

# CAN Protocol Basics

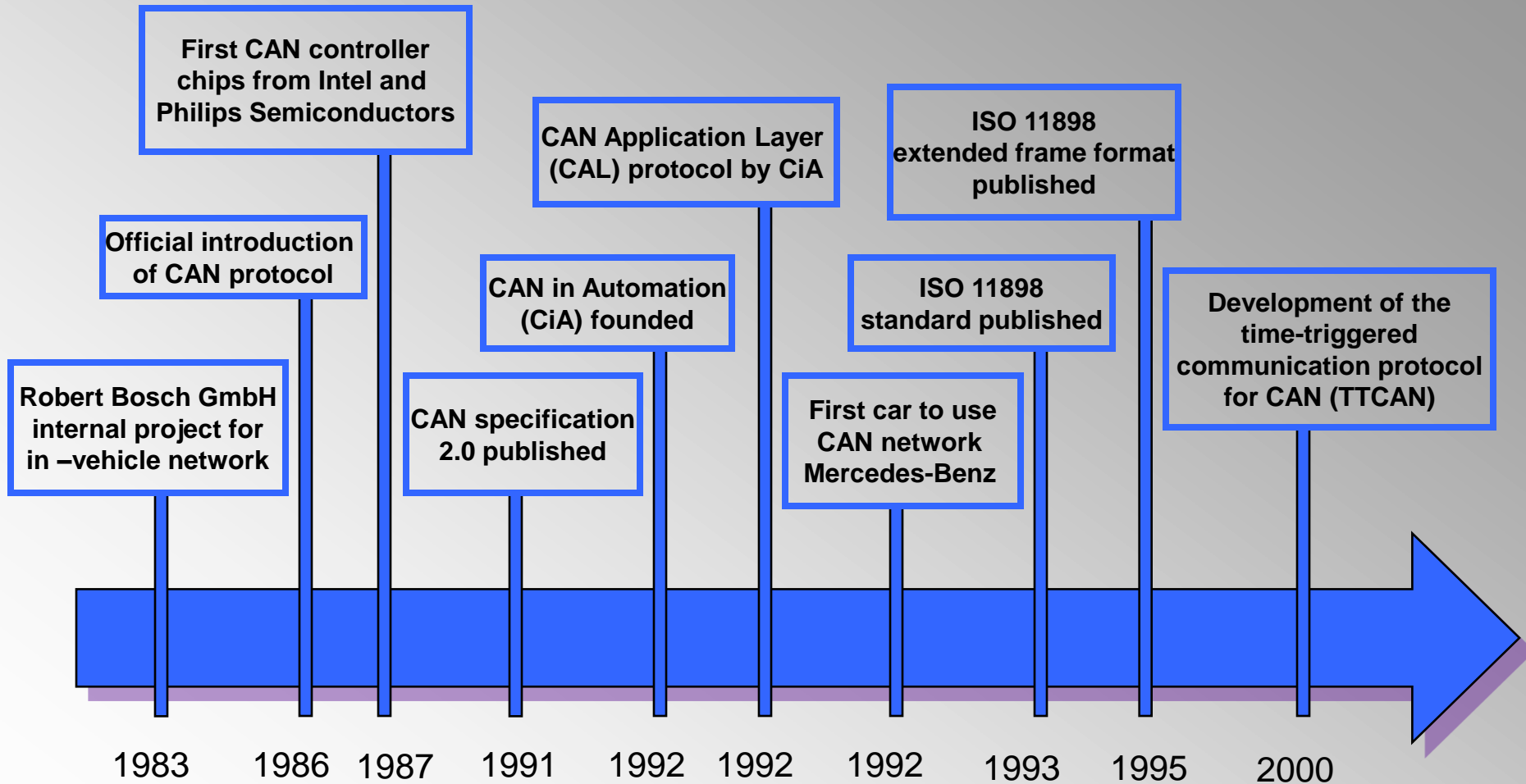
Theoretical Presentation

## Summary

---

- Protocol Description
- CAN in OSI
- Frames definition
- CAN errors
- Hardware implications
- Communication layers

# CAN History Timeline



# Application examples for the CAN Bus



*Construction Equipment*



*Domestic Appliances*



*Ships & Boats*

The many uses of  
**CAN**



*Automotive Sector*

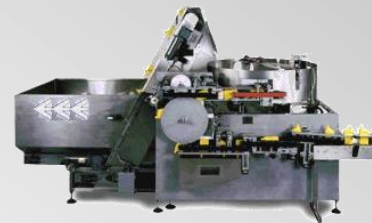


*Medical Equipment*

*Building Automation*



*Trains*



*Industry Automation*

# CAN and other protocols

Name	Usage	Topology	Type	Bitrate
<b>LIN</b> (Local Interconnect Network )	Low level communication	Bus	Single - Master	20 [KBit/s]
<b>CAN</b> (Controller Area Network)	Hard real time systems	Bus	Multi - Master	500[KBit/s]
<b>TTTCAN</b> (time-triggered communication on CAN )	Hard real time systems	Bus	Multi - Master	500[KBit/s]
<b>Bluetooth</b>	Hard real time systems	1:1, Multi point	Single - Master	1[MBit/s]
<b>FlexRay</b>	Hard real time systems	Bus/Star	Multi - Master	10[MBit/s]
<b>Byteflight</b>	Hard real time systems	Star	Single - Master	10[MBit/s]
<b>D2B</b> (domestic digital data bus)	Hard real time systems	Ring/Star	Single - Master	20[MBit/s]
<b>TTP/C</b> (time-triggered communication protocol )	Hard real time systems	Bus/Star	Multi - Master	25[MBit/s]
<b>MOST</b> (Media Oriented Systems Transport)	Hard real time systems	Ring/Star	Multi - Master	25[MBit/s]
<b>IntelliBus</b>	Hard real time systems	Bus/Star	Single/Multi - Master	25[MBit/s]
<b>IDB-1394</b>	Hard real time systems	Bus/Tree	Single - Master	100[MBit/s]

## Examples for CAN in automotive

500-1000 [Kbit/s]

- Engine
- Brakes
- Gearbox



25-125 [Kbit/s]

- Airbag
- Central locking system
- Air conditioning
- Lights

- Development by Bosch and Intel end of 80<sup>th</sup>
- Today de facto standard in automotive industry worldwide
- Use of CAN from nearly all known car manufacturer

## Bus characteristics

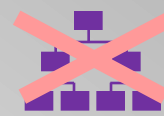
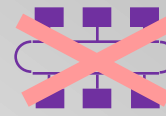
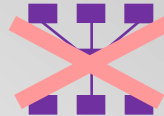
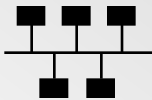
---

### ➤ Serial data communications bus

Inexpensive and simple, but slower than parallel bus.

### ➤ Linear bus structure

No star structure, no ring structure, no tree structure !



### ➤ Sensor / actor bus

Every participant is equal to all others from the protocol point of view  
=> peer-to-peer data transmission

Example: rain sensor = **sensor**, windscreen wipers = **actor**, both connected via the same bus and communicating the same way.

---

## Data rates

---

### ➤ “Good” real-time capabilities

Small latency (“fast enough”)

➔ indispensable for automotive applications.



### ➤ Data rate dependent on bus length

Class C - data rate: 1 Mbit/sec	➔	Bus length: 40 meters,
Class B - data rate: 125 kBit/sec	➔	Bus length: 500 meters,
Class A - data rate: 50 kBit/sec	➔	Bus length: 1000 meters.

Typical definitions:

<i>Low-speed:</i>	25 kBit/sec	up to	125 kBit/sec.
<i>High-speed:</i>	500 kBit/sec	up to	1 Mbit/sec.

## Transmission principles

---

### ➤ Multicast / broadcast philosophy

CAN messages do not include address of sender or receivers, but to information contents.



### ➤ Bus access principle: CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

- |                             |   |
|-----------------------------|---|
| <i>Carrier Sense:</i>       | Every node monitors the bus level, all the time => monitoring of foreign and own CAN frames!      |
| <i>Multiple Access:</i>     | Every node can start a transmission any time when the bus is free.                                |
| <i>Collision Avoidance:</i> | When several nodes start transmission at the same time, <u>all but one</u> withdraw from sending. |

## Hardware characteristics

---

### ➤ Hardware message acceptance filtering

Only messages that carry relevant information pass through

➔ reduces CPU load.

### ➤ Sophisticated hardware error management

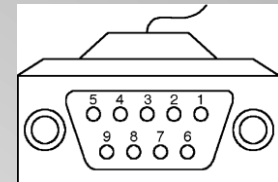
Error prevention and detection methods

Automatic re-transmission of messages detected as erroneous

➔ very high transmission security

### ➤ Standardized connector

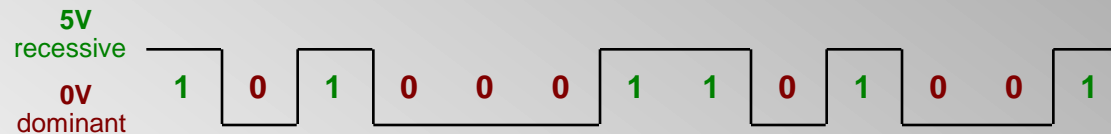
Recommended: 9-pin D-sub connectors (DIN 41652).



# Signal coding

## ➤ NRZ (non-return-to-zero) coding

Example: Voltage levels: **0V (dominant)**, **5V (recessive)**



### Characteristics of NRZ coding:

Voltage level stays the same for consecutive bits of same polarity.

### Note:

Different voltage levels are defined for different purposes.

## Bus stations (Nodes)

---

### ➤ Typically 3 to 40 nodes per bus

No limit for node number defined in CAN specification.  
Number of nodes depends on capabilities of CAN transceivers.  
Bus load usually gets higher with more nodes.

### ➤ Hot plug-in / plug-out

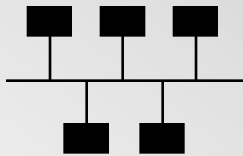
Connect / disconnect nodes while the bus is up and running.



## Advantages of the CAN Bus

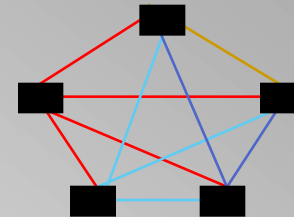
# Advantages of

the  
CAN Bus



over

conventional  
cabling

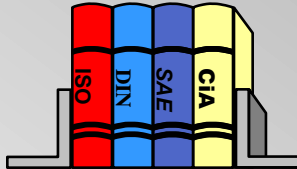


- + Less wires
- + Less connections
- + Less weight
- + Less EMI problems

- + Less space requirements
- + Easier addition of new nodes
- + Lower assembly costs
- + Increased transmission reliability

# International Standards

---



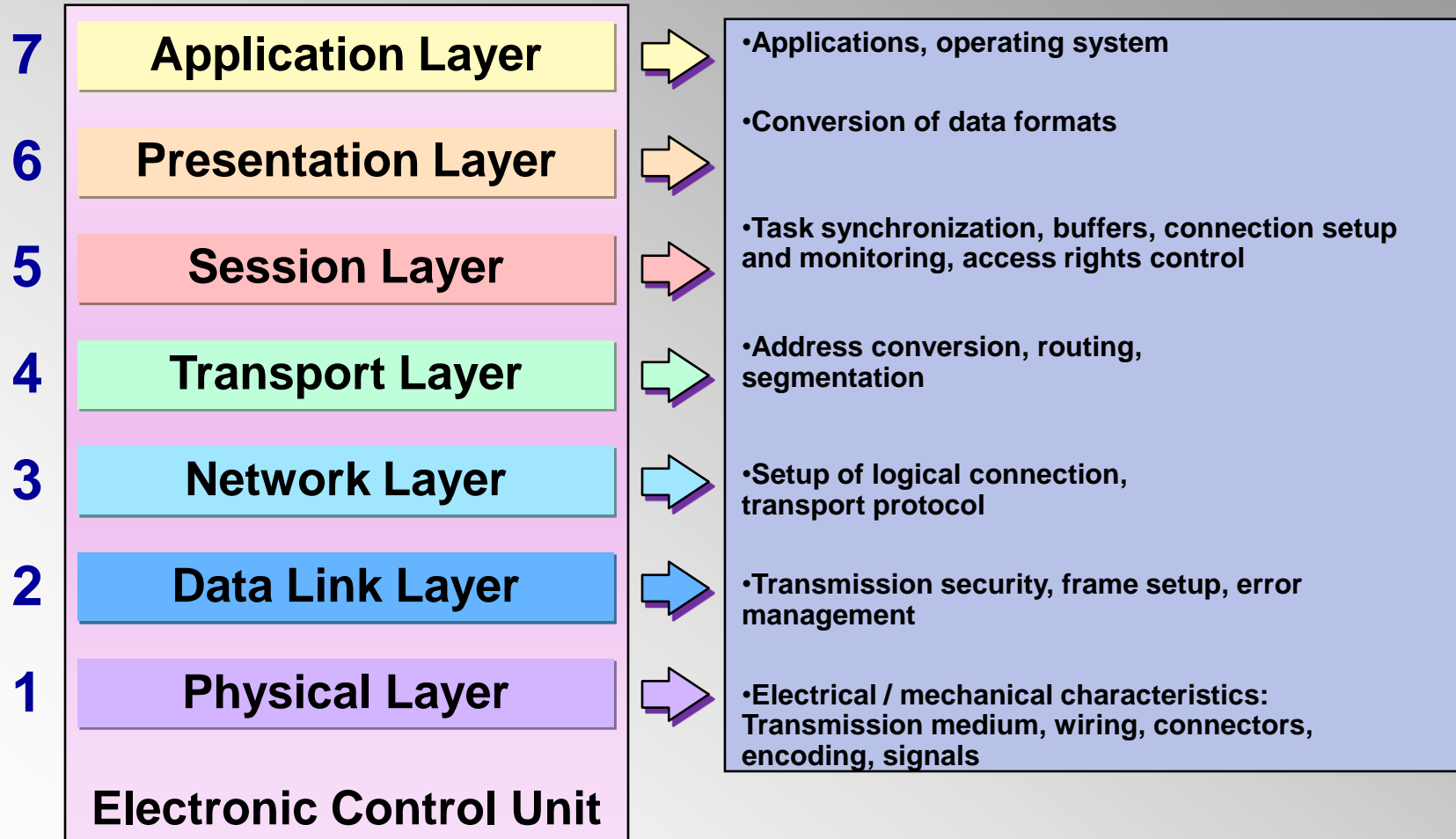
“The great thing about **standards** is that there are so many to choose from.”

## International CAN standards

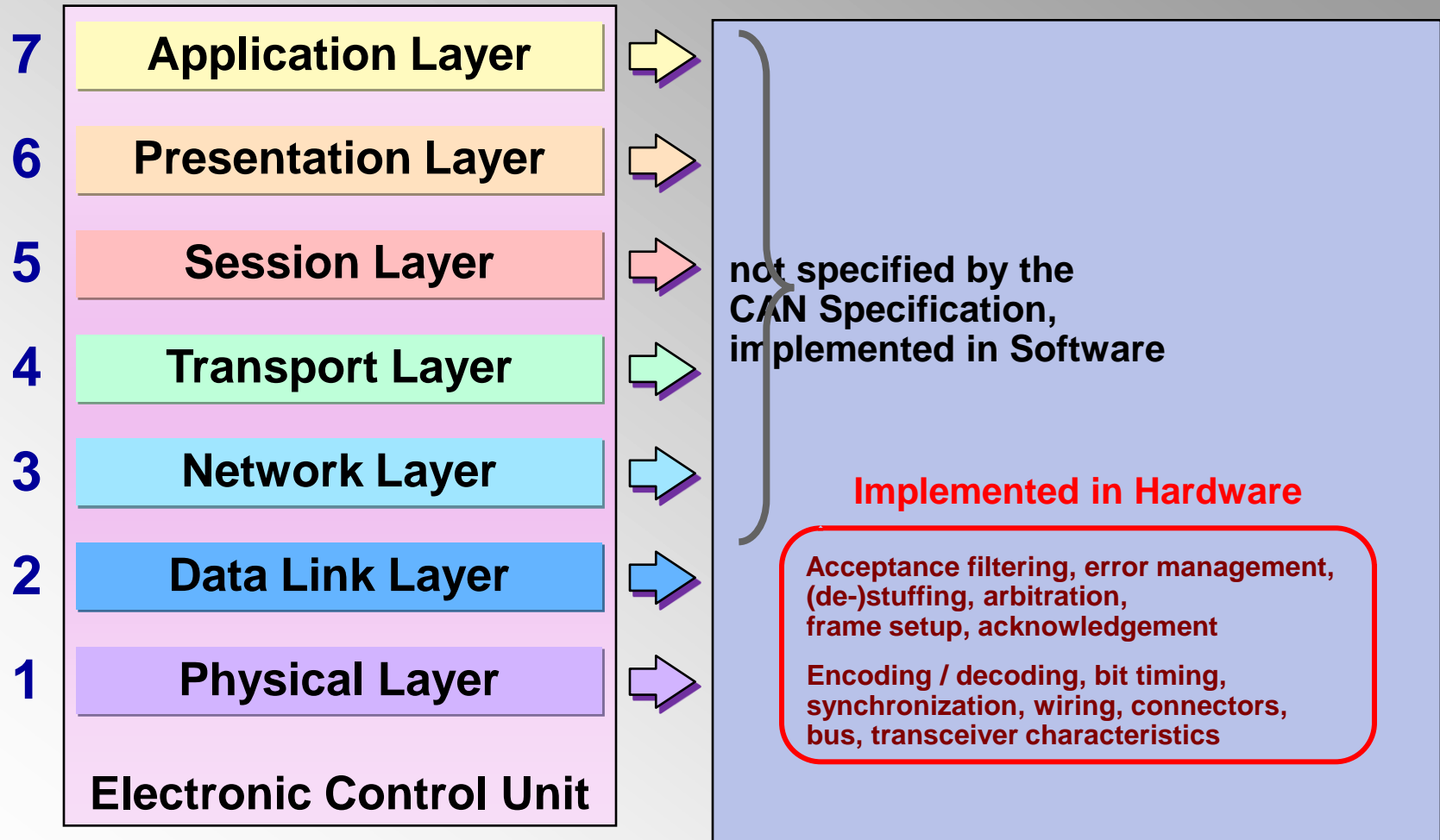
- **ISO 11898** “Road vehicles: CAN for **high-speed** communication”
- **ISO 11519** “Road vehicles: **Low-speed** serial data communication - Low-Speed CAN / VAN”
- **ISO 11992** “Road vehicles: Electrical connections between towing and towed vehicles”
- **EIA RS-485** “Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems” (formerly used for CAN Physical Layer)



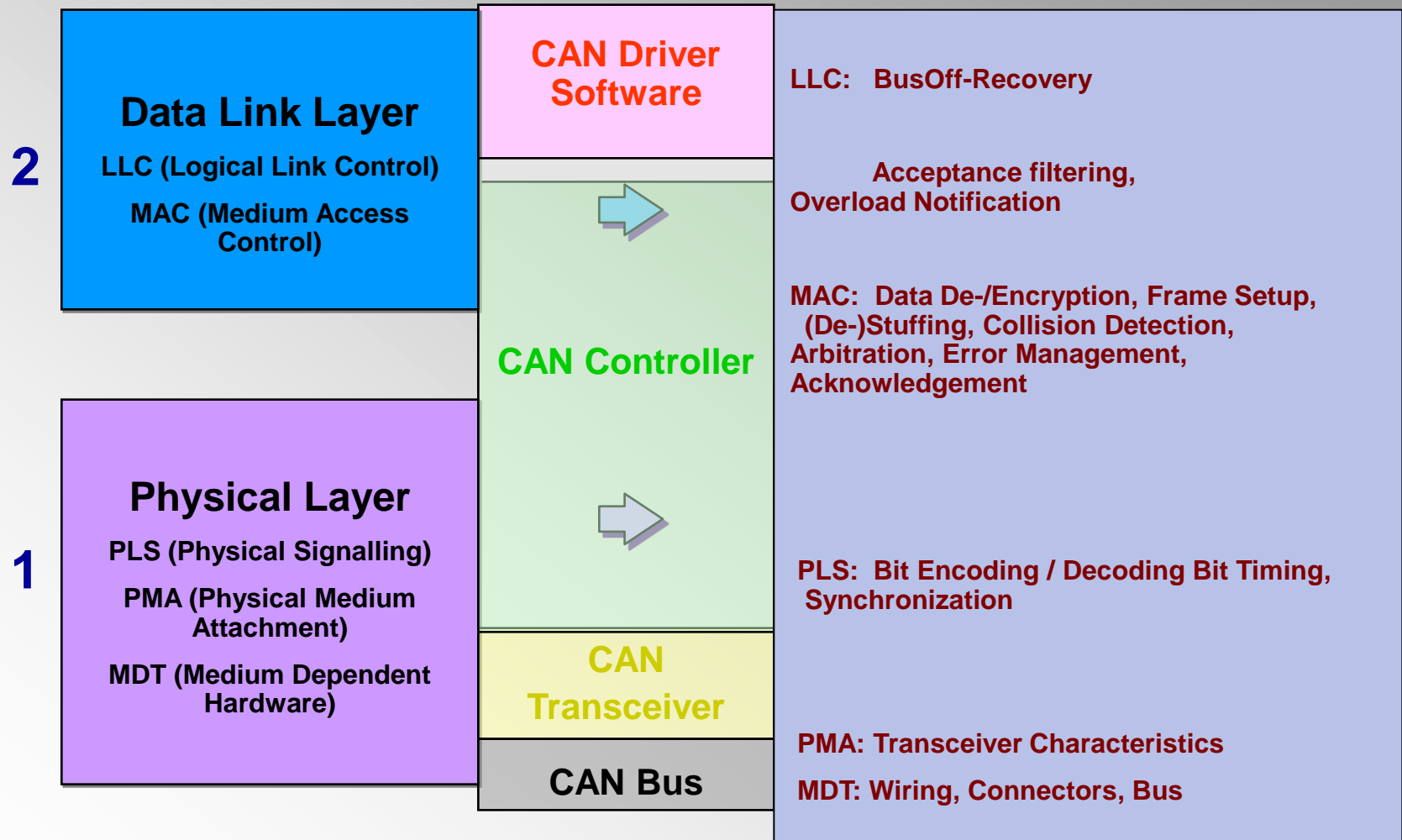
# The ISO/OSI seven-layer model



## CAN within the ISO/OSI model



# CAN ISO coverage on ISO/OSI layers 1,2



## Frames: Overview

---

➤ **Frame: “Envelope” for transmission data**

Exact frame format is defined in CAN specification

➤ **Note: CAN Frame  $\neq$  CAN Message !!!**

A CAN *message* can be spread out over several CAN *frames*

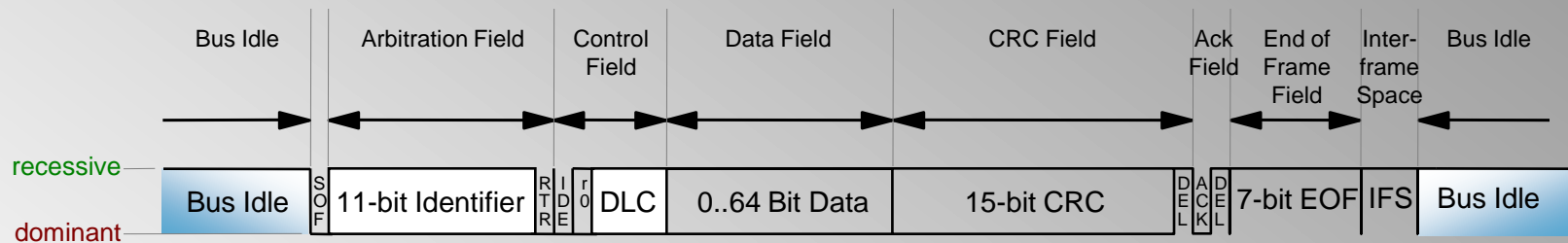
➤ **Four different frame types:**

<b>Data Frame:</b>	Transmission of regular data
<b>Remote Frame:</b>	Remote request for data transmission
<b>Overload Frame:</b>	Indication of bus overload situations
<b>Error Frame:</b>	Indication of transmission errors

## Data Frame: Formats

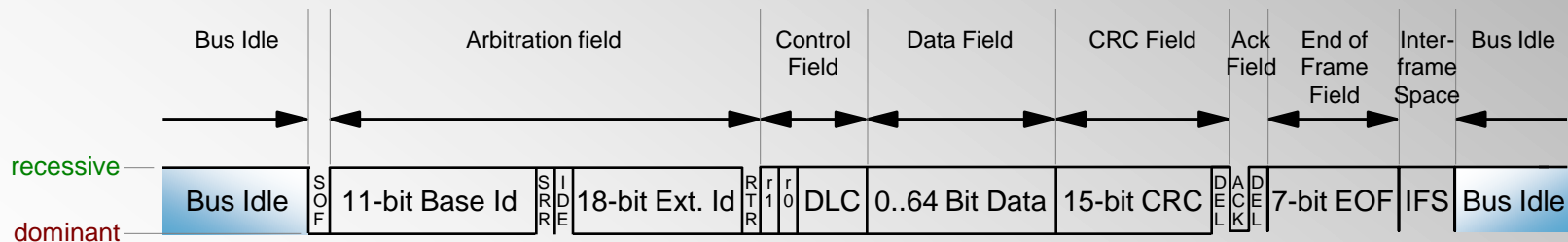
### ➤ Standard Format (CAN 2.0A): 11-bit Identifier

$2^{11} = 2048$  identifiers possible



### ➤ Extended Format (CAN 2.0B): 29-bit Identifier

$2^{29} = 536.870.912$  identifiers possible

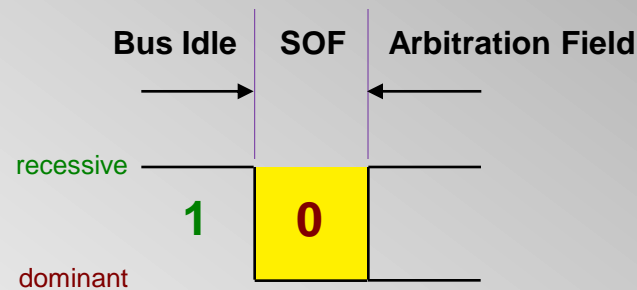


## Data Frame: Start Of Frame (SOF) bit

---



### Start Of Frame (SOF) bit

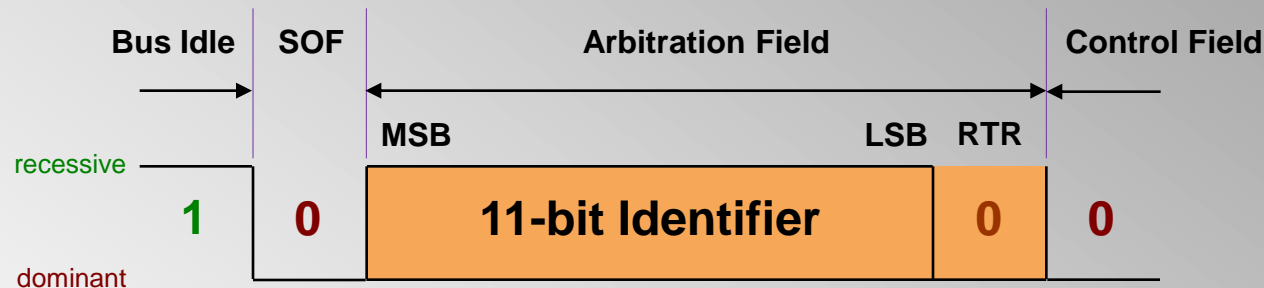


- marks the start of any CAN frame
- is always a **dominant** bit
- provides a falling edge for hard synchronization of transmitter and receivers

## Data Frame: Arbitration Field



### Arbitration Field

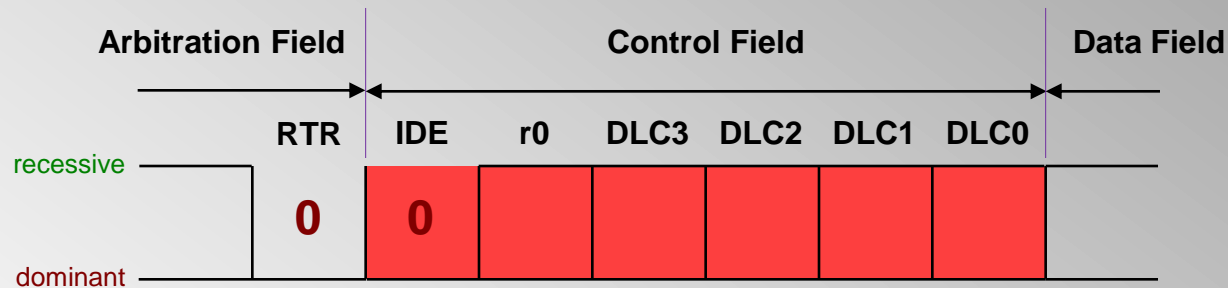


- contains the Identifier which is used for arbitration
- Identifier determines frame priority: **low** identifier = **high** priority
- Remote Transmission Request (RTR) bit is always **dominant** in a Data Frame
- Arbitration = the allocation of bus access rights
- Arbitration occurs when several nodes start transmission at the same time

# Data Frame: Control Field



## Control Field

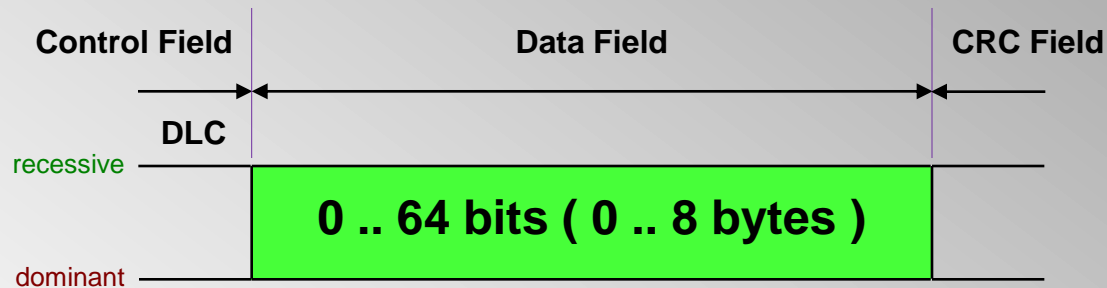


- Identifier Extension (IDE) bit is **dominant** for Standard Frames and **recessive** for Extended Frames
- r0 bit is not used (“reserved for future extensions”)
- Data Length Code (DLC, 4 bits) indicates number of data bytes in Data Field; may take values ranging from 0 to 8, other values are not allowed

## Data Frame: Data Field



### Data Field

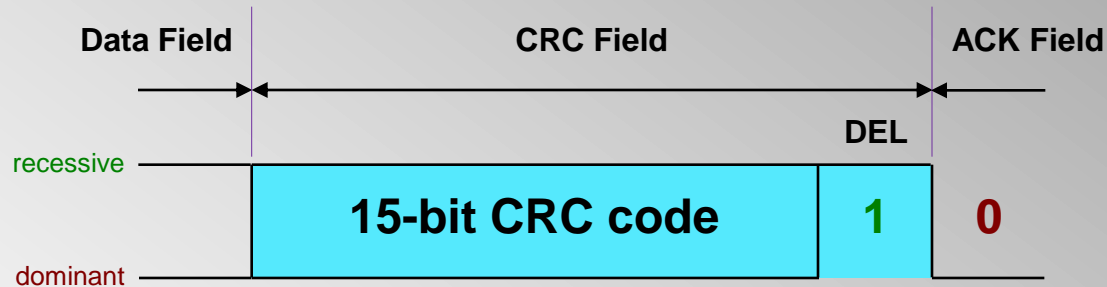


- contains the actual information which is transmitted
- number of data bytes may range from 0 to 8 in units of bytes
- number of data bytes is given in the Data Length Code (DLC)
- transmission starts with the first data byte (byte 0), MSB first

## Data Frame: CRC Field



### CRC Field

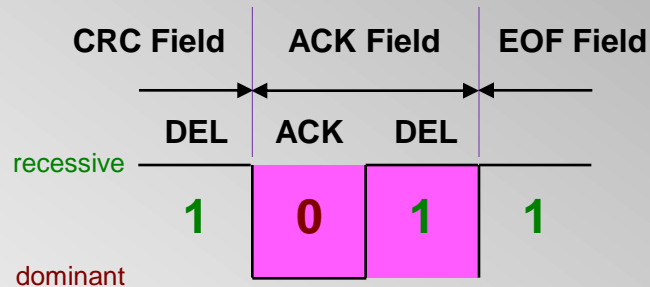


- contains the 15-bit Cyclic Redundancy Check (CRC) code
- CRC is a complex, but fast and effective error detection method
- the CRC Field Delimiter (DEL) marks the end of the CRC field
- the CRC Field Delimiter is always **recessive**

## Data Frame: Acknowledge (ACK) Field



### Acknowledge (ACK) Field



- contains the Acknowledgement (ACK) bit
- the Acknowledgement bit can be **dominant** or **recessive**
- the ACK Field Delimiter (DEL) marks the end of the ACK field
- the ACK Field Delimiter is always **recessive**

## ACK Bit: Values and interpretation

---

### ➤ Acknowledgement procedure:

1. Sender transmits a **recessive** bit in the ACK bit slot
2. Every receiver which received the frame without an error transmits simultaneously a **dominant** bit in the ACK bit slot

### ➤ A dominant ACK bit

1. means that at least one node received the frame without an error, but
2. does **not** necessarily mean that the frame was error-free

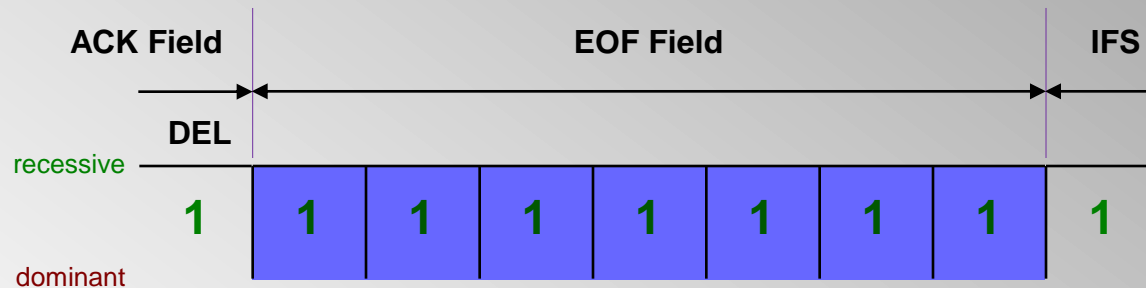
### ➤ A recessive ACK bit

1. could mean that **no** node received the frame without an error or
2. that **no** other node is connected to the bus

## Data Frame: End Of Frame (EOF) Field



### End Of Frame (EOF) Field

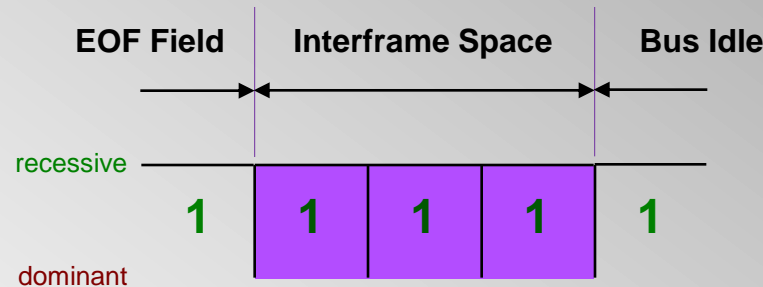


- consists of seven consecutive **recessive** bits
- marks the end of the Data Frame

## Data Frame: Interframe Space (IFS)

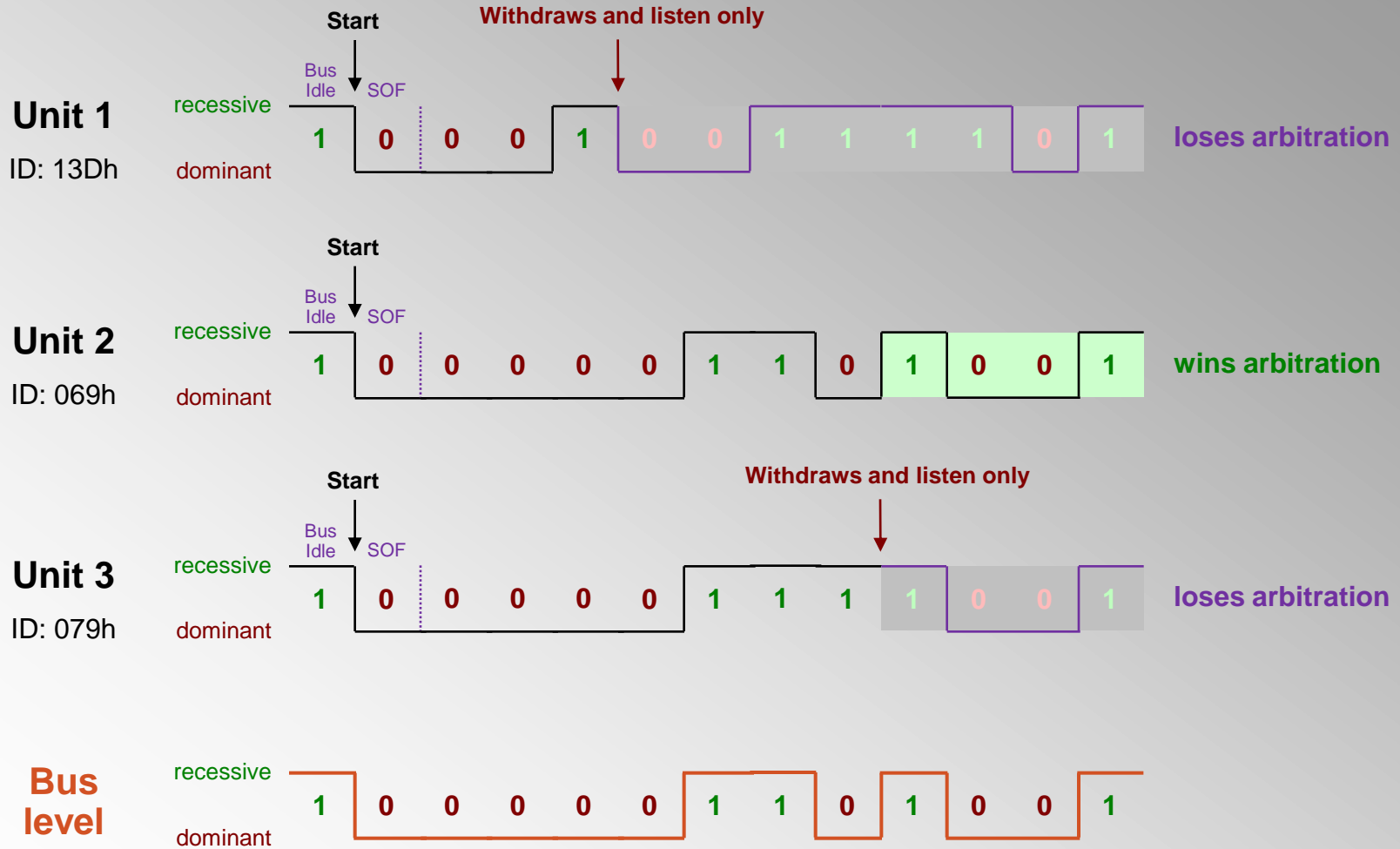


### Interframe Space (IFS)



- consists of at least three consecutive **recessive** bits
- no transmission is allowed during the Interframe Space (IFS)
- is needed by controllers to copy received frames from their Rx buffers
- ACK Field Delimiter + EOF + IFS = 11 consecutive **recessive** bits

# Arbitration: Example



## Arbitration: Identifier $\leftrightarrow$ Priority

---



**Consequence:**  
**Frames**  
**carrying information**  
**of **high** priority**  
**should have**  
**a **low** identifier**

## Bit stuffing: Motivation

---

### Problems when using NRZ coding



- long sequences of bits of the same polarity
- no changes in voltage level for a longer time
- no falling edges for synchronization
- synchronization between sender and receiver may be lost

## Bit stuffing: Approach

---

### Solution: Bit stuffing

- after **five** consecutive bits of same polarity, insert **one** bit of reverse polarity
- CRC code is calculated before bit stuffing is done
- bit stuffing is done by the sender directly before transmission
- de-stuffing is done by the receiver directly after reception
- CRC code check is done after de-stuffing the frame
- bit stuffing is applied to part of the frame from SOF to CRC field



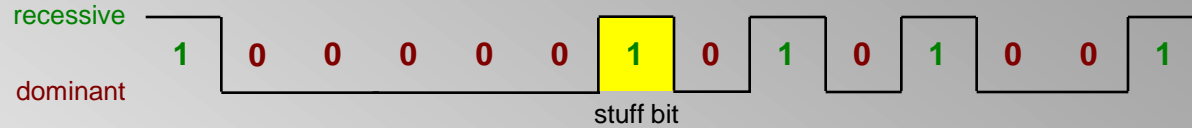
# Bit stuffing: Examples

## Example 1

original  
sequence



stuffed  
sequence

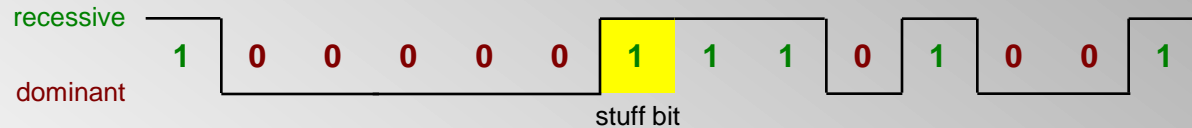


## Example 2

original  
sequence



stuffed  
sequence



## Example 3

original  
sequence



stuffed  
sequence



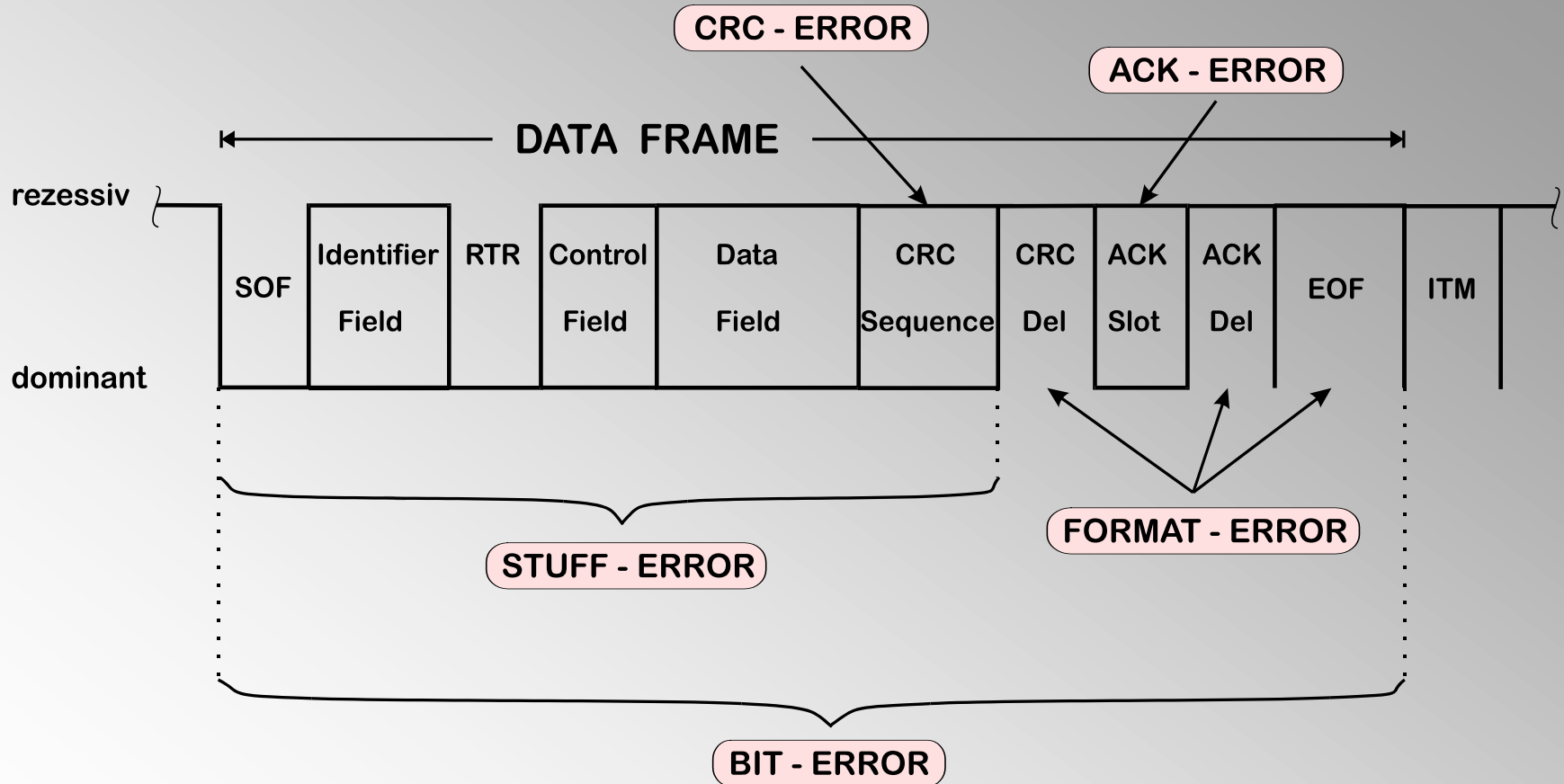
## Data Frame: Maximum size

---

### Maximum frame size (for 8 Data Bytes)

	Standard Frame CAN 2.0A (11-bit identifier)	Extended Frame CAN 2.0B (29-bit identifier)
without stuff bits	111 bits	131 bits
with stuff bits	130 bits	154 bits

## Error types (2)

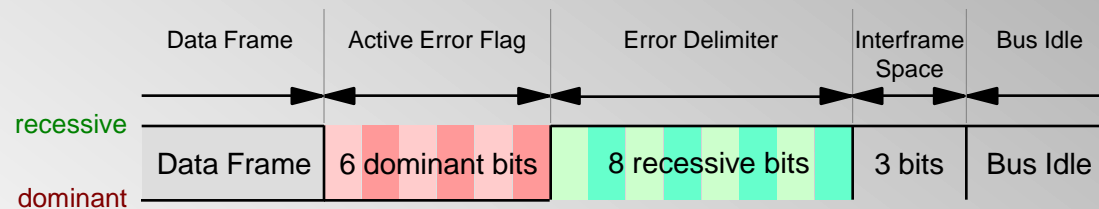


## Error management

### Procedure observed after detection of an error

- Immediate transmission of an **Error Frame**

Format of an **Active** Error Frame:



**No bit stuffing is applied to Error Frames!**

- Error Frame violates the bit stuffing rules → Other receivers are instantly informed that an error has occurred (unless they already found out)
- As a result, other units also send Error Frames
- Sender and receivers reject erroneous frame completely
- Sender retries transmission later

### Two error counters for each unit

- **Transmit Error Counter (TEC)**
- **Receive Error Counter (REC)**

### Characteristics of error counters

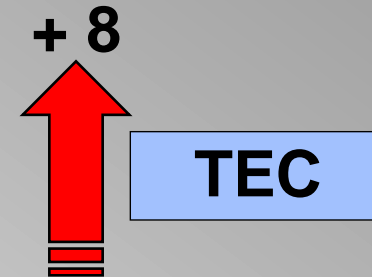
- TEC and REC start counting at 0 (zero)
- Distinction between sporadic (temporary) and permanent errors possible
- Error-prone units are deactivated automatically after a certain time
- Depending on the values of their error counters, units can assume one of three possible node states: **Error Active**, **Error Passive**, **Bus Off**.

## Transmit Error Counter (TEC)

---

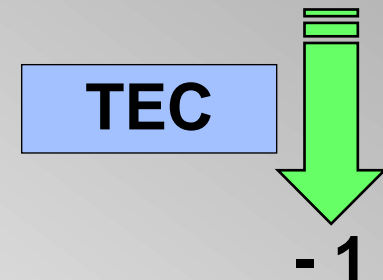
### The TEC is increased by 8 when

- the unit detected an error in the frame it just sent. It is very likely the unit itself is defective.
- there are several other conditions of lesser importance and exceptions to them. Refer to the CAN specification for details.



### The TEC is decreased by 1 when

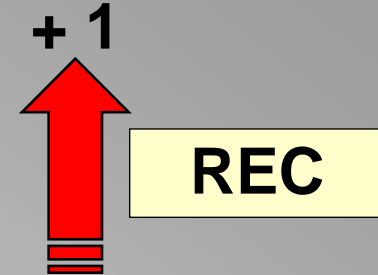
- the unit successfully transmitted a frame, i.e. a **dominant** ACK bit was observed on the bus and Error Frames were neither sent nor received.
- Note: The TEC is not decreased when TEC = 0.



## Receive Error Counter (REC)

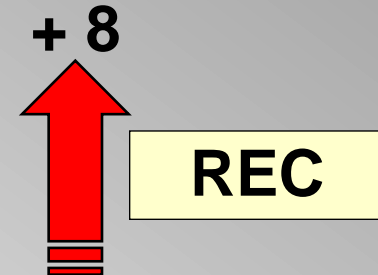
The REC is increased by **1** when

- the unit detected an error in a received frame.



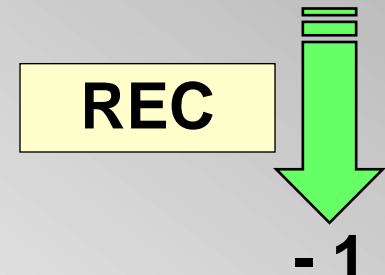
Additionally, the REC is increased by **8** when

- the unit detected this error autonomously, i.e. not through another unit's Error Frame. This is the case when the unit observes a **dominant** bit following its own Error Flag on the bus.
- Usually, the REC cannot be greater than ca. 135.



The REC is decreased by **1** when

- the unit successfully received a frame, i.e. Error Frames were neither sent nor received.
- Notes: The REC is not decreased when REC = 0. Usually, the REC is decreased by 8 when REC > 127.



## Node states: Error active

---

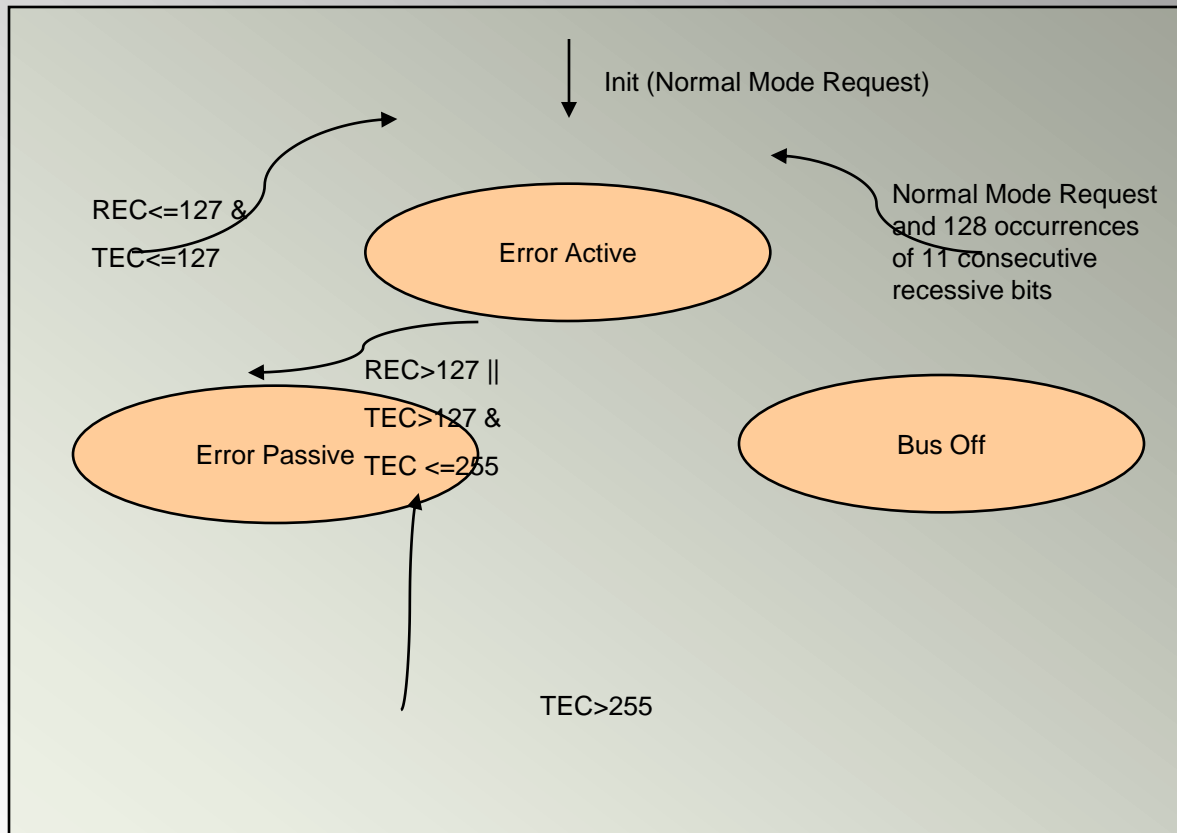
A unit is in **Error Active** state when

- its Transmit Error Counter (TEC) is less than 128:  $TEC < 128$  **AND**
- its Receive Error Counter (REC) is less than 128:  $REC < 128$

In **Error Active** state a unit

- is fully operational
- sends an Active Error Frame when it has detected an error

## Node states: Error management



## Node states: Error passive

---

### A unit is in **Error Passive** state when

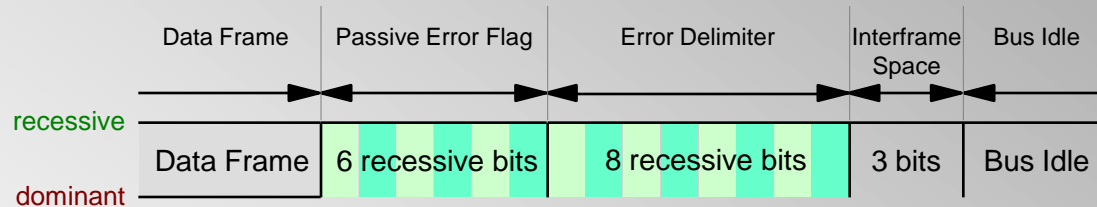
- its Transmit Error Counter (TEC) is greater than 127:  $TEC > 127$  **AND / OR**
- its Receive Error Counter (REC) is greater than 127:  $REC > 127$

### In **Error Passive** state a unit

- sends a Passive Error Frame when it has detected an error
- can still receive frames like a unit in **error active** state can
- has to wait after transmission of a Data Frame for 8 additional consecutive **recessive** bit cycles on the bus (Suspend Transmission Field) until it is permitted to transmit another Data Frame
- can go back to **error active** state for  $TEC \leq 127$  **AND**  $REC \leq 127$

## Passive Error Frame

### Passive Error Frame



- a node in **error passive** state sends a **Passive** Error Frame in case of an error
- a **Passive** Error Frame cannot destroy an ongoing transmission on the bus
- the **Passive** Error Frame might overlap with **Active** Error Frames

## Node states: Bus off

---

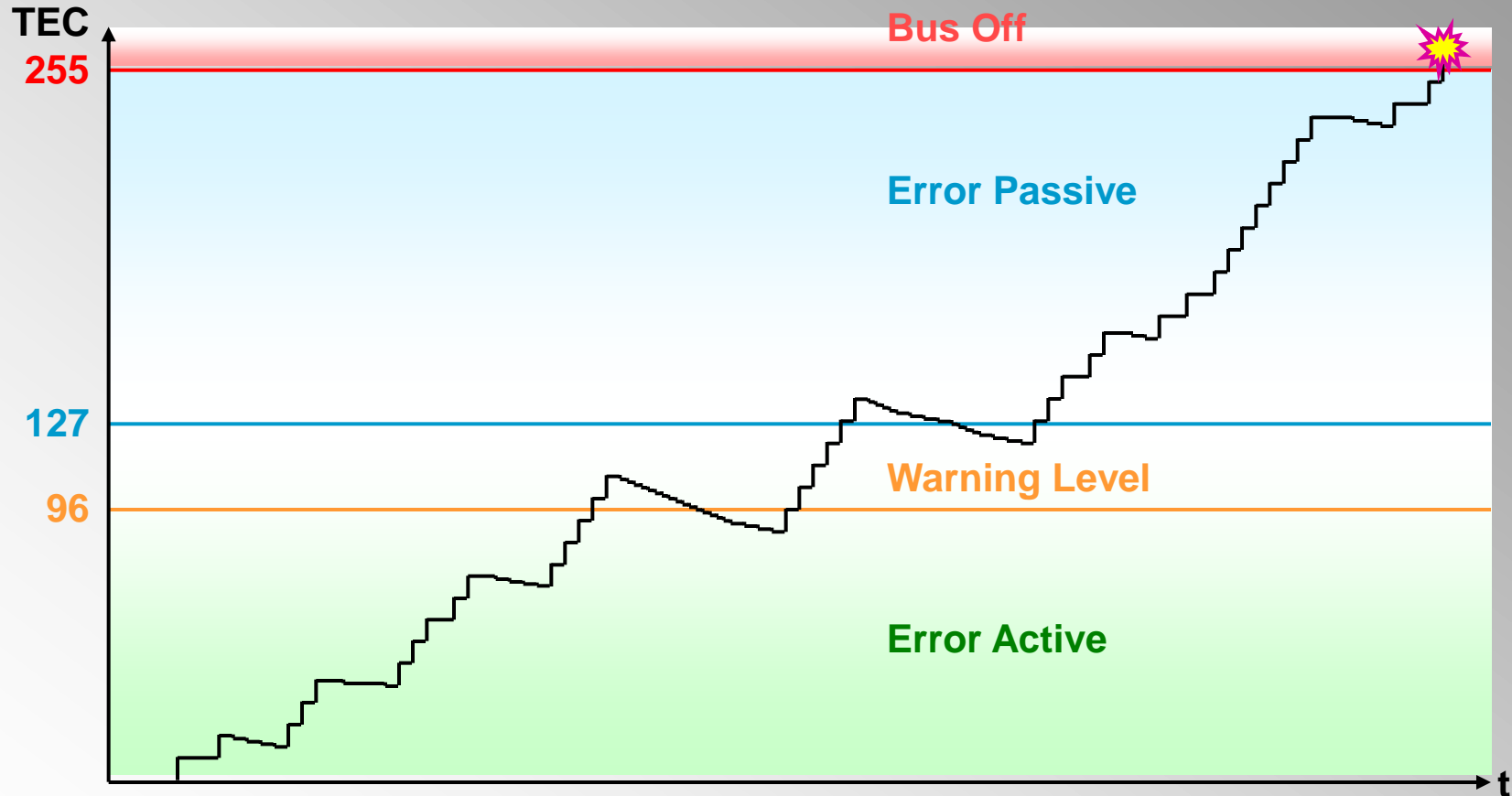
### A unit is in **Bus Off** state when

- its Transmit Error Counter (TEC) is greater than 255:  $TEC > 255$
- Note: The value of the Receive Error Counter (REC) is of no importance
- In case of an Acknowledge error a Bus Off never occurs in the sending node

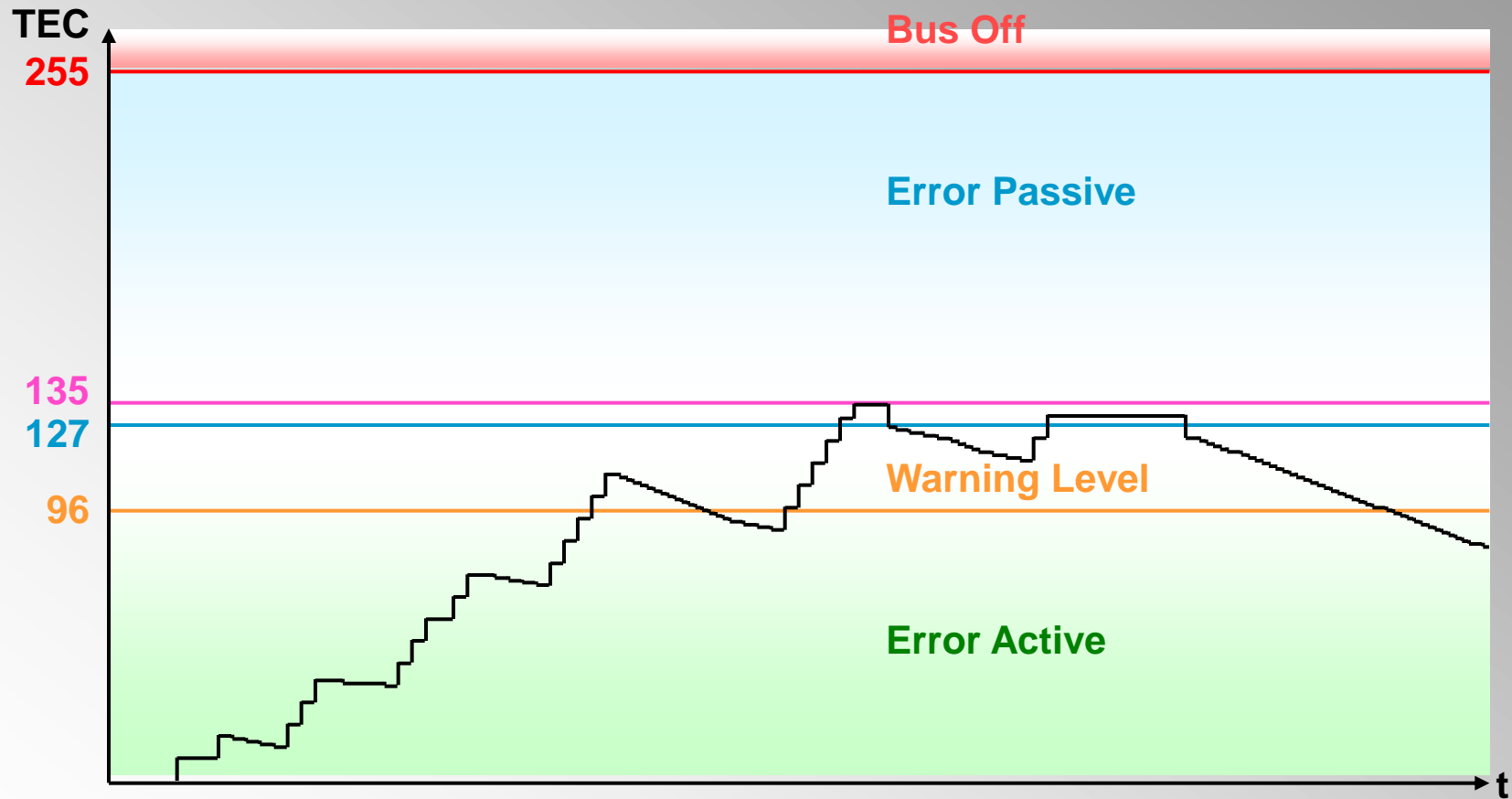
### In **Bus Off** state a unit

- is practically disconnected from the bus
- cannot receive and transmit anything any more
- can only leave **Bus Off** mode via a hardware reset **OR** a software reset and subsequent initialization carried through by the host (CAN specification: TEC and REC are set to zero and the unit must receive 128 times a field of 11 consecutive **recessive** bits)

## Transmit Error Counter (TEC): Example



## Receive Error Counter (REC): Example



## When is CAN Frame valid?

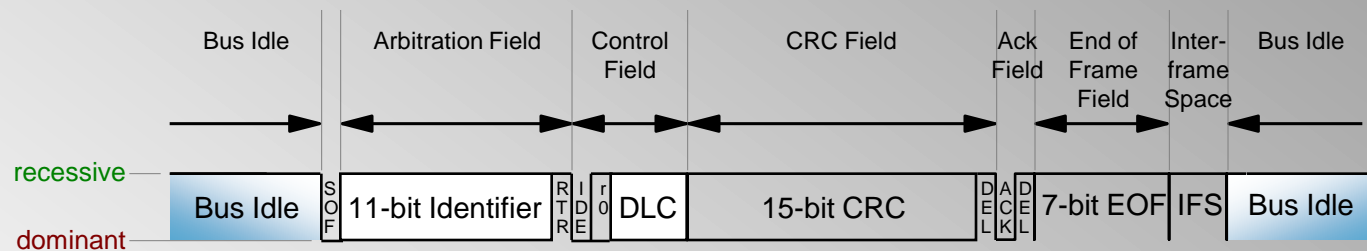
---

The CAN specification includes the following rules to this:

- The CAN Frame is valid for the transmitter, if up to the end of END OF FRAME no fault appears. Is a CAN Frame faulty, it will be repeated automatically.
- The CAN Frame is valid for the receiver, if up to the second last bit OF END OF FRAME end no fault appears.
- The receiver must reject the CAN Frame and has to force the transmitter to repeat the frame, if a local error was detected during the last bit, which was necessary for the validation of the CAN Frame.
- So that the receiver can inform the transmitter, it is clear that the transmitter must wait an additional bit time till the message for him is valid.
- The problem is independent of this, which last bit is required for the validity. Therefore nothing helps to introduce further additional bits at the end of message.

## Remote Frame

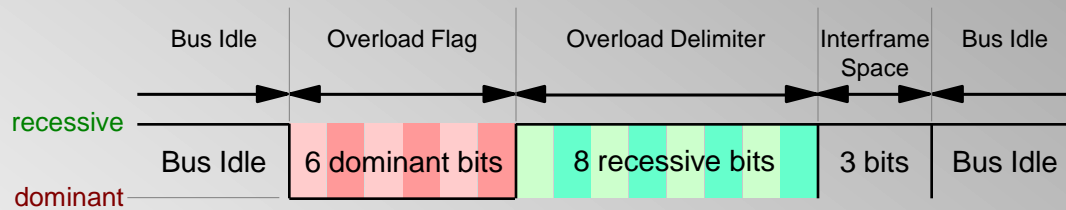
### Remote Frame



- Used to request transmission of a specific Data Frame
- Similar to a Data Frame, but without Data Field
- Remote Transmission Request (RTR) bit is **recessive**
- Same identifier as the Data Frame which is requested
- Note: When Remote Frame is transmitted at the same time as corresponding Data Frame, Data Frame wins arbitration because of **dominant** RTR bit

## Overload Frame

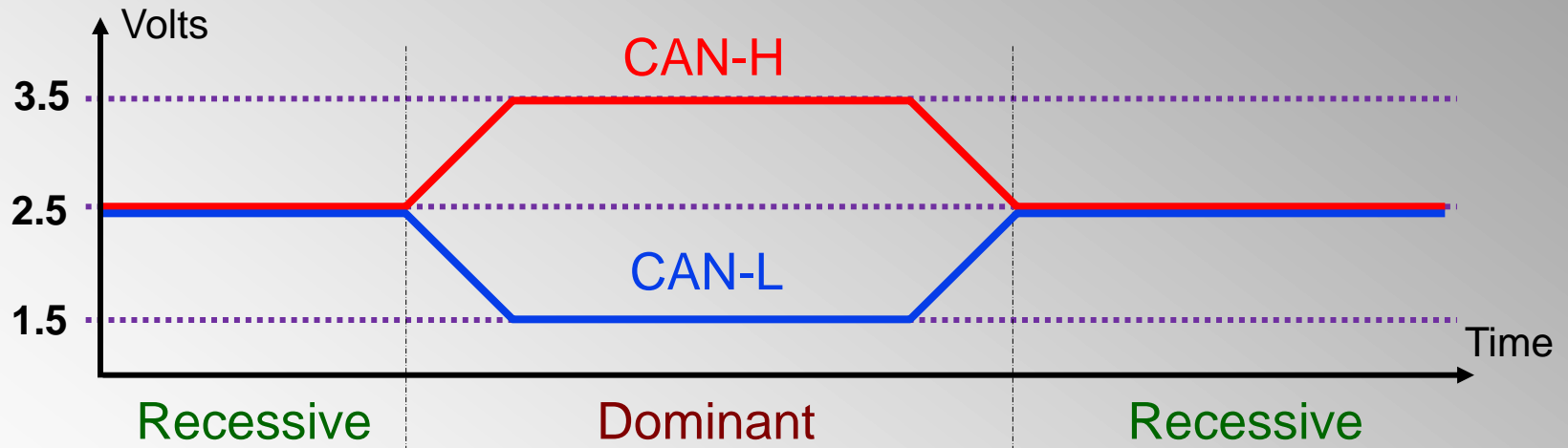
### Overload Frame



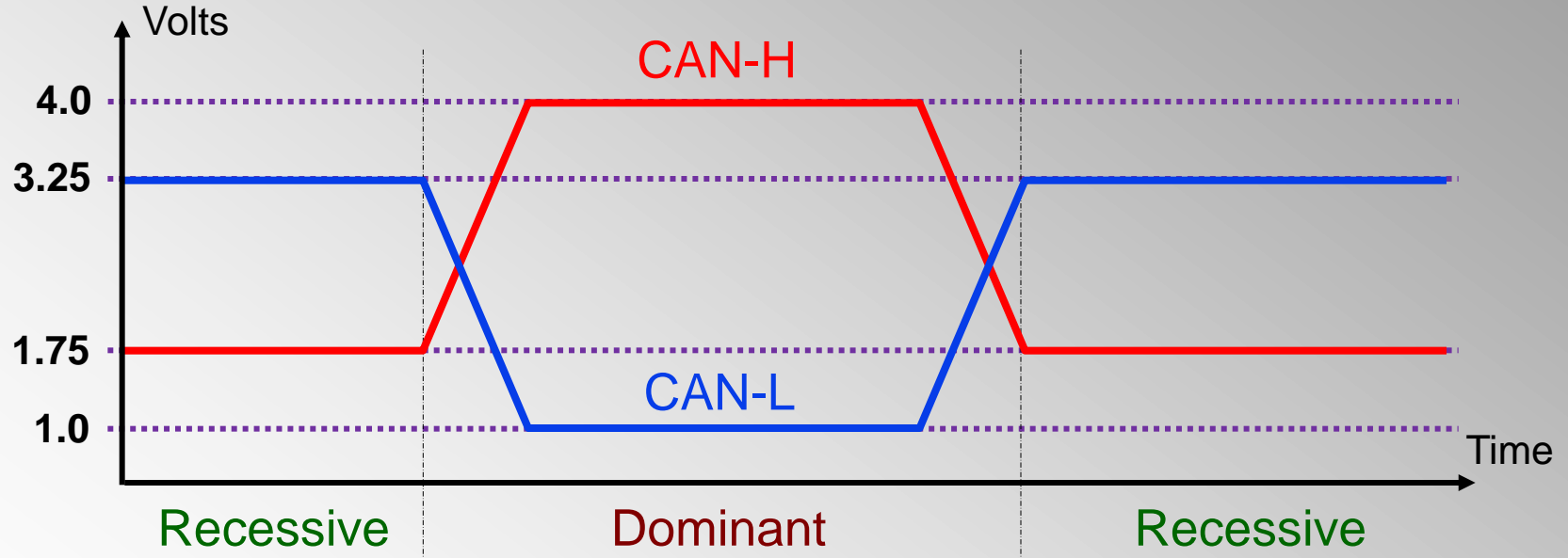
- Unit sends Overload Frame when at present it cannot receive frames any more due to high workload
- Transmission of an Overload Frame is started during the first two bits of the Interframe Space (IFS) of the preceding frame
- Other units react immediately by also transmitting Overload Frames  
⇒ Overload Flags overlap, resulting in up to 12 consecutive **dominant** bits
- Implemented in very few (mostly older) controllers, though controllers must still be able to interpret correctly Overload Frames they receive
- Overload Frames do not influence the error counters (TEC and REC)

## Double-Wire Voltage Levels: High Speed

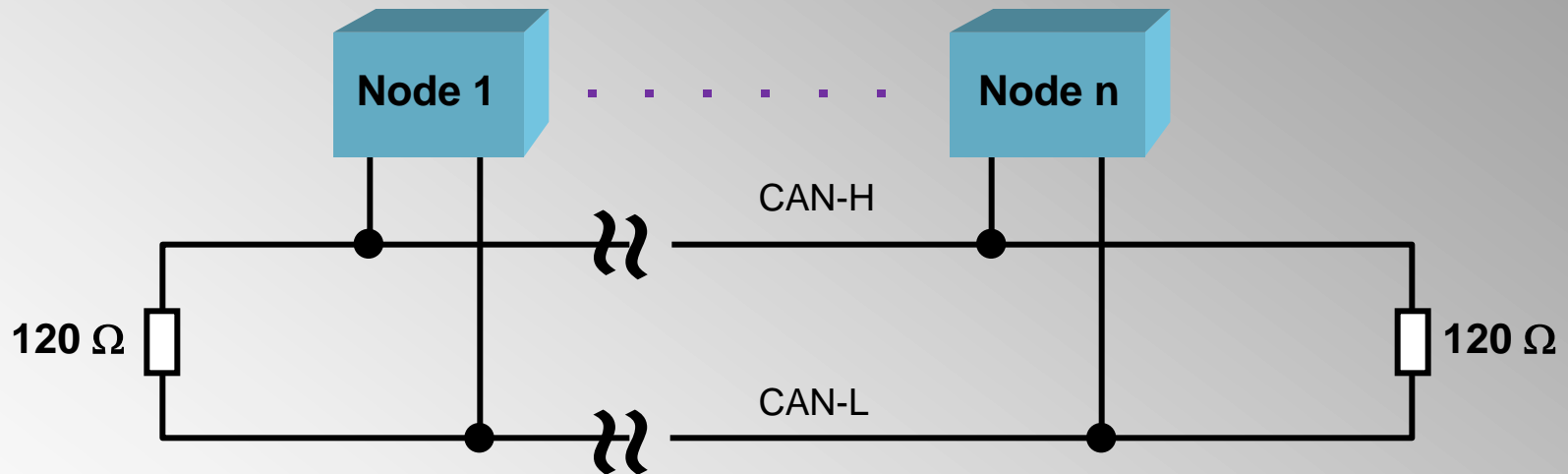
### High-Speed definitions according to ISO 11898-2



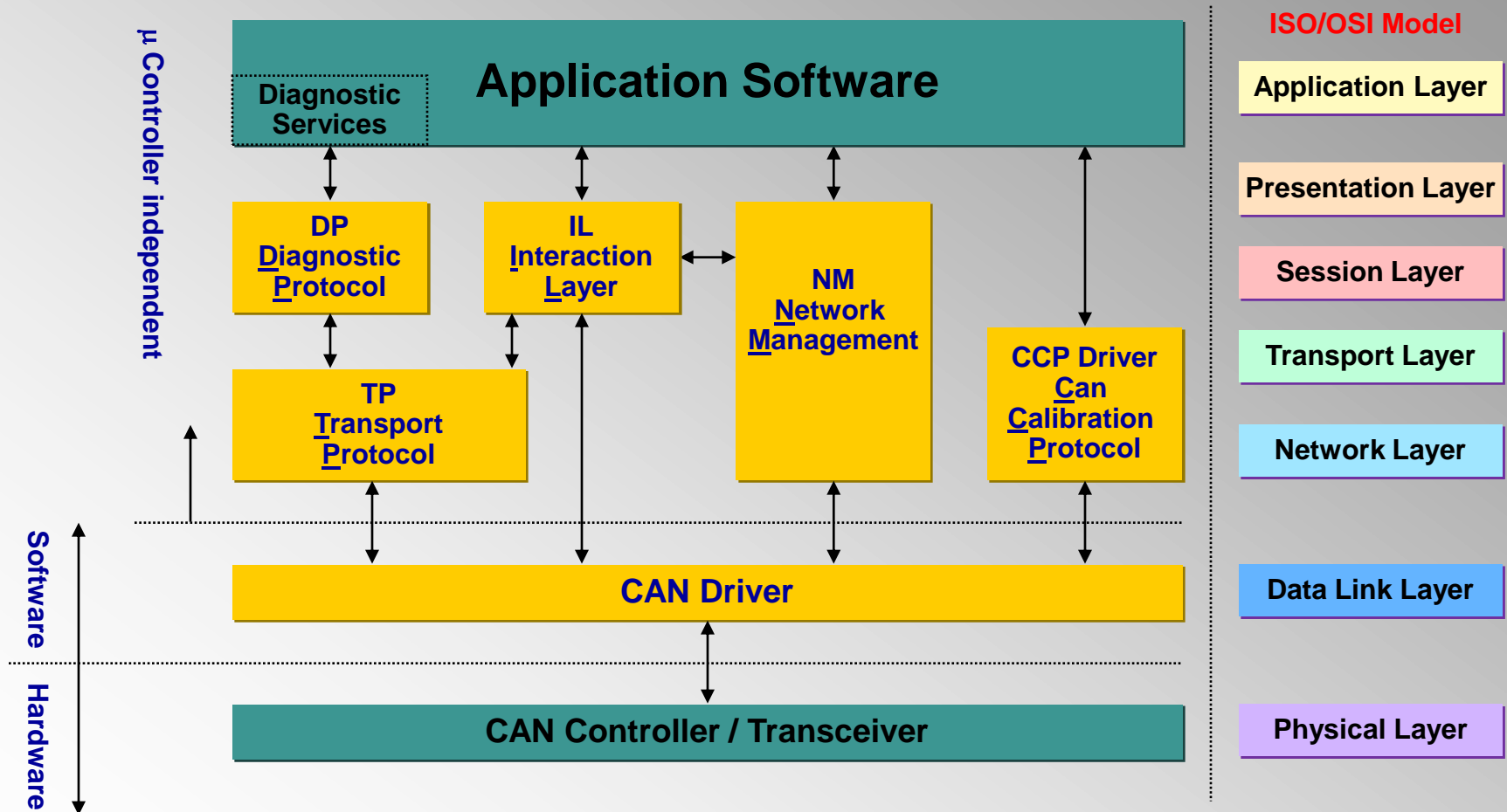
## Low-Speed definitions according to ISO 11519-2



## Double-Wire Bus High Speed Line Termination



# Embedded Communication Software Layers



## CAN Driver

- **Initializes CAN controller**
- **Transmits (Tx) and receives (Rx) frames**
- **Provides Tx and Rx frame buffers**
- **Handles Tx, Rx and error interrupts**
- **Filters messages**

### ISO/OSI Model

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer



## PDU

---

Term : **PDU** = **PROTOCOL  
DATA  
UNIT**

➤ Meaning : unit of data that has a certain meaning in a certain context and can be differentiated from another unit depending on it's purpose.

➤ Example : the *bit* is a PDU at Physical Layer

### From the perspective of ECU functionality:

- **PDU** = **Data Field** from Data Frame.
- The Data Field contains the useful information.
- All other pieces of data are useful for formatting only.

## Network Management (NM)

- Presents current status and configuration of the CAN network to the application
- Supervises CAN network during operation (Detection of plug-ins / plug-outs)
- Each node has a Network Management Frame ID reserved
- Network Management PDU = **NM\_PDU**
- Each node participates in the **Network Management** process (via it's own NM\_PDU)

### ISO/OSI Model

Application Layer

Presentation Layer

Session Layer

Transport Layer

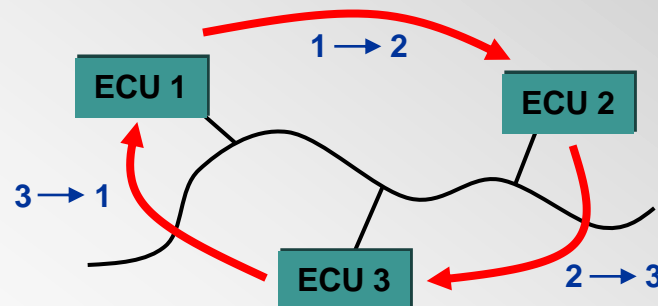
Network Layer

Data Link Layer

Physical Layer

# Direct Network Management

- ECUs send consecutively specific frames containing Network Management information  
➔ “Network Management Token Ring”
- Each ECU is always informed about other ECUs
- On ECU addition, ring is dissolved and re-forms
- On ECU failure, ring shrinks



## ISO/OSI Model

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer

## Network Management (NM)

---

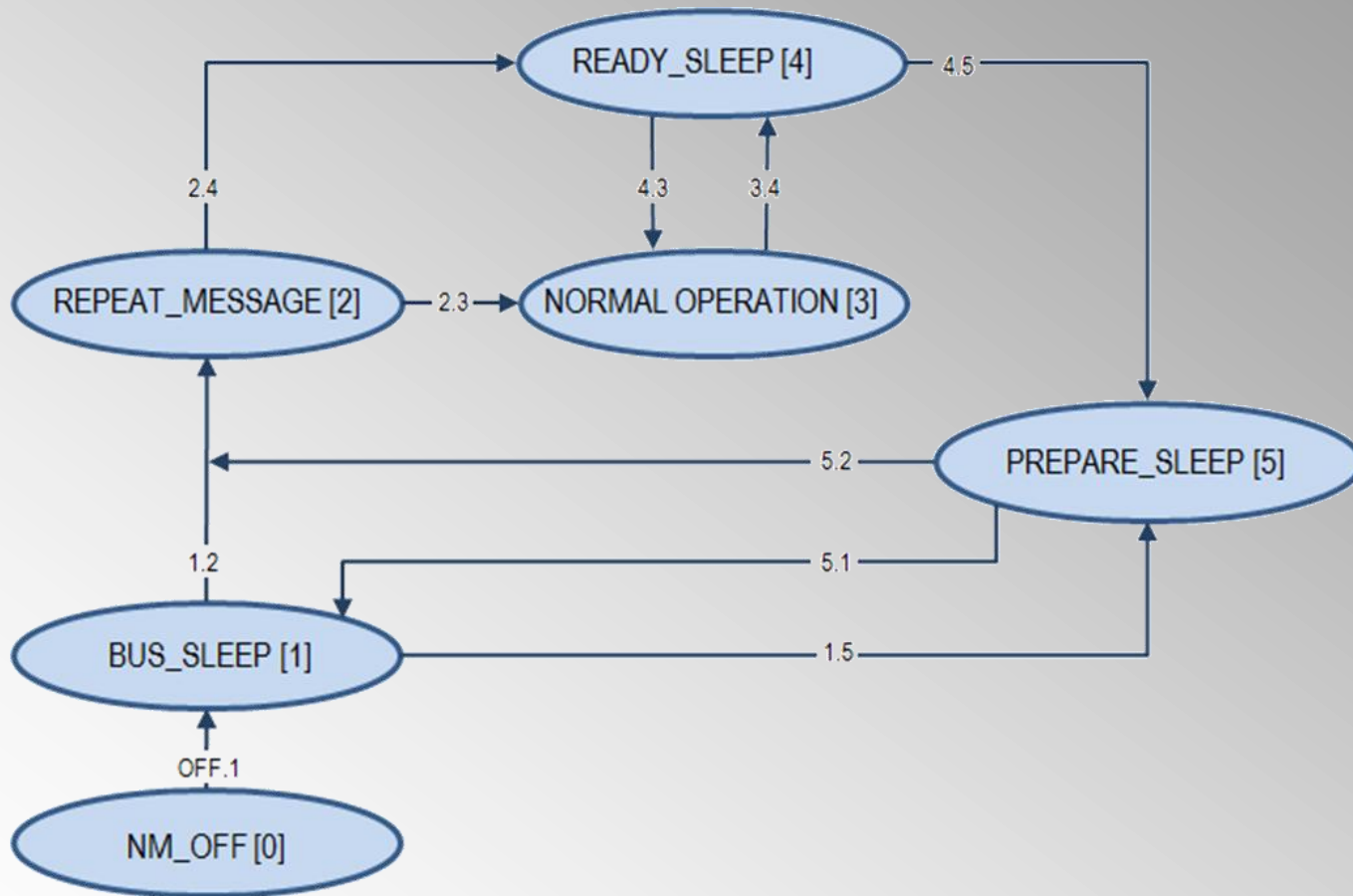
### Possible Network Management states:

#### \* Silent ,Full and No communication

- **Bus Sleep** : occurs when no messages are exchanged on bus  
=> all nodes can reduce power consumption
- **Prepare Bus Sleep** : occurs when message exchange rate is reduced  
precedes Bus Sleep state.
- **Ready Sleep**: makes transition between Normal Operation and Prepare Bus Sleep  
exists so that nodes can synchronize when entering Prepare Bus Sleep
- **Normal Operation** : occurs when nodes exchange frames almost constantly
- **Repeat Message** : occurs after Bus Sleep  
in this state the network is initialized

## Network Management (NM)

### Network Management state machine:



## Network Management (NM)

---

### Notes:

- This state machine form is not mandatory, but it is **AUTOSAR** compliant
- Additional states may be added according to customer needs
- **Network Management** frames are generally transmitted in bursts when some transitions occur.
- Entering and leaving **Bus Sleep** state is affected by **NM-PDUs** exchange
  - when it does not occur nodes will begin transition towards Sleep
  - Sleep is left when NM-PDUs are seen on the bus
- **NM states** should NOT be confused with **System states**

## Transport Protocol (TP)

- Transfer of messages larger than 8 bytes
- Segmentation into frames before transmission
- Recombination of frames after reception
- Flow control handling
- Time-out detection and handling
- Error detection and handling
- TP PDUs = **N\_PDU**s (Network)

### ISO/OSI Model

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer

## Transport Protocol (TP)

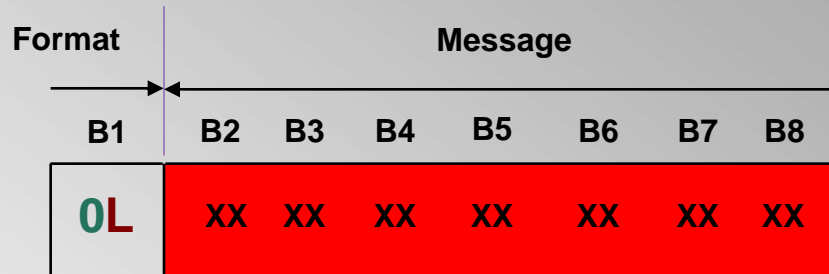
---

### Types of messages :

- **Single Frame messages** : for transmission of a maximum of 7 bytes long messages.
- **Multi Frame messages** : for transmission of at least 8 bytes long messages (maximum length is project specific).

## Transport Protocol (TP)

### Single Frame (SF) Messages:



- B1 : single byte used for formatting of the message
  - 0 : All Single Frame message will start Data Field with the first nibble 0
  - L : This nibble describes the length of the message
- B2-8 (XX) : Message bytes (the actual useful information)
- 252 08 02 10 02 xx xx xx xx xx
- 262 08 02 50 02 xx xx xx xx xx
- Unused bytes are called pad bytes

## Transport Protocol (TP)

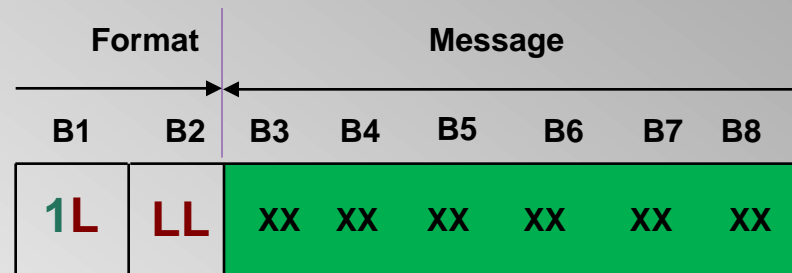
---

### A Multi Frame message requires :

- First Frame (**FF**) : contains message length and first 6 bytes of message
- Consecutive Frames (**CF**) : contains an increment used for frame discrimination and 7 bytes of the message
- Flow Control frames (**FC**) : contains data used for controlling frame flow

## Transport Protocol (TP)

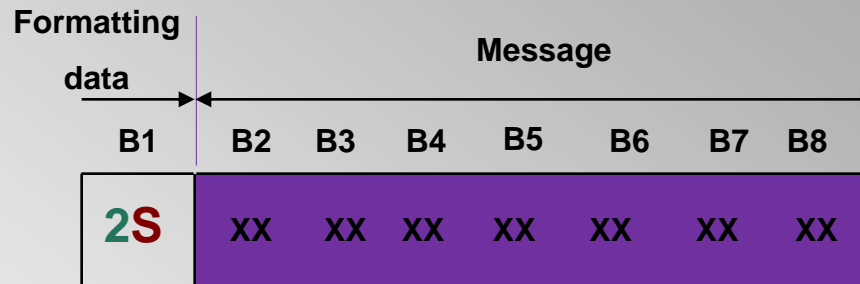
### First Frame (FF):



- B1-2 : bytes used for formatting of the message
  - 1 : All First Frames will start Data Field with the first nibble 1
  - LLL : These 3 nibbles describes the length of the message
- B3-8 (XX) : Message bytes (the actual useful information)
  - No padding here; maximum message size =  $2^{12} = 4096$

## Transport Protocol (TP)

### Consecutive Frame (FF):



- B1 : byte used for formatting of all CFs
  - 2 : All Consecutive Frames will start Data Field with the first nibble 2
  - S : An index (increment) : starts at value 1 and increments
    - After 0x0F the index goes to value 0x00 and continues incrementation
    - In ISO it is called Sequence Number (SN).
- B2-8 (XX) : Message bytes (the actual useful information)

## Transport Protocol (TP)

### Flow Control frame (FC):



➤ B1-3 : useful control data

➤ 3 : All First Frames will start Data Field with the first nibble 3

➤ F : FC flag OR  
Flow Status (FS) in ISO

➤ BS : Block Size

➤ ST : Separation Time

➤ B4-8 (XX) : Padding (usually 00)

## Transport Protocol (TP)

---

### Flow Control frame (**FC**):

- **FC flag** can have values : 0, 1 or 2
  - Value 0
    - **FC.CTS** (Flow Control Clear To Send)
    - it's format was mentioned previously
    - is transmitted after a FF
  - Value 1
    - **FC.WAIT** (Flow Control Wait)
    - does not contain BS or ST
    - it is used to tell the message receiver to wait for a FC.CTS
  - Value 2
    - **FC.OVFLW** (Flow Control Overflow)
    - does not contain BS or ST
    - it is sent when message length is too big
    - marks the abortion of message transmission

## Transport Protocol (TP)

---

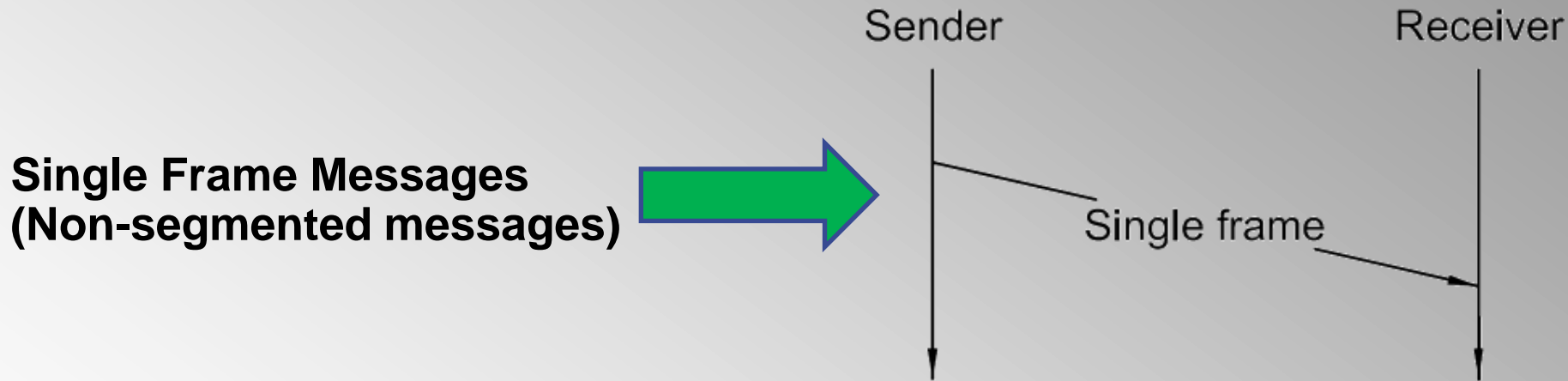
### Flow Control frame (FC):

- **BS** tells the receiver after how many CFs to send a new FC.CTS
- If the value of **BS** is equal to 00 than no FC.CTS needs to be sent after the first one.
- **ST** tells the receiver what the time between every two CFs should be.
- The FCs will always be sent by the node that receives the message, unlike the FFs and the CFs.

## Transport Protocol (TP)

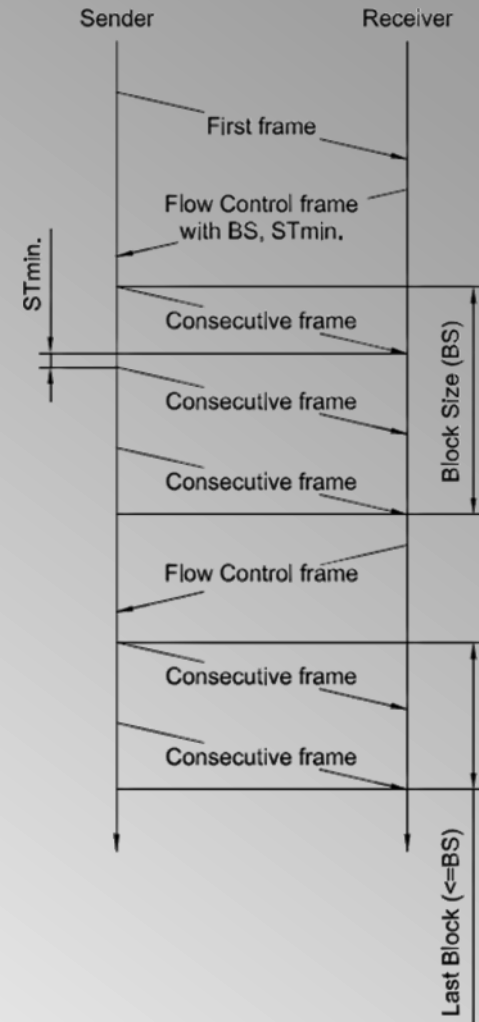
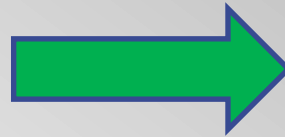
---

### Message Transmission



## Transport Protocol (TP)

**Multi Frame Messages  
(Segmented messages)**



## Transport Protocol (TP)

---

### Communication models:

- **Physical addressing :**
  - corresponds to 1-to-1 communication
  - supports both multi-frame and single-frame messages
  - each node has unique frame IDs used for communication known as Physical Addresses
- **Functional addressing :**
  - corresponds to 1-to-n communication
  - supports only single-frame messages
  - functional addresses are recognized by each node (they are not unique, and they may be multiple)

## Interaction Layer (IL) / Message Manager

- Automatic transmission of frames with the most recent data:
  - event-triggered
  - cyclic
  - cyclic and event-triggered
- Supervision of Rx/Tx time-outs
- Presentation of signals contained in received frames to the application
- Supports signal-based API
- Signal-based frames contain **I\_PDU**s

### ISO/OSI Model

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer

# Questions

---

?