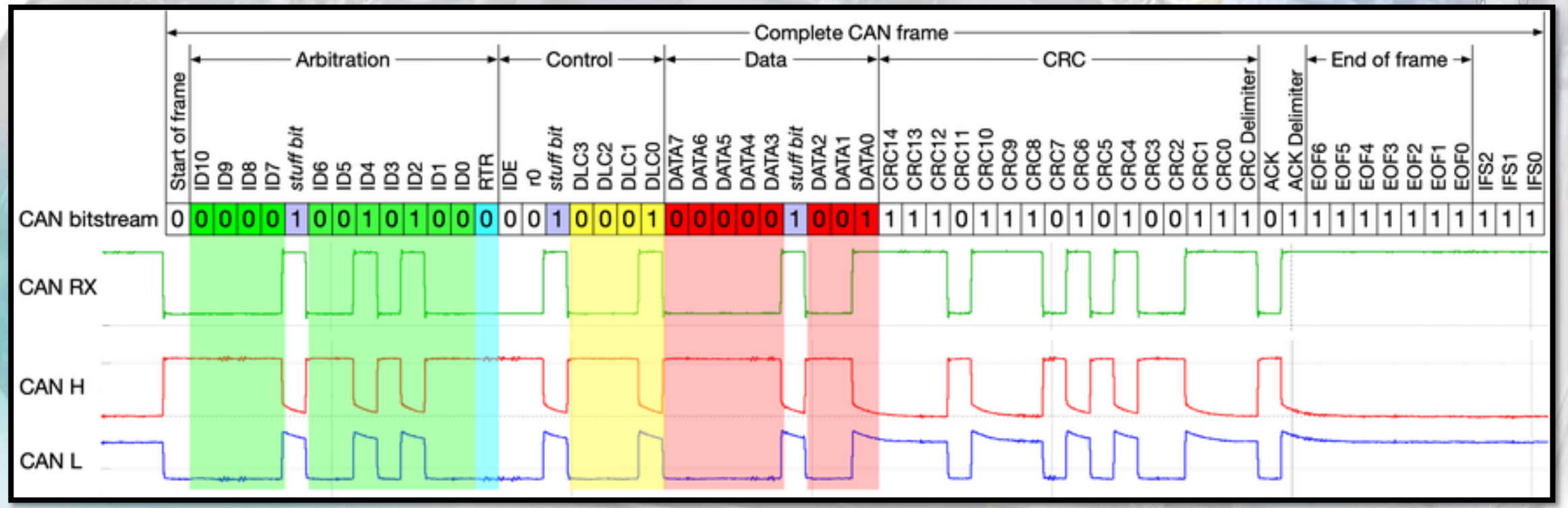
CAN "Controller Area Network"

CAN
Controller Area Network

- Overview
- General Characteristics
- Types of CAN Messages
  - Data frame
  - Remote frame
  - Error frame
  - Overload frame
- CAN bus Characteristics
- CAN Bus Errors

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

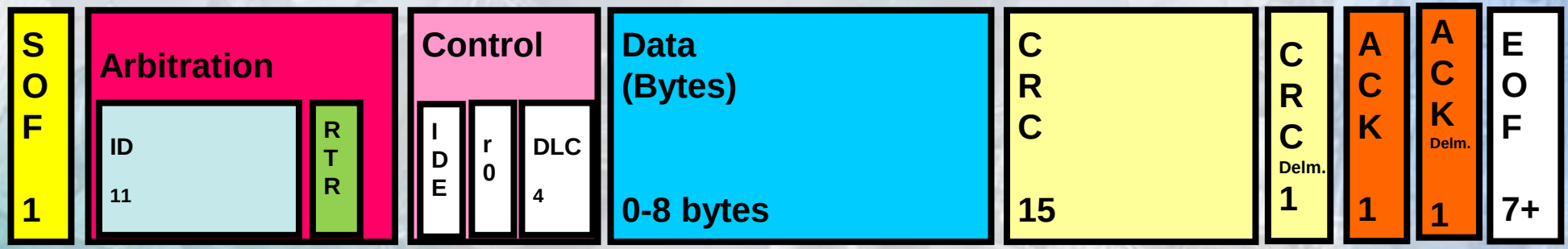
CAN Frame



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

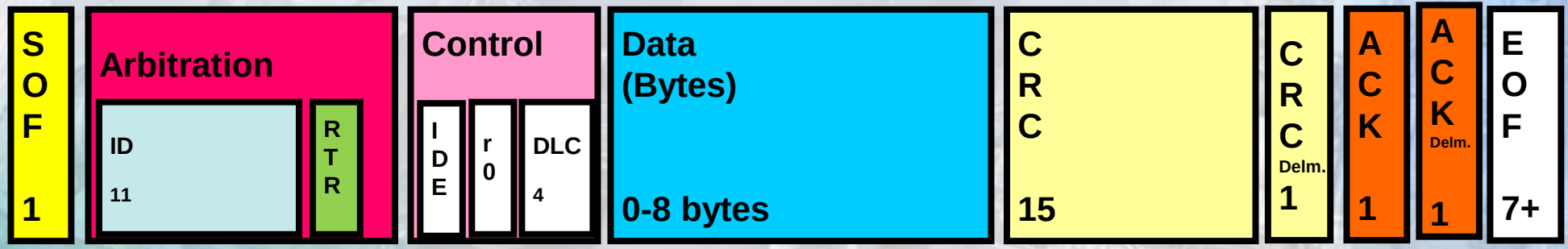
Data Frame

- A data frame consists of seven fields: start-of-frame, arbitration, control, data, CRC, ACK, and end-of-frame.



Data Frame

- A data frame consists of seven fields: start-of-frame, arbitration, control, data, CRC, ACK, and end-of-frame.



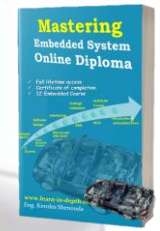
Start of Frame

- A **single dominant bit** to mark the beginning of a data frame.
- All nodes have to synchronize to the leading edge caused by this field.

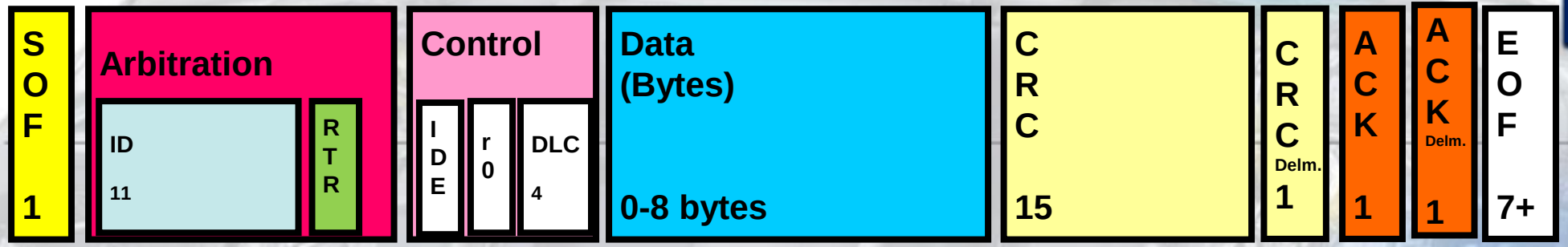
Data Frame

Arbitration Field

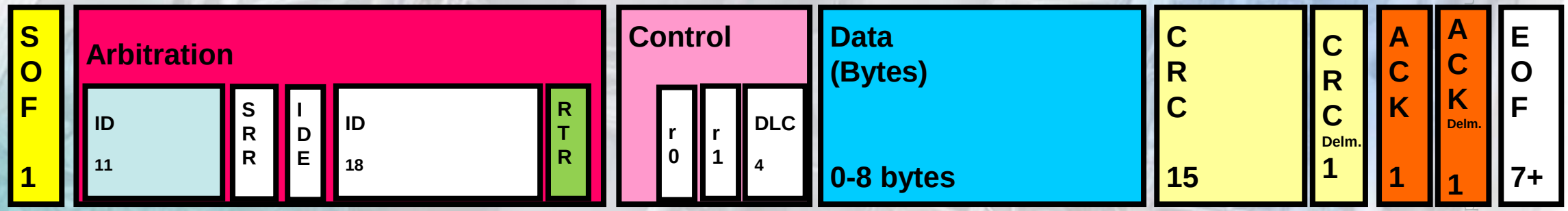
There are two formats for this field: standard format and extended format.



standard format



extended format

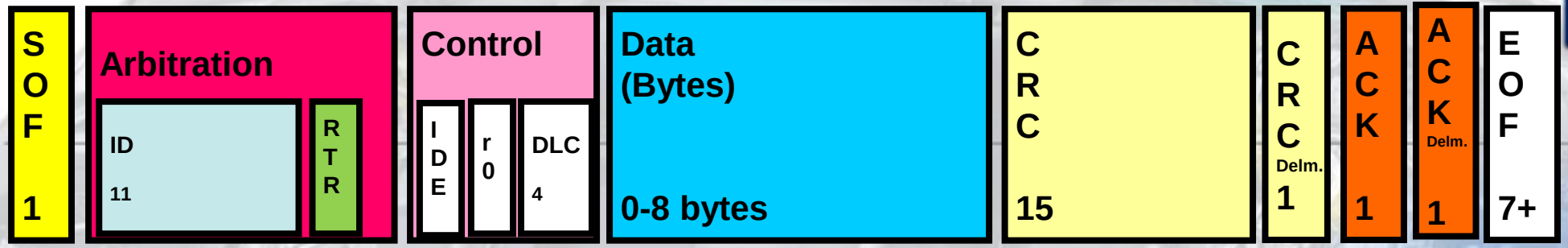


- The **RTR** bit is the **remote transmission request** and must be 0 in a data frame.
- The **SRR** bit is the substitute remote request and is **recessive** .
- The **IDE** field indicates whether the identifier is extended and should be recessive in the extended format.
- The extended format also contains the 18-bit extended identifier.

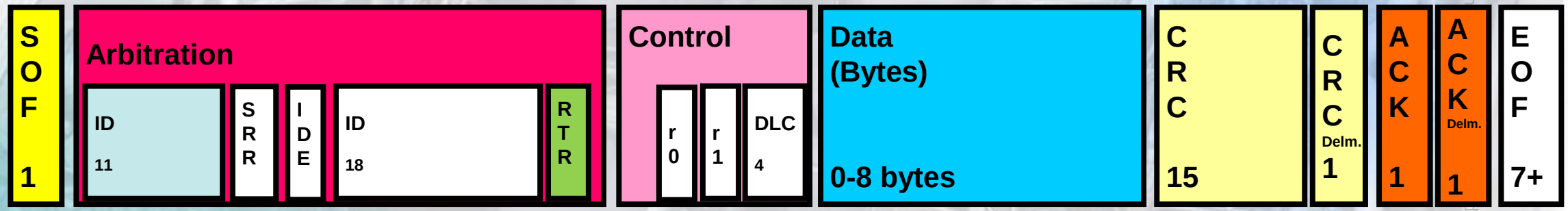
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Data Frame

standard format



extended format

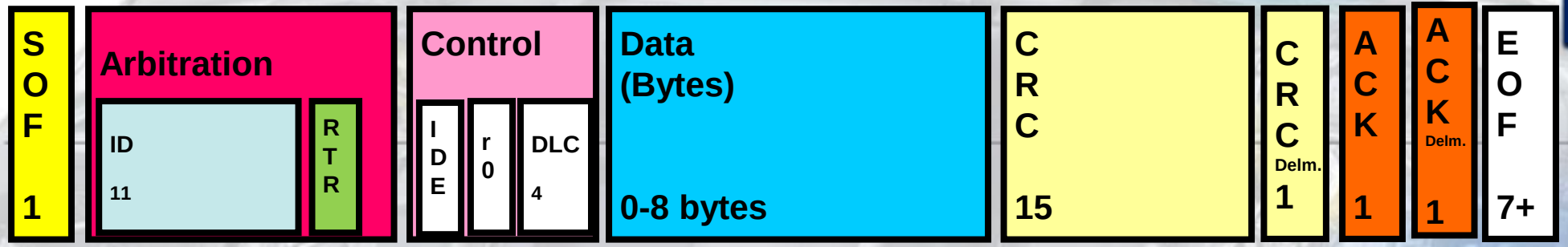


Control Field

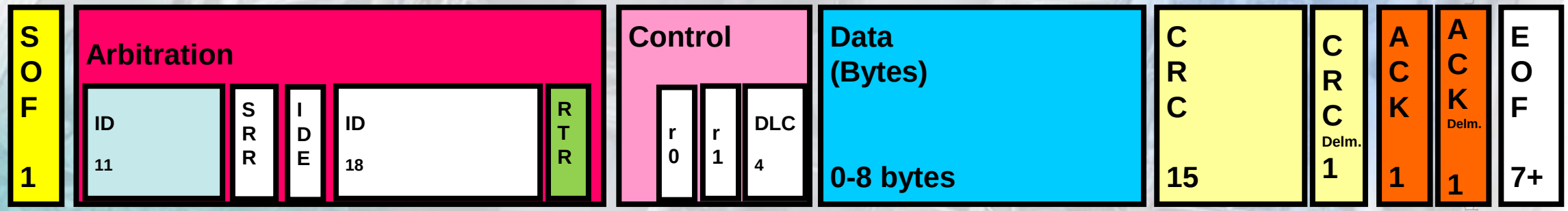
- The first bit is IDE bit for the standard format but is used as reserved bit r1 in extended format.
- r0 is reserved bit.
- DLC3…DLC0 stands for data length and can be from 0000 (0) to 1000 (8).

Data Frame

standard format

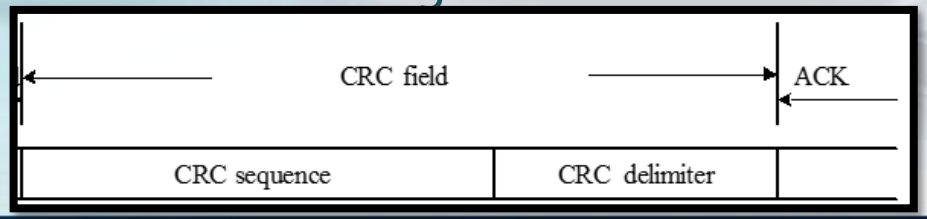


extended format

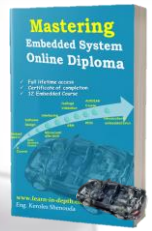


CRC Field

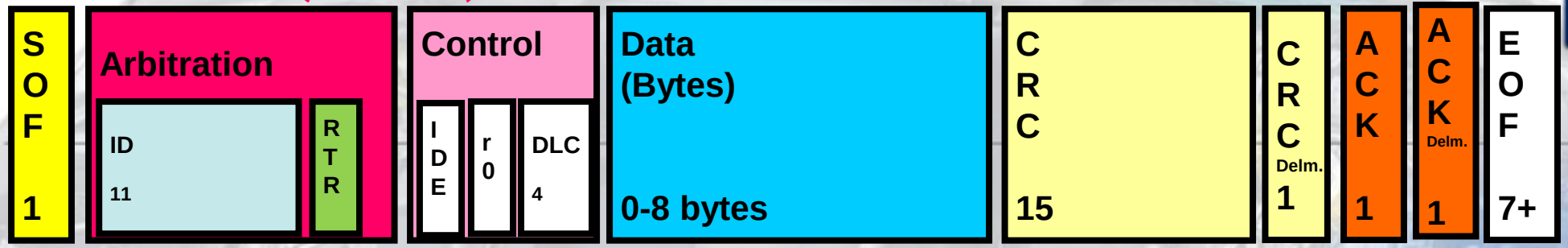
- It contains the 16-bit CRC sequence and a CRC delimiter.
- The CRC delimiter is a single **recessive** bit.



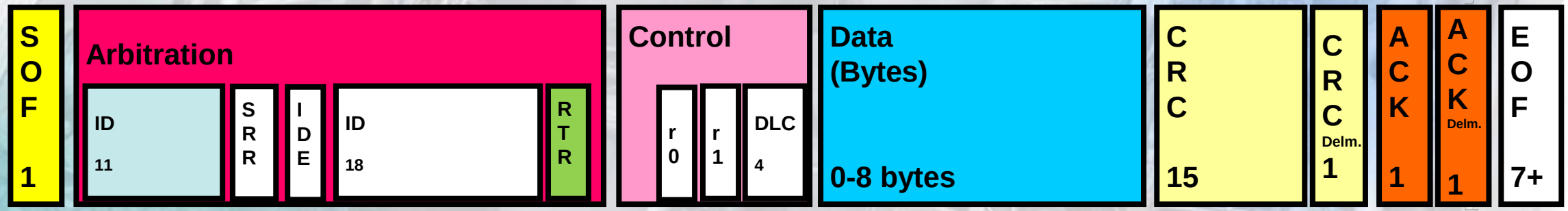
Data Frame



standard format (CAN 2.0A)



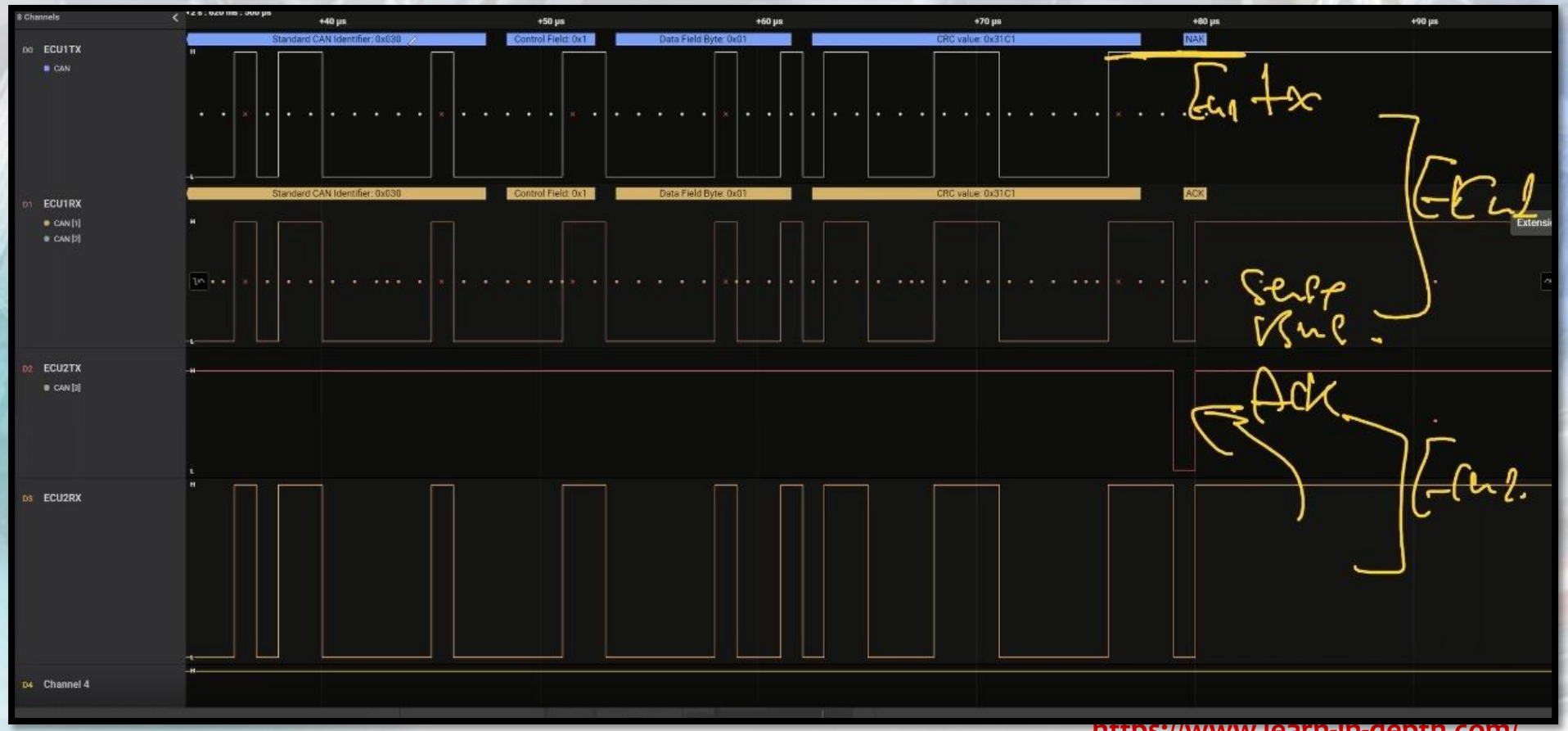
extended format (CAN 2.0B)



ACK Field

- Consists of two bits
- The first bit is the **acknowledgement bit**.
  - This bit is set to recessive by the transmitter, but will be reset to dominant if a receiver acknowledges the data frame.
- The second bit is the **ACK delimiter** and is **recessive**.

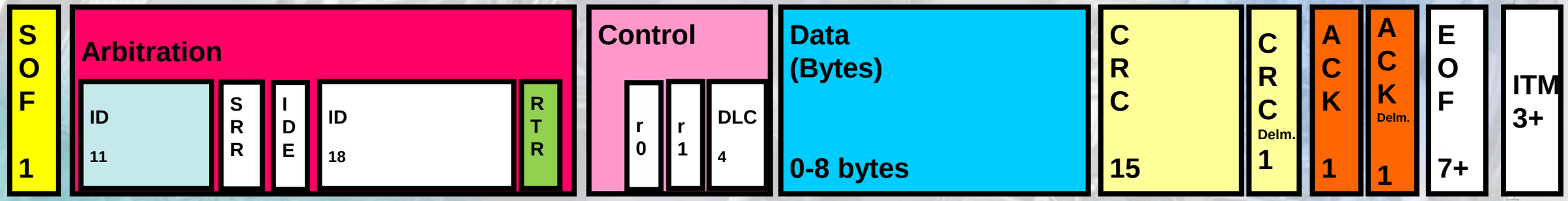
CASE Study 3



Data Frame

standard format (CAN 2.0A)

extended format (CAN 2.0B)

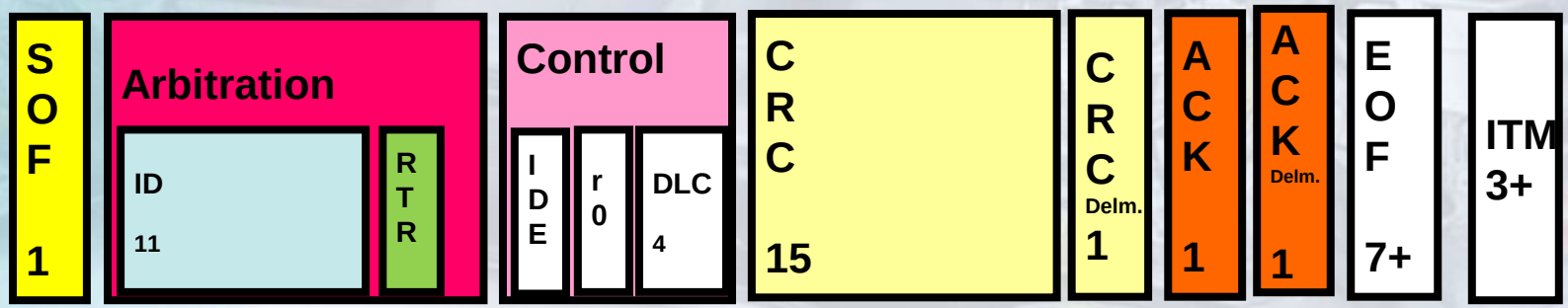


- EOF : 7 bit **recessive**
- ITM Field
  - **After frame there is IFS Inter frame Space is at least 3 recessive bits**

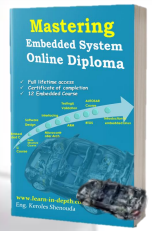
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Remote Frame

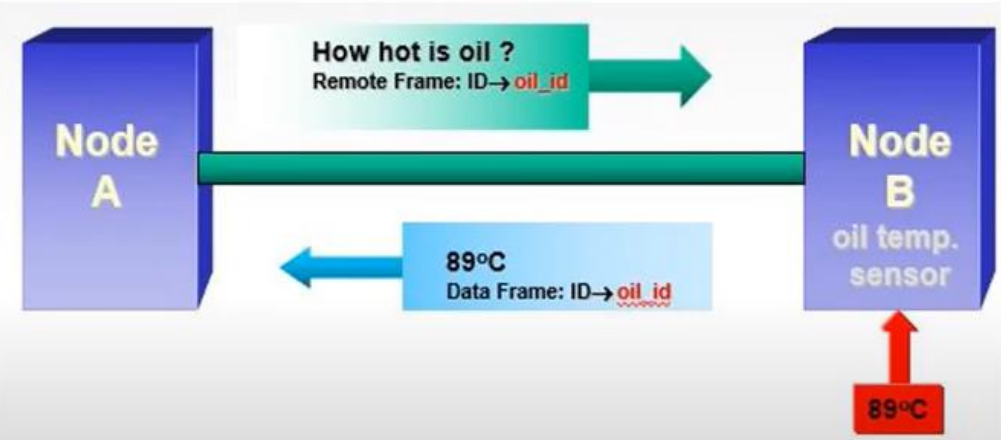
- Used by a node **to request** other nodes **to send certain type of messages**
- Has six fields
  - These fields are identical to those of a data frame with the exception that the RTR bit in the arbitration field is **recessive** in the remote frame.



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Remote Frame Example

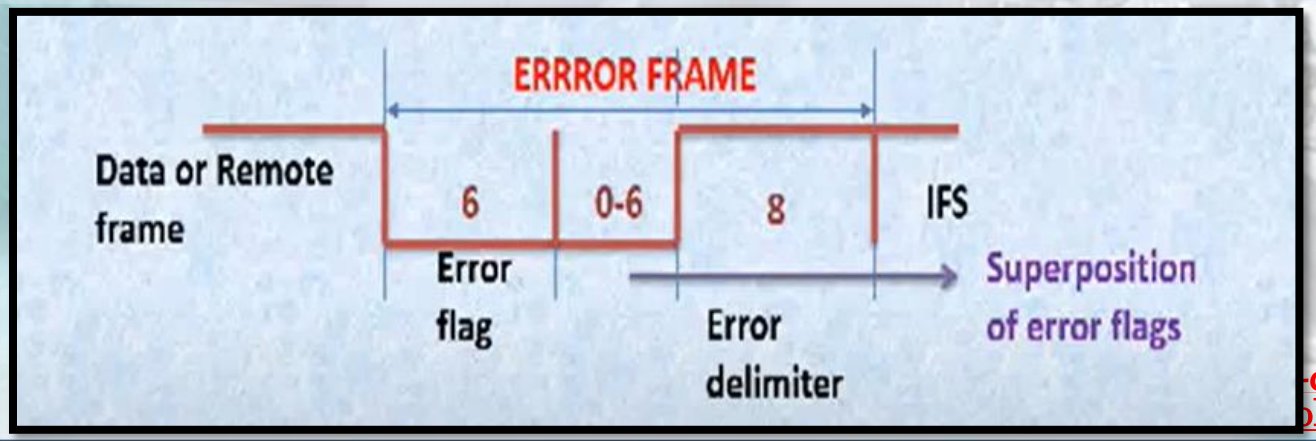


- If a node acting as a RECEIVER wishes to request the data from the source, it sends a Remote Frame with an identifier that matches the identifier of the required Data Frame. The appropriate data source node will then send a Data Frame as a response to this remote request.

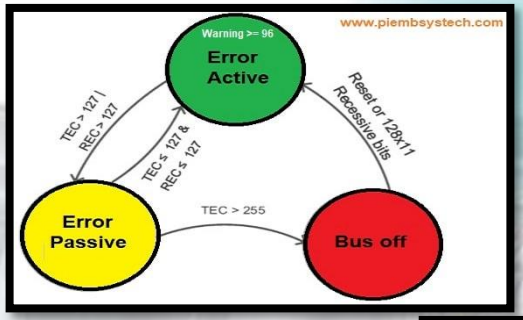
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

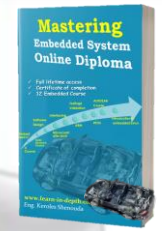
Error Frame

- This frame consists of two fields.
  - The first field is given by the superposition of error flags contributed from different nodes.
  - The second field is the **error delimiter**. (Recessive)
- Error flag can be either active-error flag or passive-error flag.
  - **Active error** flag consists of **six consecutive dominant bits**.
  - **Passive error** flag consists of **six consecutive recessive bits**.
- The **error delimiter** consists of eight recessive bits.

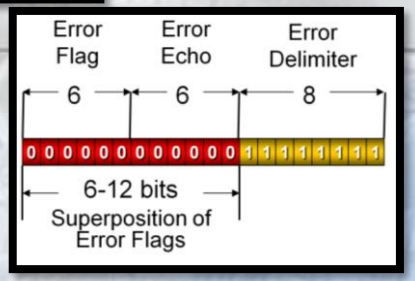


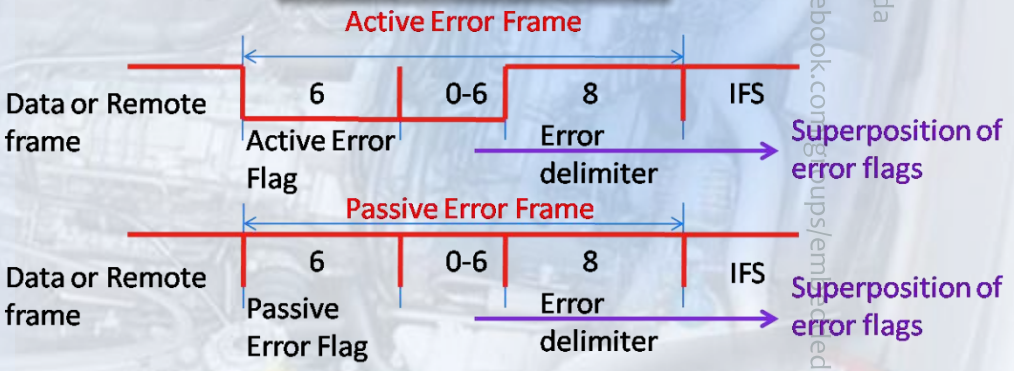
how the CAN manages error states

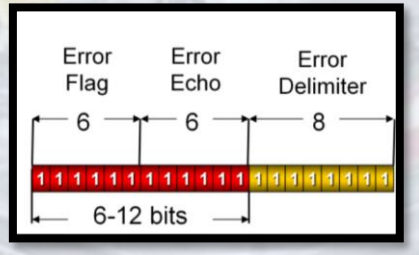




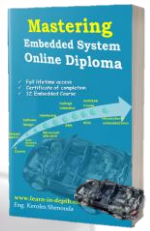
- In every CAN node, there are 2 error counters
  - **Transmit Error Counter (TEC)**
  - **Receive Error Counter (REC).**
- When **the transmitter** detects an error in the transmitted frame, it increments the **TEC by 8**.
- A **receiver detecting** an error will increment its **REC by 1**.
- On successful transmission/reception the error counters are reduced by 1.
- **Based on the error counts**, the node behavior varies.
  - By default, the **Active Error frame will be transmitted** on the bus, when **TEC and REC < 128.**
  - But when **127 < TEC \ REC > 255**, **the passive Error** frame will be transmitted on the bus
- Finally, the node enters into the **Bus off state**, when **TEC > 255**.
- If node enters into the bus off state then no frames will be transmitted.



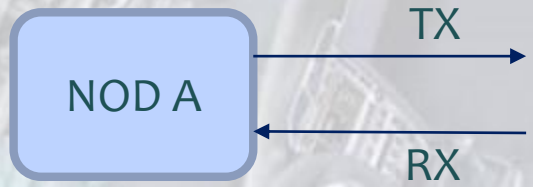




https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Case Study 4 (Nod A Send without Any other node on the BUS)

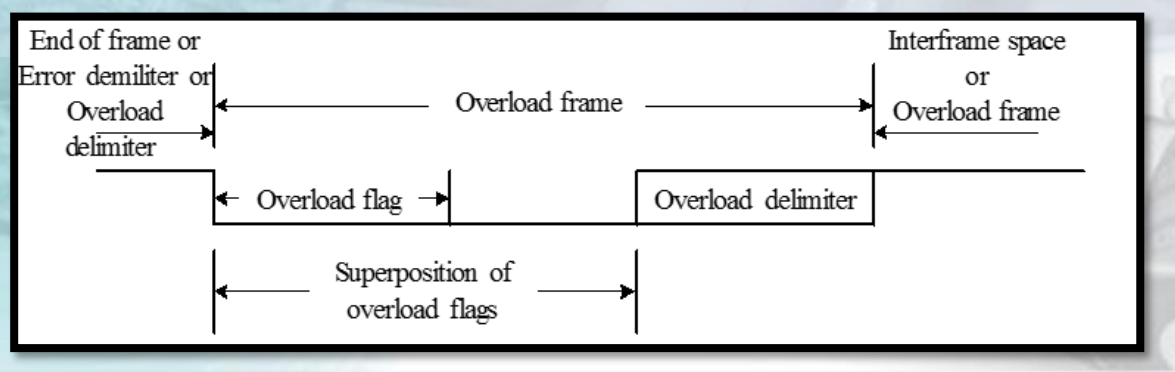


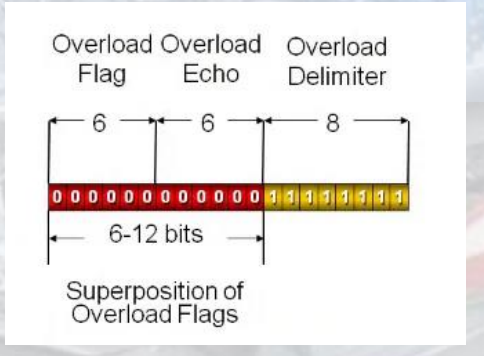

```
RX_MSG c=1, t=267115315200, type=0000, ERROR_FRAME id=00
CHIP_STATE c=1, t=267115315200, busStatus= PASSIVE, txErrCnt=128, rxErrCnt=0
```


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

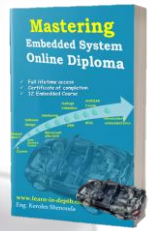
Overload Frame

- Consists of two bit fields: **overload flag** and **overload delimiter**
- The **over load frame** will be generated, **when the receiving node is overloaded** - **i.e. it is not able to detect and receive** the incoming messages
- The format is very similar to Error Frame but without the error counters incrementing.
- An Overload frame indicates that its transmitter require delay before receiving next data or remote frame The overload flag consists of six dominant bits.
- The overload delimiter consists of eight recessive bits.





https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

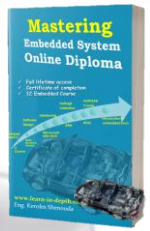


Question In Depth ☺

The Overload Frame is identical to an Active Error Frame. The only difference is that an Overload Frame does not increase the error counters and does not causes a retransmission of a frame

How does a CAN controller differentiate the overload frame and error frame ?

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

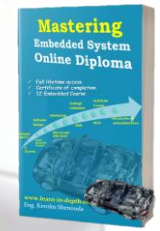


#LEARN_IN DEPTH
#Be_professional_in_embedded_system

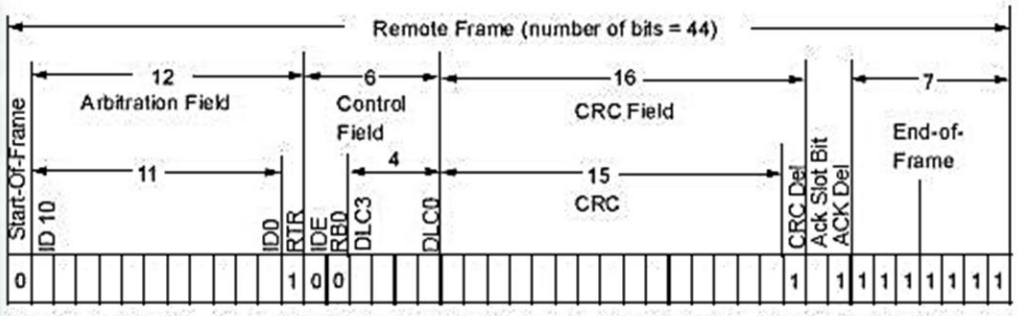
Question In Depth ☺

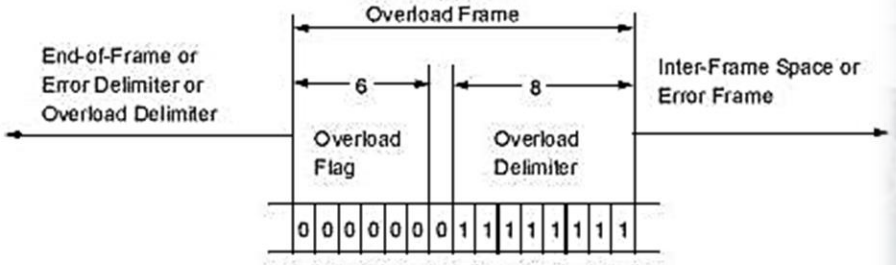
- overload frames and error frames **look the same**.
- However, a CAN node can distinguish them by **the time** when the frame was received:
  - An **overload frame** will only occur **during the interframe space**,
  - and an **error frame** only **during an actual CAN frame**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

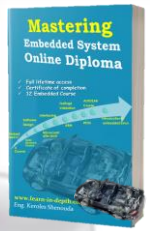


Question In Depth ☺

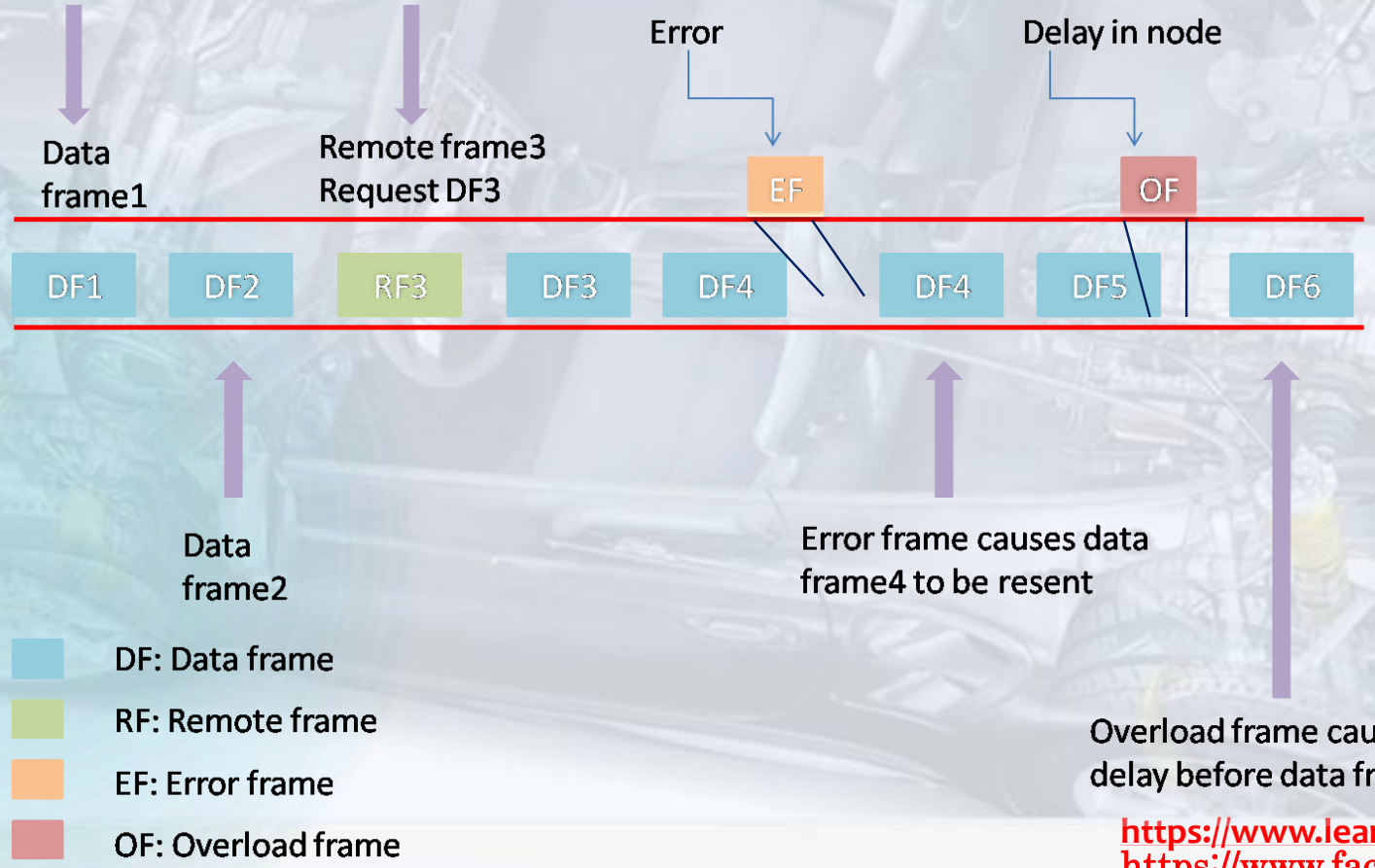




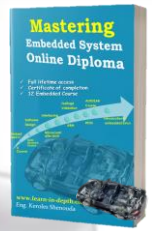
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Examples



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

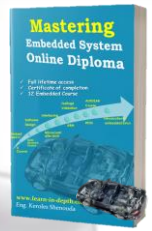


CAN "Controller Area Network"

- Overview
- General Characteristics
- Types of CAN Messages
- CAN bus Characteristics
- CAN Bus Errors

CAN
Controller Area Network

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

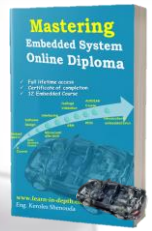


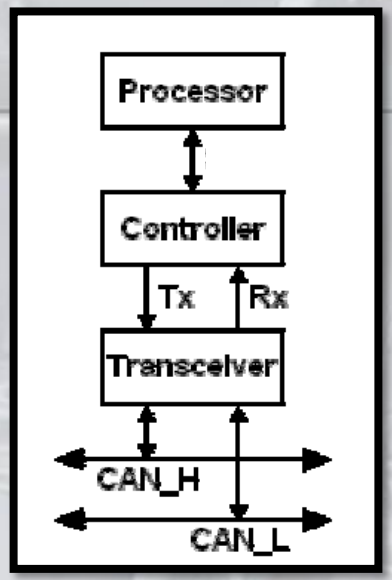
CAN "Controller Area Network"

CAN
Controller Area Network

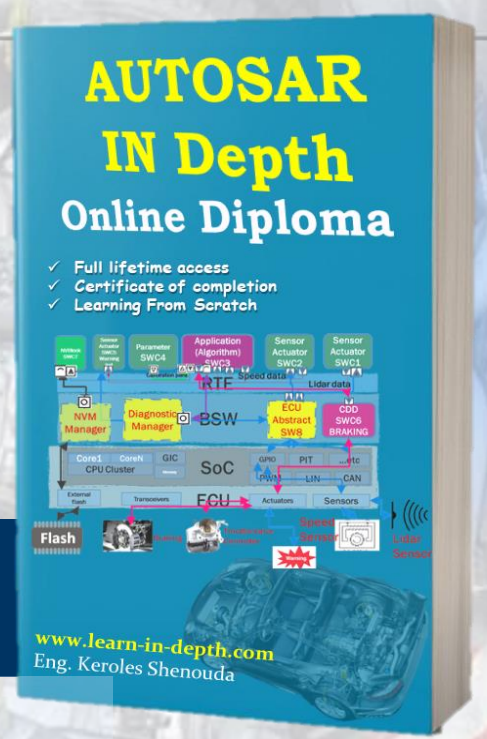
- Overview
- General Characteristics
- Types of CAN Messages
- CAN bus Characteristics
  - CAN Node
  - Single Ended Vs Differential
  - Recessive And Dominant Signals
  - Bit Rate / Bus Length
- CAN Bus Errors

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/





CAN Node



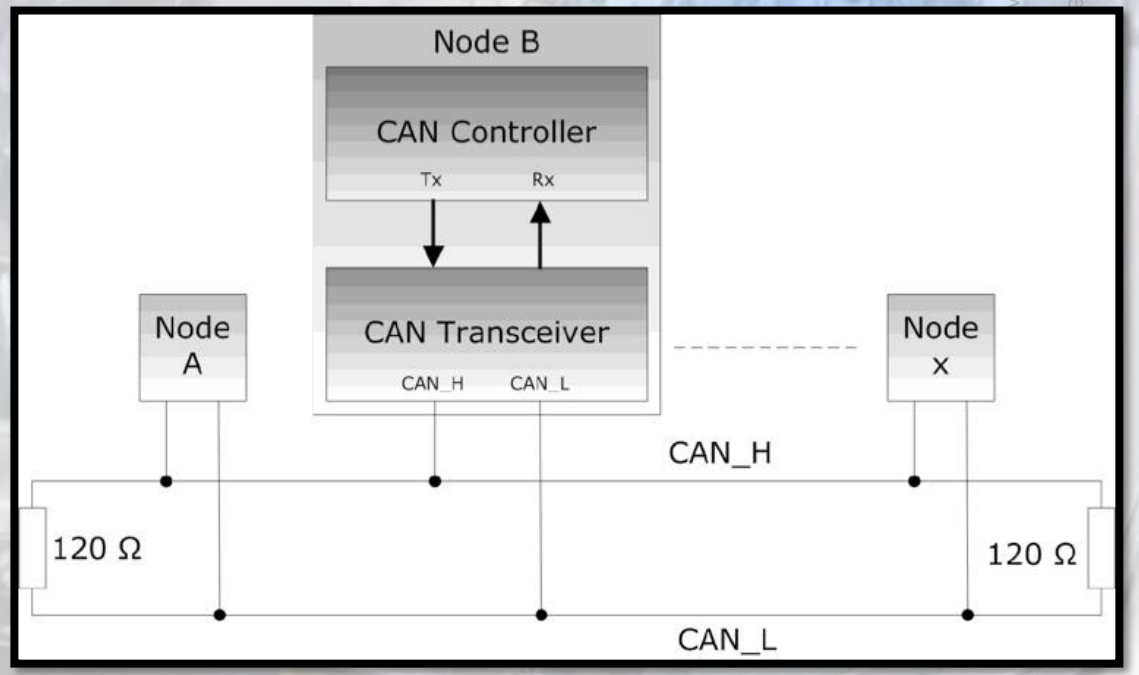
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

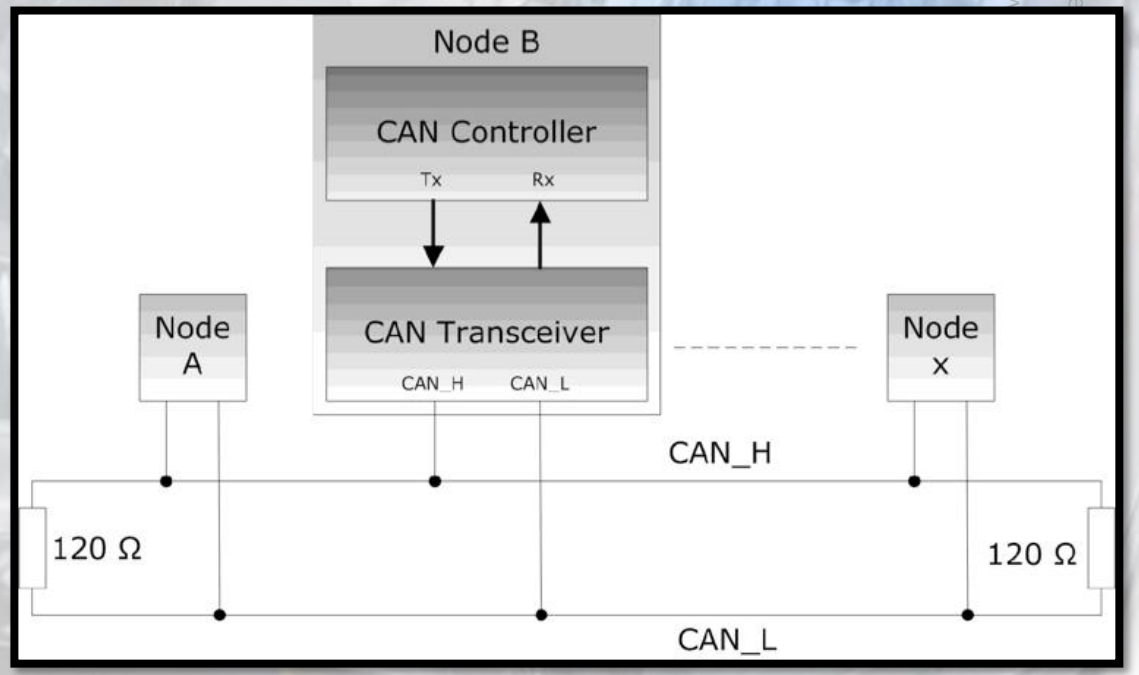
CAN Interface Hardware

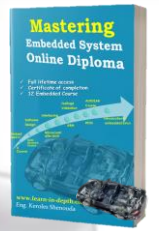
- CAN Controller
  - deals with the communication functions described by the CAN protocol.
  - It also triggers the transmission, or the reception of the CAN messages.
- CAN Transceiver
  - is responsible for the transmission or the reception of the data on the CAN bus.
  - It converts the data signal into the stream of data collected from the CAN bus that the CAN controller can understand.



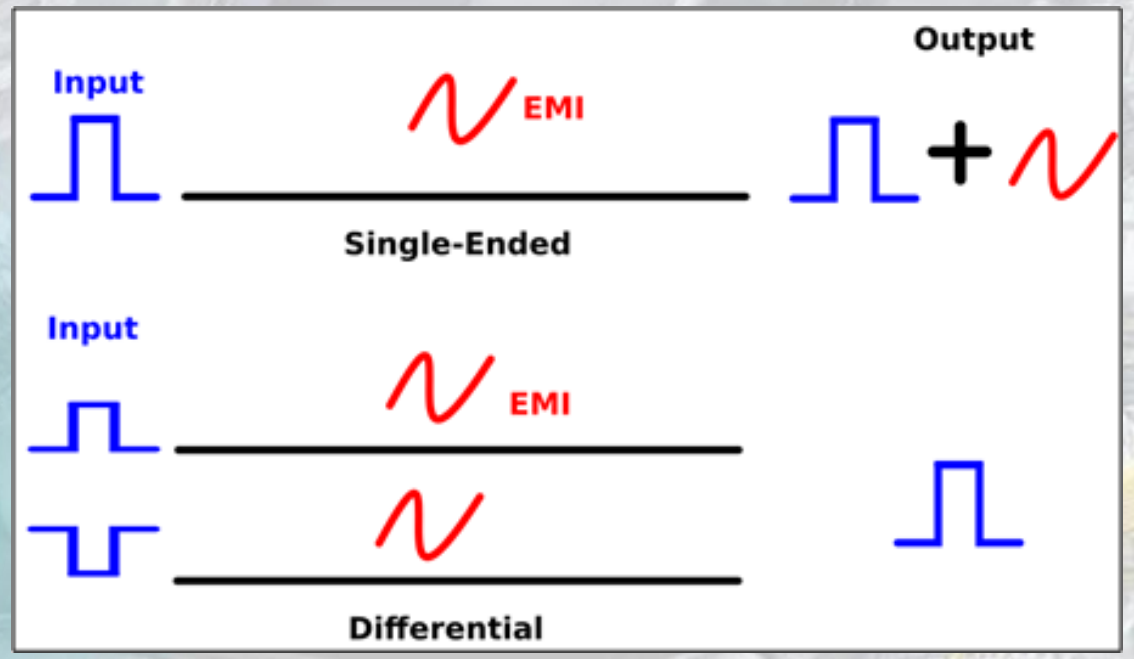
CAN Interface Hardware

- In the specific case of the CAN bus controller, we need a line driver (**transceiver**) to convert the controller's TTL signal to the actual CAN level, which is **a differential voltage**.
- The use of differential voltage contributes to the vast reliability of CAN.



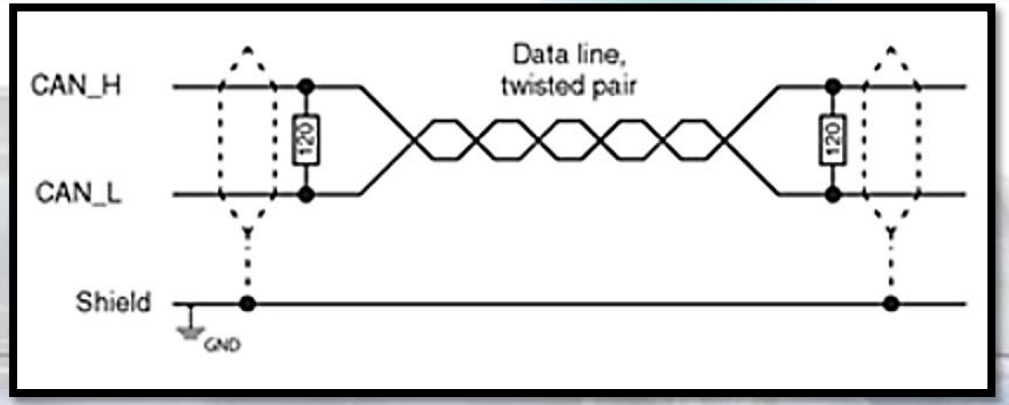


Single Ended Vs Differential

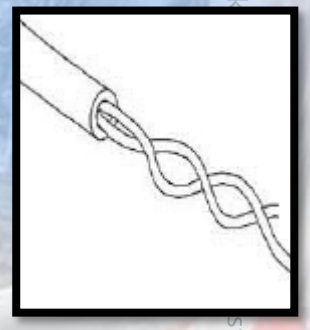


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

twisted pair cable



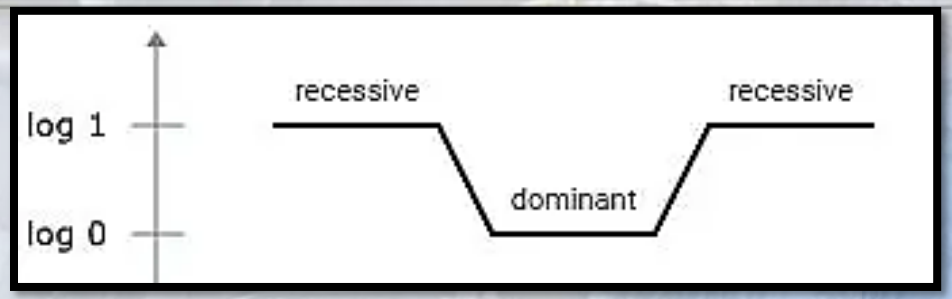
- twisted pair cable is used to transmit or receive the data. It is also known as CAN bus
- CAN bus consists of two lines, i.e., **CAN low** line and **CAN high** line, which are also known as CANH and CANL, respectively.
- The transmission occurs due to the **differential voltage** applied to **these lines**.
- The CAN uses **twisted pair cable** and **differential voltage** because of its environment.
- The twisting of the two lines also reduces the magnetic field.
- The bus is terminated with 120Ω resistance at each end.

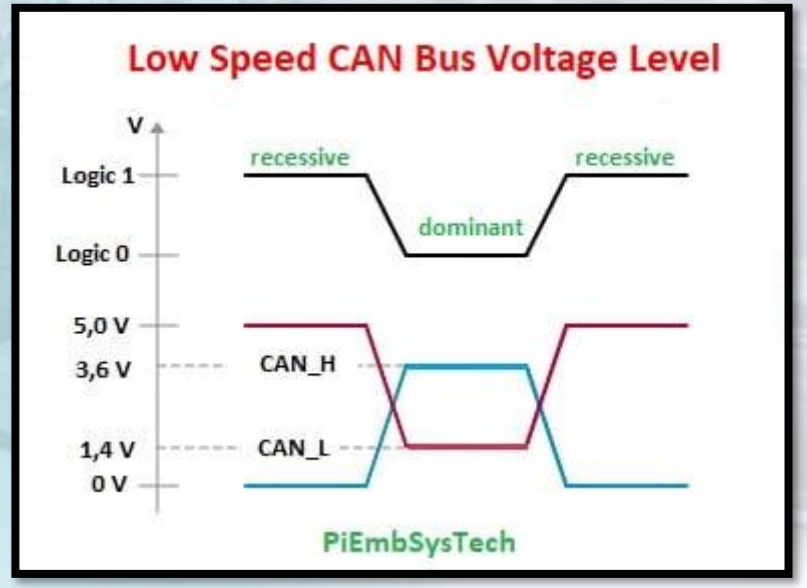


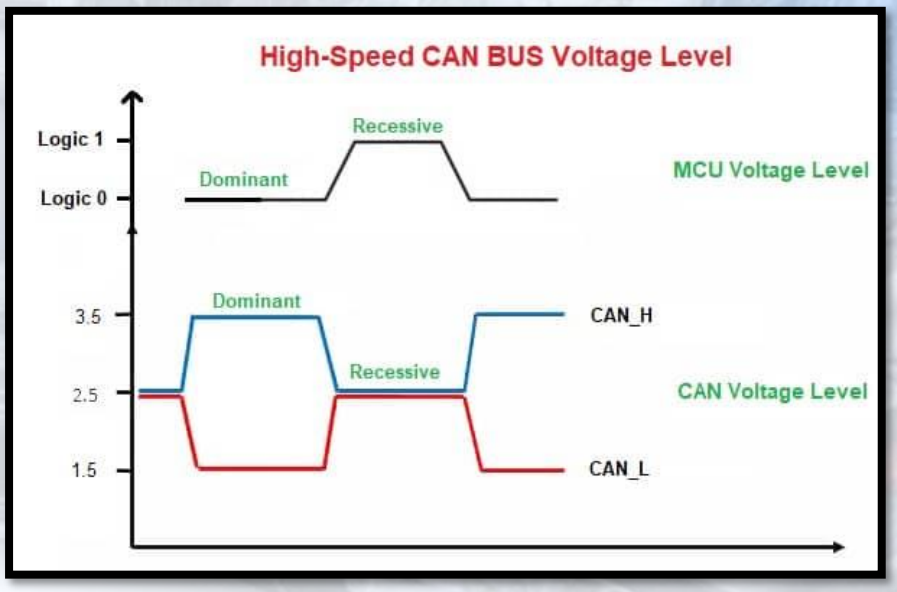
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

CAN BUS use two signal level

- Logic 1 is known as a Recessive bit
- Logic 0 is known as a Dominant bit



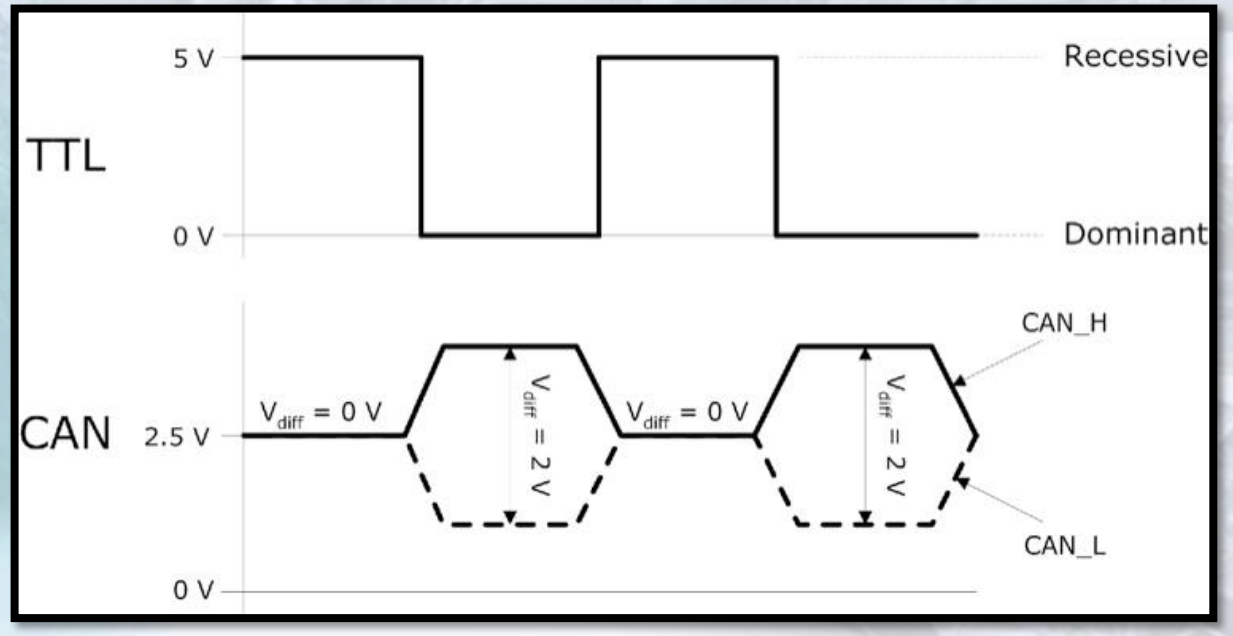




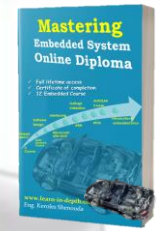
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

CAN_H and CAN_L

- The actual signal status, recessive or dominant, is based on the differential voltage between CAN_H and CAN_L (2V during dominant bit time; 0V during recessive bit time).

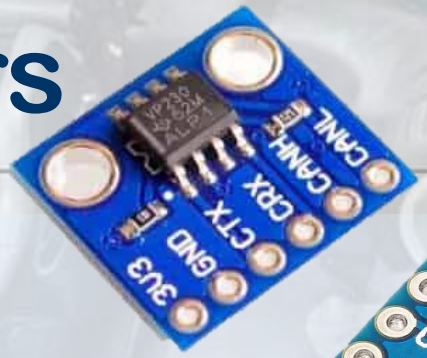


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



CAN Bus Transceivers

- ~3/3.3V
  - CJMCU-230 CAN Bus Transceiver
  - SN65HVD230 CAN Bus Transceiver
- TJA1050 CAN Bus Transceiver Module
  - Up to 1Mb/s bus speed
  - Up to 1000 meter bus length
  - 5V Operation
- MCP2551 (High-Speed CAN Bus Transceiver)
  - Connect it with 120 ohm

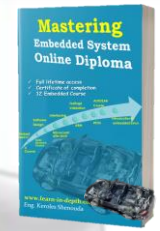


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

TJA1050 CAN BUS TRANSCEIVER MODULE

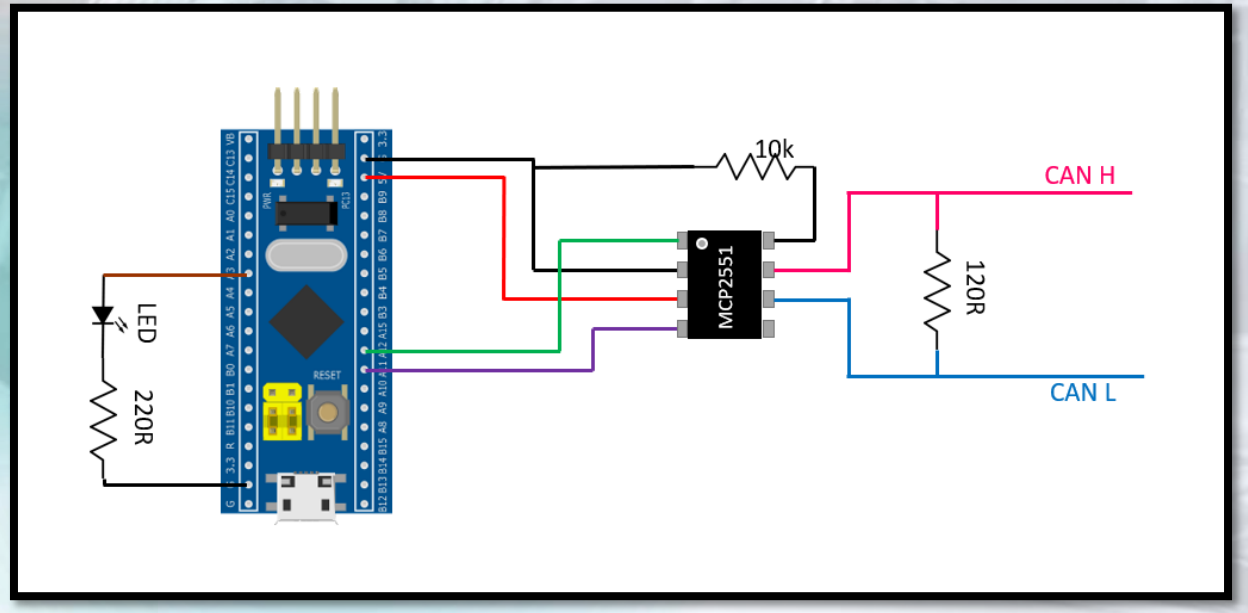
- 1 x 4 Header
  - VCC = 5V power
  - TX= Transmit data input. Reads in data from CAN controller and sends to bus lines
  - RX= Receive data output. Reads out data from bus lines to the CAN controller
  - GND= Ground, connects to MCU ground.
- 1 x 2 Header
  - CANH = CAN Bus H connection. CANH connects to CANH on other modules
  - CANL = CAN Bus L connection. CANL connects to CANL on other modules
  - The module has 2 small M2 holes for mounting.

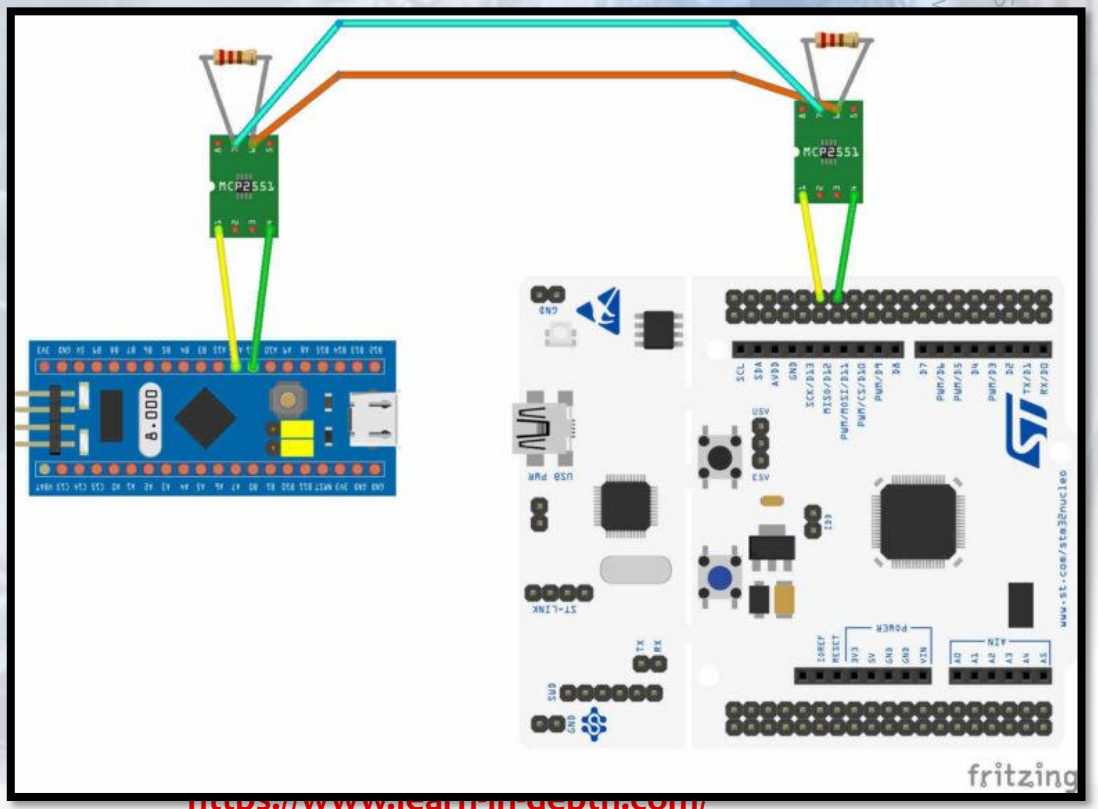
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



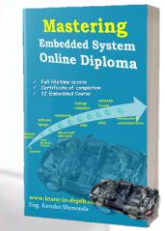
MCP2551 (High-Speed CAN Bus Transceiver)

- there is 120 ohms Resistance at each node. This is very important, or else you will not get the data.

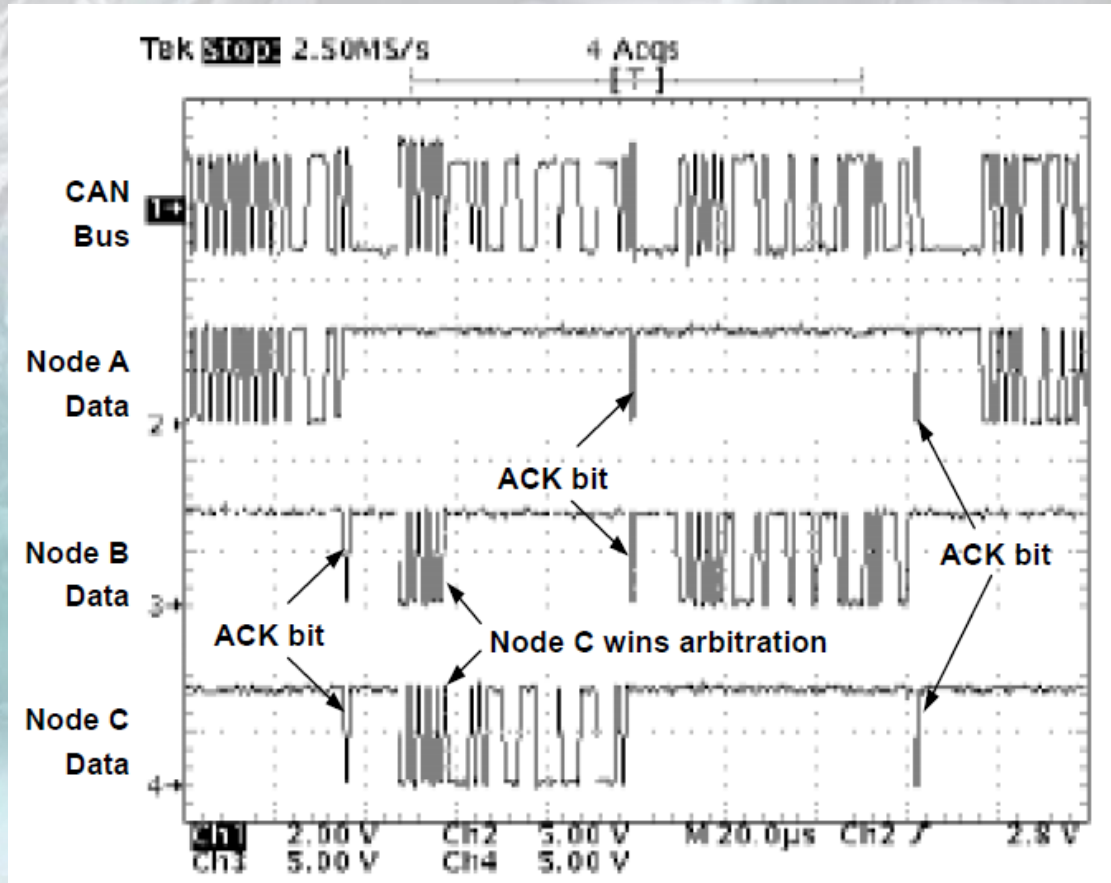




https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



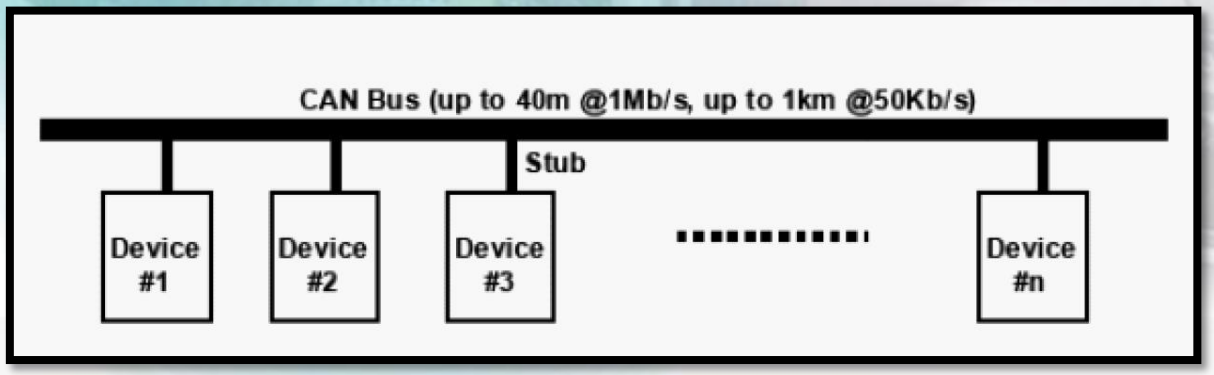
CAN Bus Traffic

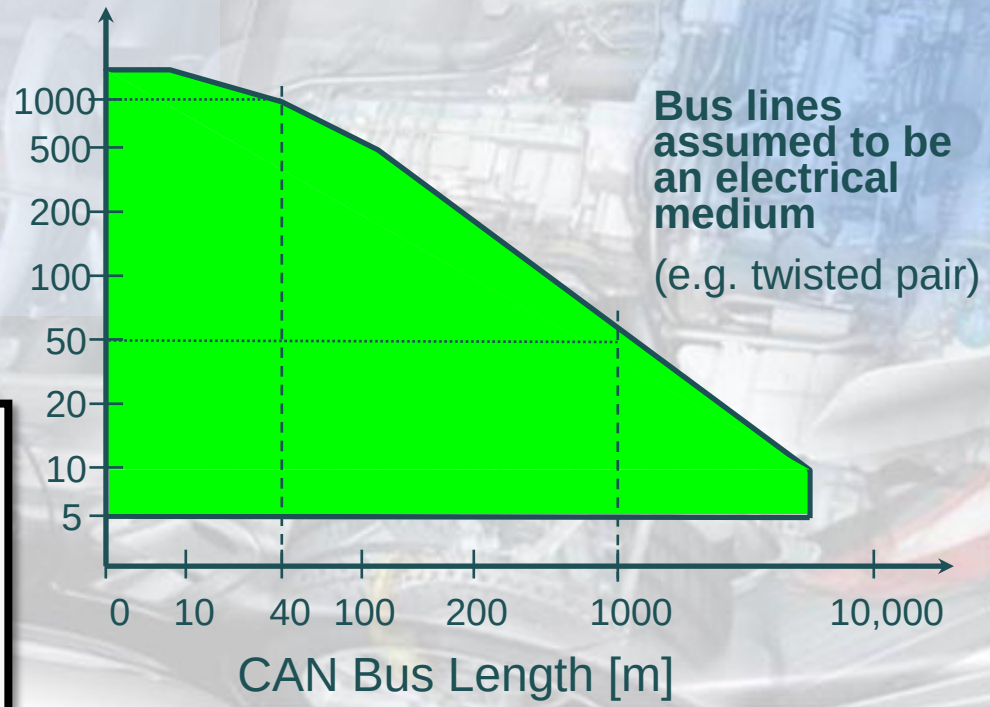


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Bit Rate / Bus Length

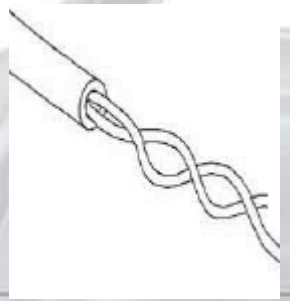
- 1M bit/sec 40 meters (131 feet)
- 500K bit/sec 100 meters (328 feet)
- 250K bit/sec 200 meters (656 feet)
- 125K bit/sec 500 meters (1640 feet)

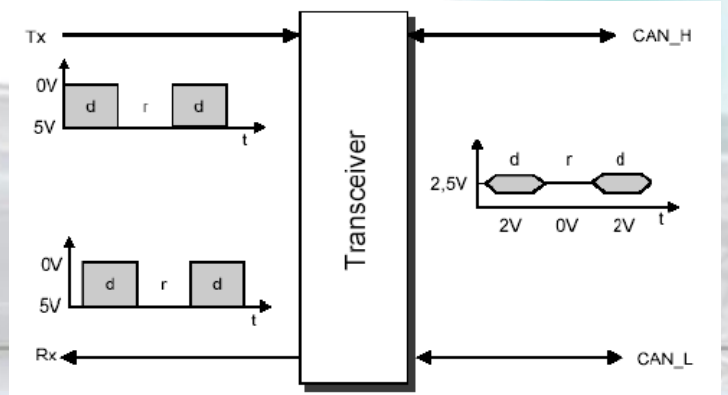




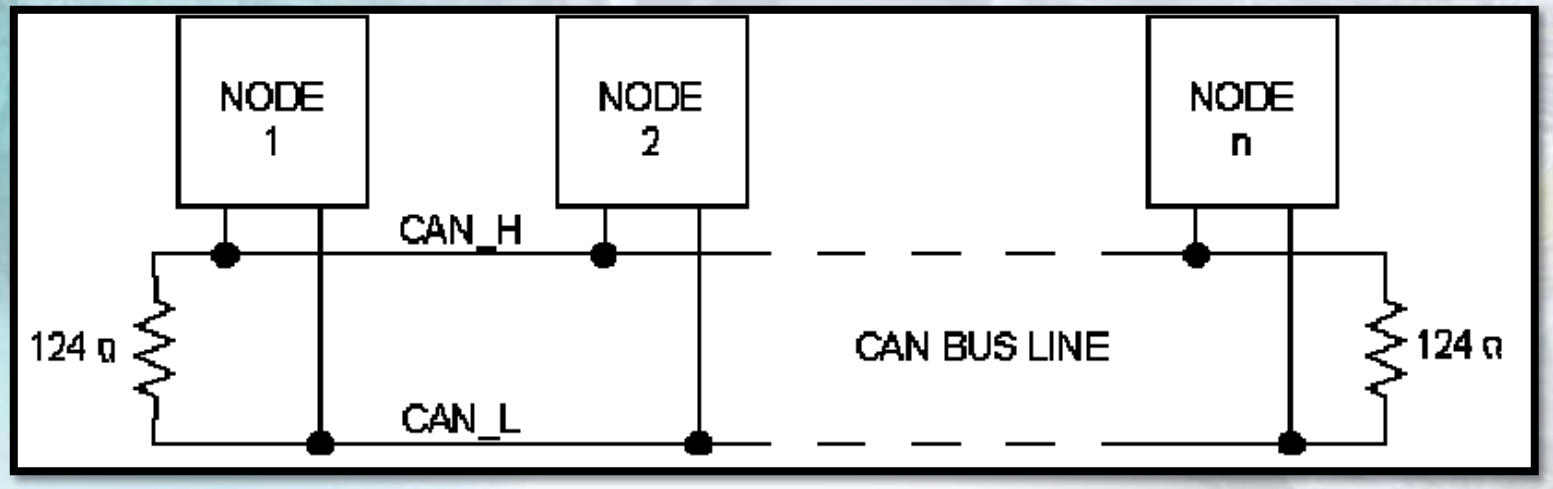
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

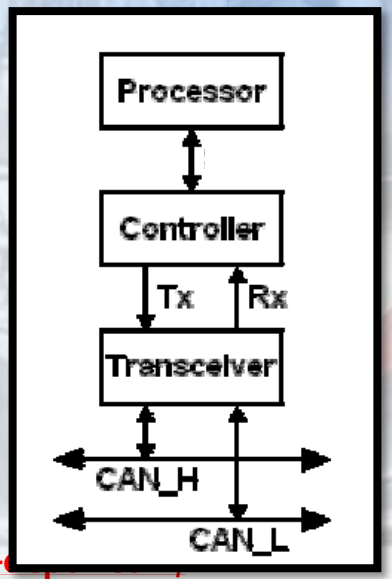
CAN Network



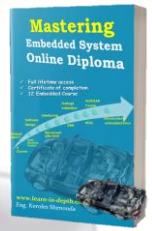


- **ISO11898** describes the physical-layer implementation of CAN. This specification describes a
  - twisted-wire pair bus with 120-Ω line impedance, and differential signaling at a rate up to 1 Mbps on a 40-meter bus with multipoint topology. Longer bus lengths are expected to have slower.





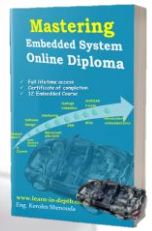
https://www.learn-in-
https://www.facebook.com/groups/embedded.system.KS/



CAN "Controller Area Network"

CAN
Controller Area Network

- Overview
- General Characteristics
- Types of CAN Messages
- CAN bus Characteristics
- CAN Bus Errors
  - Bit error
  - Stuff error
  - CRC error
  - Acknowledgment error
  - Form Error



CAN Error Detection And Fault Confinement

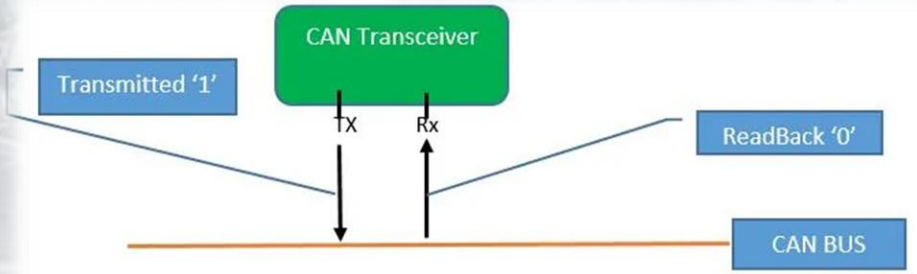
- There are five types of CAN error can introduce on CAN-BUS by receiver and transmitter node-

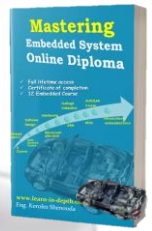
1. Bit error
2. Stuff error
3. CRC error
4. Acknowledgment error
5. Form Error

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Bit Error(Introduce By Transmitter)

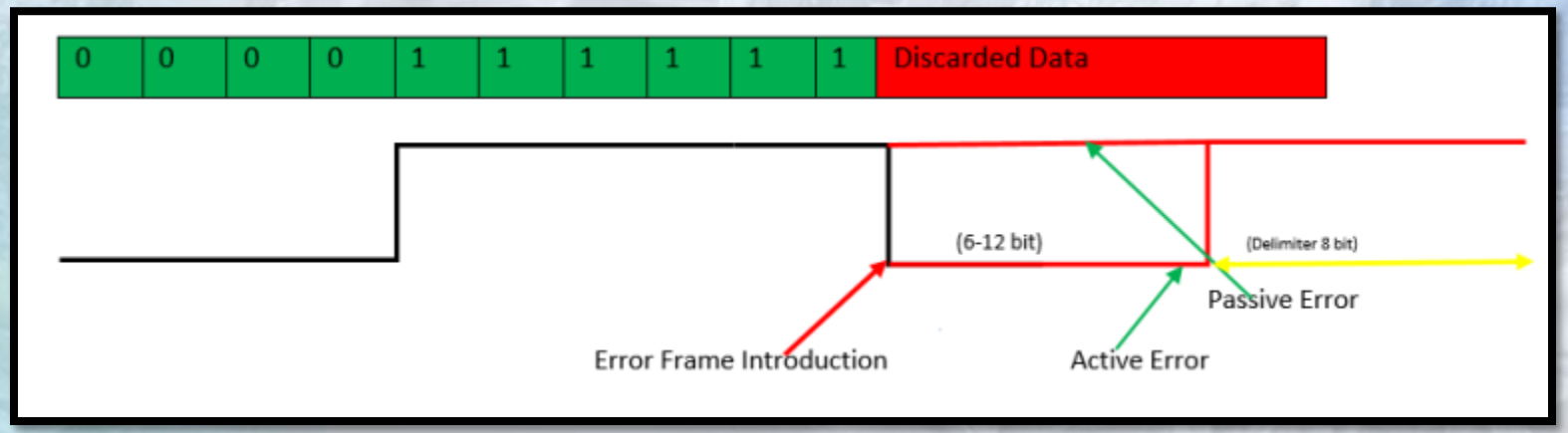
- Every transmitter node reads back its transmitted bit from CAN line along with other ECUs But if it will not get read bit same as it transmitted then transmitted node stop the further transmission and introduce an error frame on CAN line. This error is known as bit error.
- Note:
  - If **bit mismatch** **detected** **during Arbitration** field or **ACK** then transmitter **neither looks it as bit error nor introduce any error frame** because as per the CAN standard these fields has defined functionality.
  - **CRC Delimeter, ACK and EOF** are **also fixed length** fields in CAN frame so it will also not be the part of bit error frame execution if more than five consecutive bits found.



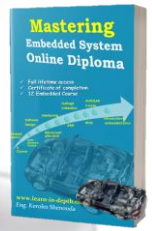


Stuff Error(Introduce By Receiver)

- **More than 5 consecutive** bits of the same polarity in CAN frame between the **start of Frame(SOF)** to **CRC field** is considered as a **faulty frame** on CAN Bus and it signaled as **stuff error** on CAN line.



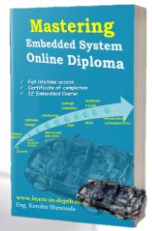
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



CRC Error(Introduce By Receiver)

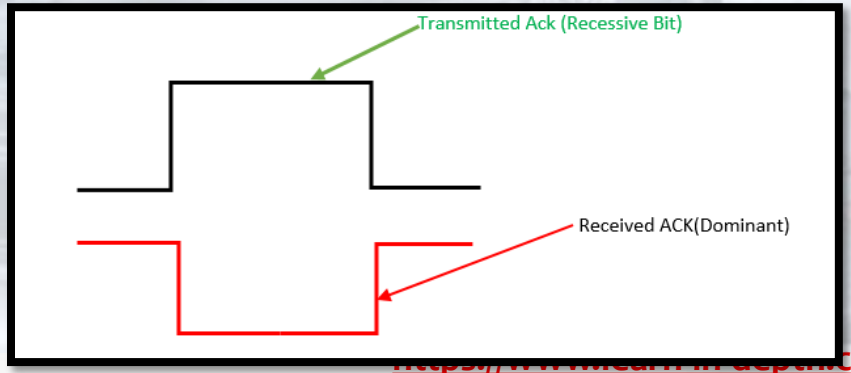
- The transmitter transmits CRC of transmitted data at CRC field of CAN frame
- and receivers also calculate CRC on received data.
- If the receiver found **Calculated CRC** is **different** from **received CRC** at CRC field then receiver signaled it as **CRC error** and introduce an **error frame on CAN line**.

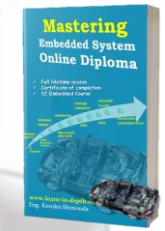
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



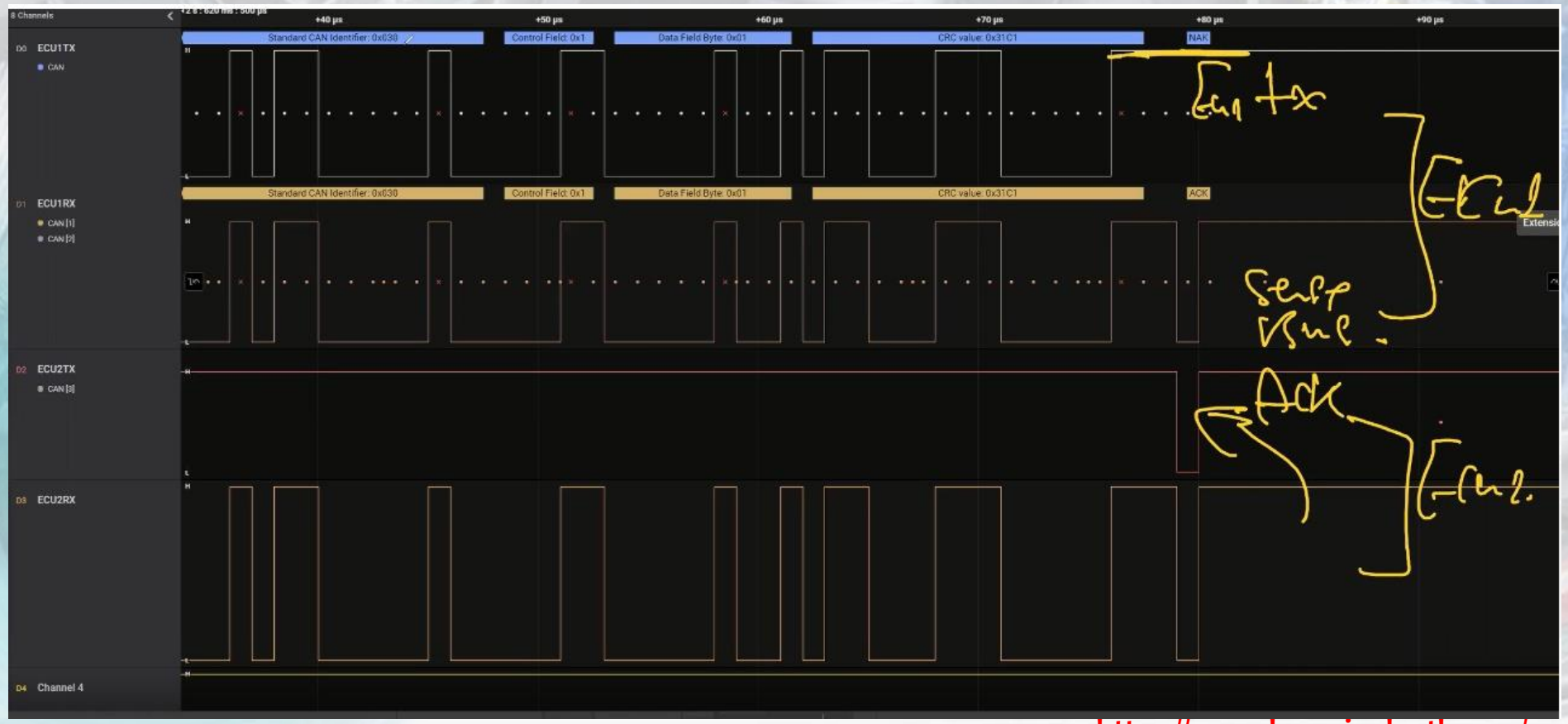
ACK Error(Introduce By Transmitter)

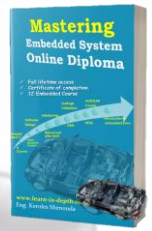
- After Transmission of CRC field of CAN frame
- Transmitter send ACK (a **recessive** bit) and **receiver** makes it **dominant** as a part of acknowledgment to the transmitter.
- During readback, transmitter found the **dominant** bit and consider it as **receiver acknowledgment** and
- if it reads **a recessive bit** then transmitter signaled it **as ACK error and introduce an error frame.**





Example

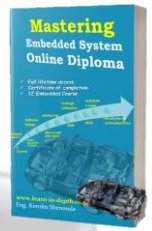




Form Error(Introduce By Receiver)

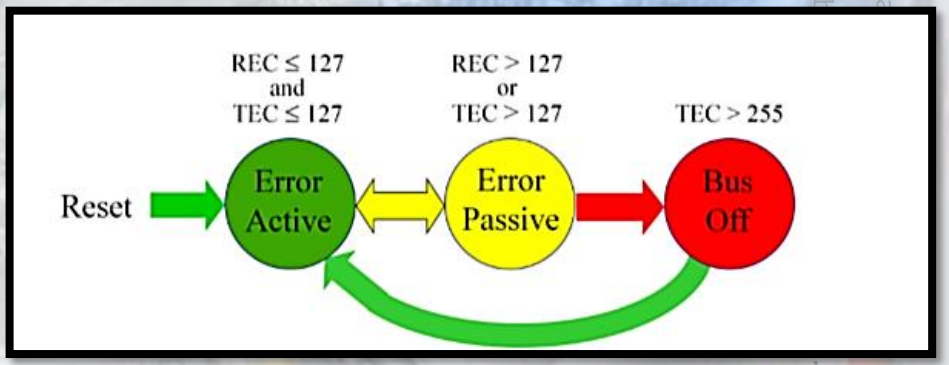
- As per CAN Frame Format, there are some fields of **fixed length** and **format** like CRC Delimeter, ACK Delimeter, EOF ,InterFrame Spaceand if it is **detected corrupted** at receiver side then it signaled **as Form Error** and Node will introduce **an error frame** on CAN line.

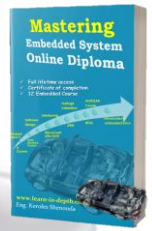
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Error Confinement Mechanism

- In CAN network Every node is very honest.
- if any node transmits/receive continuous faulty frame then **it will disconnect itself from CAN network after a threshold limit if fault not recovered.**
- To decide this **threshold** every Node has **two error counters**:
  - **Transmit error Counter(TEC)** and **Receive Error Counter(REC)** to keep the track of error generated because of above-discussed situations.
  - On the basis of these counter, Nodes are categories in three error states
- **Active Error state**(When TEC<=127 and REC<=127)
- **Passive Error state**(When TEC>127 or REC>127)
- **Bus-Off State**(When TEC>255)





Error Confinement Mechanism Cont.

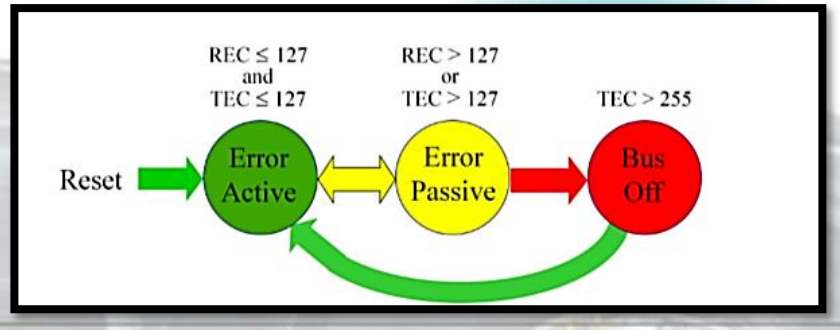
Rules for increasing and decreasing the error counters:

- Transmit errors give **8 error points**
- Receive errors give **1 error point**
- Correctly transmitted and/or received messages cause the counter(s) to **decrease**

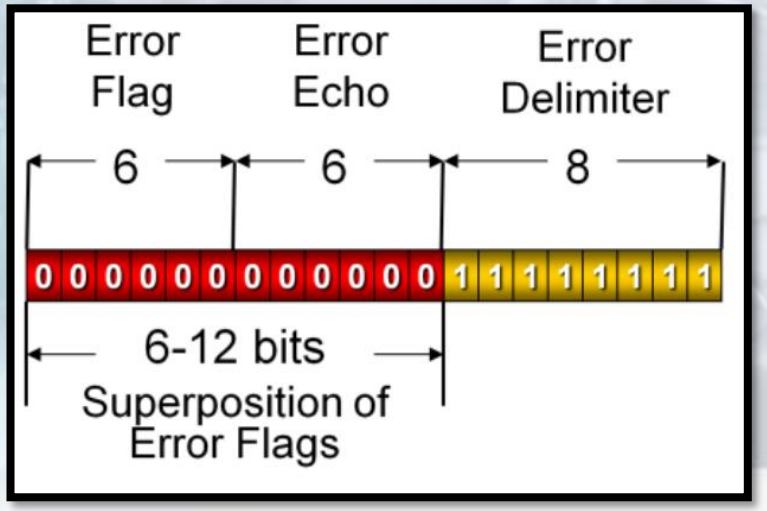
Transmit Error	Receive Error	Transmit/Receive correctly
+ 8	+ 1	-

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

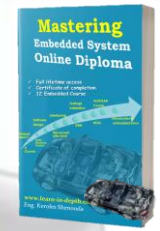
Active Error State



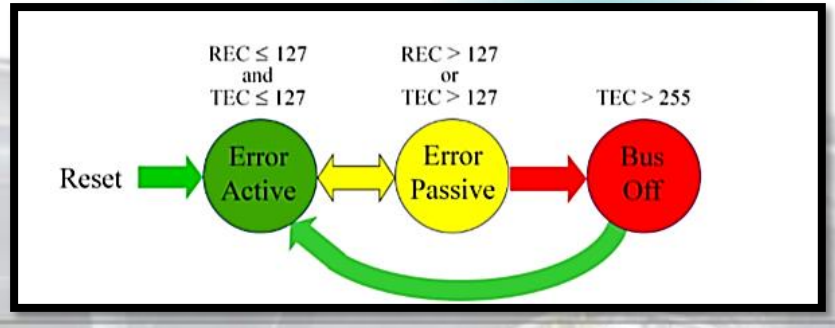
- A CAN Node enters into Active Error State when **TEC<=127** and **REC<=127**
- In This State Node **can participate in all CAN activity** like
- **A node can Transmit all type of Frames** like **Data Frame**, **Remote Frame**, **Overload Frame**, **and active Error Frame**(6-12 Dominant bit)



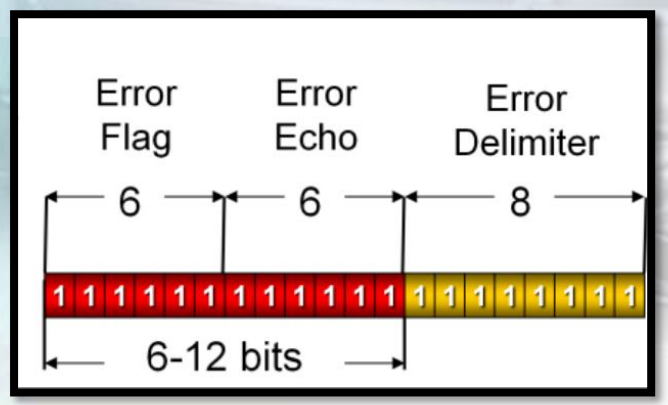
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

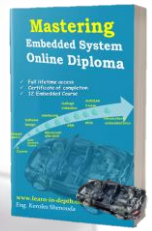


Passive Error State



- A CAN Node enters into Passive Error State when TEC>127 or REC>127.
- This state still has a chance to come back in **error-free state**.
- **In This State a Node still can participate in all CAN activity like**
- A node can Transmit all type of Frames like Data Frame, Remote Frame, Overload Frame, **but Passive Error Frame(6-12 Recessive bit)**
- A node can come back to in error active state if it starts transmission of the correct frames and its counter value comes under 127.



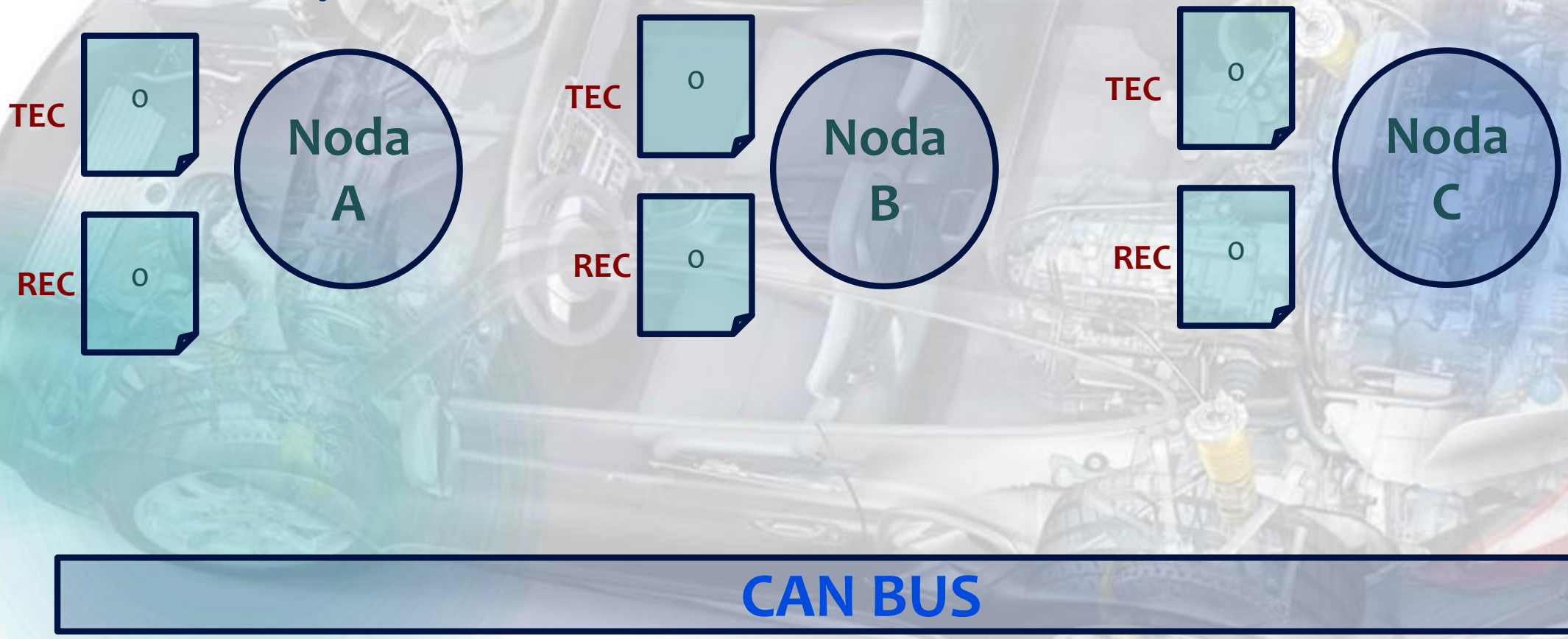


Bus-Off State

- A CAN Node enters into **Bus-Off** State When **TEC>255.**
- In this state **CAN node** will be **removed** from **CAN network** and now **it will no longer available on CAN network to participate in any CAN activity.**

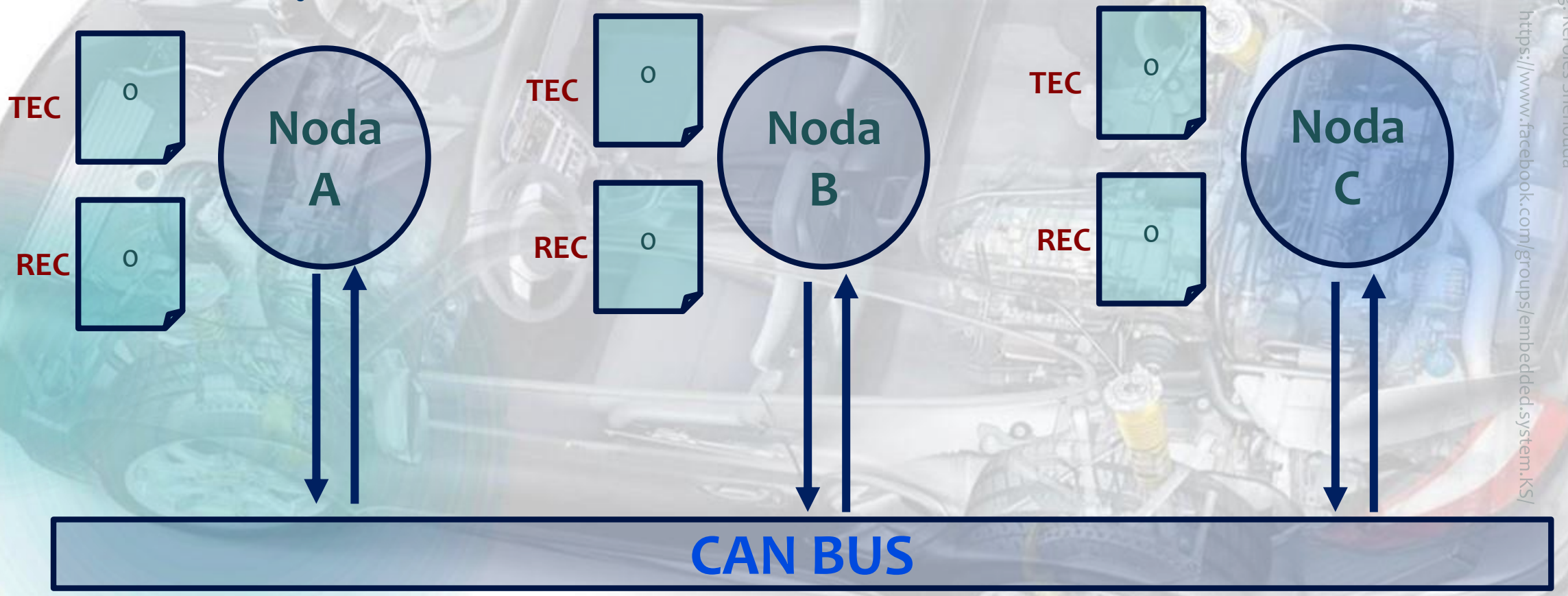
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Error Confinement Mechanism Example



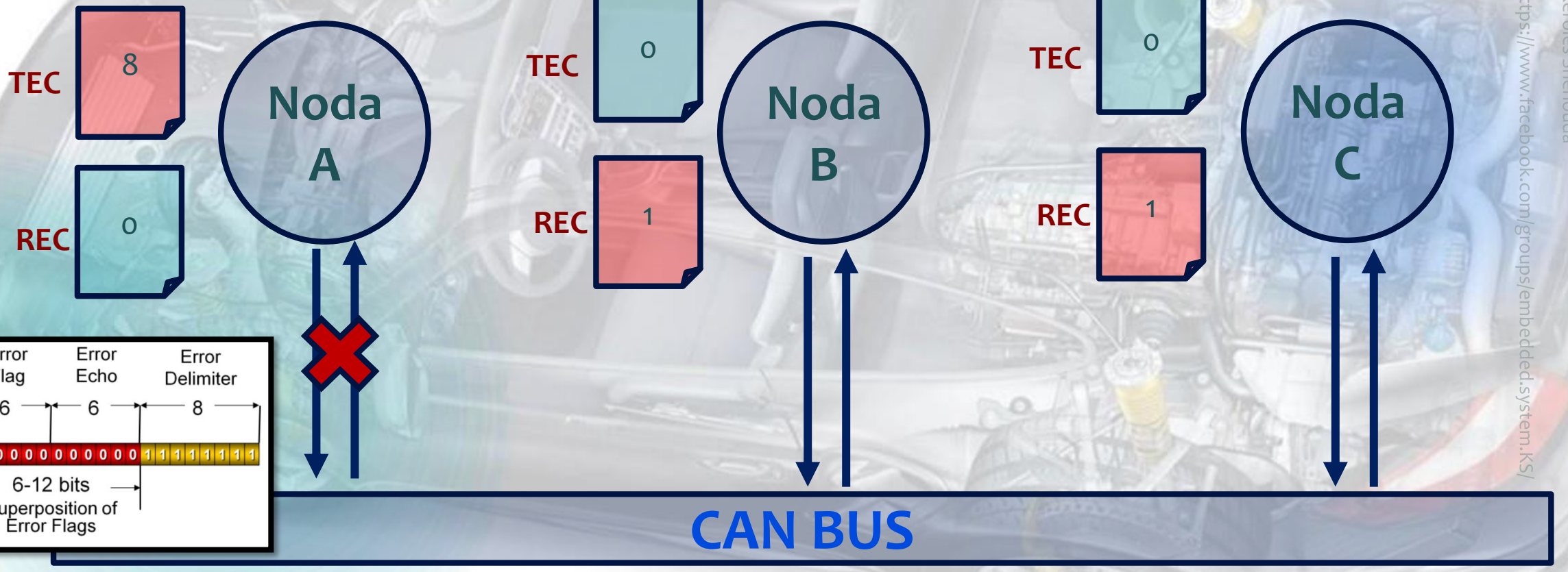
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

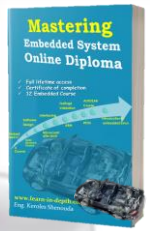
Error Confinement Mechanism Example



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Error Confinement Mechanism Example



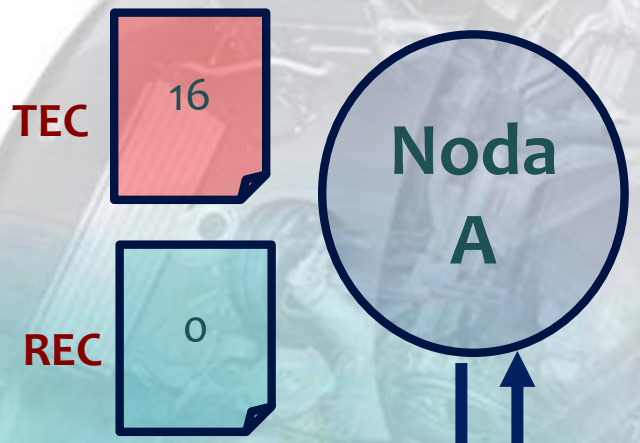


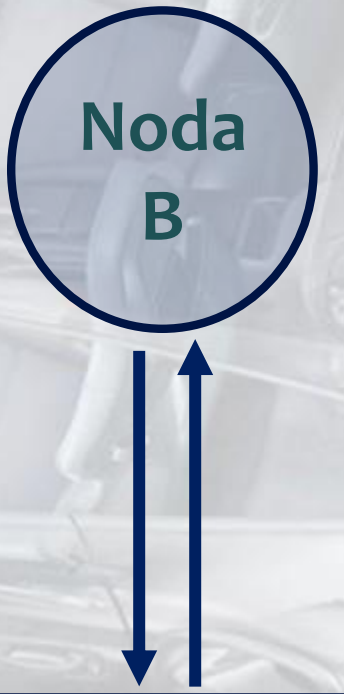
#LEARN_IN DEPTH

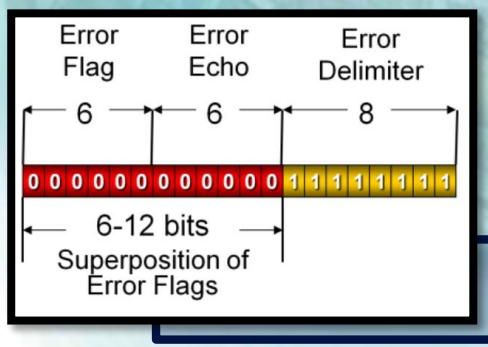
#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Error Confinement Mechanism Example



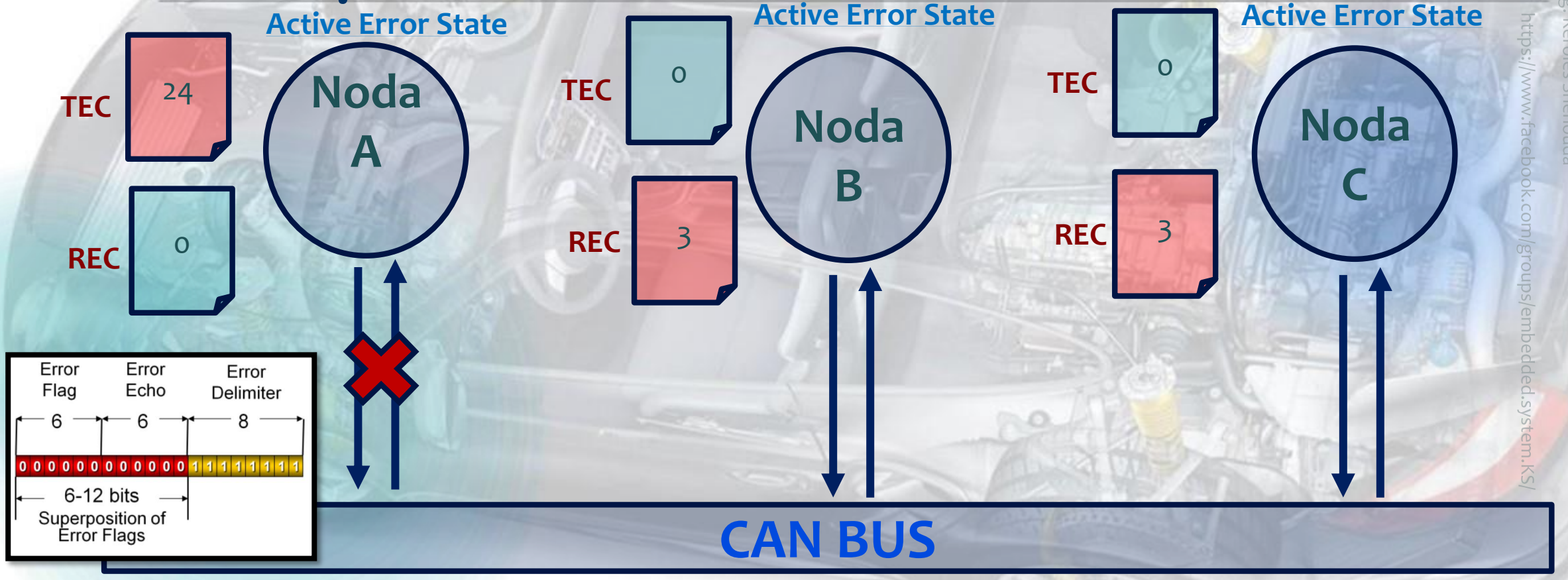




CAN BUS

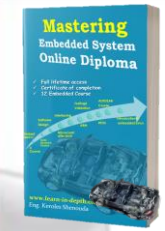
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

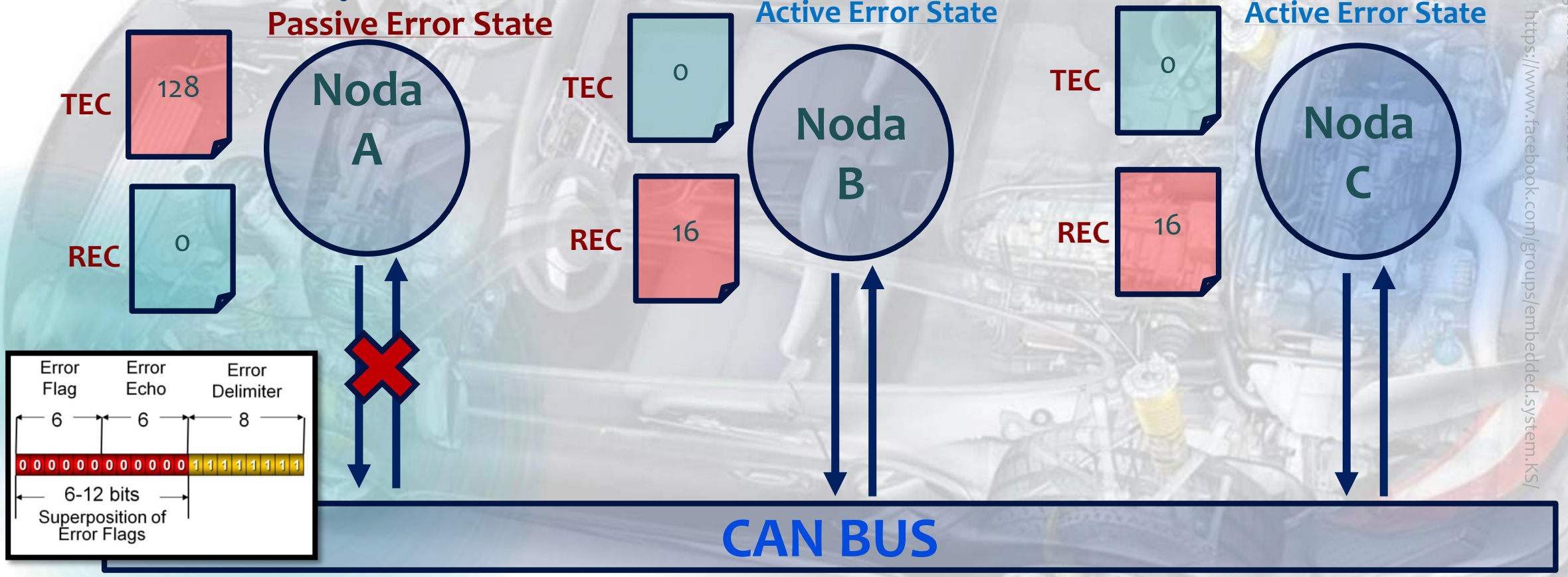
Error Confinement Mechanism Example



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

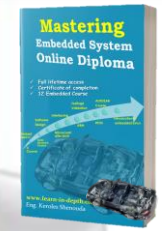
Error Confinement Mechanism Example

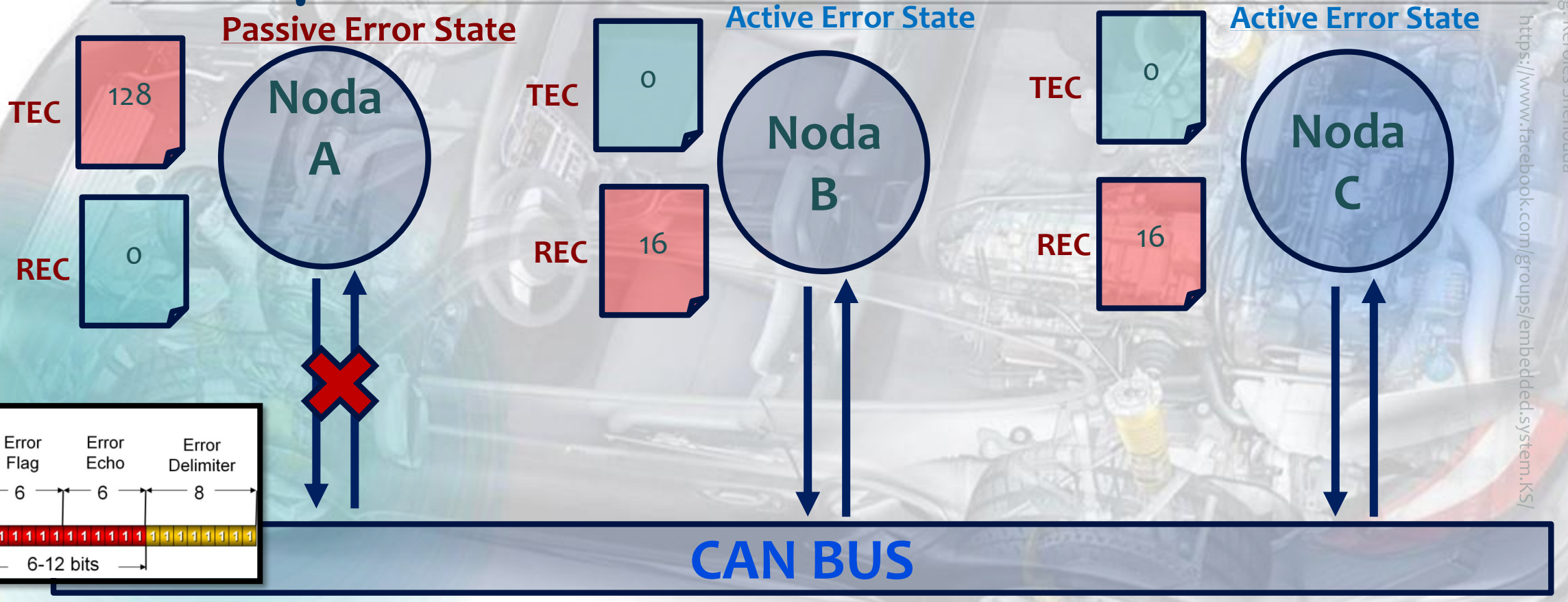




https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

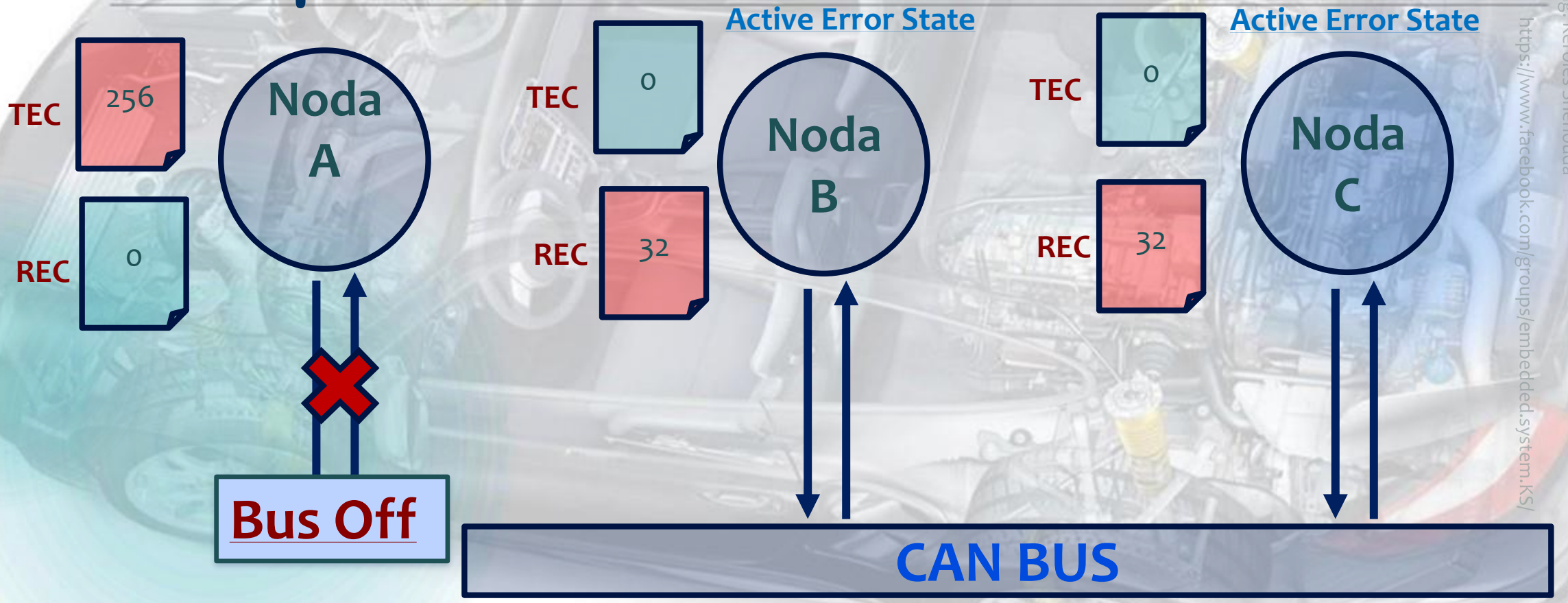
Error Confinement Mechanism Example



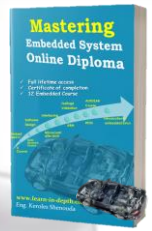


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

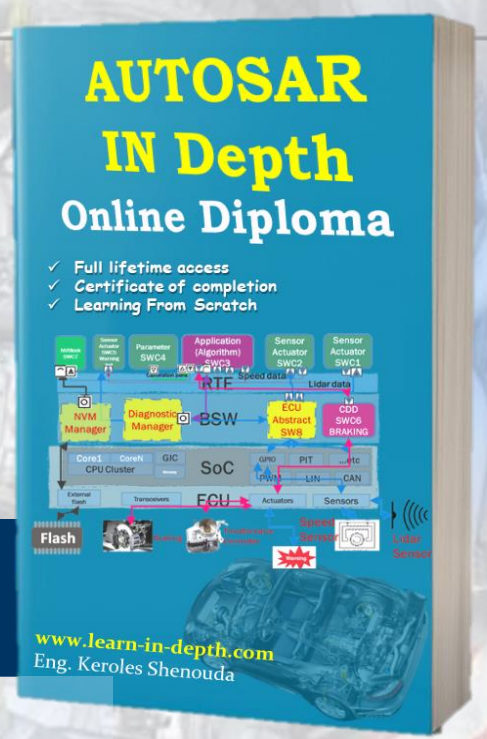
Error Confinement Mechanism Example



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

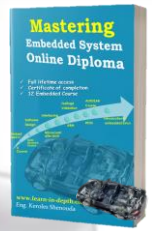


CAN Bit Monitoring and Stuffing



LEARN-IN-DEPTH
Be professional in embedded system

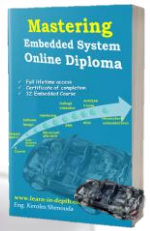
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Bit Monitoring

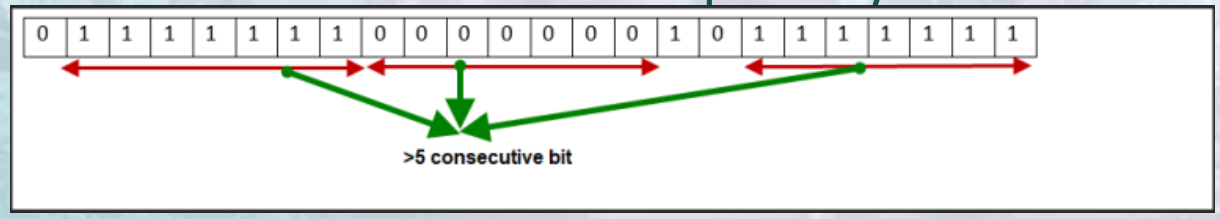
- Every transmitter reads back its transmitted bit from the CAN bus to ensure its transmitted data Integrity is **called Bit monitoring**.
- The concept of CAN bus arbitration is also decided by bit monitoring.
- when **two nodes start transmission** **at the same time** and read back its transmitted bit then one have to leave bus if it read something else what it has transmitted and thus other node wins the arbitration and continue its transmission **while looser node should wait for bus idle**
- Note: During Arbitration **no bit error** will happen in case of bit mismatch.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

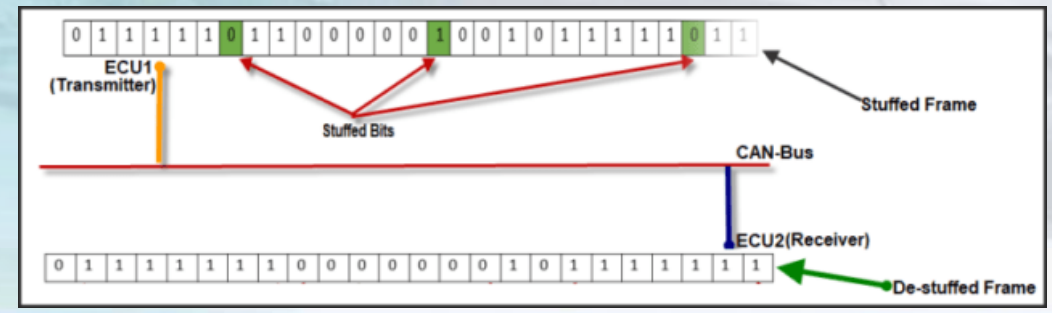


Bit Stuffing

- Insertion of **a bit of opposite polarity** **after five consecutive** bit of same polarity if that frame has more than five consecutive bits of same polarity is called bit stuffing.



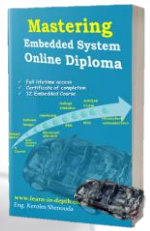
- while on receiver side these inserted bit is removed and received actual transmitted frame and it is called bit **de-stuffing**.



Note:

In a CAN frame **stuffing concept** will not apply on CRC delimiter, ACK and EOF fields because these fields have **fixed size**.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



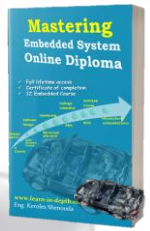
References

- https://www.softwaretestingmaterial.com/stlc-software-testing-life-cycle/
- https://www.amarinfotech.com/roles-responsibilities-agile-tester.html
- https://www.softwaretestinghelp.com/scrum-artifacts/
- https://www.visual-paradigm.com/scrum/what-are-scrum-ceremonies/
- **ISO 26262 – Software Advanced Training (Part II)**

  Dr. Julian Wolf TÜV SÜD Product Service GmbH

  http://documents.irevues.inist.fr/bitstream/handle/2042/56194/lm19_com_4D-5_062_A_Mihalache.pdf?sequence=1
- **ISO 26262-6 part 6**
  - Second edition 2018-12
- Verification and Validation: A Quick Introduction
  - https://slideplayer.com/slide/4564006/

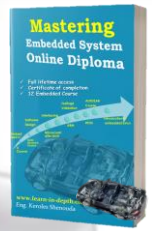
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://www.starkinfotech.com/manual-testing/
- https://www.embitel.com/blog/embedded-blog/understanding-how-iso-26262-asil-is-determined-for-automotive-applications
- Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model based Development
  - https://www.scitepress.org/papers/2013/45920/45920.pdf
- https://www.techdesignforums.com/practice/technique/formal-fault-analysis-for-iso26262-find-faults-before-they-find-you/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



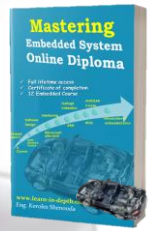
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- **Microprocessor: ATmega32 (SEE3223-10)**
  http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- **http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller**
- **http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller**
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

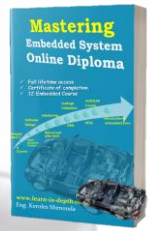
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://techdifferences.com/difference-between-interrupt-and-polling-in-os.html
- http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
- Preventing Interrupt Overload Presented by Jiyong Park Seoul National University, Korea 2005. 2. 22. John Regehr, Usit Duogsaa, School of Computing, University.
- First Steps Embedded Systems Byte Craft Limited reference
- COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION **William Stallings**
- **Getting Started with the Tiva™ TM4C123G LaunchPad Workshop**

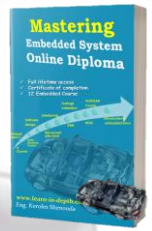
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- Tiva™ TM4C123GH6PM Microcontroller DATA SHEET
- Interrupts and Exceptions COMS W6998 Spring 2010
- THE **AVR MICROCONTROLLER**. AND EMBEDDED SYSTEMS Using Assembly and C. Muhammad Ali **Mazidi**.
- http://embedded-lab.com/blog/tinkering-ti-msp430f5529/27/
- How to use JIRA (AR)

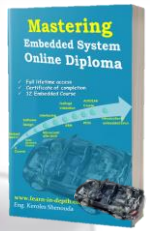
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology) 1st Edition https://www.amazon.com/AVR-Microcontroller-Embedded-Systems-Electronics/dp/0138003319

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



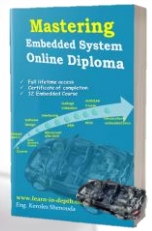
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

References

- https://www.newbiehack.com/MicrocontrollersABeginnersGuideIntroductionandInterfacinganLCD.aspx
- http://www.slideshare.net/MathivananNatarajan/asynchronous-serial-data-communication-and-standards
- https://www.slideshare.net/AnkitSingh13/uart-32550652
- **the avr** microcontroller and embedded. System using assembly and c. Muhammad Ali **Mazidi**
- Embedded Systems lectures "Engr. Rashid Farid Chishti"
- https://www.corelis.com/education/SPI_Tutorial.htm
- http://ftm.futureelectronics.com/2014/09/nxp-macronics-nor-series-quad-spi-flash-a-simpler-faster-alternative-to-standard-spi-flash-when-adding-external-memory-to-32-bit-mcu-systems/
- http://www.byteparadigm.com/products/spi-storm/spi-storm-advanced-information/
- https://stackoverflow.com/questions/17125505/what-makes-a-better-constant-in-c-a-macro-or-an-enum
- https://blog.digilentinc.com/i2c-how-does-it-work/#prettyPhoto
- https://rophoenixmakerevolution.files.wordpress.com/2015/09/spi-and-can-bus.pdf
- http://denethor.wlu.ca/cp316/lectures/Serial_Interconnect_Bus.pdf
- https://aticleworld.com/i2c-interview-questions/
- https://www.youtube.com/watch?v=7MZ-a-unAU8

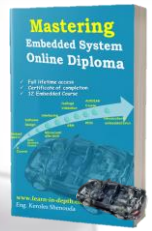
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- https://www.autosar.org
- Embedded Microcomputer Systems Real Time Interfacing *Third Edition* Jonathan W. Valvano University of Texas at Austin.
- MicroC/OS-II the real-time kernel second edition jean j.labrosse.
- RTOS Concepts http://www.embeddedcraft.org.
- OSEK/VDX Operating System Specification 2.2.3
- AUTOSAR Layered Software Architecture
- The Trampoline Handbook release 2.0
- Trampoline (OSEK/VDX OS) Test Implementation -Version 1.0, Florent PAVIN ; Jean-Luc BECHENNEC
- Autosar Architecture (Learn from Scratch with Demo) **Udemy Course**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- **Trampoline**:an open platform for (small) embedded systems based on OSEK/VDX and AUTOSAR

http://trampoline.rts-software.org/

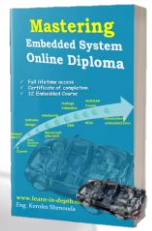
Jean-Luc Béchennec1;2, **Sébastien Faucou**1;3

1IRCCyN (Institute of Research in Communications and Cybernetics of Nantes)

2CNRS (National Center for Scientific Research) / 3University of Nantes

10th Libre Software Meeting

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- Real Time Systems (RETSY)
  Jean-Luc Béchennec - Jean-Luc.Bechennec@irccyn.ec-nantes.fr
  Sébastien Faucou - Sebastien.Faucou@univ-nantes.fr
  jeudi 12 novembre 15
- AUTOSAR Specification of Operating System V5.0.0 R4.0 Rev 3
- **OSEK - Basics** http://taisnotes.blogspot.com.eg/2016/07/osek-basic-task-vs-extended-task.html
- OSEK OS Session Speaker Deepak V.

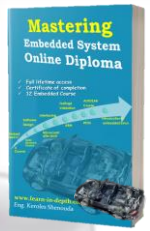
M.S Ramaiah School of Advanced Studies - Bangalore 1

- **Introducción a OSEK-OS - El Sistema Operativo del CIAA-Firmware**

Programación de Sistemas Embebidos

MSc. Ing. Mariano Cerdeiro

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

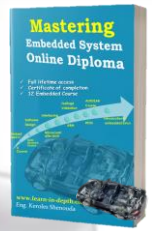
- **Introduction to AUTOSAR, Stephen Waldron, Vector webinar** Wednesday 7th May 2014

https://vector.com/portal/medien/cmc/events/Webinars/2014/Vector_Webinar_AUTOSAR_Introduction_20140507_EN.pdf

- Introduction to AUTOSAR, Stephen Waldron, **Vector webinar** Tuesday 5th May 2015

https://vector.com/portal/medien/cmc/events/Webinars/2015/Vector_Webinar_AUTOSAR_Introduction_20150505_EN.pdf

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



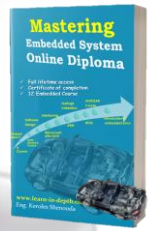
References

- Applying AUTOSAR in Practice *Available Development Tools and Migration Paths Master Thesis, Computer Science Authors*: Jesper Melin

http://www.idt.mdh.se/utbildning/exjobb/files/TR1171.pdf

Freescale AUTOSAR Software Overview.pdf

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- AUTOSAR Method, Vector Webinar 2013-04-17

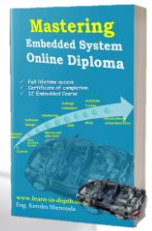
https://vector.com/portal/medien/cmc/events/Webinars/2013/Vector_Webinar_AUTOSAR_Method_20130417.pdf

- AUTOSAR Configuration Process - How to handle 1000s of parameters Vector Webinar 2013-04-19

https://vector.com/portal/medien/cmc/events/Webinars/2013/Vector_Webinar_AUTOSAR_Configuration_Process_20130419_EN.pdf

- AUTOSAR Runtime Environment and Virtual Function Bus, Nico Naumann

https://hpi.de/fileadmin/user_upload/fachgebiete/giese/Ausarbeitungen_AUTOSAR0809/NicoNaumann_RTE_VFB.pdf



References

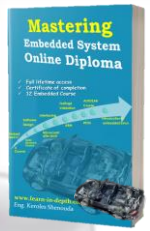
- **The AUTOSAR Adaptive Platform for Connected and Autonomous Vehicles,** Simon Fürst, AUTOSAR Steering Committee 8th Vector Congress 29-Nov-2016, Alte Stuttgarter Reithalle, Stuttgart, Germany

https://vector.com/congress/files/presentations/VeCo16_06_29Nov_Reithalle_Fuerst_BMW.pdf

- A Review of Embedded Automotive Protocols, Nicolas Navet1, Françoise Simonot-Lion2 April 14, 2008

https://www.realtimeatwork.com/wp-content/uploads/chapter4_CRC_2008.pdf

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- AUTOSAR Adaptive Platform

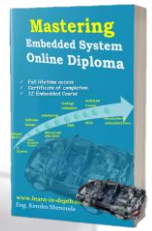
https://vector.com/conference_india/files/presentations/Day1/3_AUTOSAR%20Adaptive%20Platform.pdf

- AUTOAR Specification of Diagnostic Communication Manager

https://www.autosar.org/fileadmin/user_upload/standards/classic/3-1/AUTOSAR_SWS_DCM.pdf

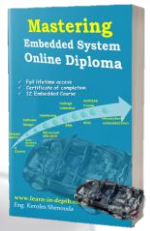
- **Automotive & Embedded Info**

https://automotiveembeddedsite.wordpress.com/memory-stack/



References

- http://www.autosar.org/about/technical-overview/ecu-software-architecture/autosar-basic-software/
- http://www.autosar.org/standards/classic-platform/
- https://automotivetechis.files.wordpress.com/2012/05/communicationstack_gosda.pdf
- https://automotivetechis.files.wordpress.com/2012/05/autosar_ppt.pdf
- https://automotivetechis.wordpress.com/autosar-concepts/
- https://automotivetechis.files.wordpress.com/2012/05/autosar_exp_layeredsoftwarearchitecture.pdf
- http://www.slideshare.net/FarzadSadeghi1/autosar-software-component
- https://www.renesas.com/en-us/solutions/automotive/technology/autosar/autosar-mcal.html
- https://github.com/parai/OpenSAR/blob/master/include/Std_Types.h



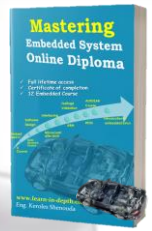
References

- https://www.softwaretestingmaterial.com/stlc-software-testing-life-cycle/
- https://www.amarinfotech.com/roles-responsibilities-agile-tester.html
- https://www.softwaretestinghelp.com/scrum-artifacts/
- https://www.visual-paradigm.com/scrum/what-are-scrum-ceremonies/
- **ISO 26262 – Software Advanced Training (Part II)**

  Dr. Julian Wolf TÜV SÜD Product Service GmbH

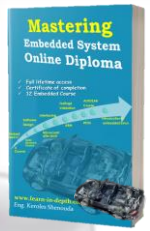
  http://documents.irevues.inist.fr/bitstream/handle/2042/56194/lm19_com_4D-5_062_A_Mihalache.pdf?sequence=1
- **ISO 26262-6 part 6**
  - Second edition 2018-12
- Verification and Validation: A Quick Introduction
  - https://slideplayer.com/slide/4564006/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://www.starkinfotech.com/manual-testing/
- https://www.embitel.com/blog/embedded-blog/understanding-how-iso-26262-asil-is-determined-for-automotive-applications
- Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model based Development
  - https://www.scitepress.org/papers/2013/45920/45920.pdf
- https://www.techdesignforums.com/practice/technique/formal-fault-analysis-for-iso26262-find-faults-before-they-find-you/



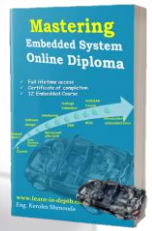
#LEARN_IN DEPTH

#Be_professional_in embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- **Microprocessor: ATmega32 (SEE3223-10)**
  http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- **http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller**
- **http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller**
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

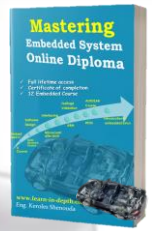
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

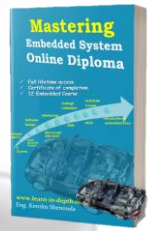
- http://techdifferences.com/difference-between-interrupt-and-polling-in-os.html
- http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
- Preventing Interrupt Overload Presented by Jiyong Park Seoul National University, Korea 2005. 2. 22. John Regehr, Usit Duogsaa, School of Computing, University.
- First Steps Embedded Systems Byte Craft Limited reference
- COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION **William Stallings**
- **Getting Started with the Tiva™ TM4C123G LaunchPad Workshop**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

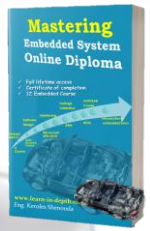
- Tiva™ TM4C123GH6PM Microcontroller DATA SHEET
- Interrupts and Exceptions COMS W6998 Spring 2010
- THE **AVR MICROCONTROLLER**. AND EMBEDDED SYSTEMS Using Assembly and C. Muhammad Ali **Mazidi**.
- http://embedded-lab.com/blog/tinkering-ti-msp430f5529/27/
- How to use JIRA (AR)



References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology) 1st Edition https://www.amazon.com/AVR-Microcontroller-Embedded-Systems-Electronics/dp/0138003319

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



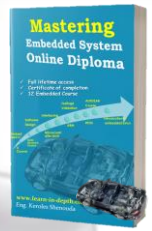
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

References

- https://www.newbiehack.com/MicrocontrollersABeginnersGuideIntroductionandInterfacinganLCD.aspx
- http://www.slideshare.net/MathivananNatarajan/asynchronous-serial-data-communication-and-standards
- https://www.slideshare.net/AnkitSingh13/uart-32550652
- **the avr** microcontroller and embedded. System using assembly and c. Muhammad Ali **Mazidi**
- Embedded Systems lectures "Engr. Rashid Farid Chishti"
- https://www.corelis.com/education/SPI_Tutorial.htm
- http://ftm.futureelectronics.com/2014/09/nxp-macronics-nor-series-quad-spi-flash-a-simpler-faster-alternative-to-standard-spi-flash-when-adding-external-memory-to-32-bit-mcu-systems/
- http://www.byteparadigm.com/products/spi-storm/spi-storm-advanced-information/
- https://stackoverflow.com/questions/17125505/what-makes-a-better-constant-in-c-a-macro-or-an-enum
- https://blog.digilentinc.com/i2c-how-does-it-work/#prettyPhoto
- https://rophoenixmakerevolution.files.wordpress.com/2015/09/spi-and-can-bus.pdf
- http://denethor.wlu.ca/cp316/lectures/Serial_Interconnect_Bus.pdf
- https://aticleworld.com/i2c-interview-questions/
- https://www.youtube.com/watch?v=7MZ-a-unAU8

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- https://www.autosar.org
- Embedded Microcomputer Systems Real Time Interfacing *Third Edition* Jonathan W. Valvano University of Texas at Austin.
- MicroC/OS-II the real-time kernel second edition jean j.labrosse.
- RTOS Concepts http://www.embeddedcraft.org.
- OSEK/VDX Operating System Specification 2.2.3
- AUTOSAR Layered Software Architecture
- The Trampoline Handbook release 2.0
- Trampoline (OSEK/VDX OS) Test Implementation -Version 1.0, Florent PAVIN ; Jean-Luc BECHENNEC

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

References

- **Trampoline**:an open platform for (small) embedded systems based on OSEK/VDX and AUTOSAR

http://trampoline.rts-software.org/

Jean-Luc Béchennec1;2, **Sébastien Faucou**1;3

1IRCCyN (Institute of Research in Communications and Cybernetics of Nantes)

2CNRS (National Center for Scientific Research) / 3University of Nantes

10th Libre Software Meeting

References

- Real Time Systems (RETSY)
  Jean-Luc Béchennec - Jean-Luc.Bechennec@irccyn.ec-nantes.fr
  Sébastien Faucou - Sebastien.Faucou@univ-nantes.fr
  jeudi 12 novembre 15
- AUTOSAR Specification of Operating System V5.0.0 R4.0 Rev 3
- **OSEK - Basics** http://taisnotes.blogspot.com.eg/2016/07/osek-basic-task-vs-extended-task.html
- OSEK OS Session Speaker Deepak V.

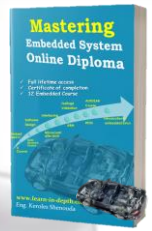
M.S Ramaiah School of Advanced Studies - Bangalore 1

- **Introducción a OSEK-OS - El Sistema Operativo del CIAA-Firmware**

Programación de Sistemas Embebidos

MSc. Ing. Mariano Cerdeiro

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

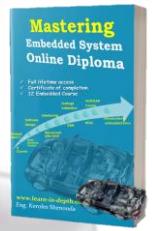
- **Introduction to AUTOSAR, Stephen Waldron, Vector webinar** Wednesday 7th May 2014

https://vector.com/portal/medien/cmc/events/Webinars/2014/Vector_Webinar_AUTOSAR_Introduction_20140507_EN.pdf

- Introduction to AUTOSAR, Stephen Waldron, **Vector webinar** Tuesday 5th May 2015

https://vector.com/portal/medien/cmc/events/Webinars/2015/Vector_Webinar_AUTOSAR_Introduction_20150505_EN.pdf

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



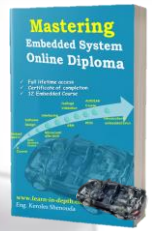
References

- Automatic Generation of AUTOSAR Software Component Descriptions

Study Thesis in Computer Sciences by Christopher Mutschler

https://www2.informatik.uni-erlangen.de/EN/teaching/thesis/download/i2S00428.pdf

- AutoSAR Overview FESA Workshop at KTH 2010-04-12
  - Prof. Jakob Axelsson
  - http://www.artist-embedded.org/docs/Events/2010/FESA/slides/3_Autosar_Axelsson.pdf



References

- AUTOSAR – An open standardized software architecture for the automotive industry Simon Fürst, BMW 1st AUTOSAR Open Conference & 8th AUTOSAR Premium Member Conference October 23rd, 2008, Cobo Center, Detroit, MI, USA
  - http://st.inf.tu-dresden.de/files/teaching/ws08/ase/03_AUTOSAR_Tutorial.pdf
- Institutionen för systemteknik Department of Electrical Engineering Examensarbete Implementation of CAN Communication Stack in AUTOSAR Examensarbete utfört i Datorteknik

vid Tekniska högskolan vid Linköpings universitet Av Johan Alexandersson och Olle Nordin

http://liu.diva-portal.org/smash/get/diva2:822343/FULLTEXT01.pdf

References

- Applying AUTOSAR in Practice *Available Development Tools and Migration Paths Master Thesis, Computer Science Authors:* Jesper Melin

http://www.idt.mdh.se/utbildning/exjobb/files/TR1171.pdf

Freescale AUTOSAR Software Overview.pdf

- ***Introduction to the Controller Area Network (CAN)***

TEXAS INSTRUMENTS

- ***CAN Learning From Vector***
  - https://elearning.vector.com/index.php?&wbt_ls_seite_id=490352&root=378422&seite=vl_can_introduction_en

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

References

- AUTOSAR Method, Vector Webinar 2013-04-17

https://vector.com/portal/medien/cmc/events/Webinars/2013/Vector_Webinar_AUTOSAR_Method_20130417.pdf

- AUTOSAR Configuration Process - How to handle 1000s of parameters Vector Webinar 2013-04-19

https://vector.com/portal/medien/cmc/events/Webinars/2013/Vector_Webinar_AUTOSAR_Configuration_Process_20130419_EN.pdf

- AUTOSAR Runtime Environment and Virtual Function Bus, Nico Naumann

https://hpi.de/fileadmin/user_upload/fachgebiete/giese/Ausarbeitungen_AUTOSAR0809/NicoNaumann_RTE_VFB.pdf

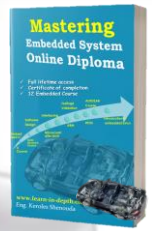
References

- **The AUTOSAR Adaptive Platform for Connected and Autonomous Vehicles,** Simon Fürst, AUTOSAR Steering Committee 8th Vector Congress 29-Nov-2016, Alte Stuttgarter Reithalle, Stuttgart, Germany

https://vector.com/congress/files/presentations/VeCo16_06_29Nov_Reithalle_Fuerst_BMW.pdf

- A Review of Embedded Automotive Protocols, Nicolas Navet1, Françoise Simonot-Lion2 April 14, 2008

https://www.realtimeatwork.com/wp-content/uploads/chapter4_CRC_2008.pdf



References

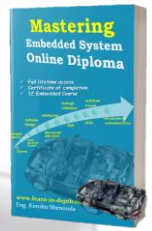
- AUTOSAR Adaptive Platform

https://vector.com/conference_india/files/presentations/Day1/3_AUTOSAR%20Adaptive%20Platform.pdf

- CAN Bus Protocol
  - By Abhinaw Tiwari CSE-12010330
- Basics of In-Vehicle Networking (IVN) Protocols

ON Semiconductor®

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://www.autosar.org/about/technical-overview/ecu-software-architecture/autosar-basic-software/
- http://www.autosar.org/standards/classic-platform/
- https://automotivetechis.files.wordpress.com/2012/05/communicationstack_gosda.pdf
- https://automotivetechis.files.wordpress.com/2012/05/autosar_ppt.pdf
- https://automotivetechis.wordpress.com/autosar-concepts/
- https://automotivetechis.files.wordpress.com/2012/05/autosar_exp_layeredsoftwarearchitecture.pdf
- http://www.slideshare.net/FarzadSadeghi1/autosar-software-component
- https://www.renesas.com/en-us/solutions/automotive/technology/autosar/autosar-mcal.html
- https://github.com/parai/OpenSAR/blob/master/include/Std_Types.h

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/