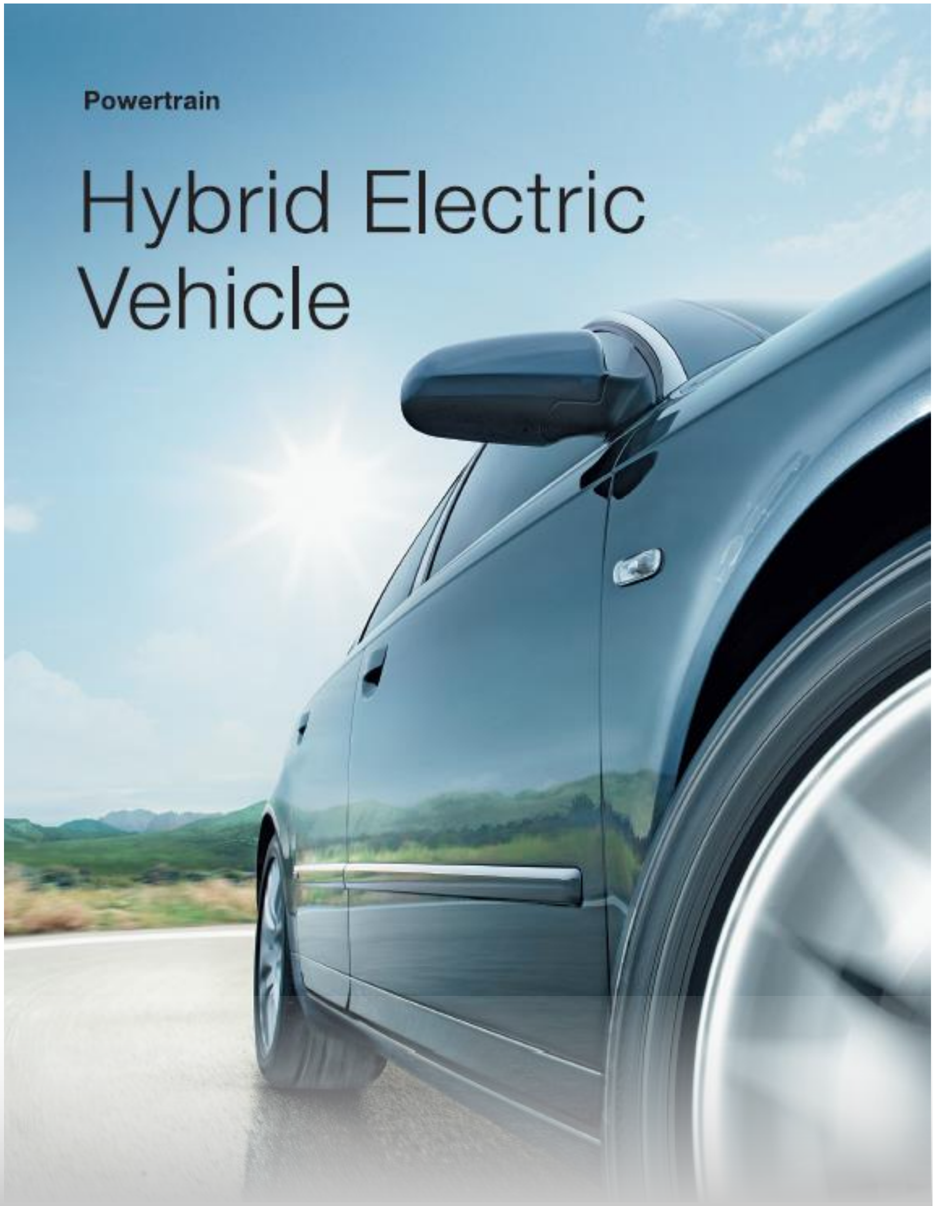
Controller Area Network Training

Abraham Tezmol

P ES FSD GDL

Controller Area Network - CAN

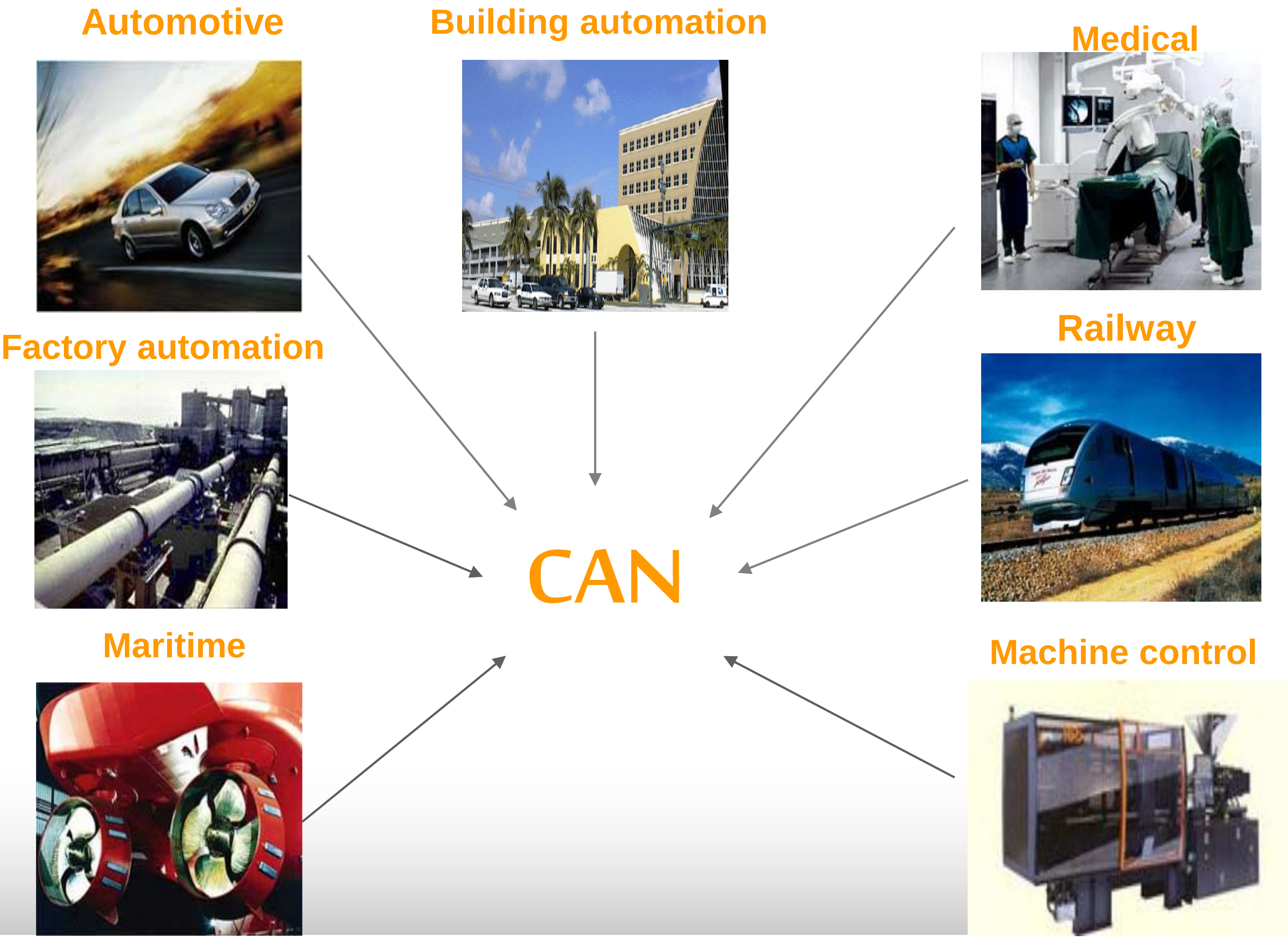
Introduction



Abraham Tezmol, P ES FSD GDL

Application

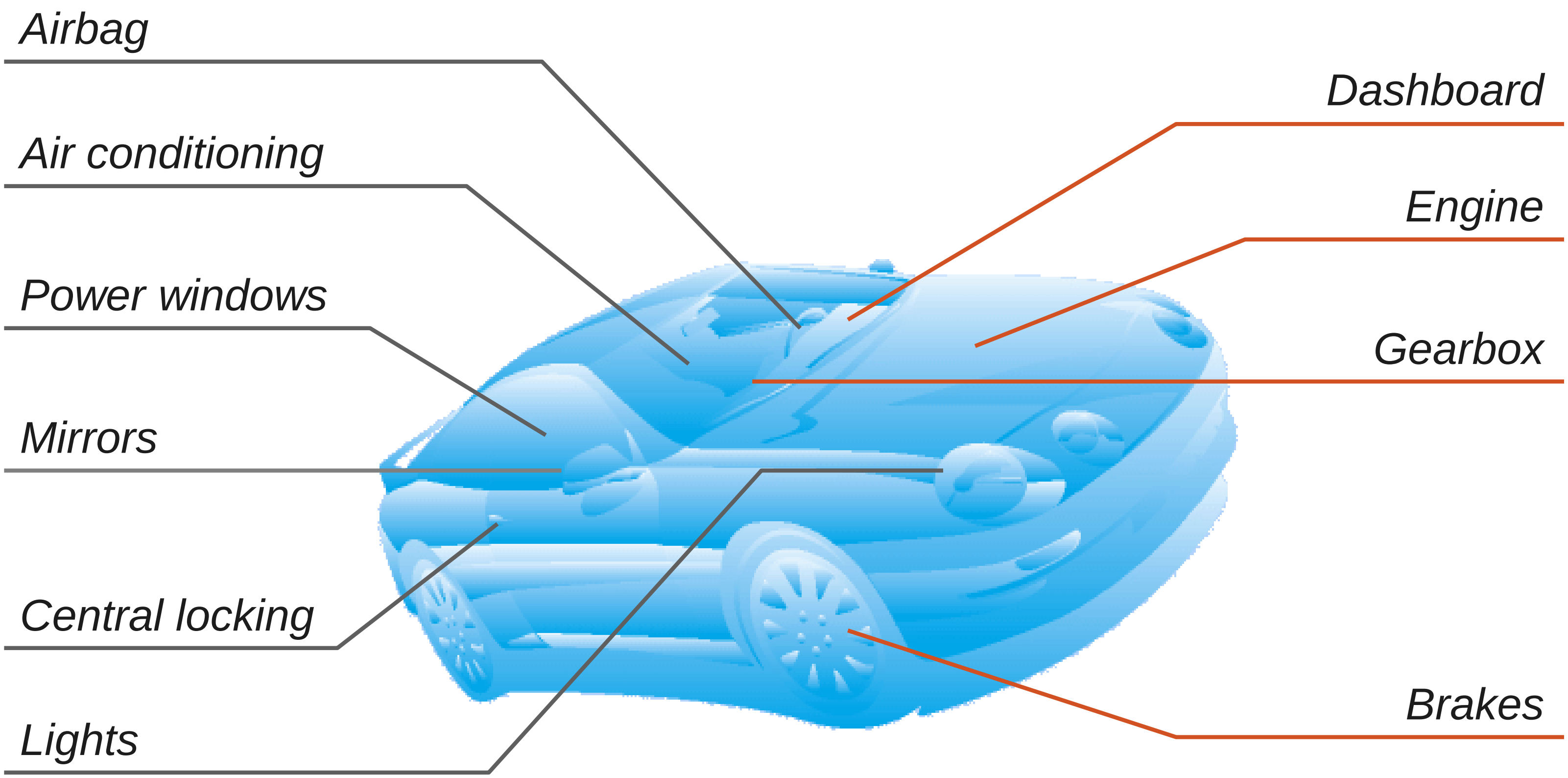
CAN is used in a wide area of applications



Abraham Tezmol, P ES FSD GDL

Application

CAN



Car Body components

Typically low-speed CAN Bus

Power Train components

Typically high-speed CAN Bus

Controller Area Network - CAN

History

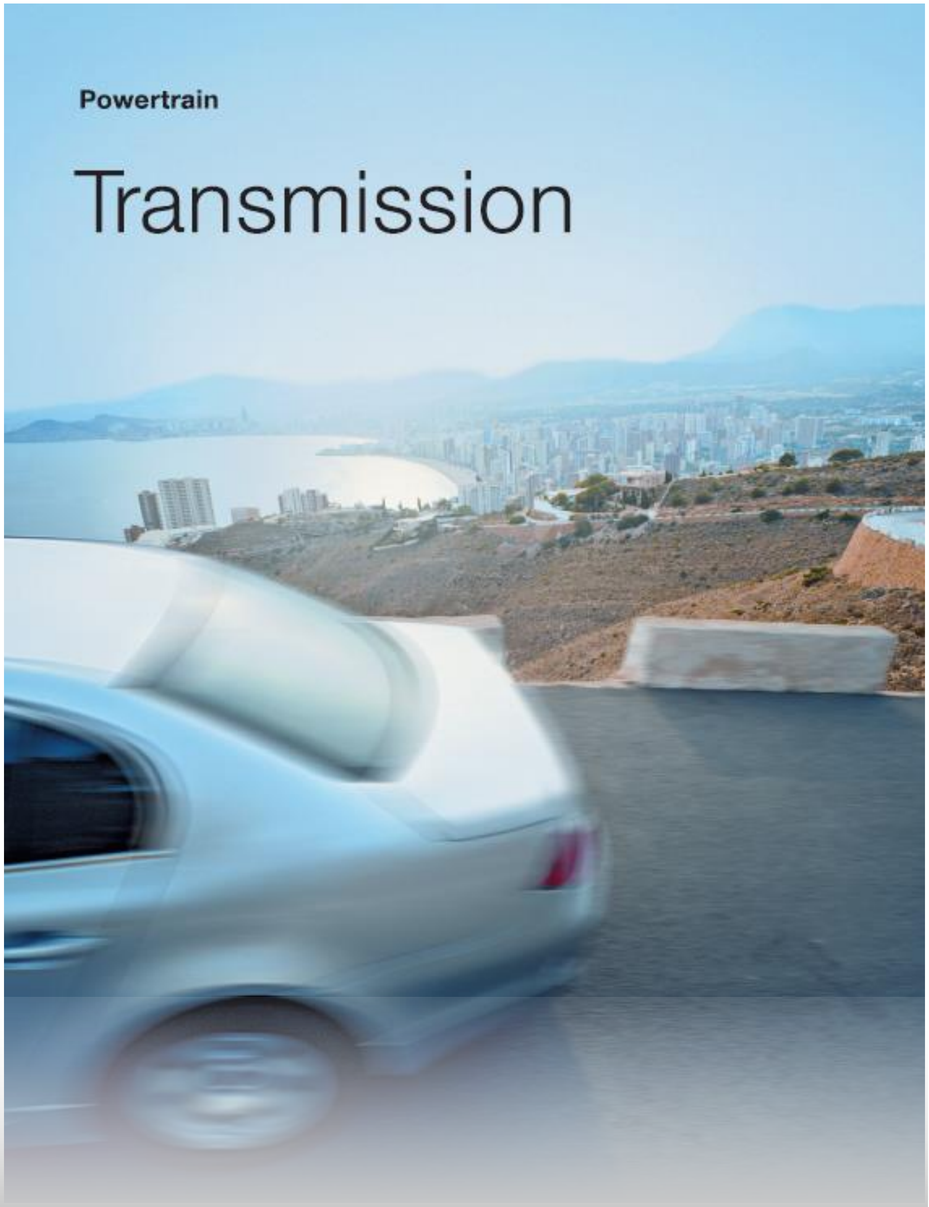
Abraham Tezmol, P ES FSD GDL

History

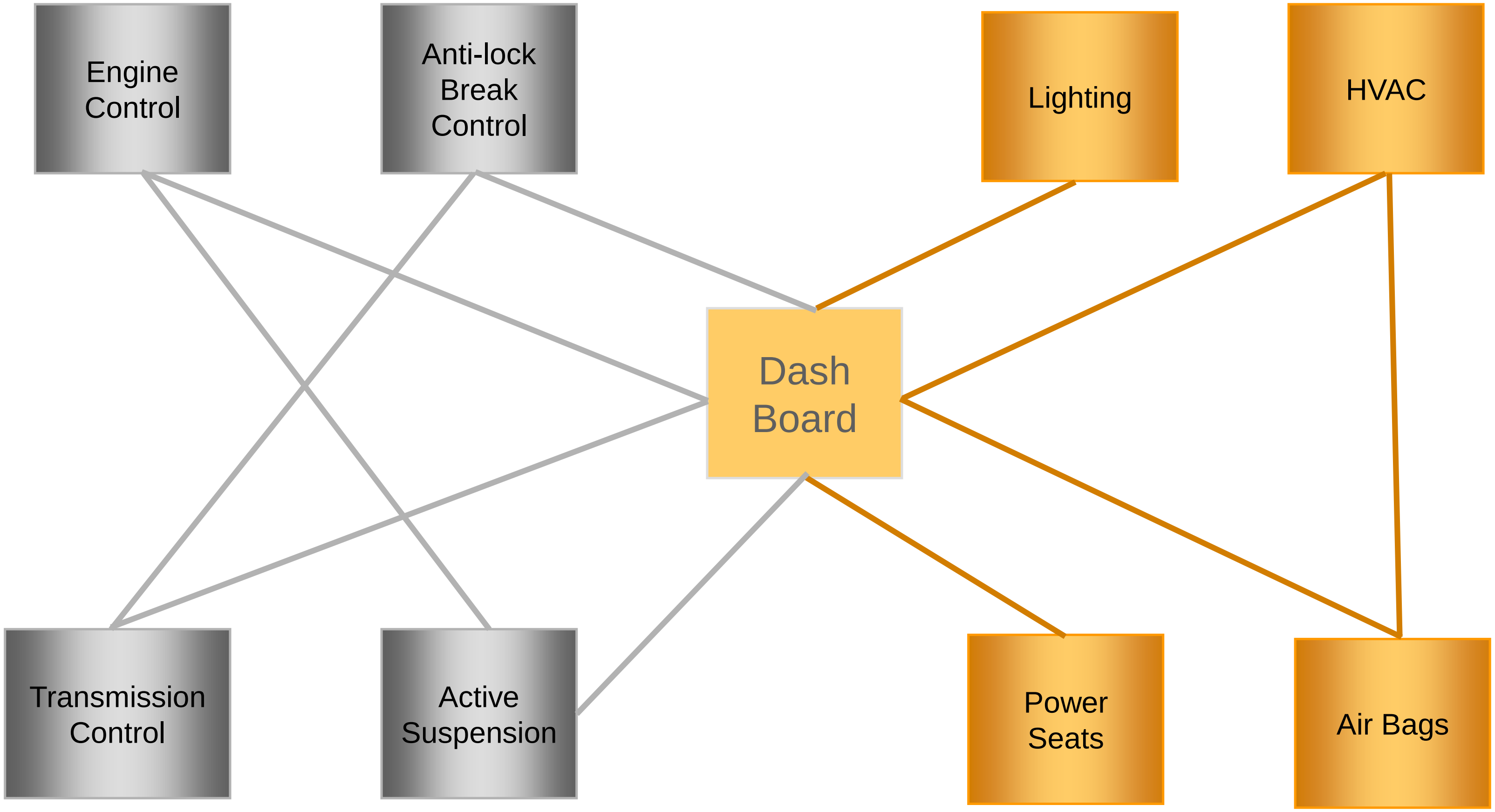
1983	Start of the Bosch internal project to develop an in-vehicle network
1986	Official introduction of CAN protocol
1987	First CAN controller chips from Intel and Philips Semiconductors
1991	Bosch’s CAN specification 2.0 published
1991	CAN Kingdom CAN-based higher-layer protocol introduced by Kvaser
1992	CAN in Automation (CiA) international users and manufacturers group established
1992	CAN Application Layer (CAL) protocol published by CiA
1992	First cars from Mercedes-Benz used CAN network
1993	ISO 11898 standard published
1994	1st international CAN Conference (iCC) organized by CiA
1994	DeviceNet protocol introduction by Allen-Bradley
1995	ISO 11898 amendment (extended frame format) published
1995	CANopen protocol published by CiA
2000	Development of the time-triggered communication protocol for CAN (TTCAN)

Controller Area Network - CAN

Overview



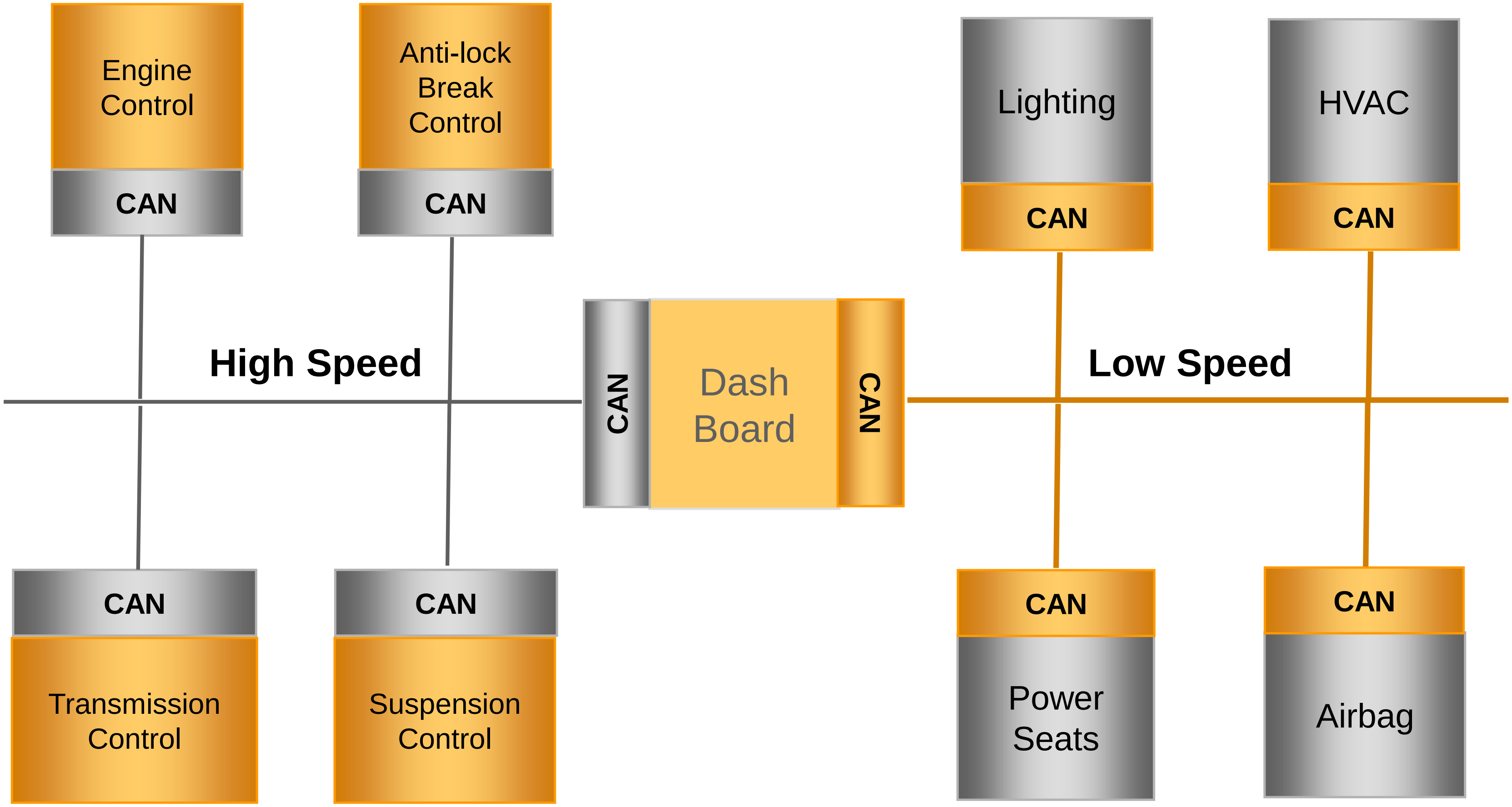
Overview – In the beginning



- The different control systems (and their sensors) to exchange
- A cable network with a length of up to several miles and many connectors was required.
- Growing problems concerning material cost, production time and reliability.

Overview - Solution

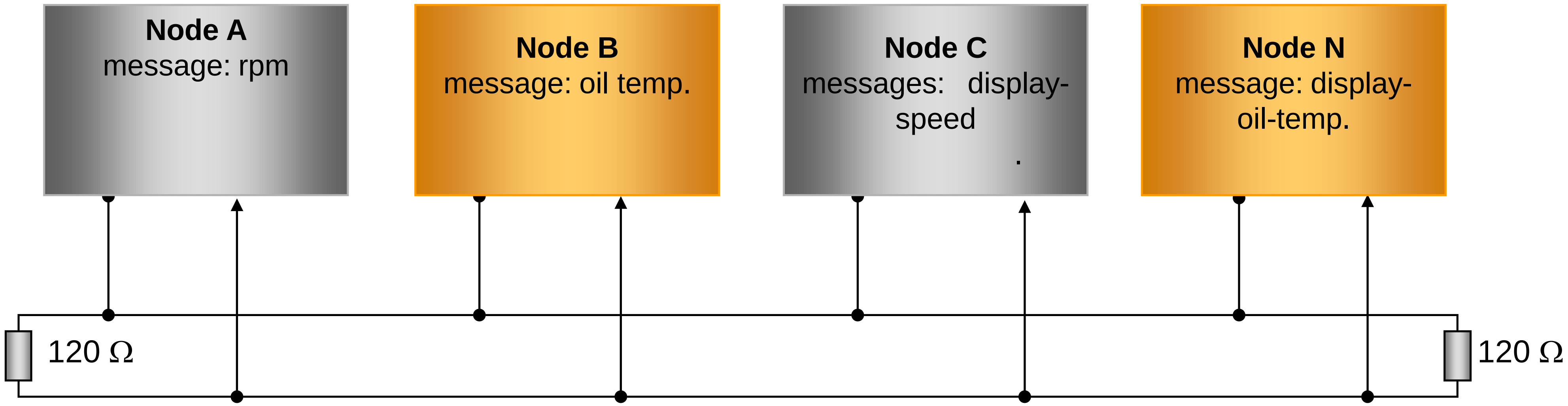
CAN



- Point-to-point wiring is replaced by one serial bus connecting all control systems

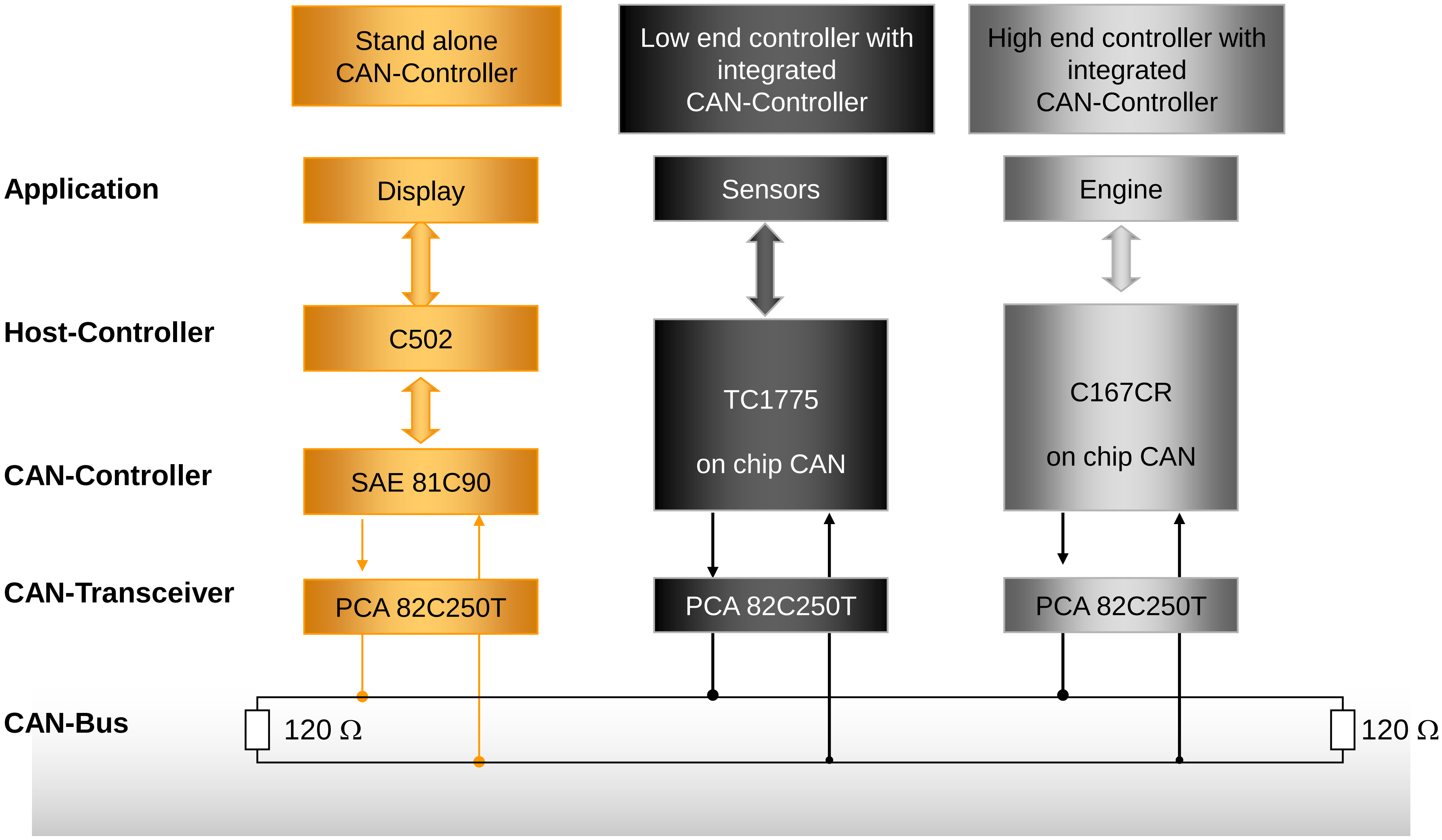
Overview – basic concept

CAN



- In the CAN protocol, the bus nodes do not have a specific address. Instead, the address information is contained in the identifiers of the transmitted messages, indicating the message content and the priority of the message
- The number of nodes may be changed dynamically without disturbing the communication of the other nodes.
- Multicasting and Broadcasting is supported by CAN.

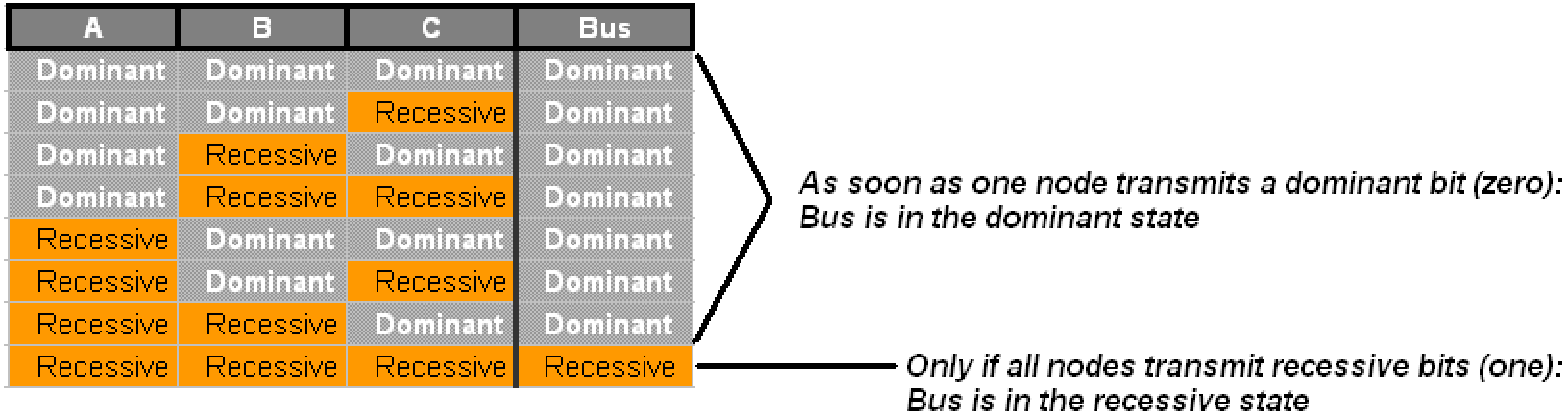
Overview – basic concept



Overview – basic concept

CAN

Two logical states possible in the bus:
"1" = Recessive
"0" = Dominant



A	B	C	Bus
Dominant	Dominant	Dominant	Dominant
Dominant	Dominant	Recessive	Dominant
Dominant	Recessive	Dominant	Dominant
Dominant	Recessive	Recessive	Dominant
Recessive	Dominant	Dominant	Dominant
Recessive	Dominant	Recessive	Dominant
Recessive	Recessive	Dominant	Dominant
Recessive	Recessive	Recessive	Recessive

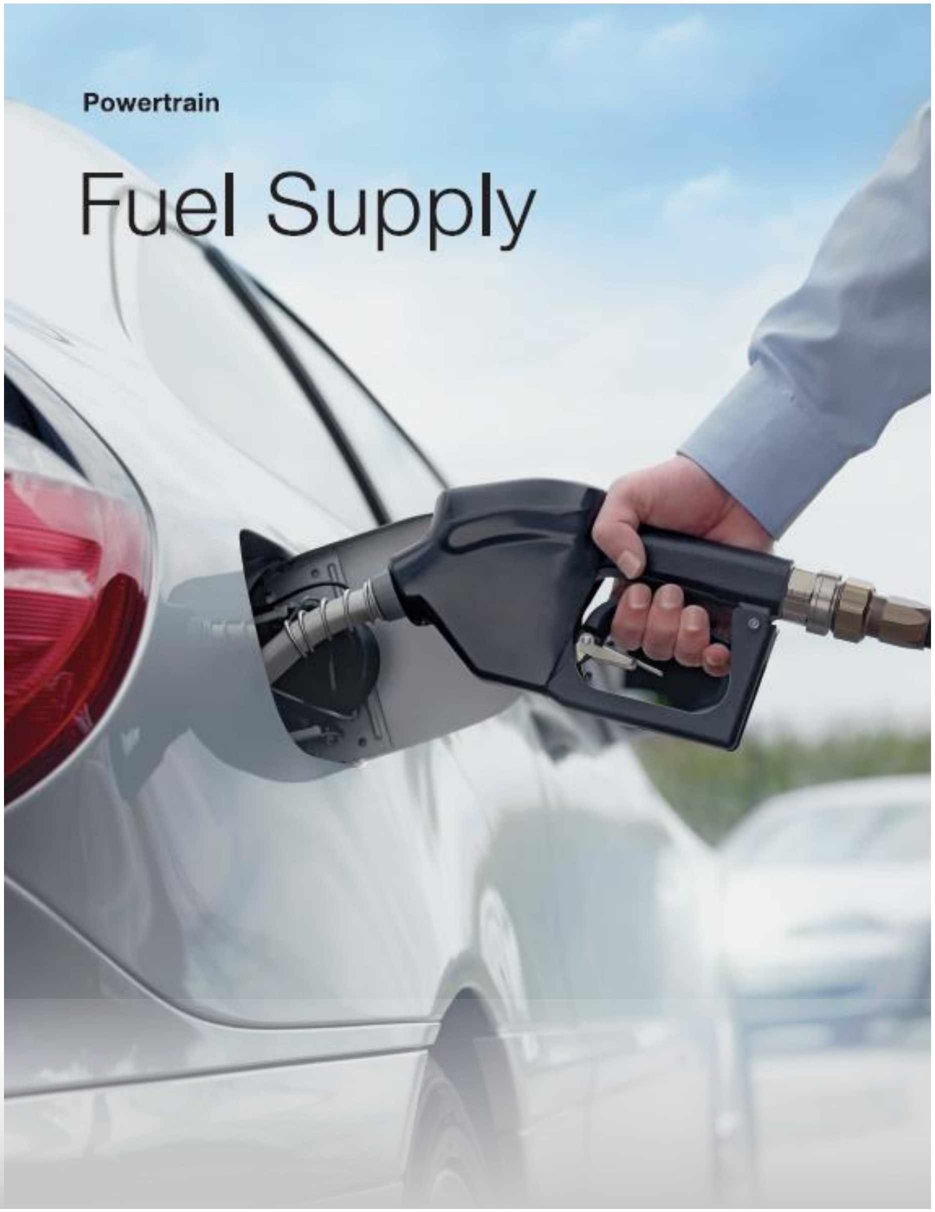
As soon as one node transmits a dominant bit (zero): Bus is in the dominant state

Only if all nodes transmit recessive bits (one): Bus is in the recessive state

- There are two bus states, called "dominant" and "recessive".
- The bus logic uses a "Wired-AND" mechanism
  - "Dominant bits" (equivalent to the logic level "Zero") overwrite the "Recessive" bits (equivalent to the logic level "One").
- Physical states (e.g. electrical voltage, light) that represent the logical levels are not defined in Bosch specification

Controller Area Network - CAN

Features



Features

Bus-access by message priority

- Carrier Sense Multiple Access / Collision Resolution

Bus access conflicts resolved by arbitration

- Bit-wise
- Non – destructive
- Allows for guaranteed latency time

Message identifier

- CAN has no node addresses

Extensive Error Checking

- Five different checks
- Every connected node participates

Features

CAN

Data Consistency Secured

- A message is accepted by all nodes or none

A Higher Layer Protocol is always required

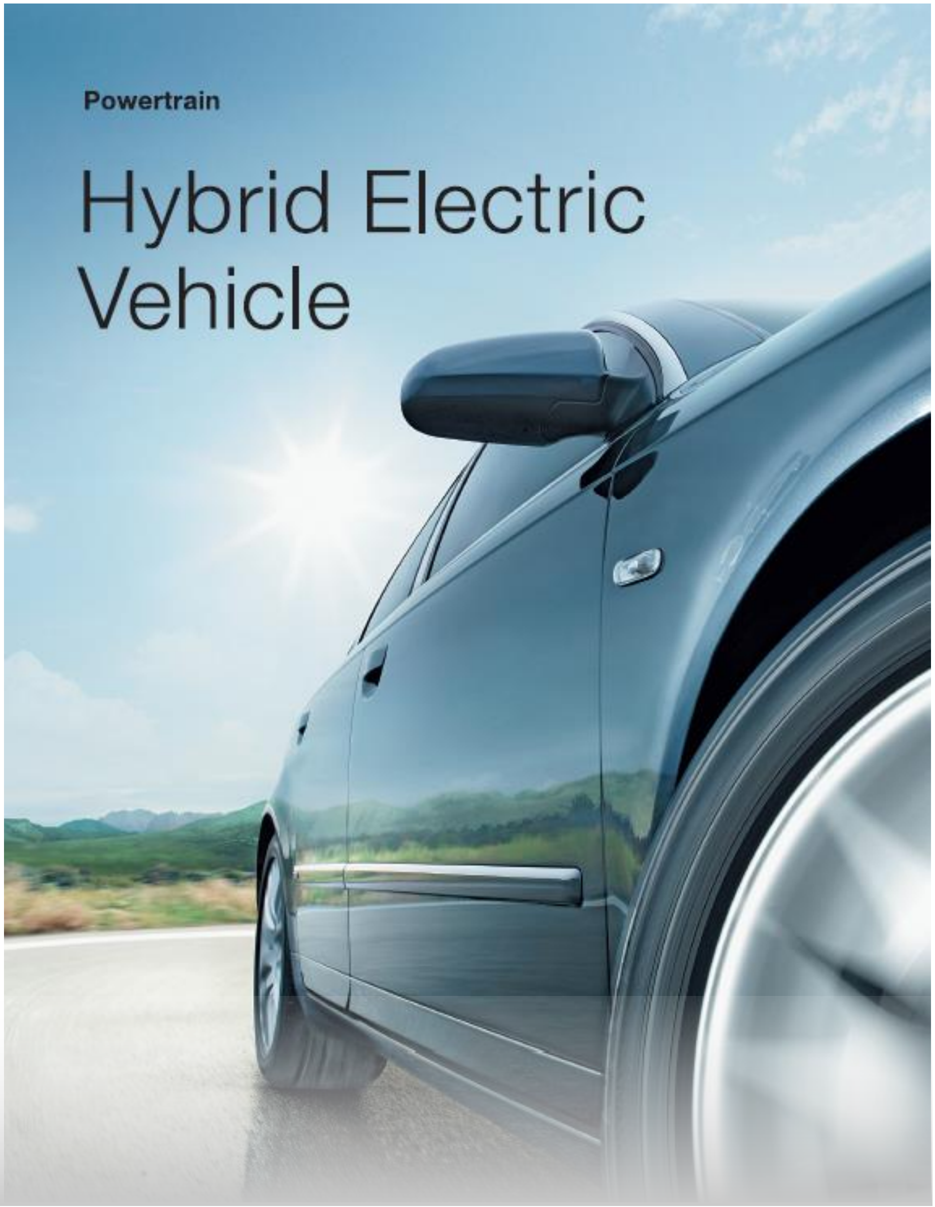
- CAN is only a low level specification

The capability of CAN is restricted by the Higher Layer Protocol chosen

- Market segment
- Real time requirements
- Product administration requirements
- Etc.

Controller Area Network - CAN

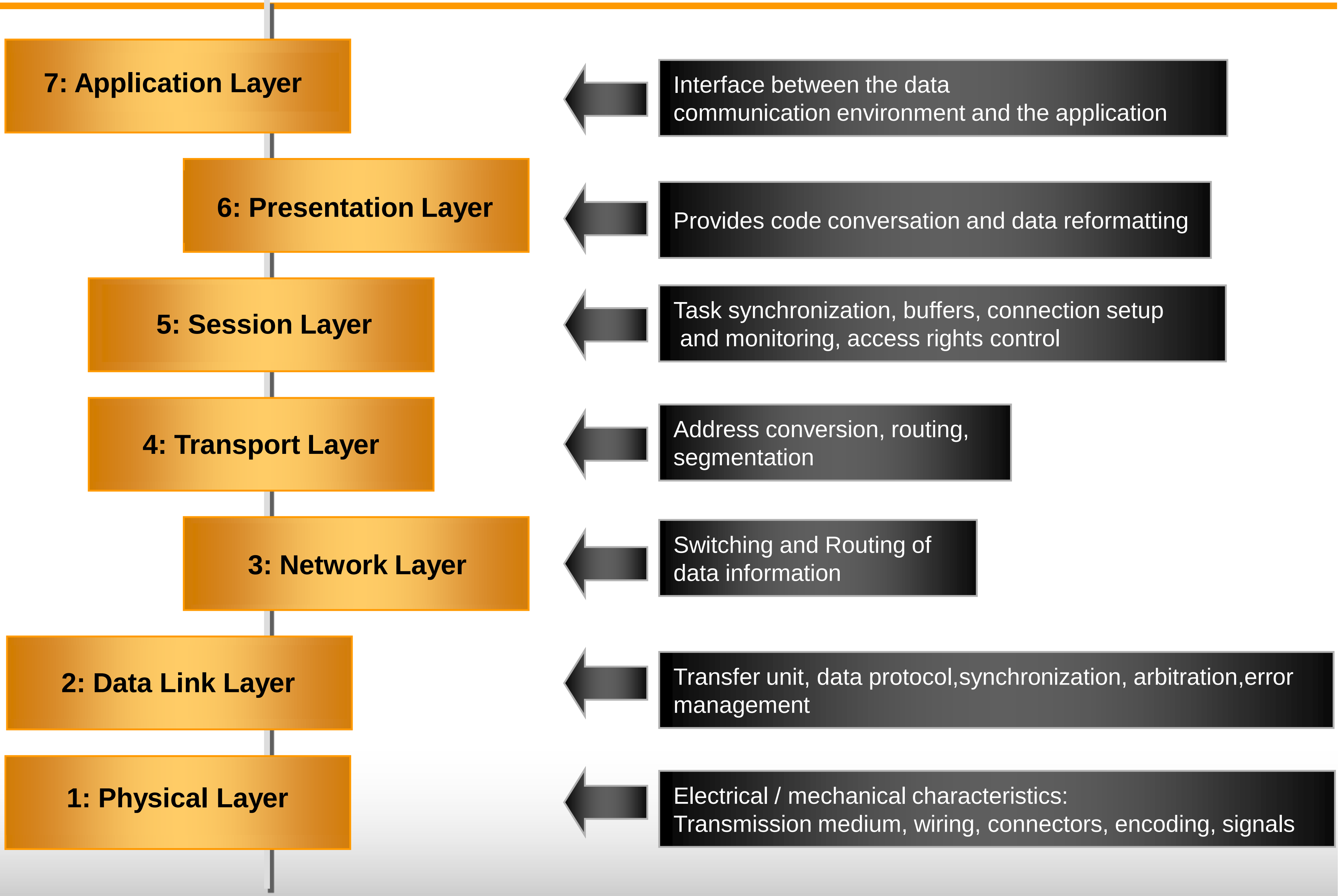
ISO / OSI Layer Structure



SO/OSI: International Standardization Organization/Open Systems Interconnect

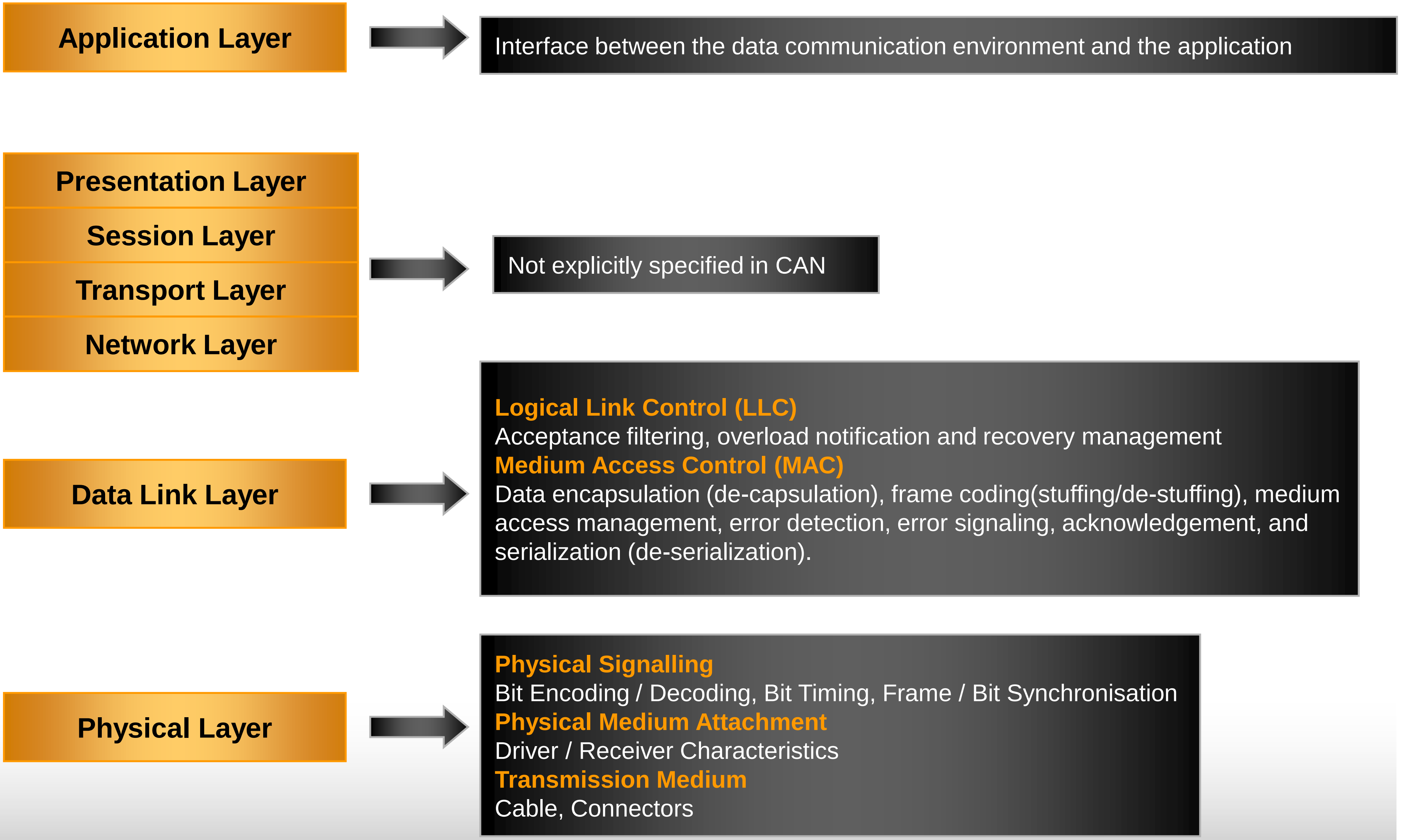
Layer Structure

CAN



7: Application Layer	Interface between the data communication environment and the application
6: Presentation Layer	Provides code conversation and data reformatting
5: Session Layer	Task synchronization, buffers, connection setup and monitoring, access rights control
4: Transport Layer	Address conversion, routing, segmentation
3: Network Layer	Switching and Routing of data information
2: Data Link Layer	Transfer unit, data protocol,synchronization, arbitration,error management
1: Physical Layer	Electrical / mechanical characteristics: Transmission medium, wiring, connectors, encoding, signals

Layer Structure



Application Layer → Interface between the data communication environment and the application

Presentation Layer
Session Layer
Transport Layer
Network Layer → Not explicitly specified in CAN

Data Link Layer →

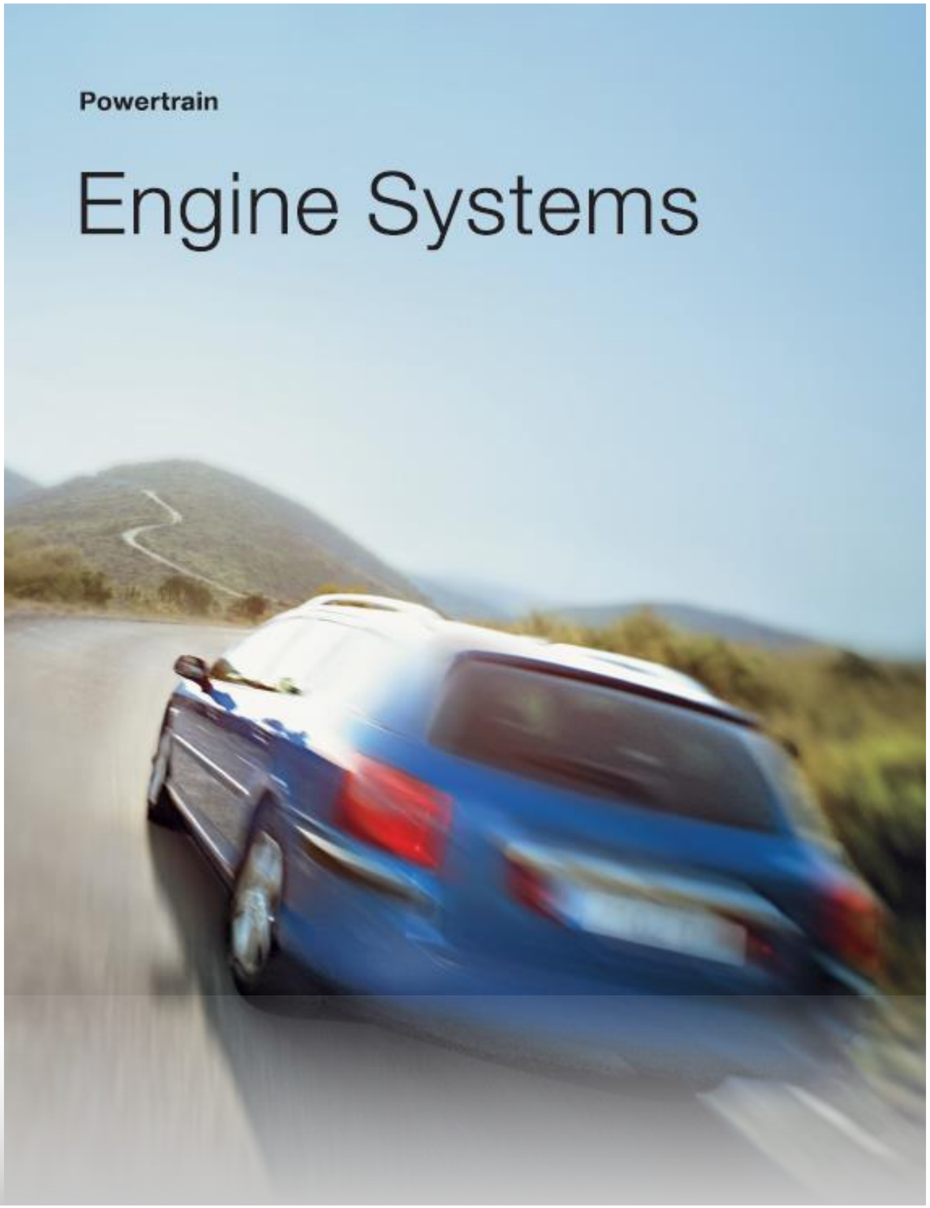
Logical Link Control (LLC)
Acceptance filtering, overload notification and recovery management
Medium Access Control (MAC)
Data encapsulation (de-capsulation), frame coding(stuffing/de-stuffing), medium access management, error detection, error signaling, acknowledgement, and serialization (de-serialization).

Physical Layer →

Physical Signalling
Bit Encoding / Decoding, Bit Timing, Frame / Bit Synchronisation
Physical Medium Attachment
Driver / Receiver Characteristics
Transmission Medium
Cable, Connectors

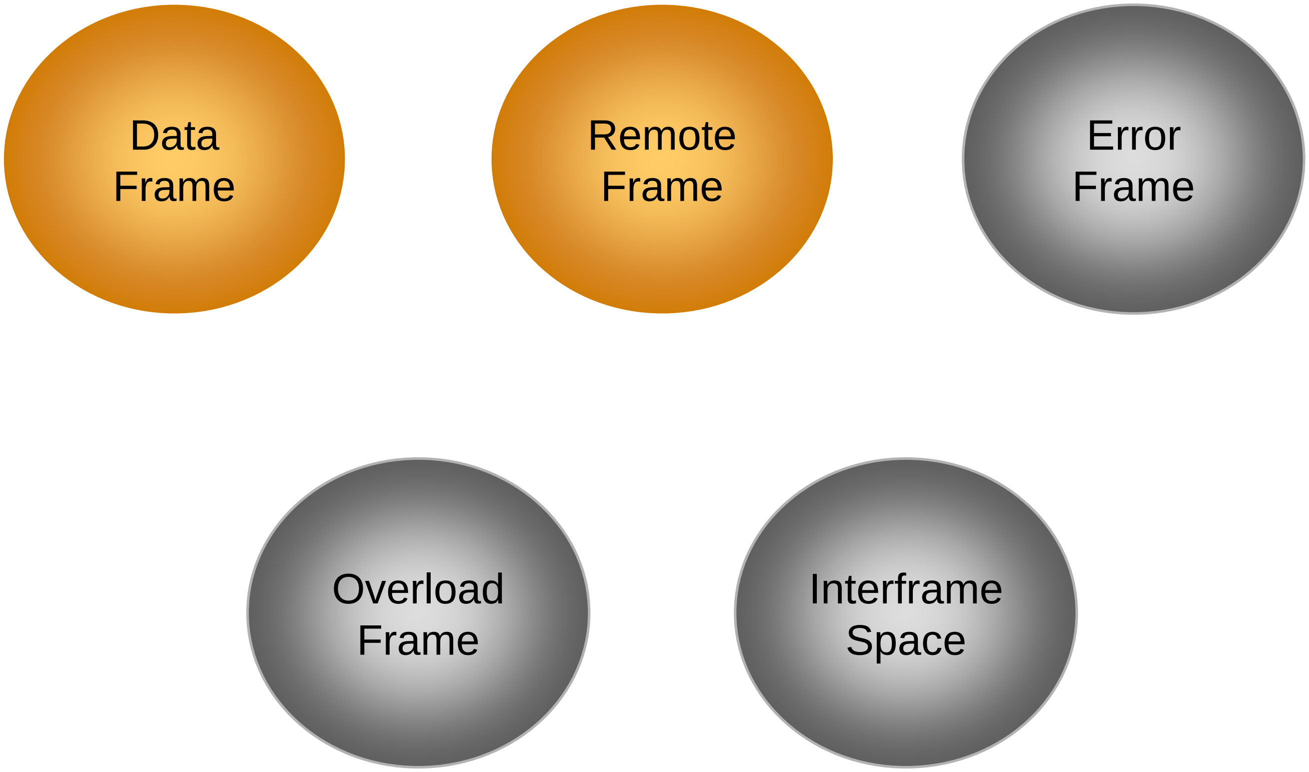
Controller Area Network - CAN

Frames



Abraham Tezmol, P ES FSD GDL

Frames - Types



Data Frame

Remote Frame

Error Frame

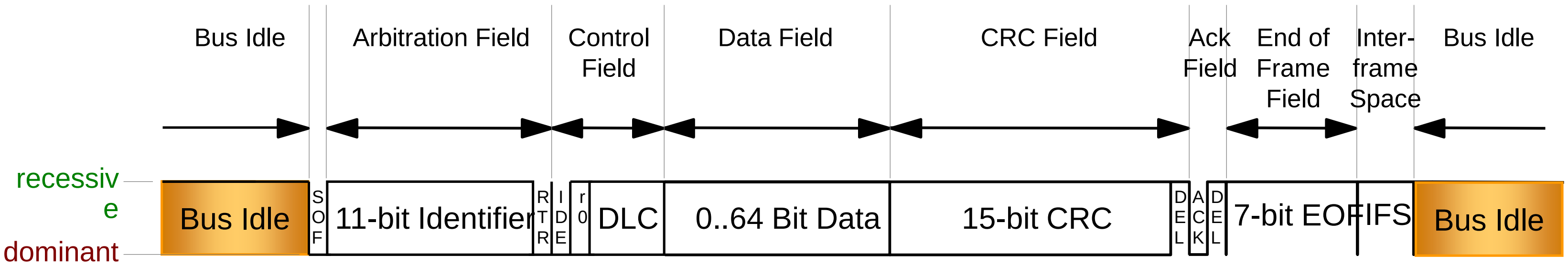
Overload Frame

Interframe Space

Frames – Data Frame

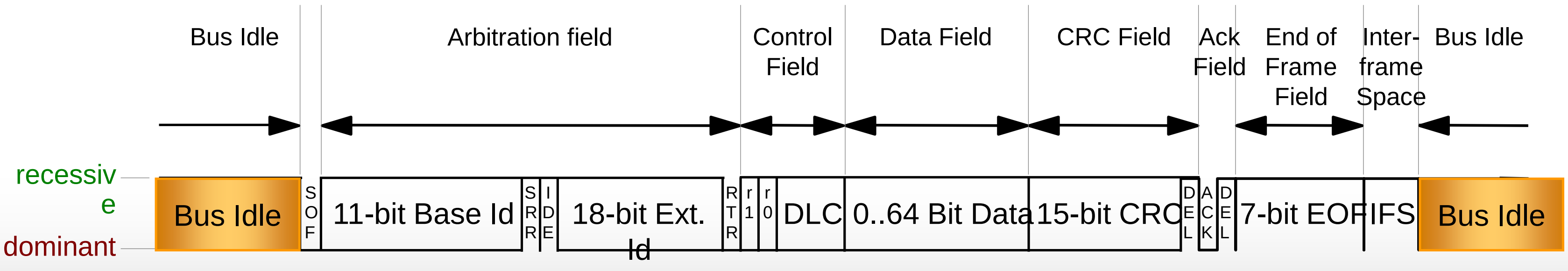
Standard Format (CAN 2.0A): 11-bit Identifier

2^{11} identifiers possible



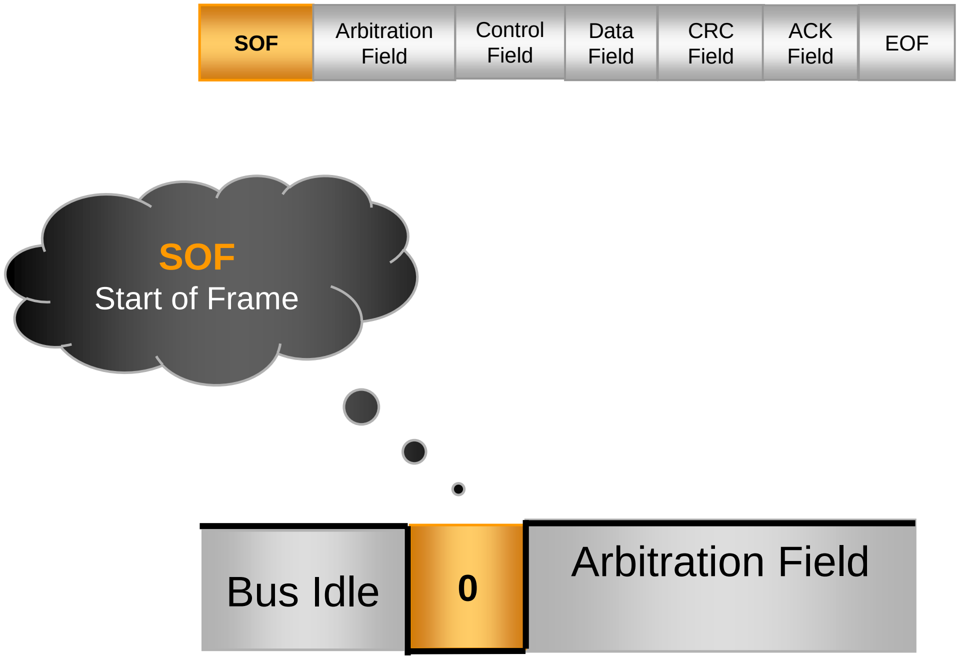
Extended Format (CAN 2.0B): 29-bit Identifier

2^{29} identifiers possible



Frames – Data Frame

CAN



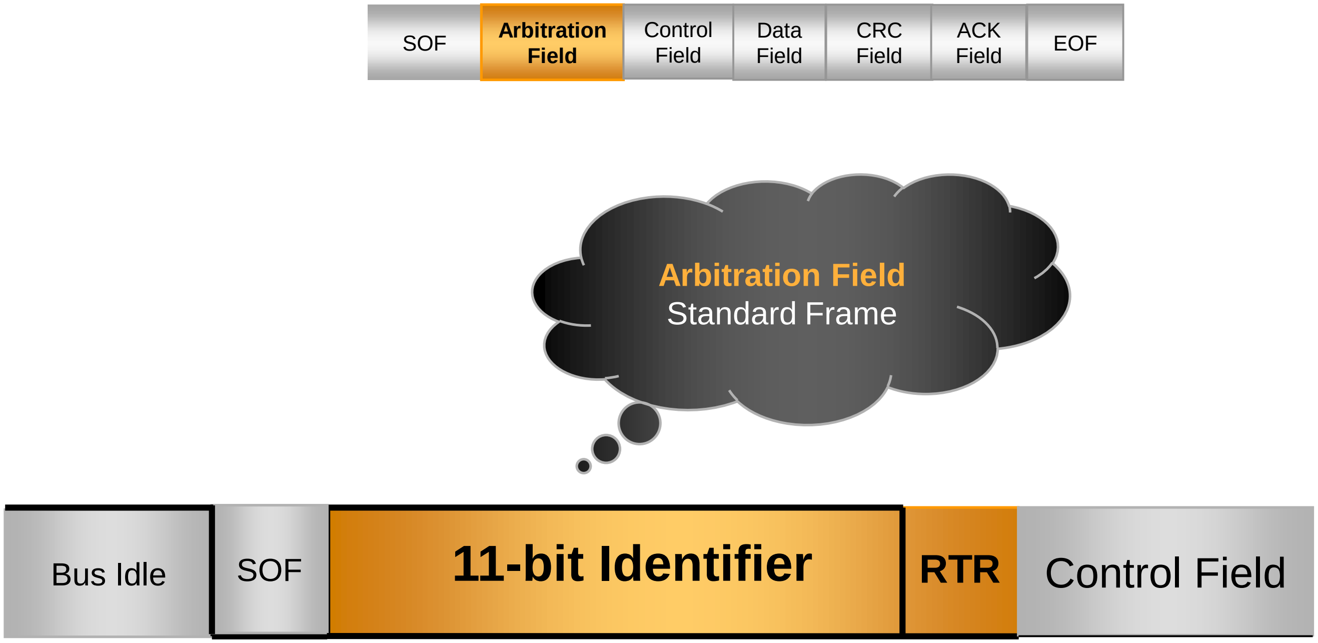
SOF	Arbitration Field	Control Field	Data Field	CRC Field	ACK Field	EOF

SOF
Start of Frame

Bus Idle | 0 | Arbitration Field

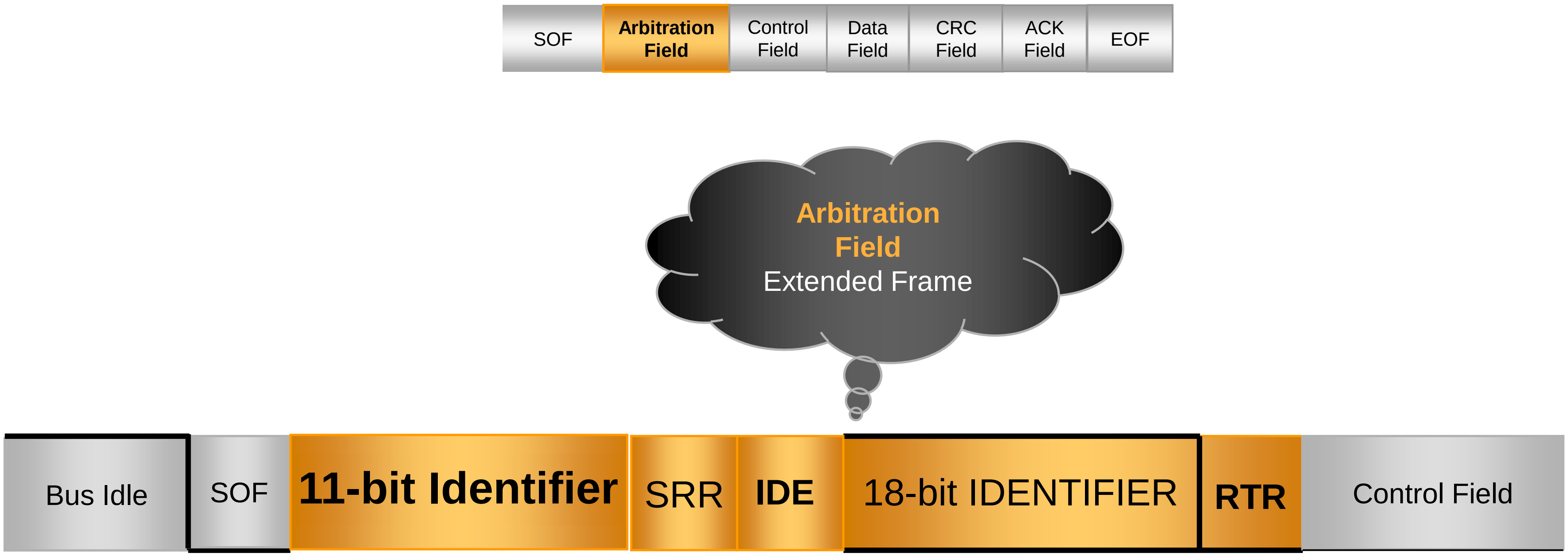
- Marks the start of any CAN frame
- It is always a dominant bit
- Provides a falling edge for hard synchronization of transmitter and receivers

Frames – Data Frame



- Contains the identifier (11 bit for CAN 2.0A) which is used for arbitration
- Identifier determines frame priority: low identifier = high priority
- The highest seven bits of the ID must not be all recessive
- Remote Transmission Request (RTR) bit is always ***dominant*** in a Data Frame

Frames – Data Frame



SOF	**Arbitration Field**	Control Field	Data Field	CRC Field	ACK Field	EOF

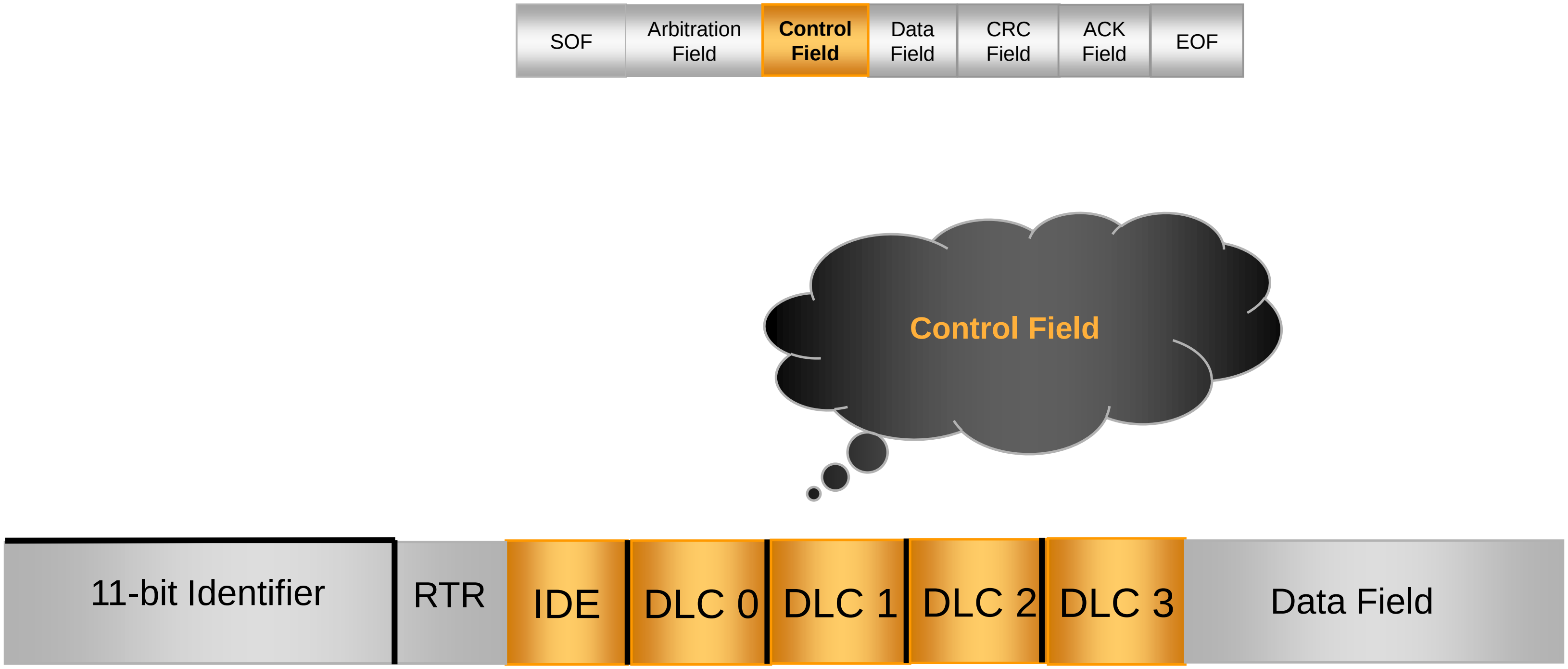
Arbitration Field
Extended Frame

Bus Idle	SOF	**11-bit Identifier**	SRR	**IDE**	18-bit IDENTIFIER	**RTR**	Control Field

- Contains the identifier (29 bit for CAN 2.0B) which is used for arbitration
- Identifier determines frame priority: **low** identifier = **high** priority
- The highest seven bits of the identifier must not be all recessive
- Remote Transmission Request (RTR) bit is always dominant in a Data Frame
- Substiture Remote Request (SRR) bit is always *recessive* in a Data Frame

Frames – Data Frame

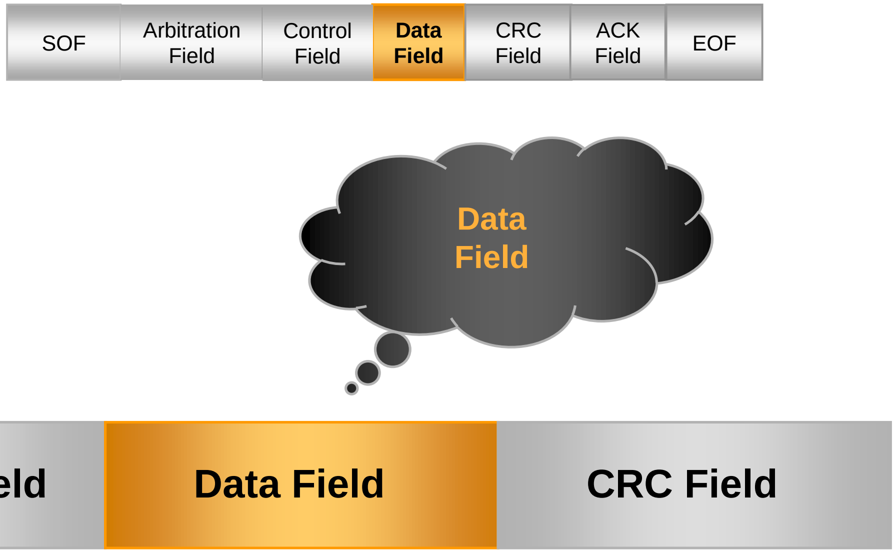
CAN



- 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 (DCL, 4 bits) indicates number of data bytes in Data field; may take values ranging from 0 to 8, other values are not allowed

Frames – Data Frame

CAN



SOF | Arbitration Field | Control Field | **Data Field** | CRC Field | ACK Field | EOF

Data Field

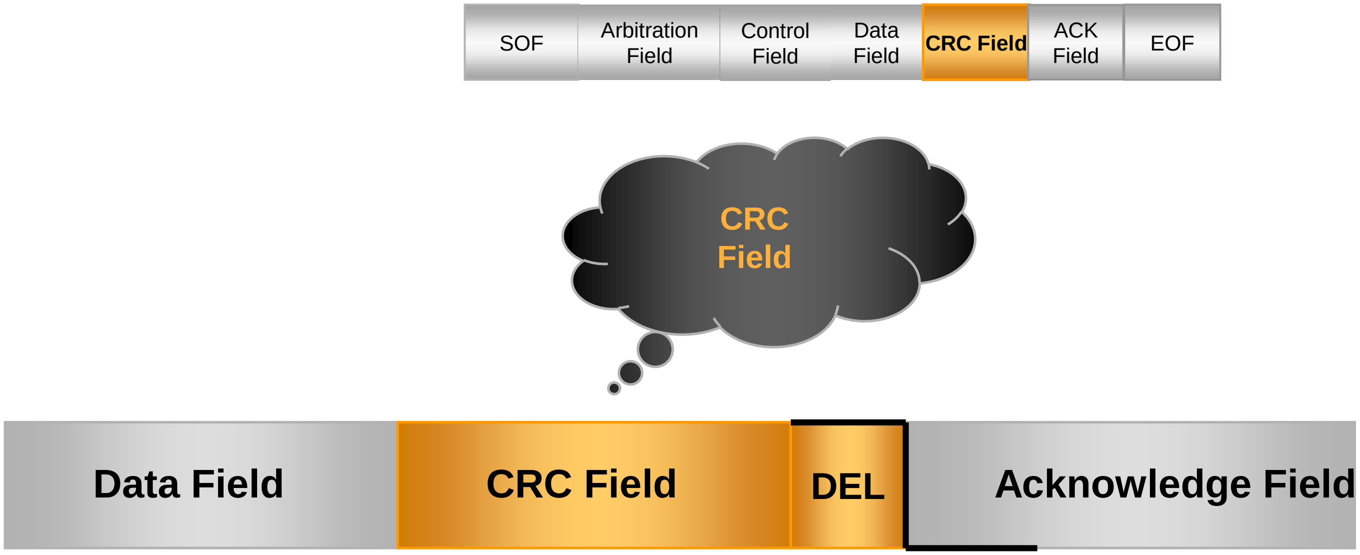
Control Field | Data Field | CRC Field

0 Data Bytes	**e.g. to indicate an event**
1 Data Byte	**low net data rate on the CAN bus**
8 Data Bytes	**high net data rate on the CAN bus**

- 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**

Frames – Data Frame

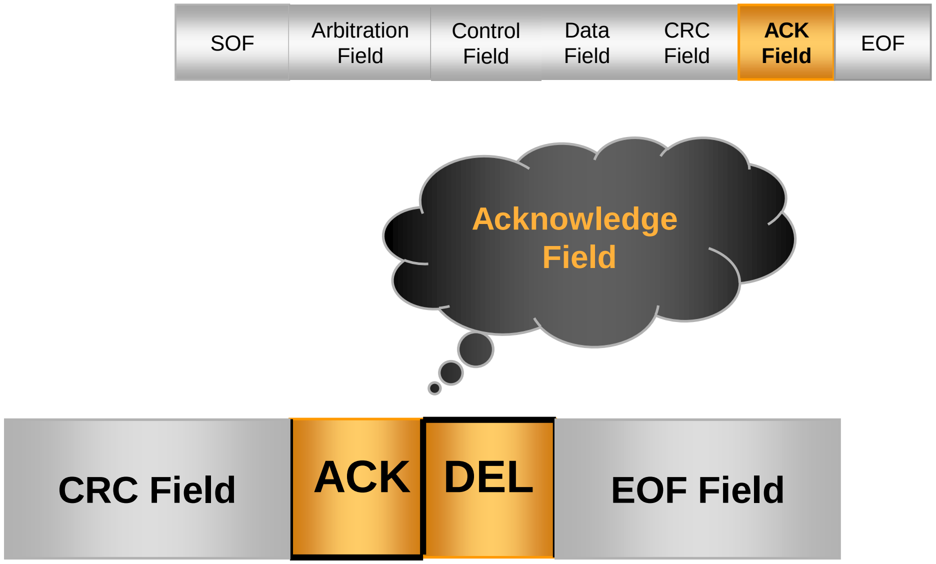
CAN



- Contains the 15-bit Cyclic Redundancy Check (CRC) code
- Use polynomial generator: $\mathbf{x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1}$
- 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**

Frames – Data Frame

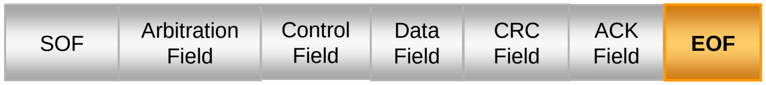
CAN

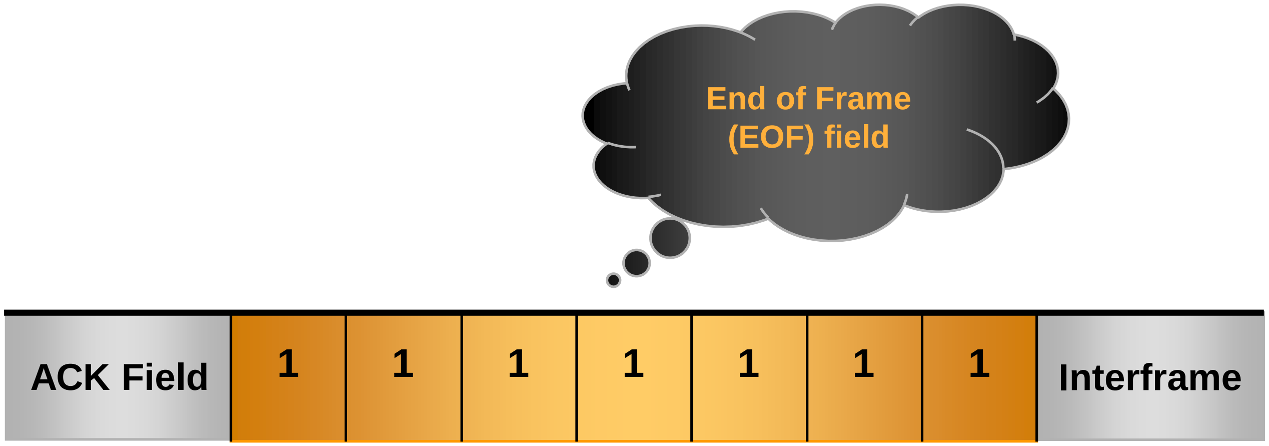


- Contains the Acknowledgement (ACK Slot) bit
- The Acknowledgement bit can be *dominant* (TX) or *recessive* (RX)
- The ACK Field Delimiter (DEL) marks the end of the ACK field
- The ACK Field Delimiter is always **recessive**

Frames – Data Frame

CAN

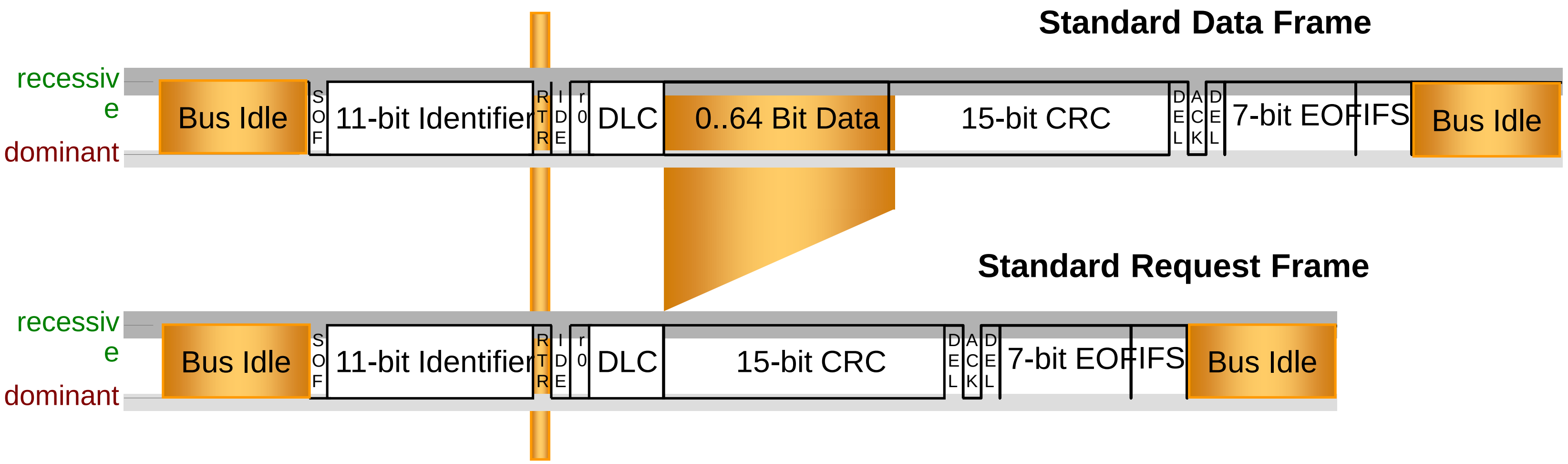




- Consists of seven (7) consecutive **recessive** bits
- Marks the end of the Data Frame
- Follows the Acknowledge (ACK) field
- Is followed by the Interframe Space (IFS)

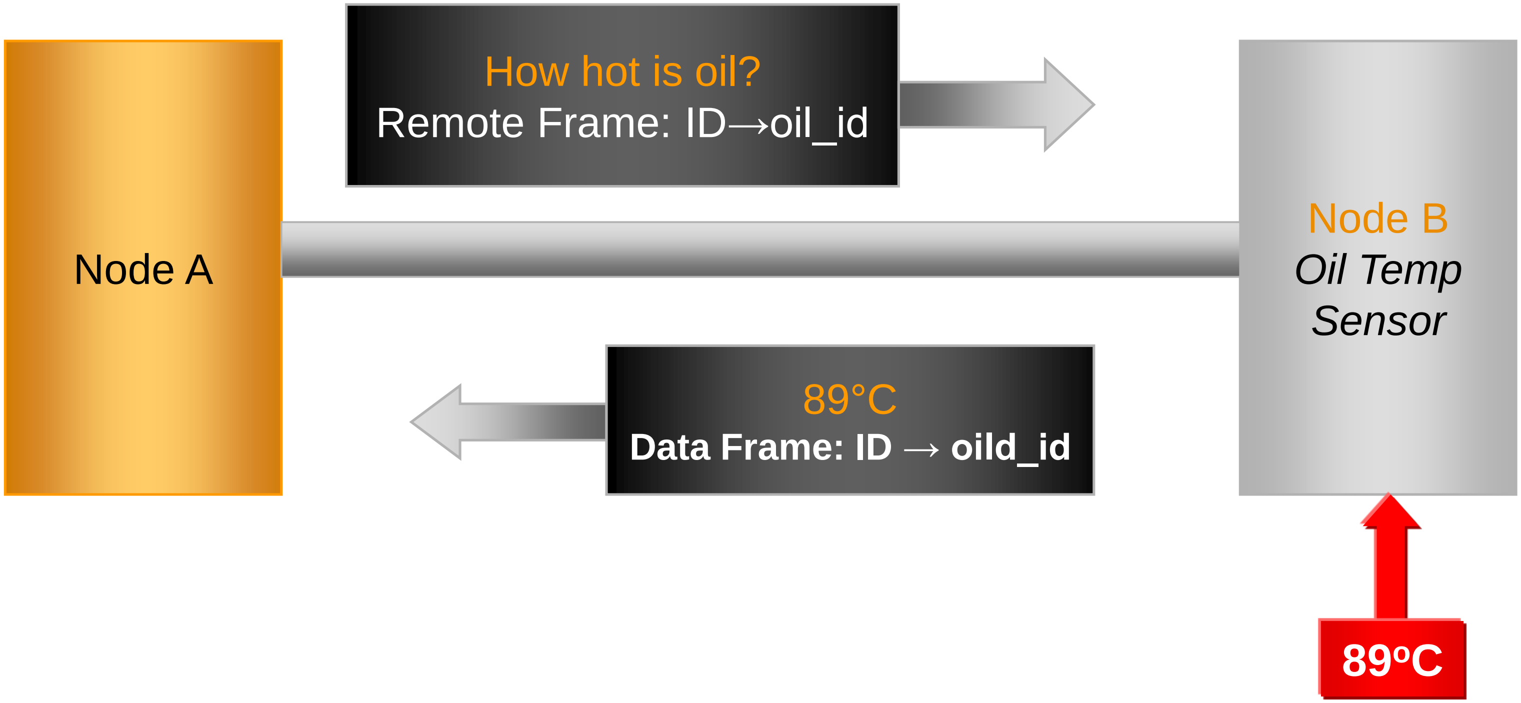
Frames – Remote Frame

CAN



- 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

Frames – How is a Remote Frame used?



How hot is oil?
Remote Frame: ID→oil_id

Node A

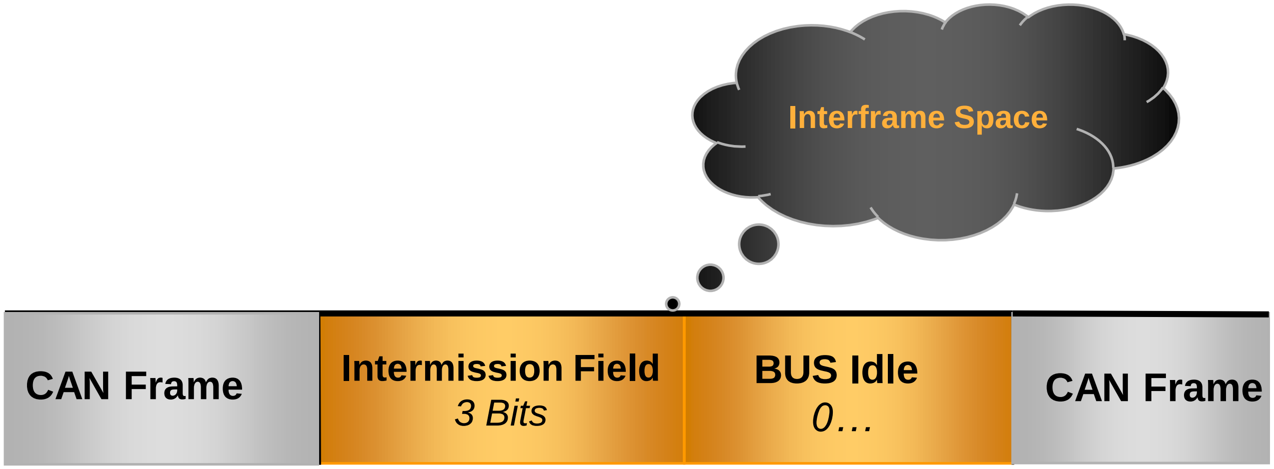
Node B
Oil Temp Sensor

89°C
Data Frame: ID → oild_id

89^{o}C

- If a node 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.

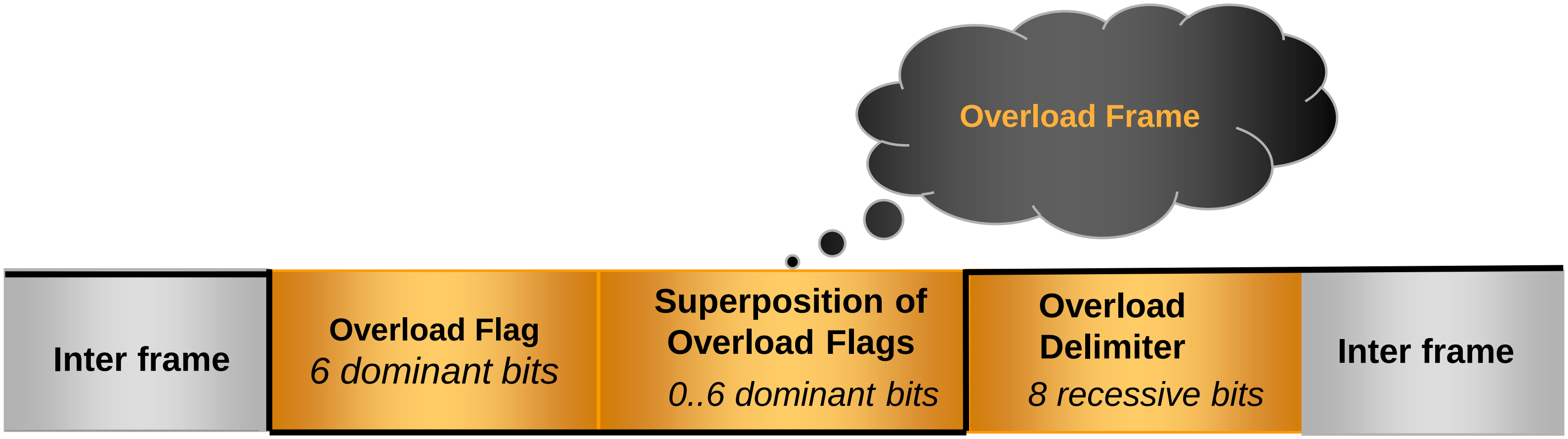
Frames – Interframe Space



- Consists of three consecutive **recessive** bits
- Separates two consecutive frames from each other
- No transmission is allowed during the Inter frame Space (IFS)
- It is needed by controllers to copy received frames from their Rx buffers
- ACK Field Delimiter + EOF + IFS = 11 consecutive recessive bits
- It might be followed by “Bus Idle” sequence of indefinite length

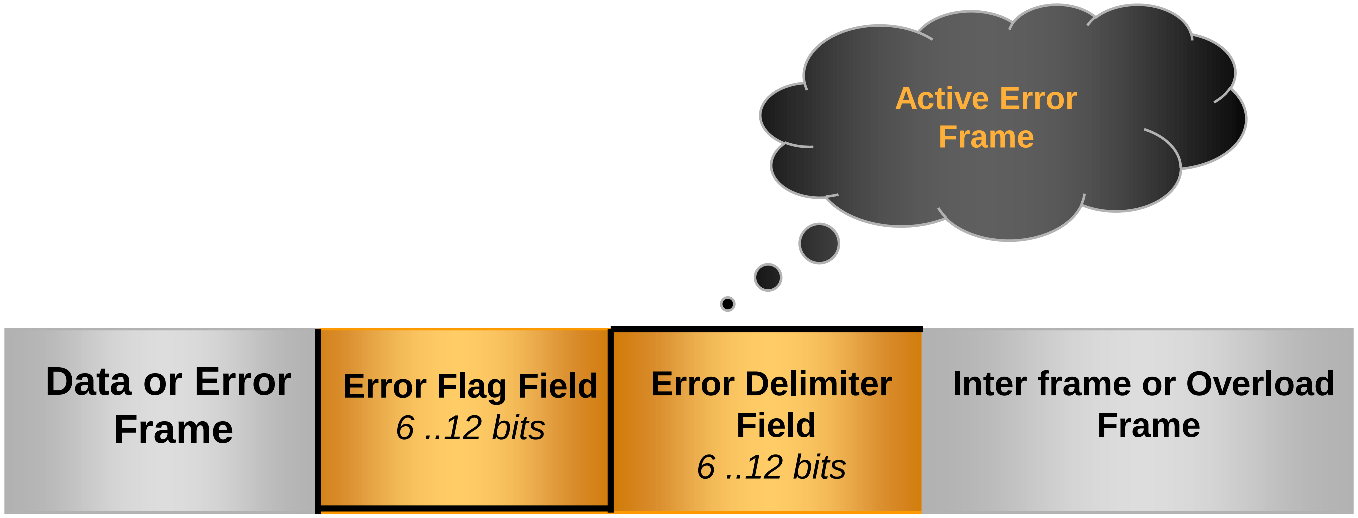
Abraham Tezmol, P ES FSD GDL

Frames – Overload Frame



- Unit sends Overload Frame when at present it cannot receive frames
- Transmission of an Overload Frame is started during the first two bits
- 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)

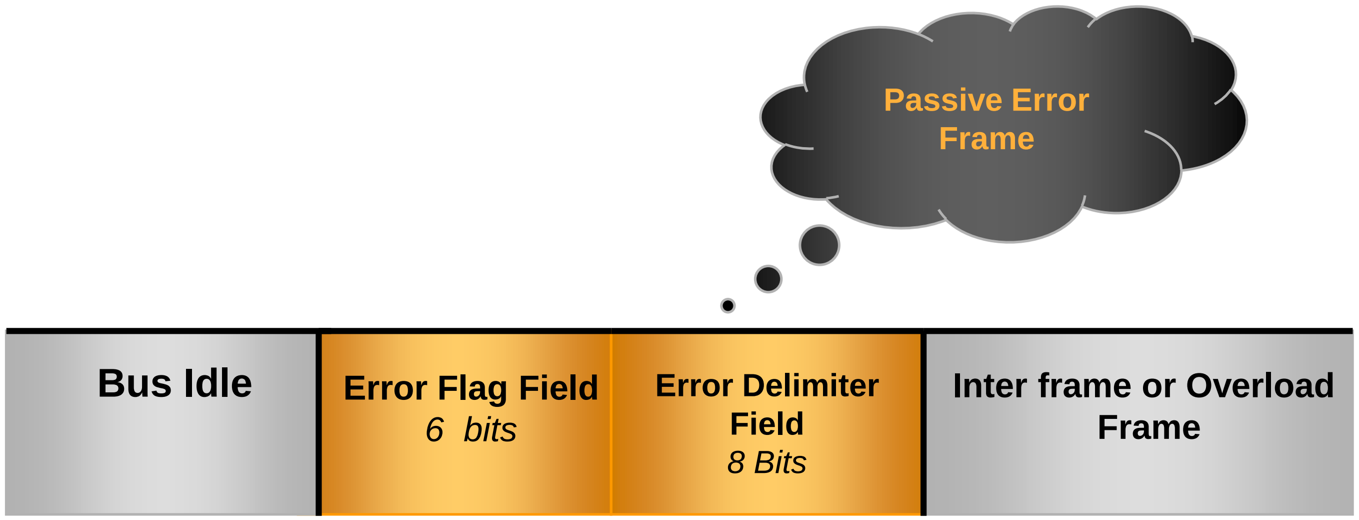
Frames – Active Error Frame



- Sender and receivers reject erroneous frame completely and do not process it any further
- Error flag actively violates the bit stuffing rule
- After the Error Delimiter bus activity returns to normal and the interrupted node attempts to re send the aborted message

Frames – Passive Error Frame

CAN



Passive Error Frame

Bus Idle

Error Flag Field
6 bits

Error Delimiter Field
8 Bits

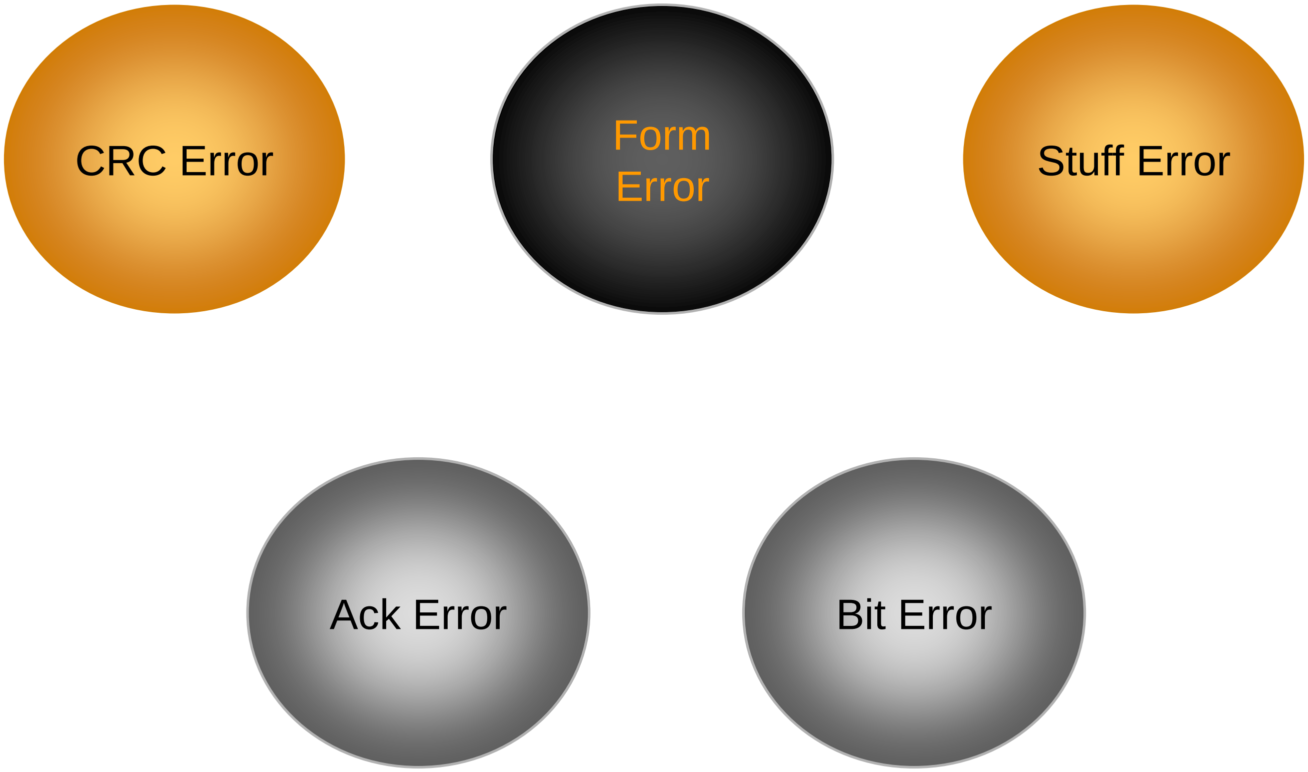
Inter frame or Overload Frame

- Can still receive frames like a unit in error active state Error flag actively violates the **bit stuffing rule**
- Has to wait after transmission of a Data Frame for **8 recessive bit cycles** on the bus until it is permitted to transmit another Data Frame
- Can go back to error active state for TEC <= 127 **AND** REC <= 127

Controller Area Network - CAN

Error Detection

Error Detection - Overview



CRC Error

Form Error

Stuff Error

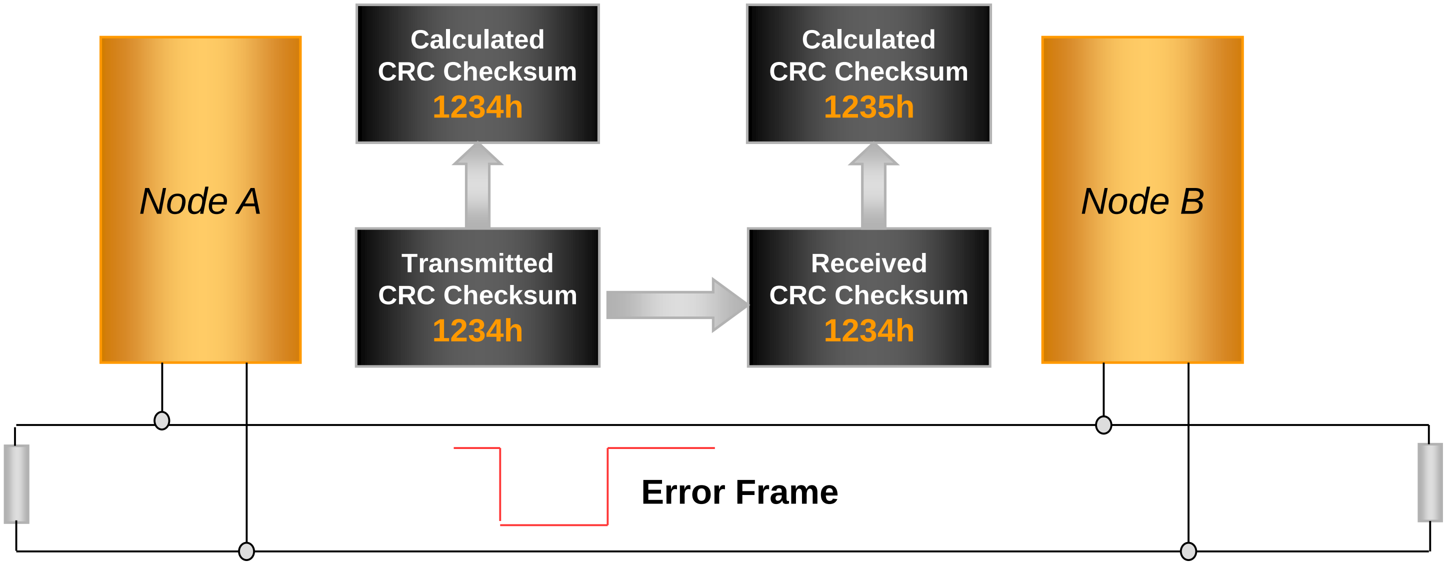
Ack Error

Bit Error

Error Detection – Error Types

- **Bit error** → transmitted and received bit are different
  - except in arbitration, acknoledge and passive error
- **Bit stuffing error** → More than five bits of equal polarity inside of a frame are detected
- **CRC error** → Received CRC code does not match with the calculated code
- **ACK error** → Transmitting node receives no dominant acknowledgement bit
  - no receiver node accepts the transmission message
- **Form error** → Fixed-form bit field contains one or more illegal bits
  - e.g. violation of end of frame EOF format, CRC or ACK delimiter

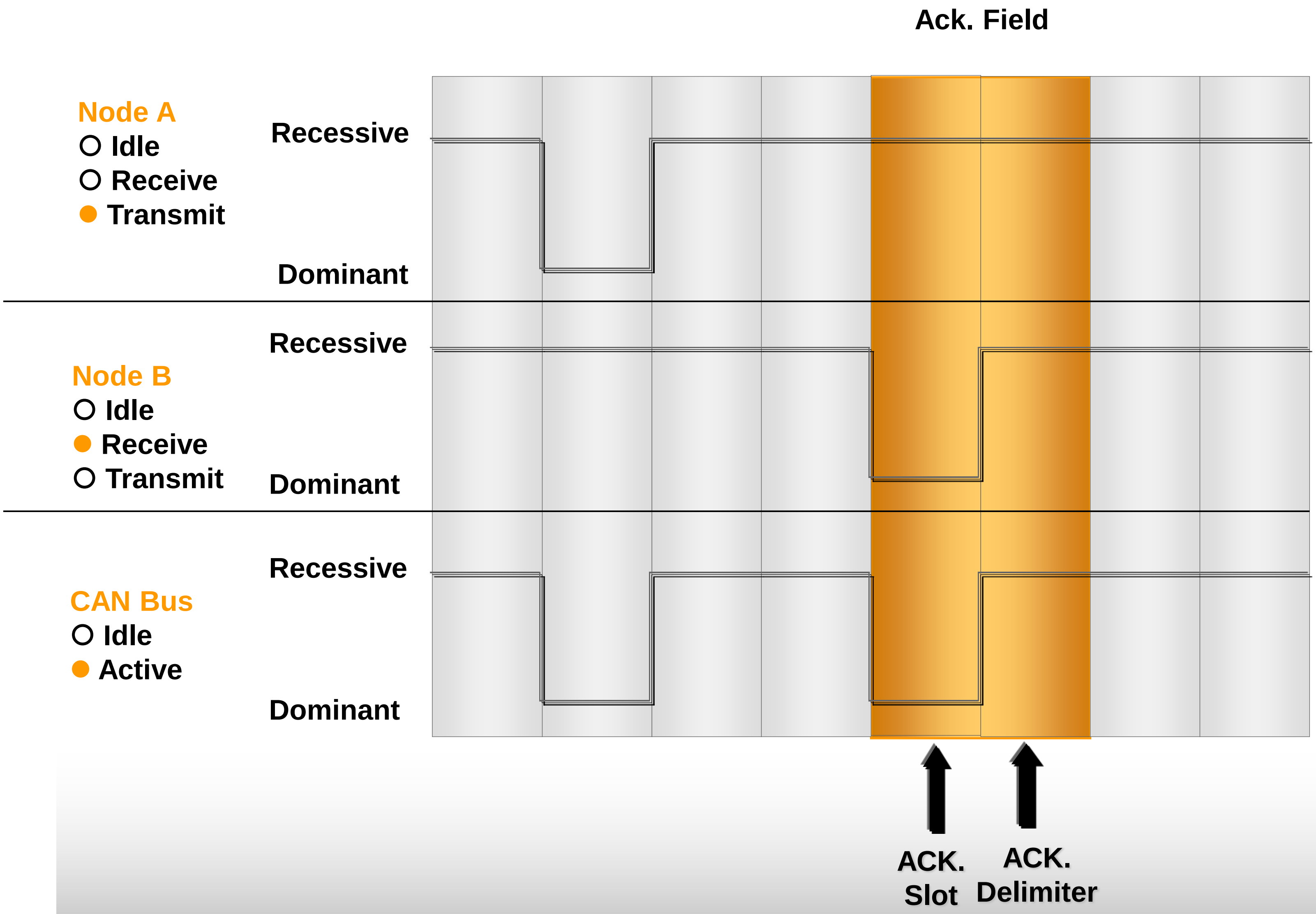
Error Detection – Cyclic Redundancy Check


Node A
Calculated CRC Checksum 1234h
Transmitted CRC Checksum 1234h
Calculated CRC Checksum 1235h
Received CRC Checksum 1234h
Node B
Error Frame

- This CRC sequence is transmitted in the CRC Field of the CAN frame.
- The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence.
- If node B detects a mismatch between the calculated and the received CRC sequence, then a CRC error has occurred.
- Node B discards the message and transmits an Error Frame to request retransmission of the garbled frame.

Error Detection – Acknowledge Procedure

CAN



Ack. Field

Node A
○ Idle
○ Receive
● Transmit

Recessive
Dominant

Node B
○ Idle
● Receive
○ Transmit

Recessive
Dominant

CAN Bus
○ Idle
● Active

Recessive
Dominant

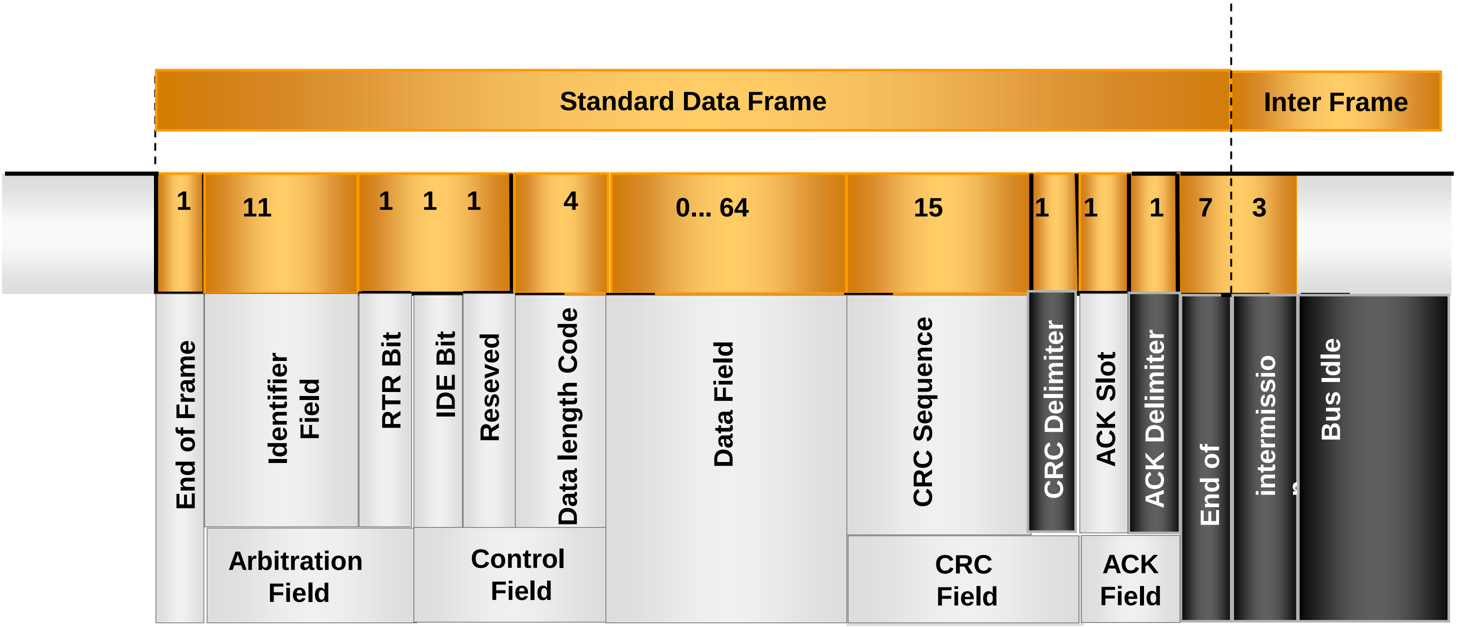
ACK. Slot

ACK. Delimiter

- ● A frame must be acknowledged by at least one other node. Otherwise ACK-Error

Error Detection – Frame Check

CAN



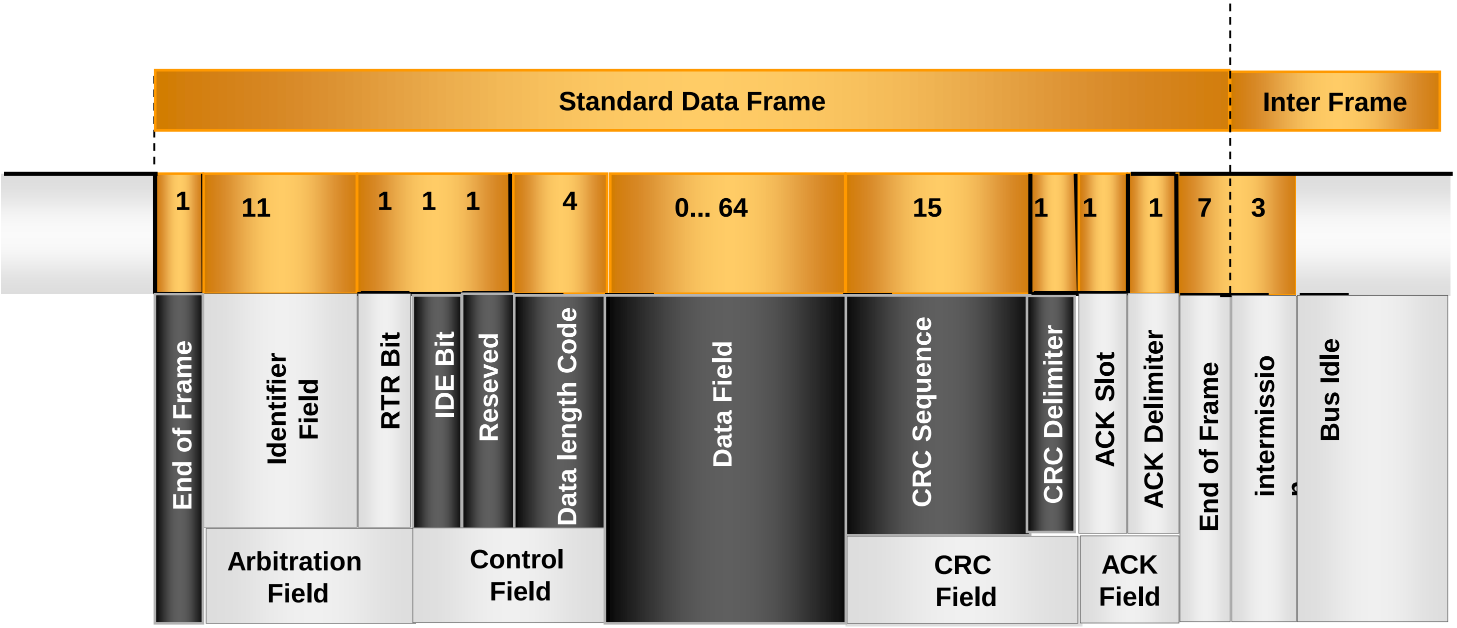
If a transmitter detects a dominant bit in one of the four segments:

- CRC Delimiter
- Acknowledge Delimiter
- End of Frame
- Interframe Space

Then a Form Error has occured and an **Error Frame** is generated. The message will then be **repeated**

Error Detection – Bit Monitoring

CAN

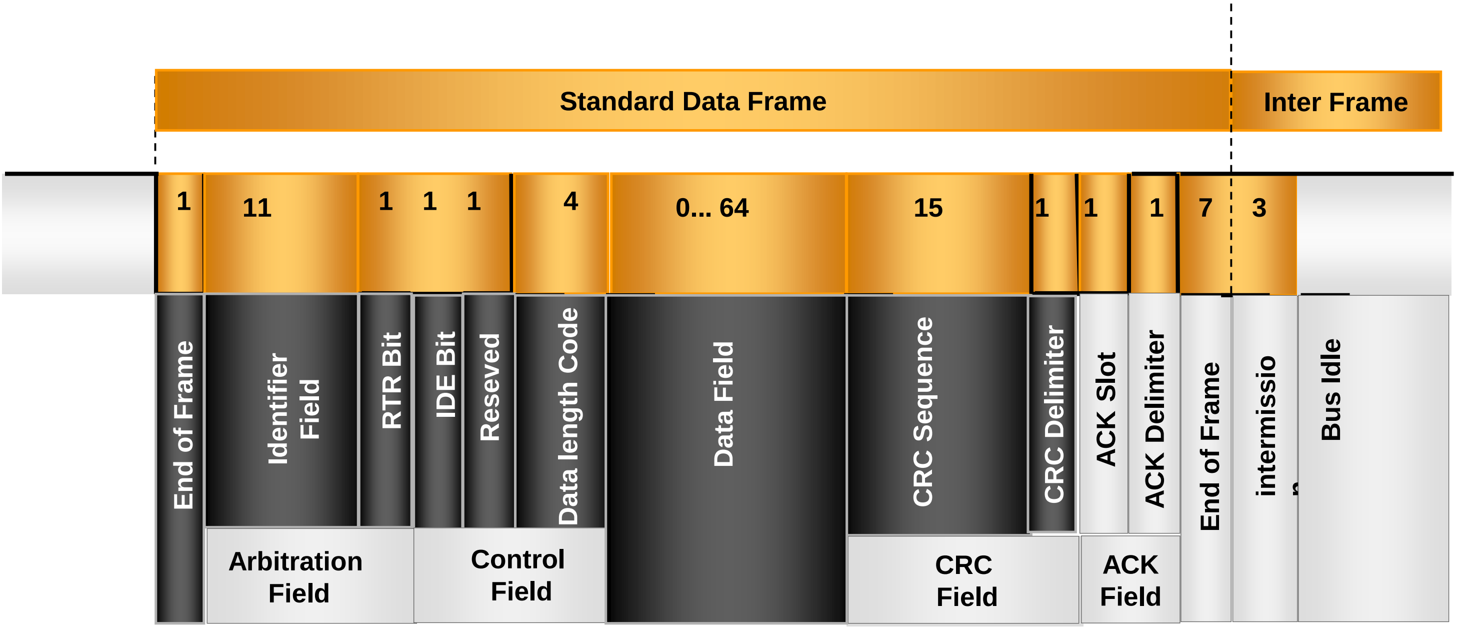


Bit error occurs if a transmitter

- Sends a dominant bit but detects a recessive bit on the bus line or
- Sends a recessive bit but detects a dominant bit on the bus line
- An Error Frame is generated and the message is repeated
- When a dominant bit is detected instead of a recessive bit, no error occurs during the **Arbitration Field** or the **Acknowledge Slot** because these fields must be able to be overwritten by a dominant bit in order to achieve arbitration and acknowledge functionality

Abraham Tezmol, P ES FSD GDL

Error Detection – Bit Stuffing Check



Standard Data Frame

Inter Frame

1
11
1
1
1
4
0... 64
15
1
1
1
7
3

End of Frame

Identifier Field

RTR Bit

IDE Bit

Reseved

Data length Code

Data Field

CRC Sequence

CRC Delimiter

ACK Slot

ACK Delimiter

End of Frame

intermissio n

Bus Idle

Arbitration Field

Control Field

CRC Field

ACK Field

- If six consecutive bits with the same polarity are detected between Start of Frame and the CRC Delimiter, the bit stuffing rule has been violated
- A stuff error occurs and an Error Frame is generated. The message is then repeated

Error Detection – Error Management Unit

- The error management unit is used to:
  - **Cancel erroneous** messages at all CAN nodes
  - **Disconnect nodes** from the CAN bus, if they detect/generate too many errors
- The error management unit uses two error counters:
  - A receive error counter **REC**
  - A transmit error counter **TEC**
- The CAN controller may work in three different states:
  - **Error active state** → for each detected error an active error flag is generated
    - Queue of six consecutive dominant bits
  - **Error passive state** → for each detected error a passive error flag is generated
    - Queue of six consecutive recessive bits
  - **Bus off state** → if too many errors are detected by one node, this node is automatically disconnected from the bus

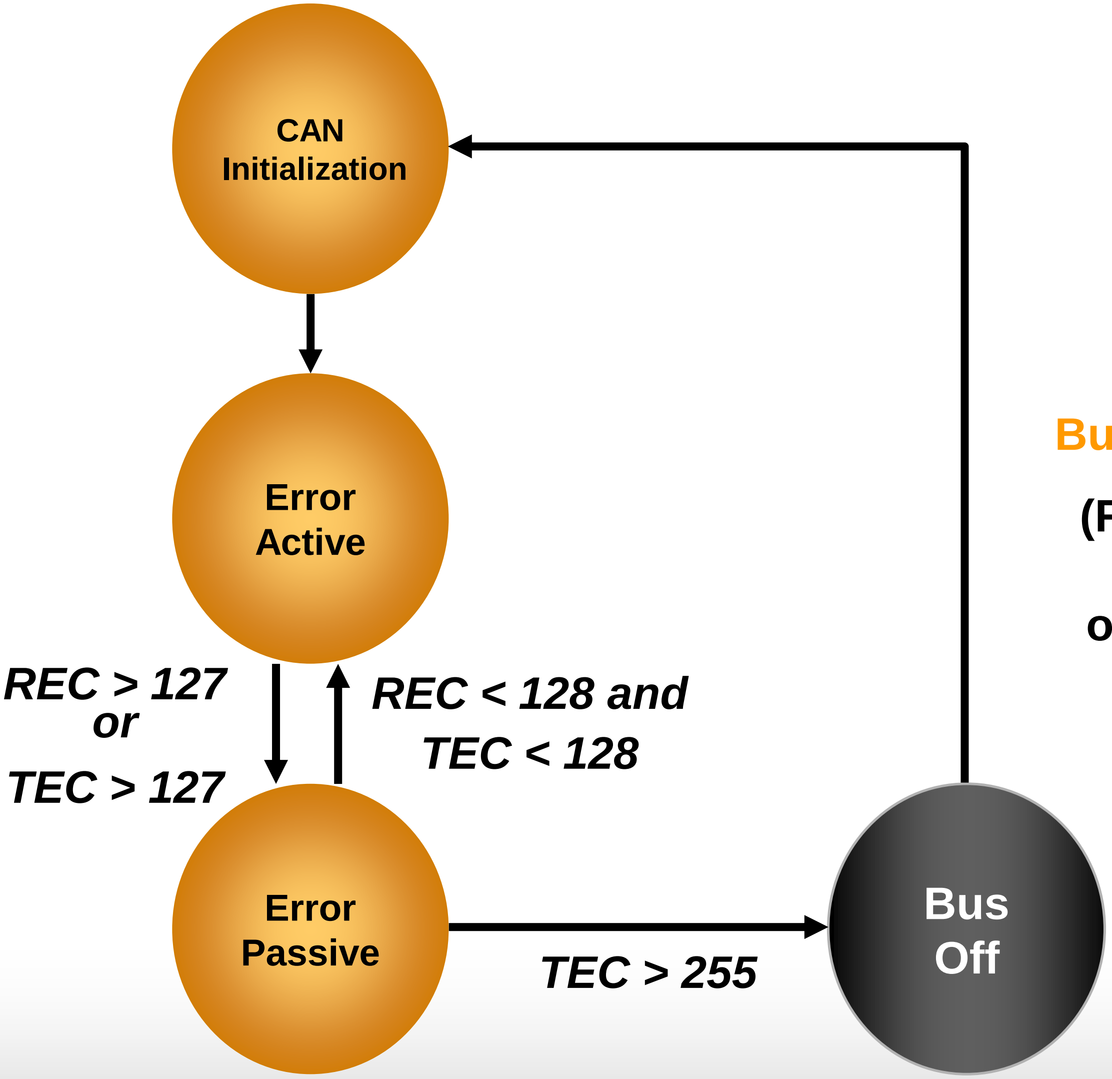
Error Detection – Status of a CAN Node

- The CAN node status levels are switched depending upon the values of TEC and REC:

  - Error active — Transmit Error Counter TEC and Receive Error Counter REC are less than 128 (CAN node status after reset: TEC = REC = 0)
  - Error passive — TEC or REC is greater than 127 and TEC is less than 255
  - Bus off — TEC is greater than 255

Status of Node	Error Counter	Generated Error Flag
Error active	TEC and REC ≤ 127	6 dominant bits
Error passive	TEC or REC > 127 and ≤ 255	6 recessive bits
Bus off	TEC > 255	

Error Detection – Operational States



Bus off recovery sequence:

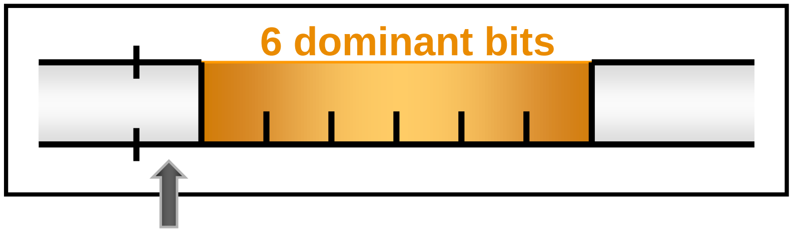
(Reset + Configuration or Configuration) and occurrence of 128 times 11 recessive bits

Error Detection – Steps of Error Handling

CAN

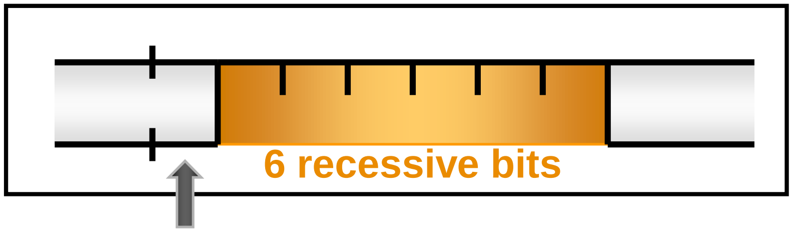
- Each CAN node which detects an error - local or global error - sends an error flag to inform all other stations about this error (globalization)
- An error flag is a queue of six consecutive bits

Error active node :



Error condition

Error passive node :

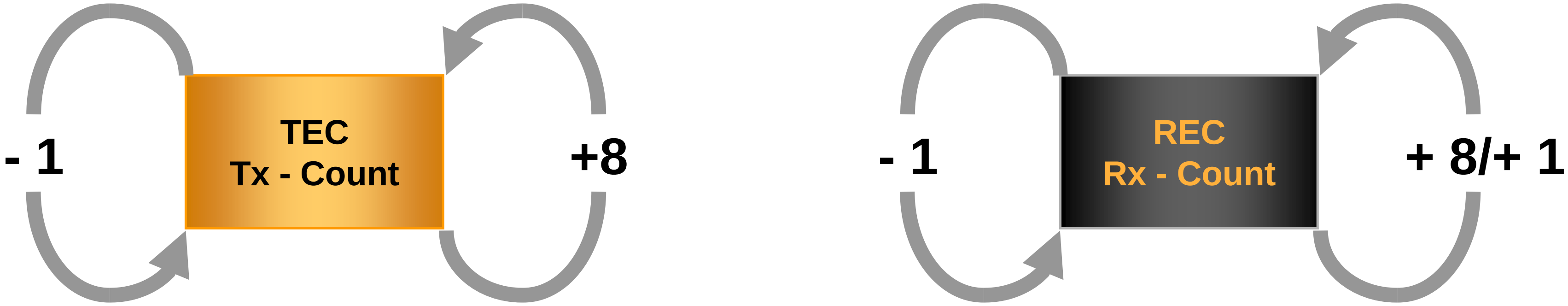


Error condition

- In the case of one node detects a local error it will send an error flag. Other nodes may detect an error because of this error flag. They send also an error flag.
- After receiving/sending an error flag all nodes cancel the message.
- Each node which detects an error increments his error counter(s).
- The transmitter of a cancelled message retransmits automatically this message.

Error Detection – Error Counters

- The **ERROR MANAGEMENT UNIT** serves two **ERROR COUNTER**



- **Successful transmission** (incl. ACK)

- **Bit Error** detected (e.g. can't write dominant bit)
- **8th consecutive dominant bit following an Error Flag**
- **no Acknowledge bit** (only in Error Active mode)
- **error detected in EOF**

- **Successful reception** (incl. ACK)

- **node detects an error** (+1)
- **error detected in** first 6 **bits of EOF** (+1)
- **transmit node is sending an Error Flag** (+8)
- **Bit Error** detected (+8)
- **8th consecutive dominant bit following an Error Flag** (+8)

Error Detection – 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 **bus off** state a unit
  - Is practically disconnected from the bus
  - Can not receive and transmit anything any more
  - Can only leave **bus off** mode via a hardware reset **OR**
  - Via a software reset and subsequent initialization carried through by the host (BOSH specification: TEC and REC are set to zero and the unit must receive 128 times a field of 11 consecutive **recessive** bits)

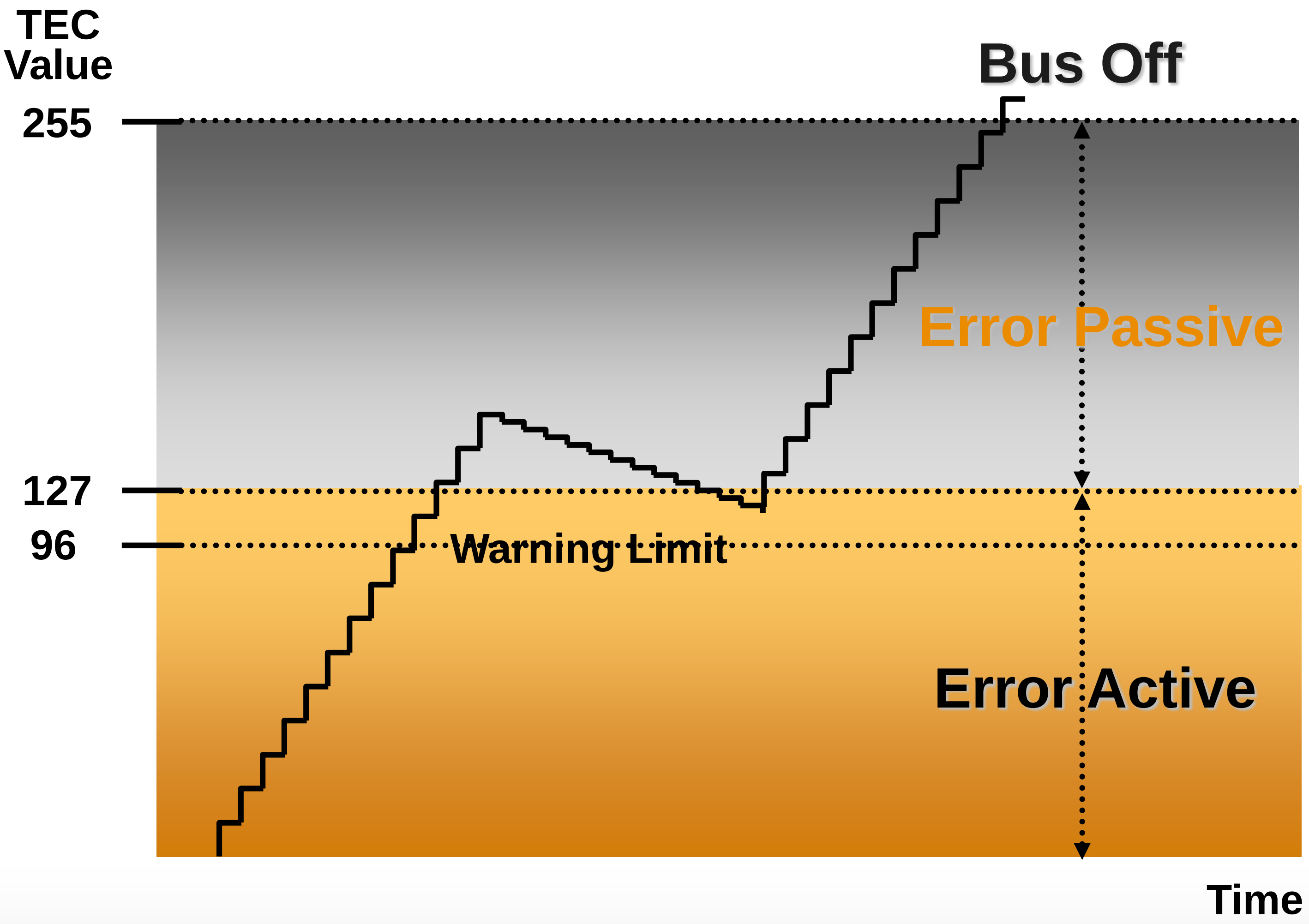
Error Detection – Warning Level

- The **Warning Level** for a unit is set when
  - its Transmit Error Counter **(TEC)** is greater than **96**: **TEC > 96** AND / OR
  - its Receive Error Counter **(REC)** is greater than **96**: **REC > 96**

- When a unit reaches the **Warning Level**
  - it is indicated to the controller CPU that the unit is “heavily disturbed”
  - an “Error Warning Flag” is set
  - optionally, an interrupt might result from this
  - the “Error Warning Flag” can be cleared for **TEC <= 96 AND REC <= 96**

- Practical use of the Warning Level
  - By constantly checking the “**Error Warning Flag**”, it can be determined whether a unit gets near the threshold to the **error passive state**
  - unfortunately, by checking this flag, one cannot determine whether a unit is still in error active state or already in error passive state

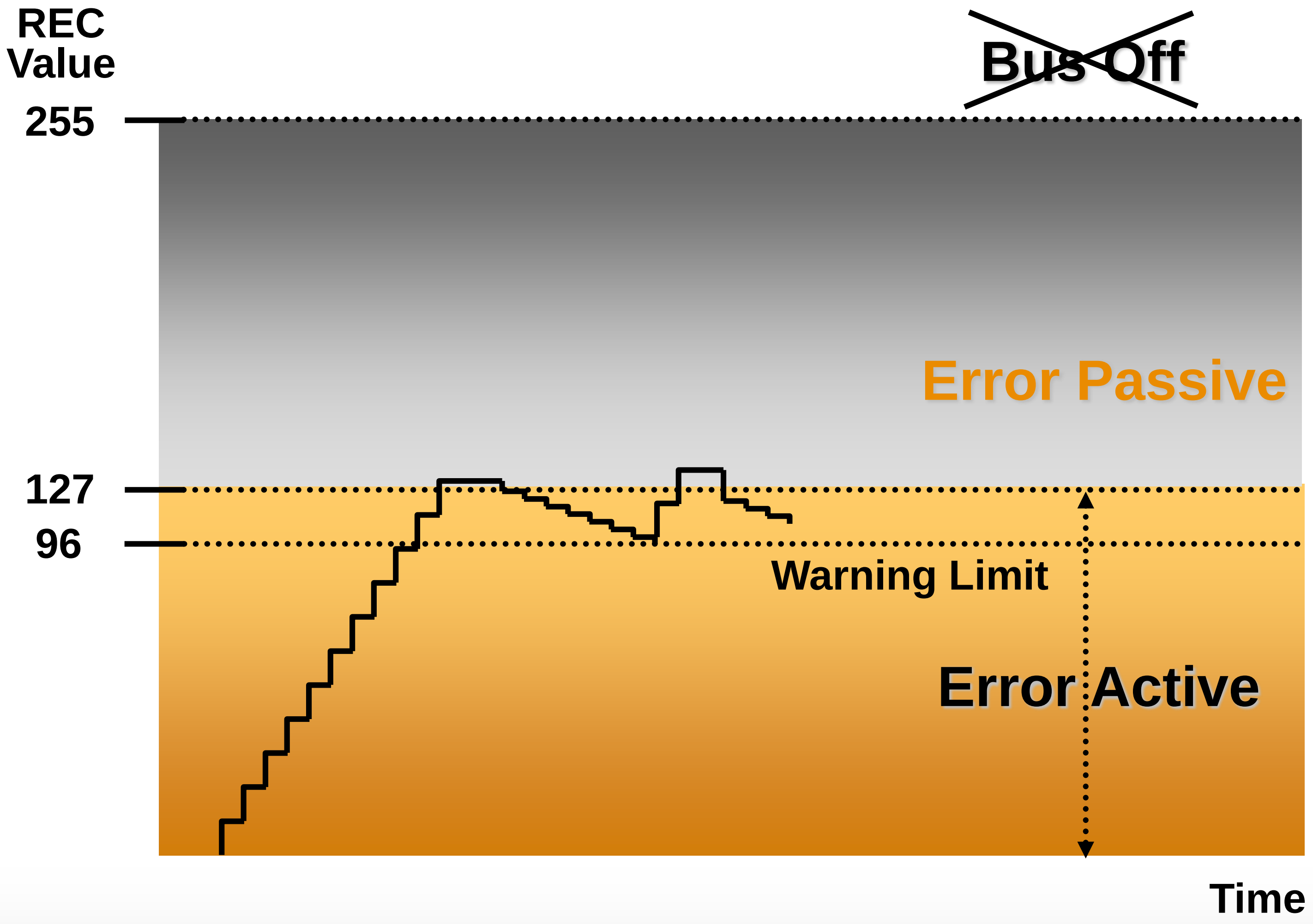
Error Detection – Tx Error Counter TEC



Abraham Tezmol, P ES FSD GDL

Error Detection – Receive Error Counter REC

CAN



Abraham Tezmol, P ES FSD GDL

Error Detection

The probability for not discovering an error is

4.7 ∗ 10-11

Example 1*

A CAN bus is used	365 days / year
	8 hours / day
with a transmission speed of	500 kBit / sec
and errors arise every	0.7 seconds

⇒ in 1.000 years, only one error remains undiscovered

Example 2**

A CAN bus in a car is runing at	500 kBit / sec
with an average bus load of	15 %
an average data frame size of	110 bits
for an average operating time of	4000 hours

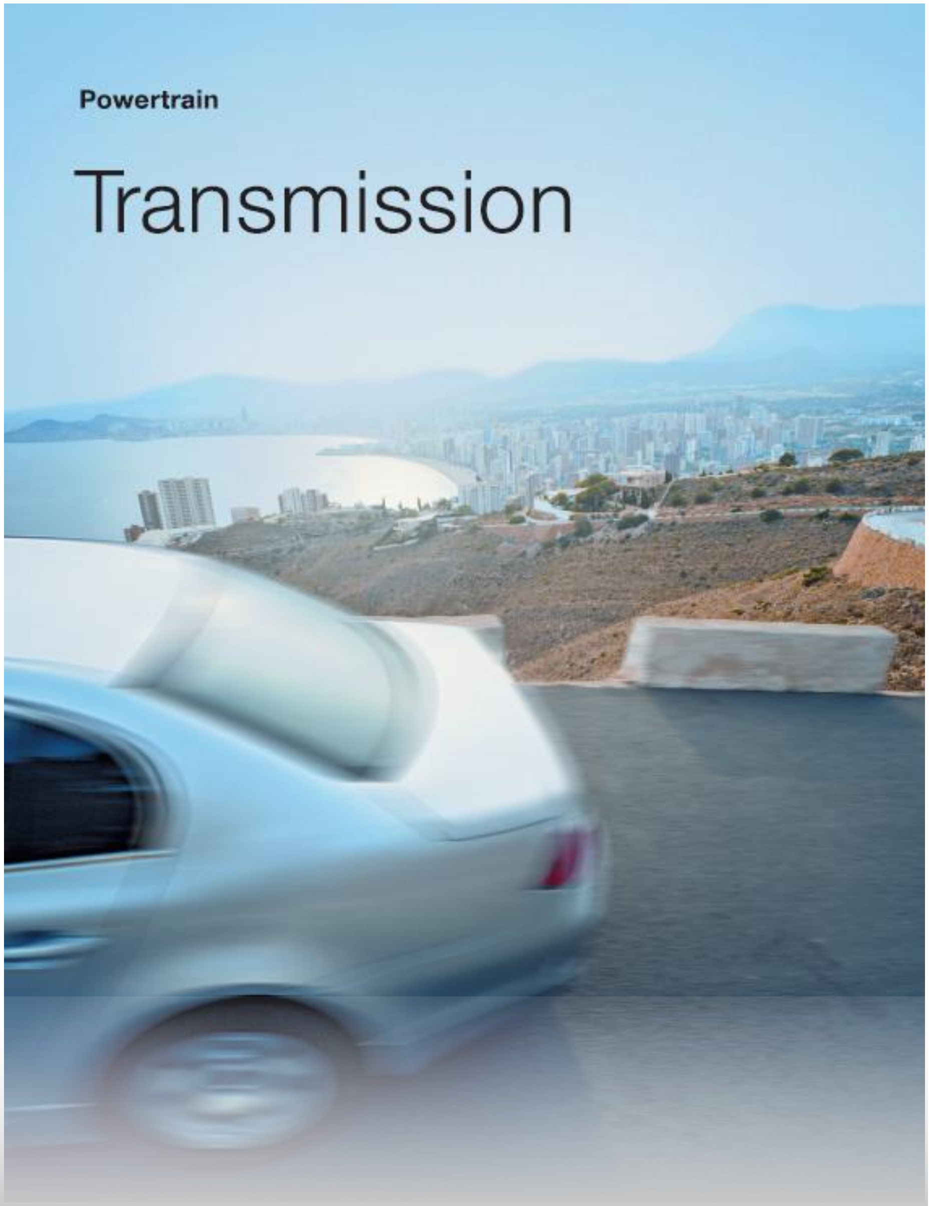
⇒ only one error in 100.000 automobiles remains undetected

*Source: CiA

**Source: Kaiser, Schröder: “Maßnahmen zur Sicherung der Daten beim CAN-Bus”

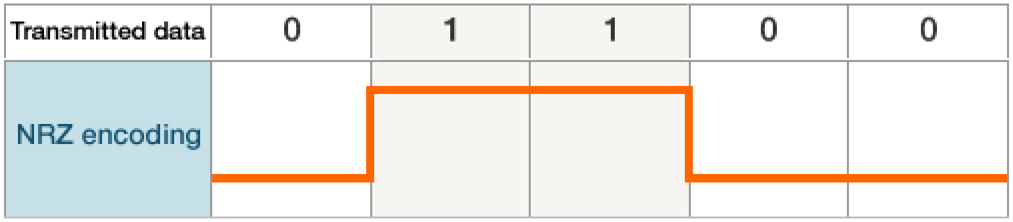
Controller Area Network - CAN

Synchronization



Synchronization – Non return to zero

CAN

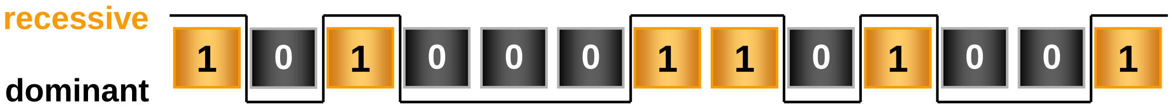


Transmitted data	0	1	1	0	0
NRZ encoding					

- The CAN protocol uses Non-Return-to-Zero or **NRZ** bit coding.
  - This means that the signal is constant for one whole bit time and only onetime segment is needed to represent one bit.

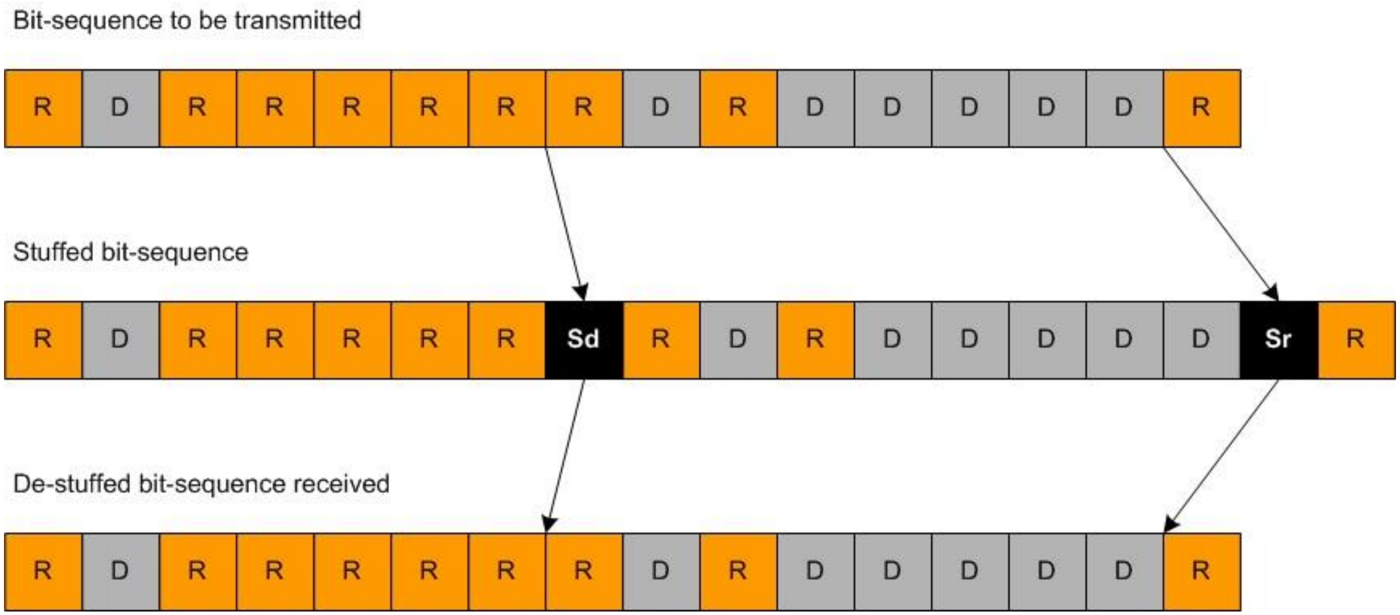
- Usually, but not always,
  - a "**zero**" corresponds to a **dominant** bit, placing the bus in the dominant state
  - a "**one**" corresponds to a **recessive** bit, placing the bus in the recessive state.

Synchronization – Problems with NRZ



- 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

Synchronization – Bit Stuffing



Bit-sequence to be transmitted

R D R R R R R R D R D D D D D R

Stuffed bit-sequence

R D R R R R R Sd R D R D D D D D Sr R

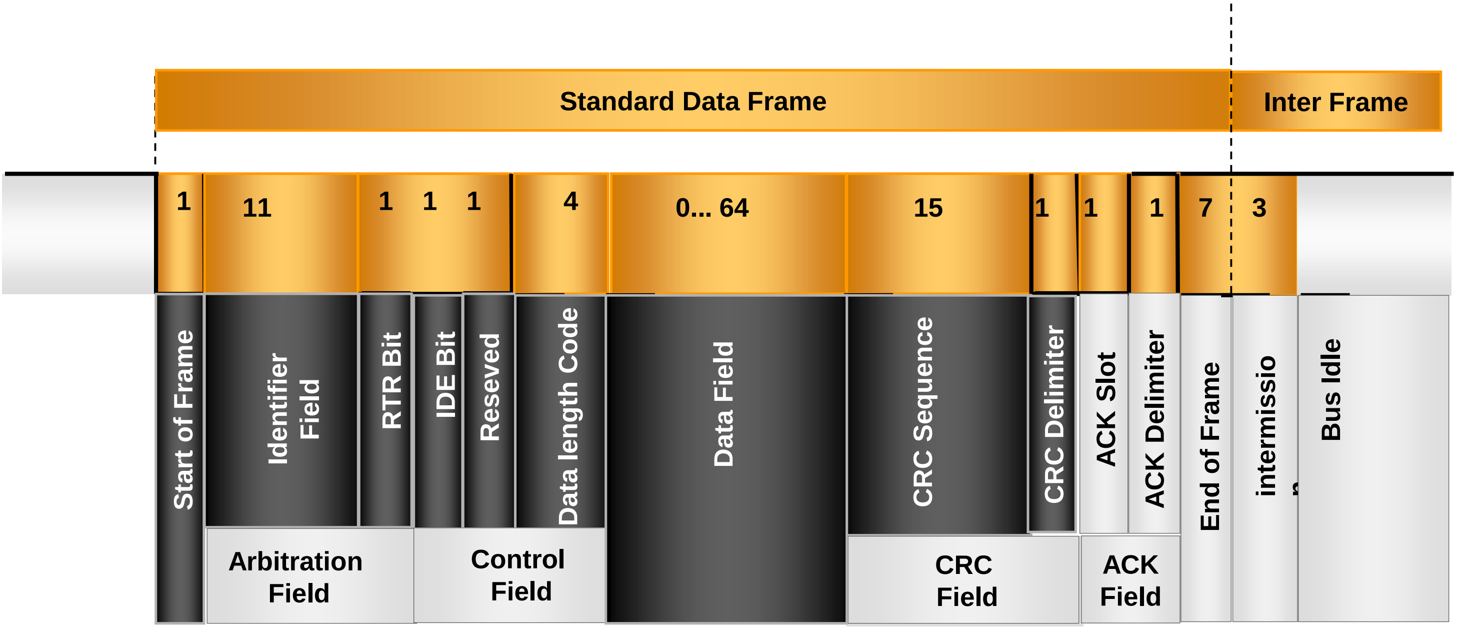
De-stuffed bit-sequence received

R D R R R R R R D R D D D D D R

- Bit stuffing is used to ensure synchronization of all busnodes
- During the transmission of a message, a maximum of five consecutive bits may have the same polarity.
- The transmitter will insert one additional bit of the opposite polarity. into the bit stream before transmitting further bits.
- The receiver also checks the number of bits with the same polarity and removes the stuff bits again from the bit stream. This is called "destuffing".

Synchronization – Bit Stuffing

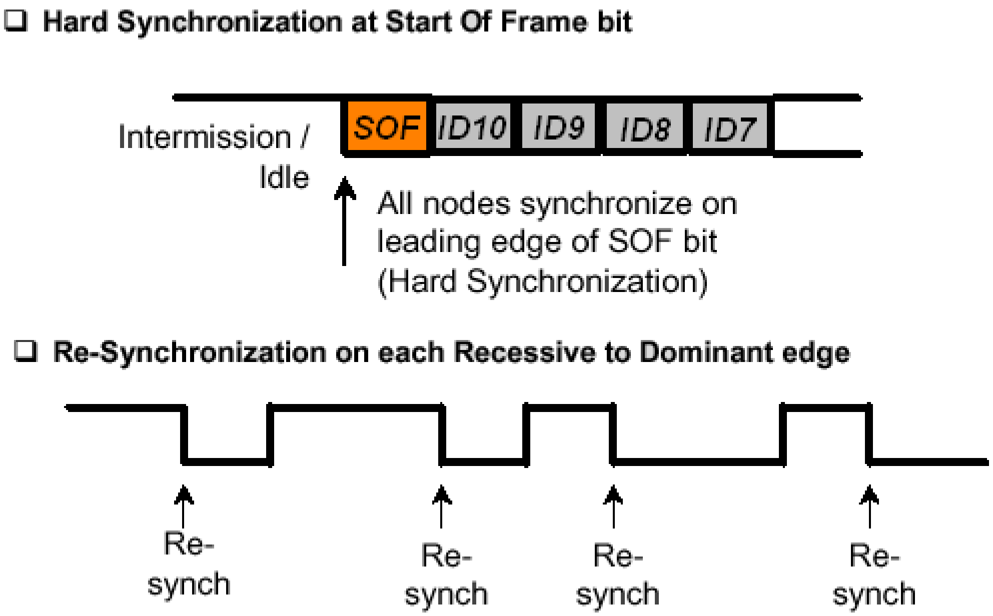
CAN



- After five consecutive bits of same polarity, insert one bit of reverse polarity
- CRC code is calculated before bit stuffing is done
- 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

Abraham Tezmol, P ES FSD GDL

Synchronization – Bus Synch

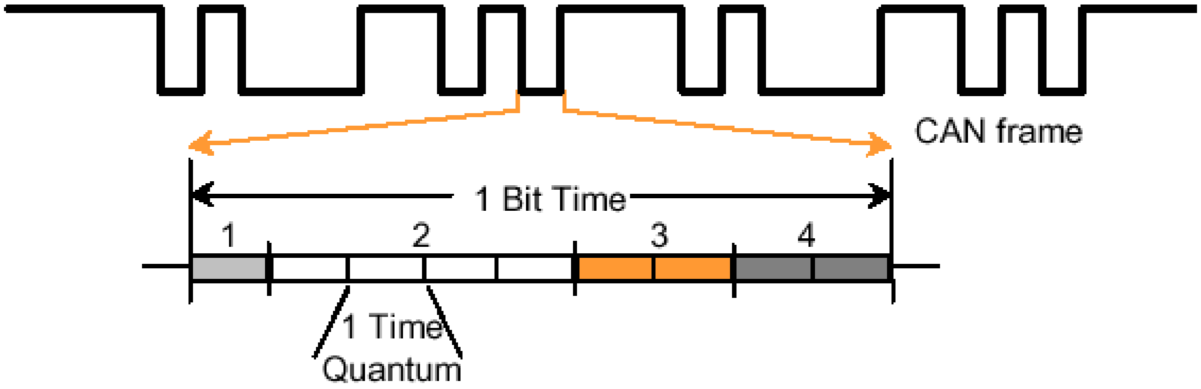

❑ Hard Synchronization at Start Of Frame bit
Intermission /
Idle
SOF
ID10
ID9
ID8
ID7
All nodes synchronize on leading edge of SOF bit (Hard Synchronization)
❑ Re-Synchronization on each Recessive to Dominant edge
Re-synch
Re-synch
Re-synch
Re-synch

- CAN handles message transfers synchronously
- All nodes are synchronized at the beginning of each message with the first falling edge of a frame which belongs to the Start Of Frame bit ➔ **Hard Synchronization**.
- To ensure correct sampling up to the last bit, the CAN nodes need to **re-synchronize** throughout the entire frame. This is done on each recessive to dominant edge.

Synchronization – Bit Construction

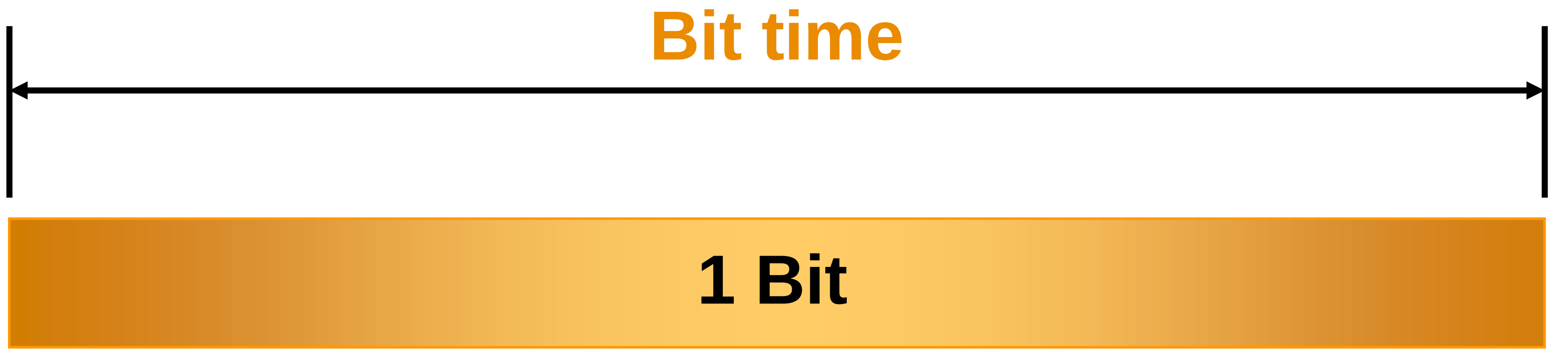
CAN

- 4 Segments, 8-25 Time Quanta (TQ) per bit time
- Time Quanta generated by programable divide of Oscillator
- CAN Baud Rate (=1 / Bit Time) programmed by selection of appropriate TQ length + appropriate number of TQ per bit



Abraham Tezmol, P ES FSD GDL

Synchronization – Bit Time

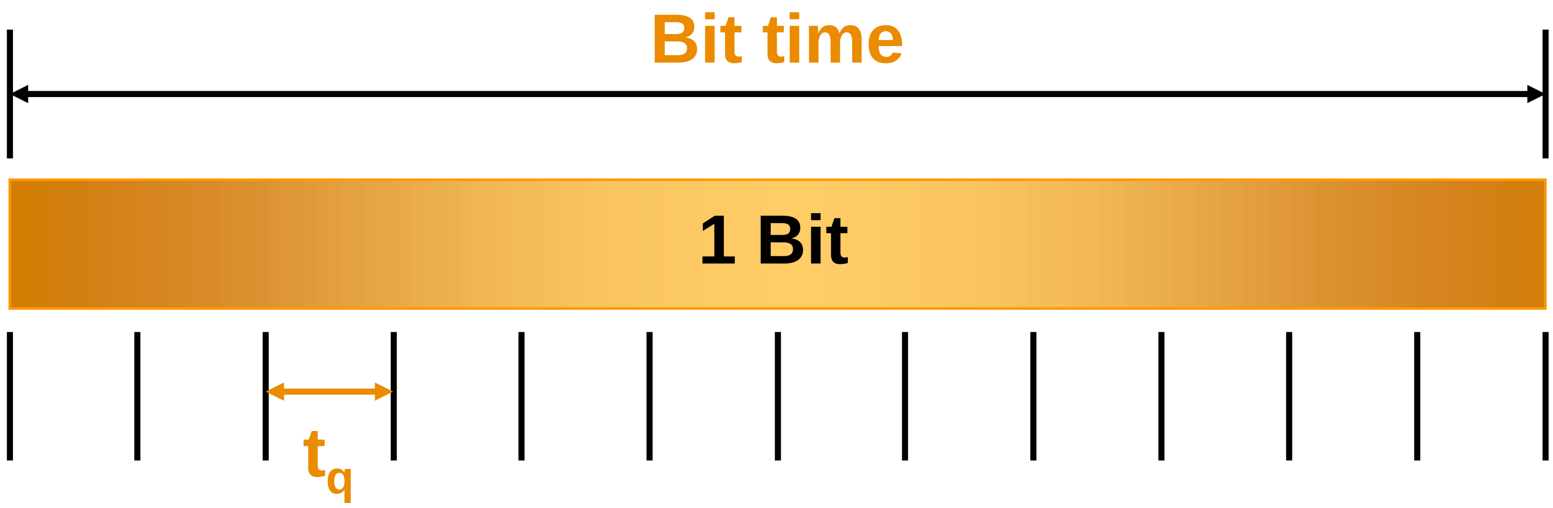


The **bit time t_{Bit}** is the time it takes to transmit one bit.

Example: Data rate: $f = 500$ kBit / s

→ Bit time: $t_{Bit} = 2$ µs

Synchronization – Time Quantum



The bit time is divided up into **time quanta t_q.**

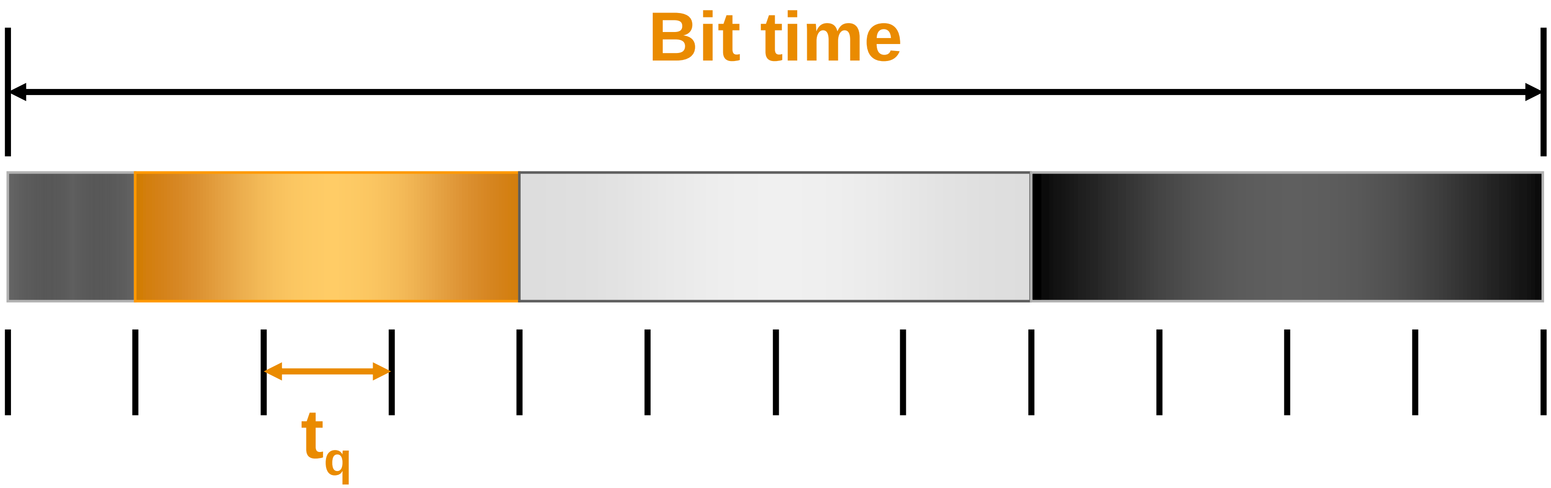
Length of one time quantum: $t_q = \text{BRP} / f_{Sys}$

Allowed number of time quanta: $8 \leq n_q \leq 25$

BRP: Baud Rate Prescaler, f_{Sys}: System clock

Synchronization – Bit time segments

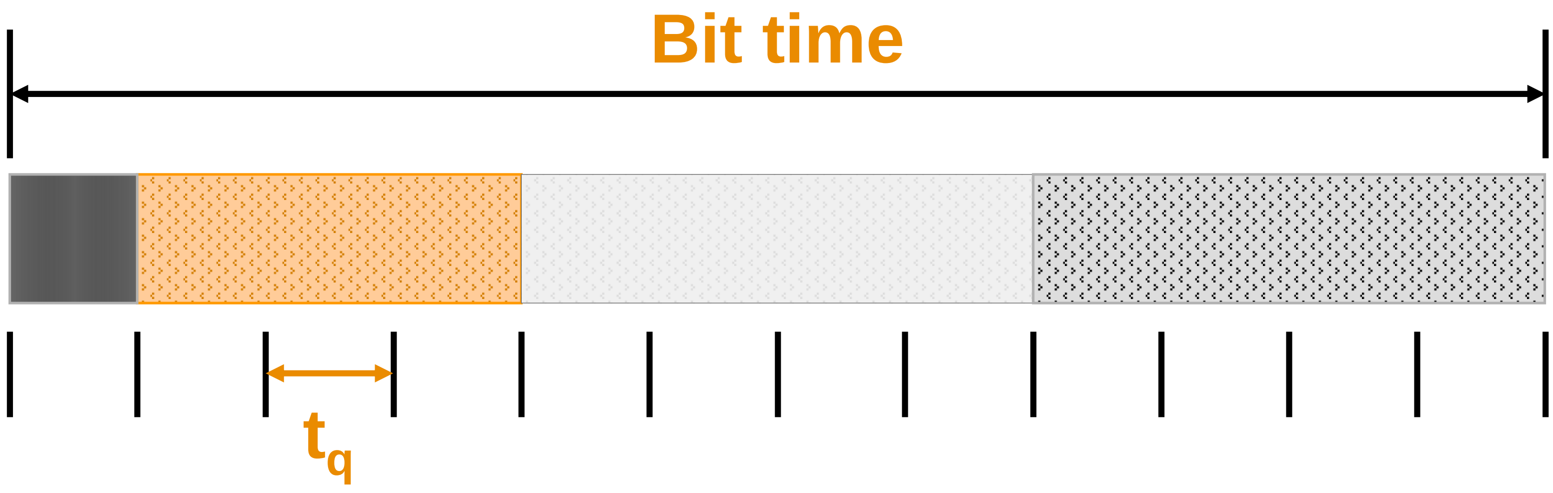
CAN



■	*Sync_Seg*	**Synchronization Segment**
🟧	*Prop_Seg*	**Propagation Time Segment**
□	*Phase_Seg1*	**Phase Buffer Segment 1**
■	*Phase_Seg2*	**Phase Buffer Segment 2**

Synchronization – Synchronization segment

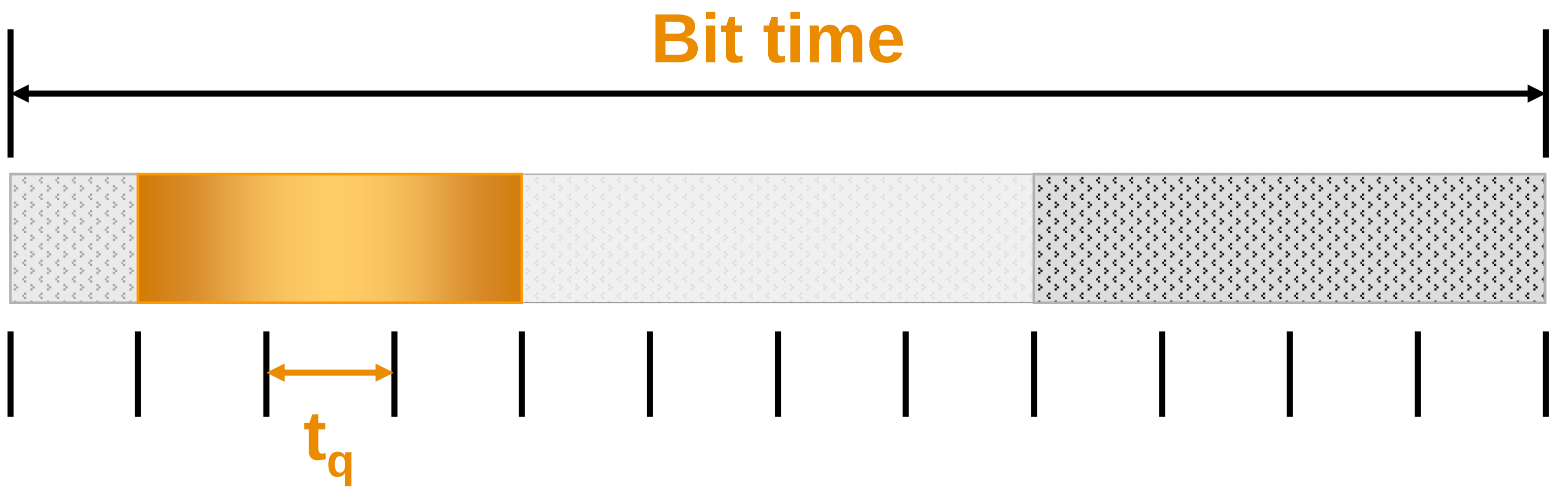
CAN



Edges in CAN bus level are expected to occur here.

Synchronization Segment has fixed length: $\boxed{t_{Sync\_Seg} = 1\ t_q}$

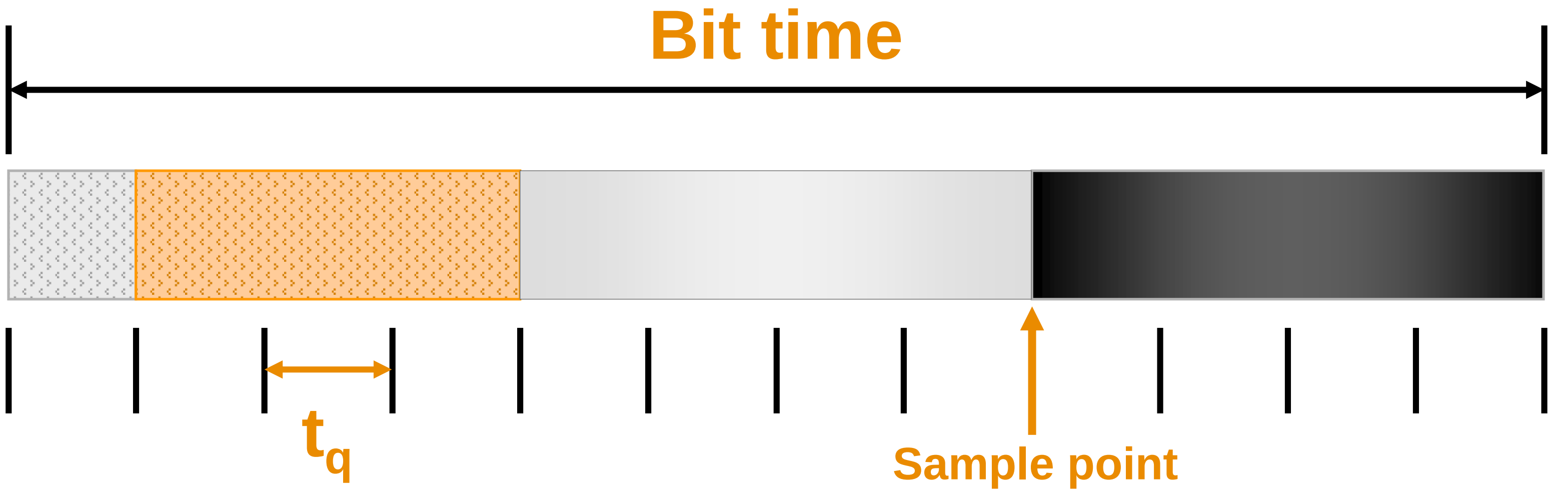
Synchronization – Propagation Time Segment



The **Propagation Time Segment** compensates for physical delay times within the CAN network.

Length: $\boxed{1\ t_q \leq t_{Prop\_Seg} \leq 8\ t_q}$

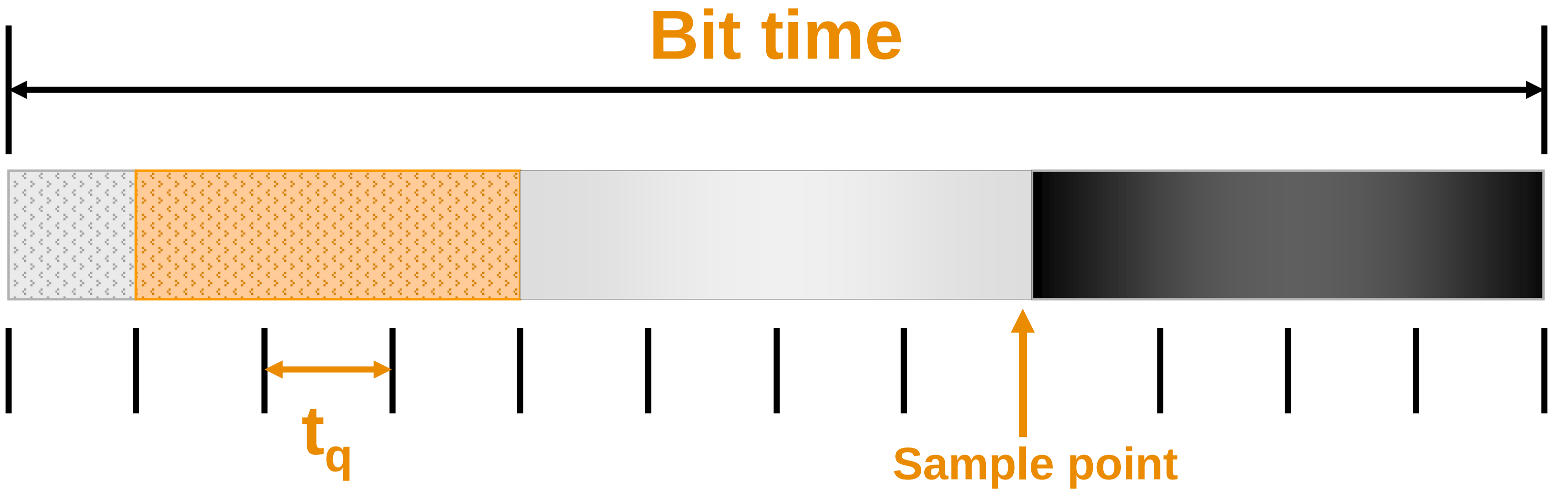
Synchronization – Sample point



□	*Phase_Seg1*	**Phase Buffer Segment 1**
■	*Phase_Seg2*	**Phase Buffer Segment 2**

- ***Phase Buffer Segment 1*** **may be lengthened during resynchronization**
- ***Phase Buffer Segment 2*** **may be shortened during resynchronization**

Synchronization – Sample point

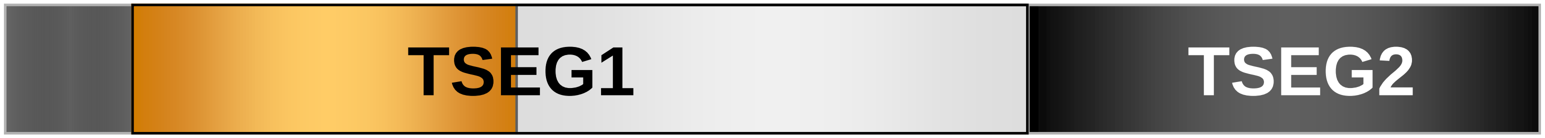


The **Phase Buffer Segments** surround the **sample point**.

Lengths:

$$1\ t_q \leq t_{Phase\_Seg1} \leq 8\ t_q$$
$$2\ t_q \leq t_{Phase\_Seg2} \leq 8\ t_q$$

Synchronization – Adjustable parameters



TSEG1 Time Segment 1

TSEG1 = Prop_Seg + Phase_Seg1

TSEG2 Time Segment 2

TSEG2 = Phase_Seg2

BRP Baud Rate Prescaler

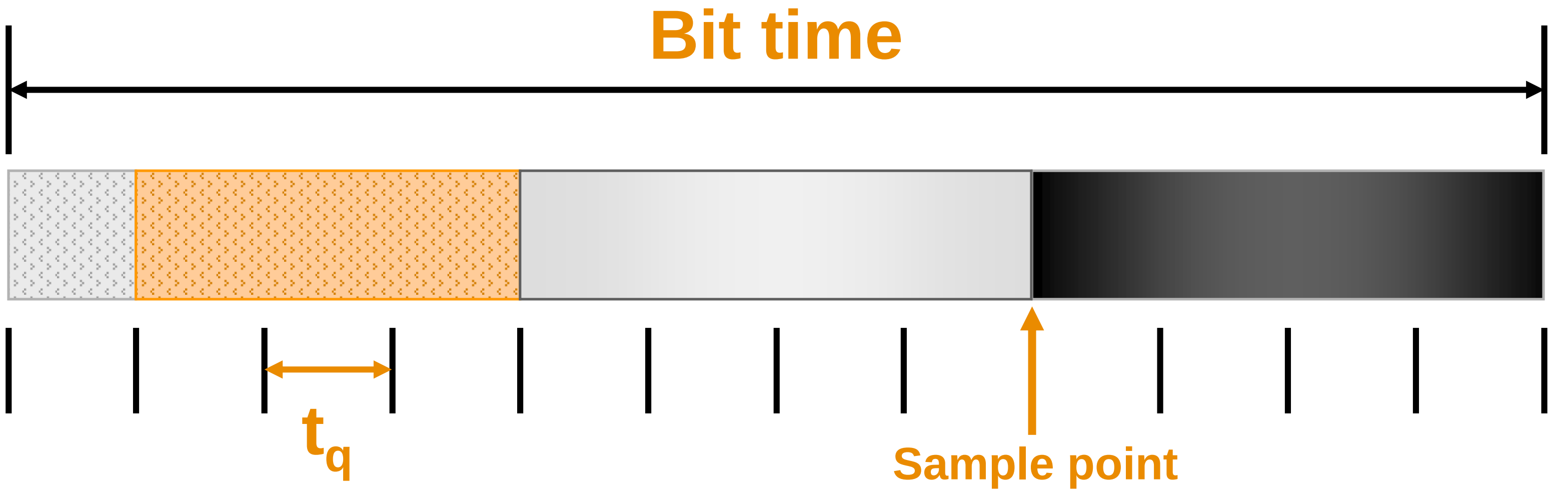
SJW Synchronization Jump Width

SJW ≤ min (Phase_Seg1, Phase_Seg2) **AND** **SJW ≤ 4**

SJW indicates the maximum number of time quanta t_q
by which TSEG1 may be lengthened or TSEG2 may be shortened.

Abraham Tezmol, P ES FSD GDL

Synchronization – Sample point



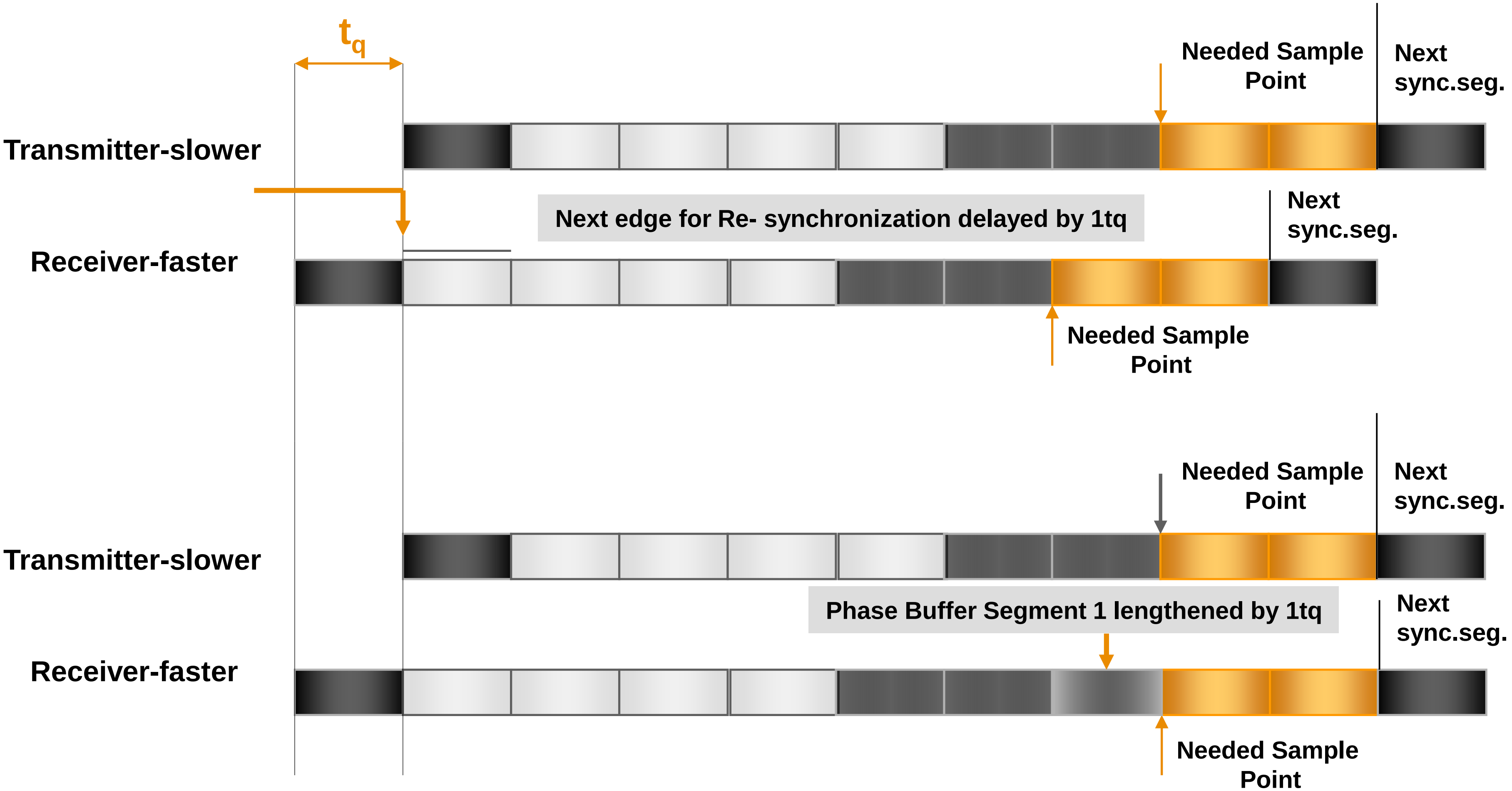
The **Phase Buffer Segments** surround the **sample point**.

Lengths:

$$1\ t_q \leq t_{Phase\_Seg1} \leq 8\ t_q$$
$$2\ t_q \leq t_{Phase\_Seg2} \leq 8\ t_q$$

Synchronization – Bit lengthening

CAN



t_q

Transmitter-slower

Needed Sample Point

Next sync.seg.

Next edge for Re- synchronization delayed by 1tq

Receiver-faster

Next sync.seg.

Needed Sample Point

Transmitter-slower

Needed Sample Point

Next sync.seg.

Phase Buffer Segment 1 lengthened by 1tq

Receiver-faster

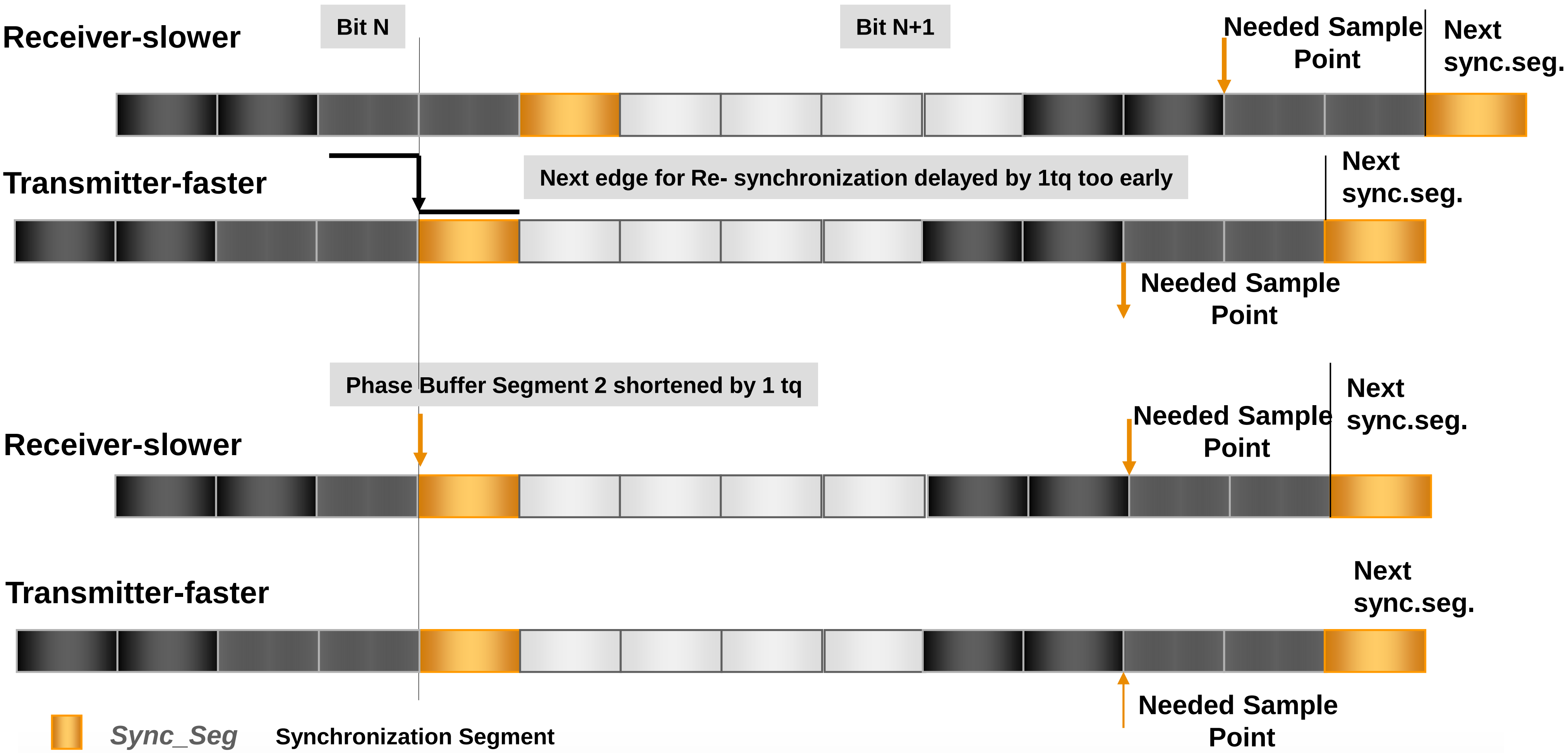
Next sync.seg.

Needed Sample Point

■	*Sync_Seg*	Synchronization Segment
□	*Prop_Seg*	Propagation Time Segment
▨	*Phase_Seg1*	Phase Buffer Segment 1
🟧	*Phase_Seg2*	Phase Buffer Segment 2

Synchronization – Bit shortening

CAN



Receiver-slower

Bit N

Bit N+1

Needed Sample Point

Next sync.seg.

Transmitter-faster

Next edge for Re- synchronization delayed by 1tq too early

Next sync.seg.

Needed Sample Point

Phase Buffer Segment 2 shortened by 1 tq

Needed Sample Point

Next sync.seg.

Receiver-slower

Transmitter-faster

Next sync.seg.

Needed Sample Point

🟧	*Sync_Seg*	Synchronization Segment
⬜	*Prop_Seg*	Propagation Time Segment
⬛	*Phase_Seg1*	Phase Buffer Segment 1
⬛	*Phase_Seg2*	Phase Buffer Segment 2

Synchronization – Parameter calculation (1)

1. Define bit rate f_{Bit}

   or bit time $\boxed{\text{tBit} = 1 / \text{fBit}}$

2. Define number of time quanta n_q per bit time t_{Bit}. Remember that:

$$\boxed{8 \leq nq \leq 25}$$

Synchronization – Parameter calculation (2)

3. Calculate length of **time quantum t_q**

$$\boxed{t_q = t_{Bit} / n_q}$$

4. Calculate value of **Baud Rate Prescaler (BRP)**

$$\boxed{BRP = t_q * f_{Sys}}$$

➔ First Bit Timing parameter: **BRP**

Synchronization – Parameter calculation (3)

5. The length of the Synchronization Segment Sync_Seg is fixed:

$$\boxed{t_{Sync\_Seg} = 1\ t_q}$$

6. To define the length of the Propagation Segment Prop_Seg, measure the delay times in the system.

Rule of thumb: **The longer the CAN Bus,**
the longer the Propagation Segment.

Synchronization – Parameter calculation (4)

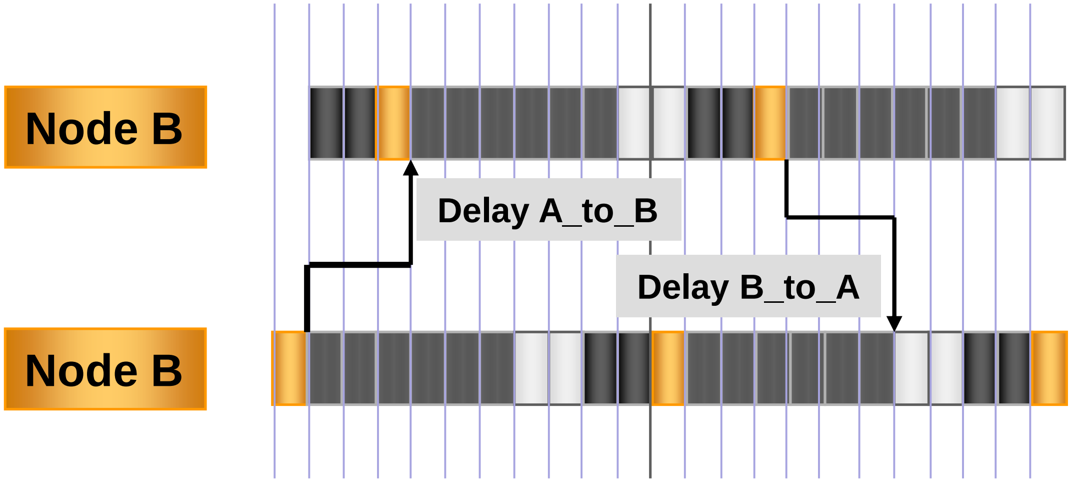
7. Define the length of the Propagation Segment Prop_Seg. Remember that:

$$\boxed{1\ t_q \leq t_{Prop\_Seg} \leq 14\ t_q}$$

Additionally, the following rule applies:

$$\boxed{3\ t_q \leq t_{Bit} - t_{Sync\_Seg} - t_{Prop\_Seg} \leq 16\ t_q}$$

Synchronization



🟧	*Sync_Seg*	Synchronization Segment
▨	*Prop_Seg*	Propagation Time Segment
■	*Phase_Seg1*	Phase Buffer Segment 1
□	*Phase_Seg2*	Phase Buffer Segment 2

- **Prop_Seg >= Delay A_to_B + Delay B_to_A**
- **Prop_Seg >= 2 • [max(node output delay+ bus line delay + node input delay)]**
- **Delay A_to_B >= node output delay(A) + bus line delay(A→B) + node input delay(B)**

Synchronization – Parameter calculation (5)

8. If the remaining number of time quanta

$$(t_{Bit} - t_{Sync\_Seg} - t_{Prop\_Seg}) / t_q$$

is an odd number, choose:

$$\boxed{t_{Phase\_Seg2} = t_{Phase\_Seg1} + 1\ t_q}$$

else choose:

$$\boxed{t_{Phase\_Seg2} = t_{Phase\_Seg1}}$$

Synchronization – Parameter calculation (6)

9. Calculate values of TSEG1 and TSEG2:

$$TSEG1 = (t_{Prop\_Seg} + t_{Phase\_Seg1}) / t_q$$

$$TSEG2 = t_{Phase\_Seg2} / t_q$$

➔ Second Bit Timing Parameter: TSEG1

Third Bit Timing Parameter: TSEG2

Synchronization – Parameter calculation (7)

10. Choose the **Synchronization Jump Width (SJW)**.

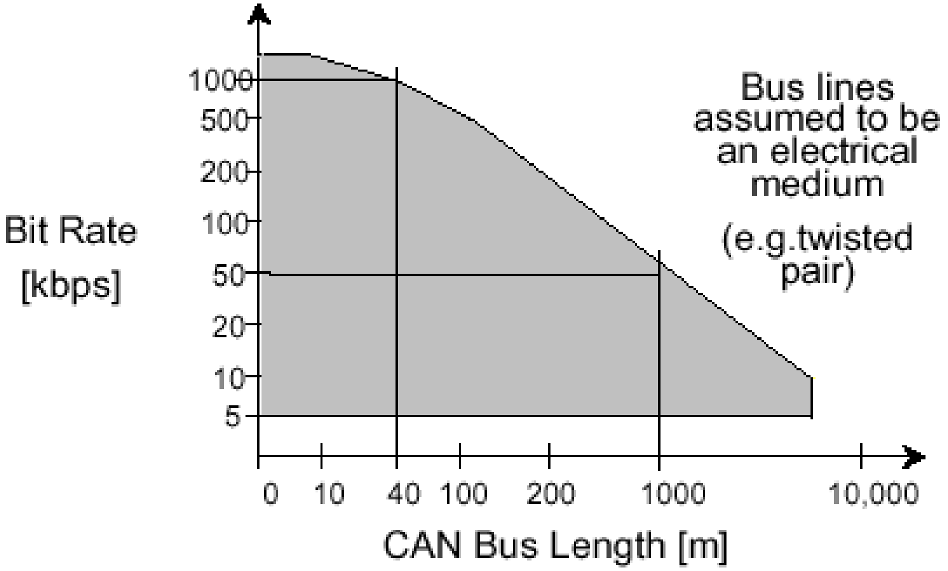
Remember that:

$$\boxed{\begin{array}{l} SJW \leq \min(Phase\_Seg1, Phase\_Seg2) \\ SJW \leq 4 \end{array}}$$

Rule of thumb: **The larger the oscillator tolerance, the larger the Synchronization Jump Width.**

→ Fourth Bit Timing Parameter: **SJW**

Synchronization – Baud Rate vs Bus Length



Bus lines assumed to be an electrical medium

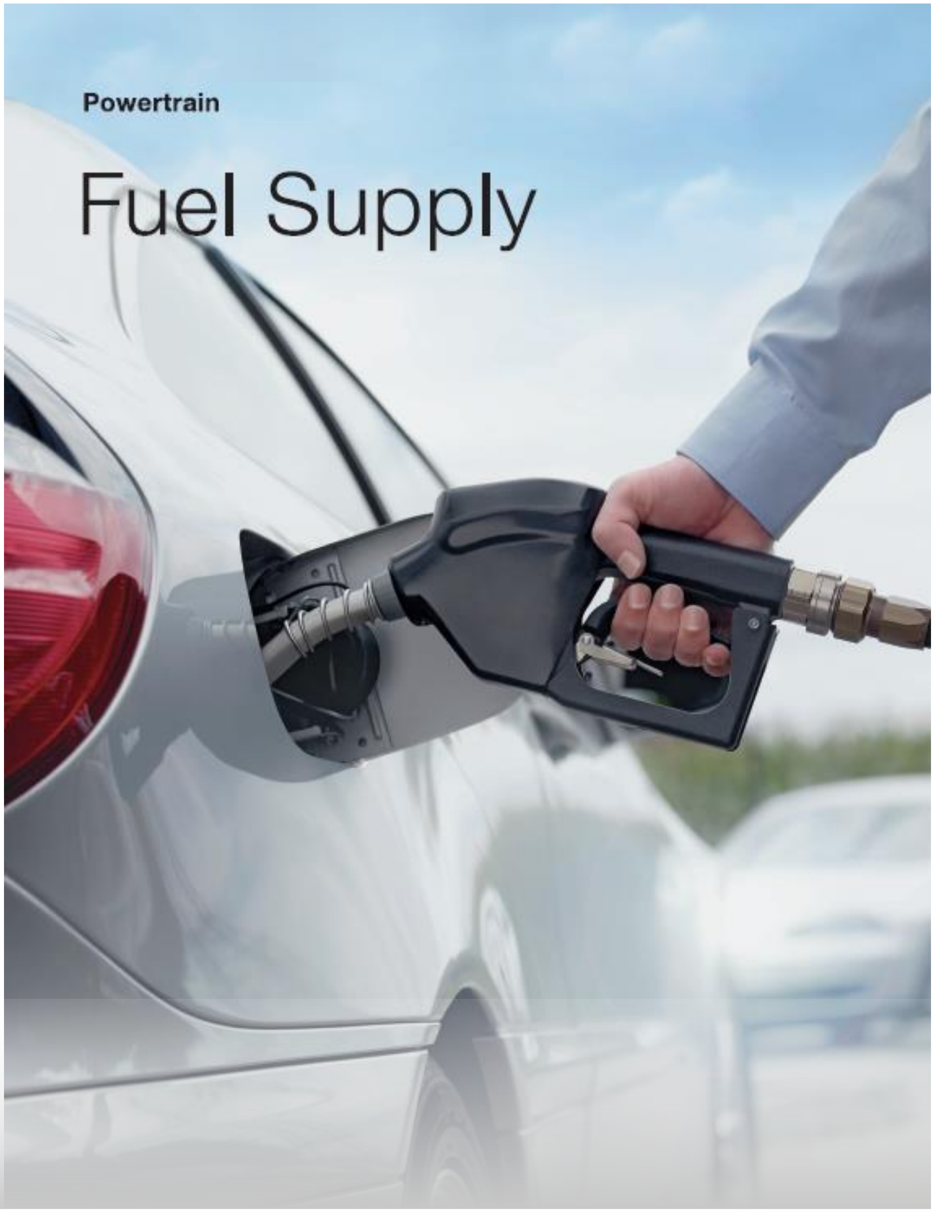
(e.g.twisted pair)

1MB → 40m

50KB → 1000m

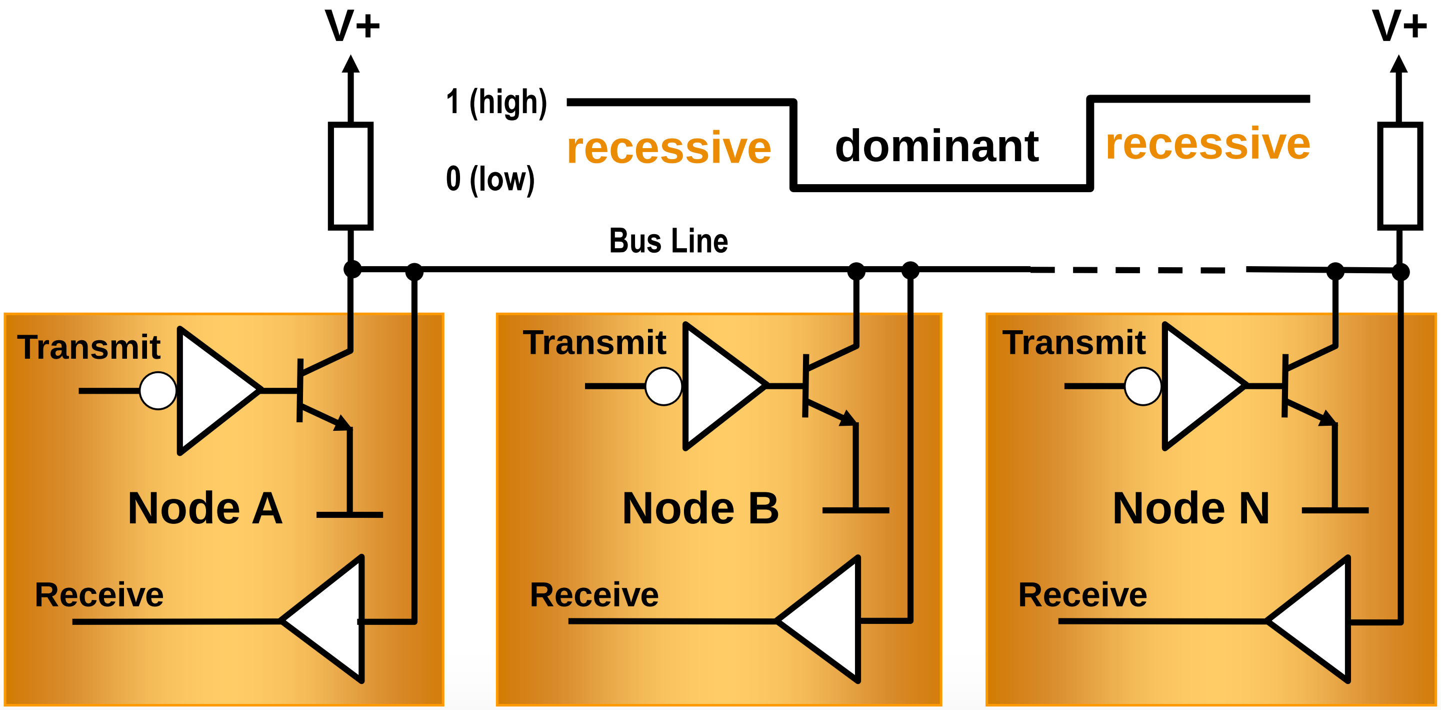
Controller Area Network - CAN

Arbitration



Arbitration – Physical Transmission

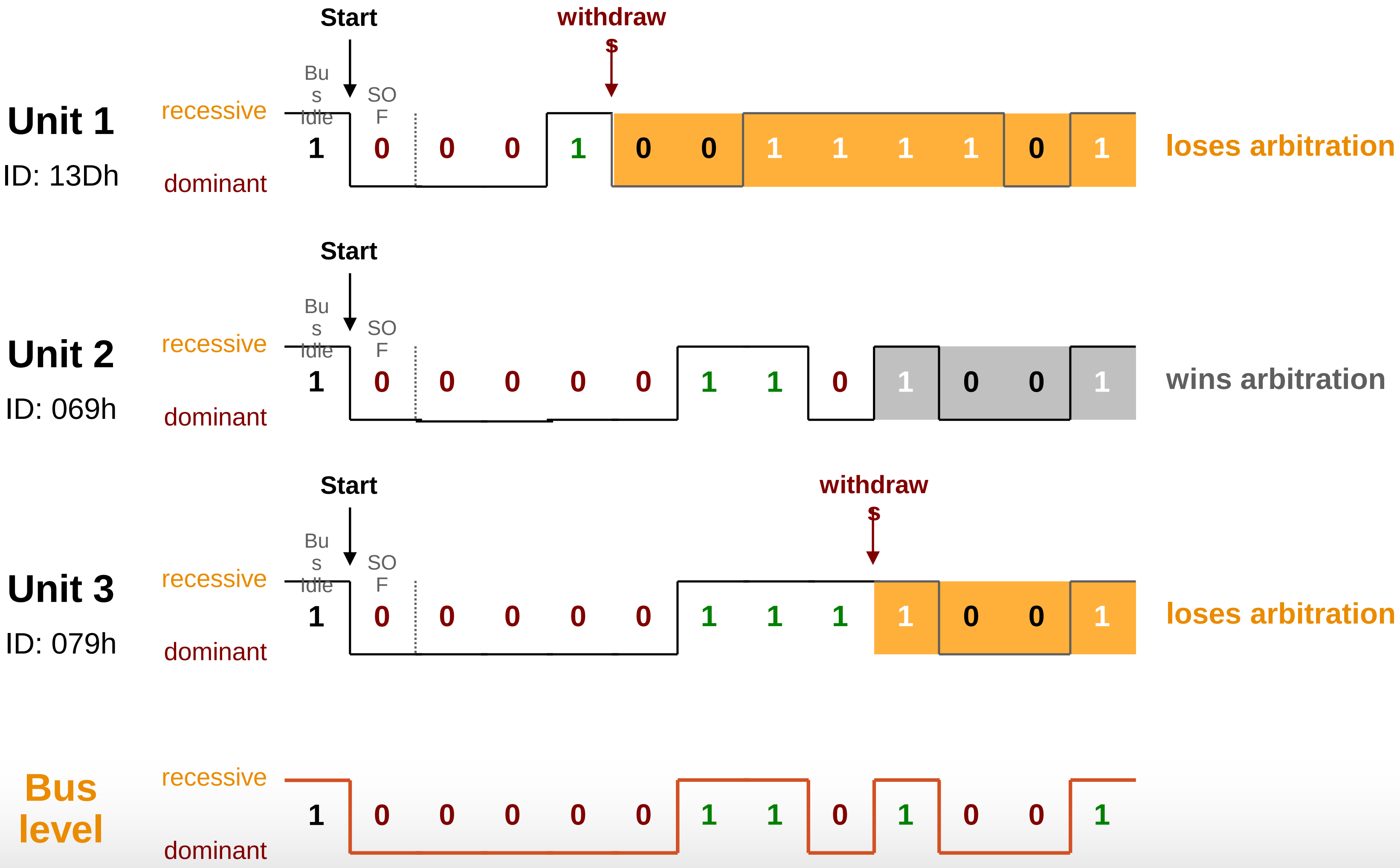
- Transmission with Carrier Sense Multiple Access with Collision Avoidance **CSMA/CA**
- **Dominant** Bits (low) win against ➔ **Recessive** Bits (high)



Abraham Tezmol, P ES FSD GDL

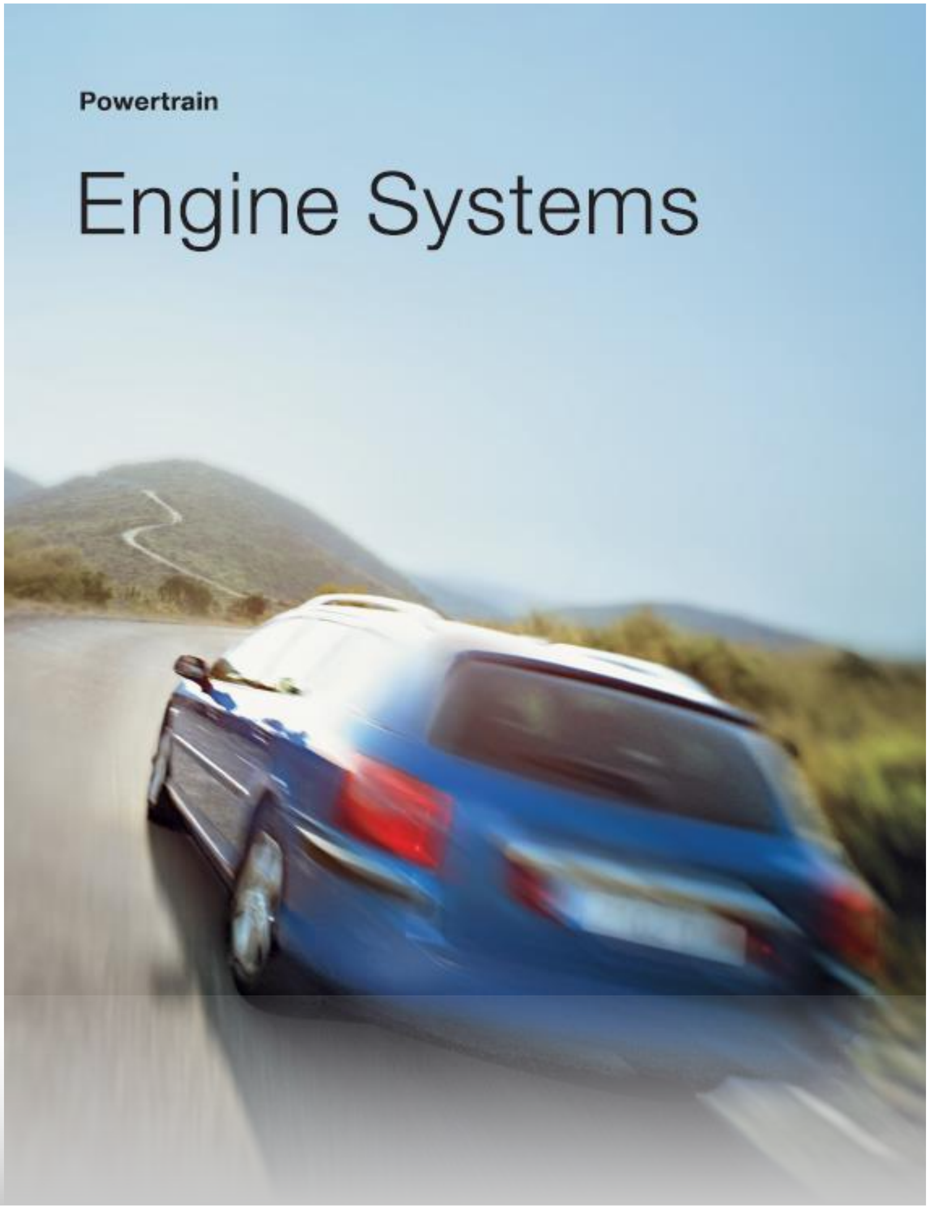
Arbitration – Example

CAN

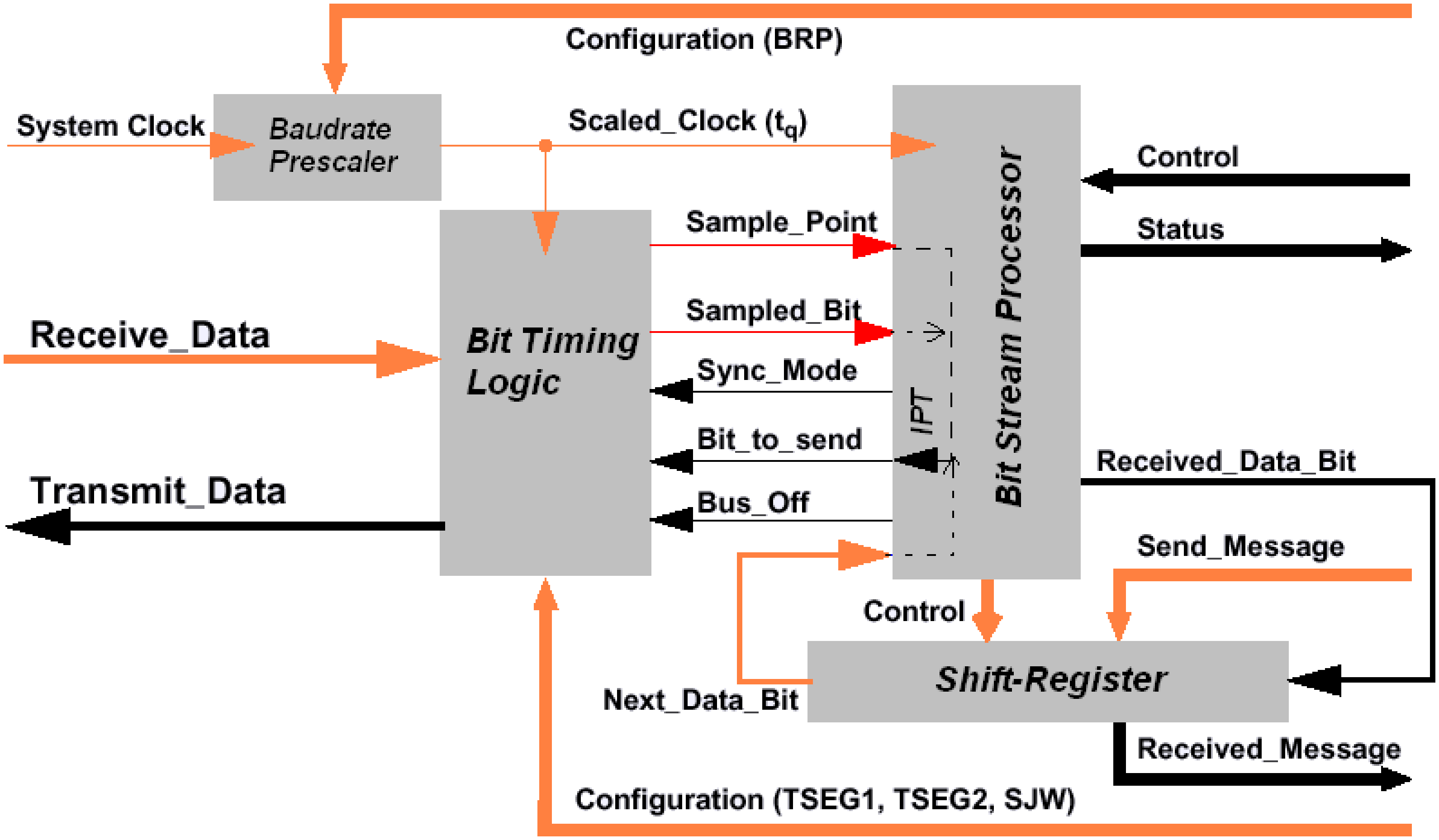


Controller Area Network - CAN

Controller



Controller – Structure of CAN Controller



Controller

CAN

Bit Stream Processor

translates messages into frames and vice versa.
inserts and extracts stuff bits
calculates and checks the CRC code
evaluate at the Sample Point and processes the sampled bus input bit.

Bit Timing Logic

number of time quanta in the bit time..
synchronization but occasional
Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the
number of time quanta in the bit time.

The Shift Register

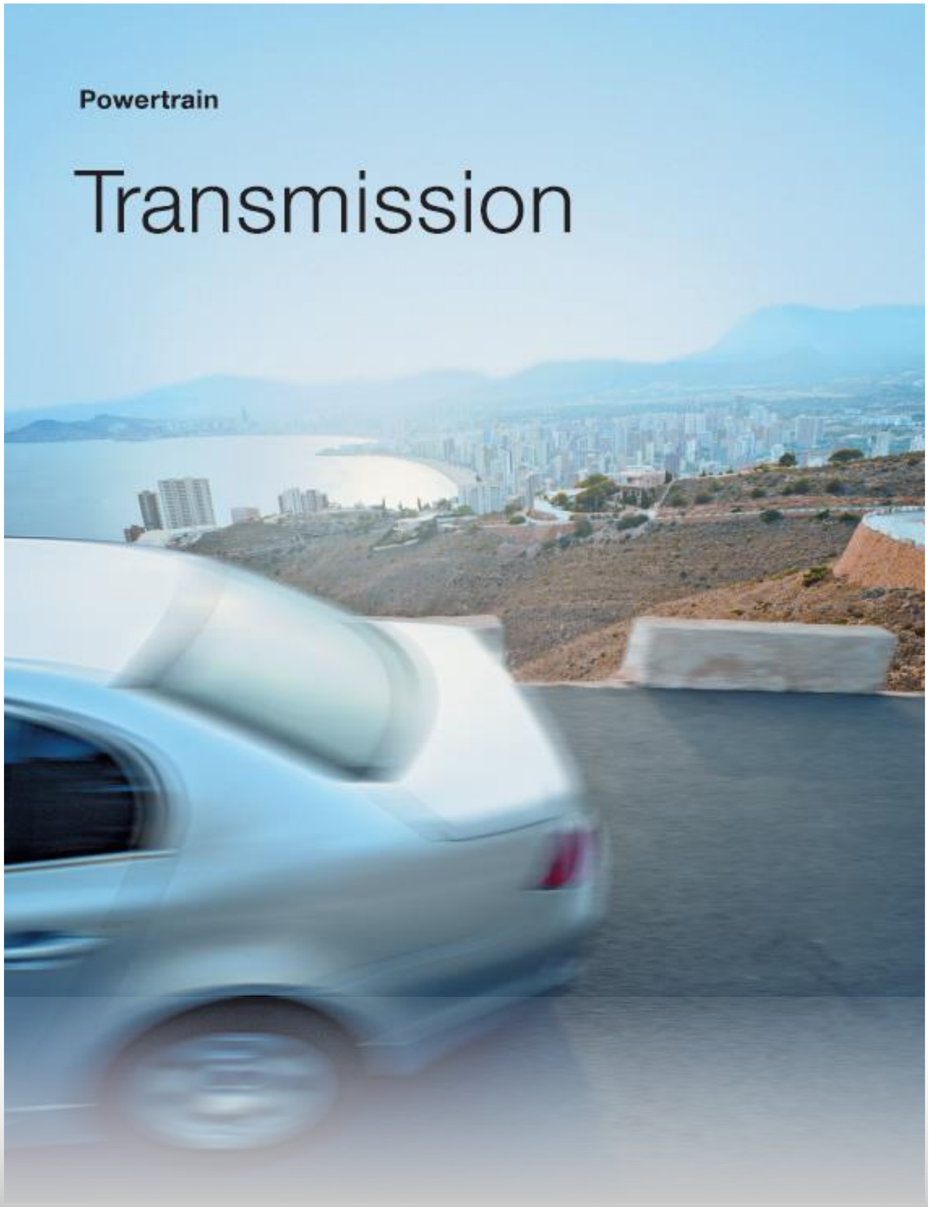
serializes the messages to be sent and parallelizes received messages
Its loading and shifting is controlled by the BSP.

Baud Rate Prescaler

defines the length of the time quantum,
the basic time unit of the bit time; the

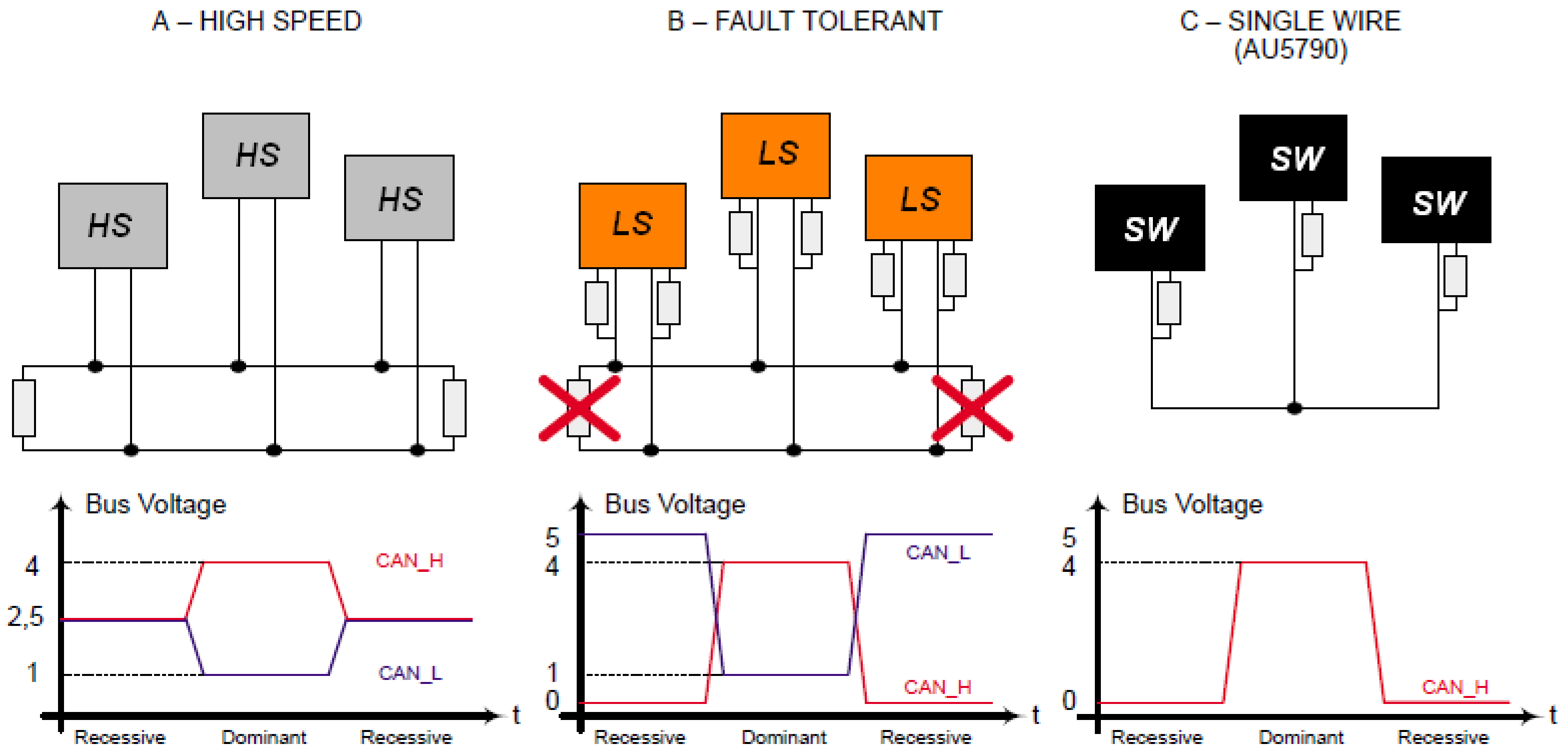
Controller Area Network - CAN

Physical Layer



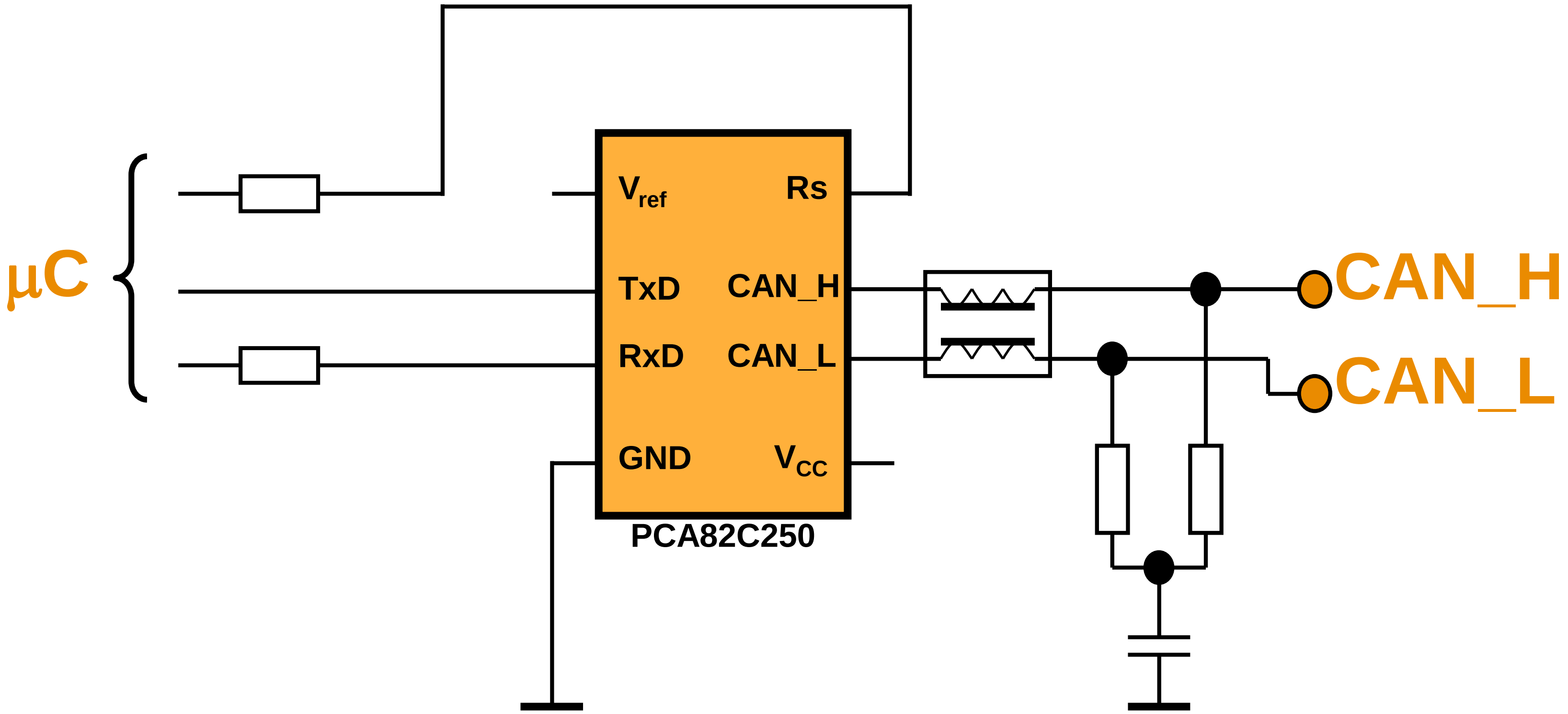
Physical Layer – TxRx Comparison

CAN



Abraham Tezmol, P ES FSD GDL

Physical Layer – Transceiver HSCAN



Vref	reference voltage output
TxD	transmit data input
RxD	receive data output
GND	ground

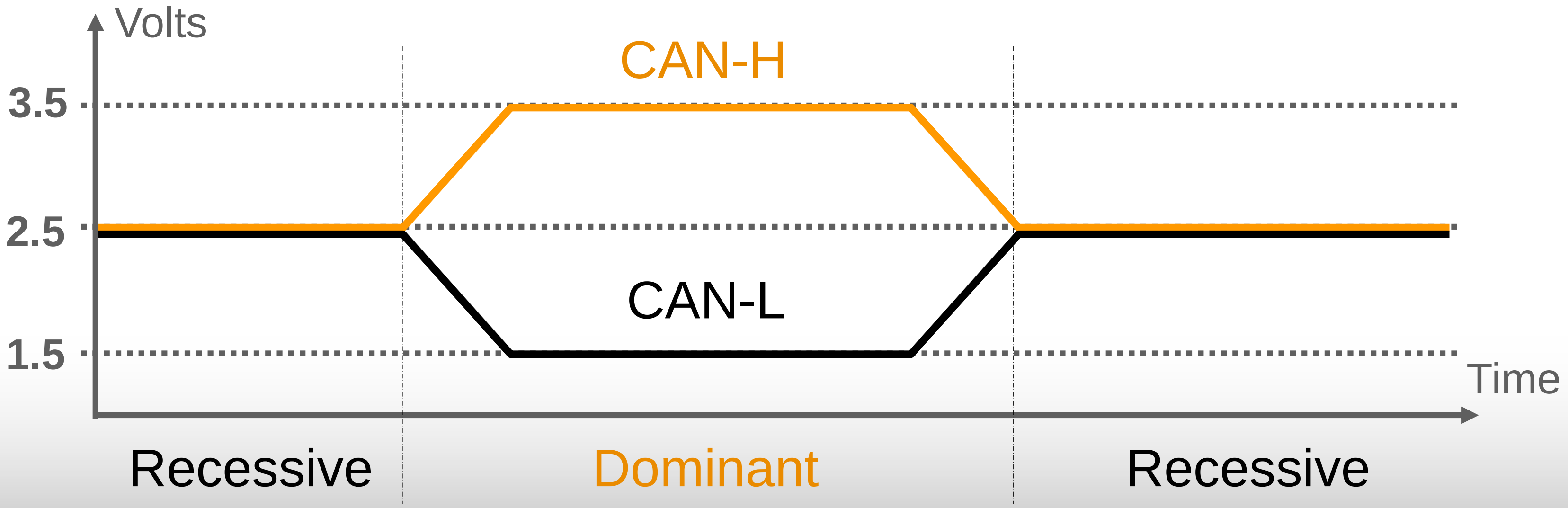
Rs	slope resistor input
CAN_H	HIGH level CAN voltage input / output
CAN_L	LOW level CAN voltage input / output
Vcc	supply voltage

Physical Layer – Physical Aspects

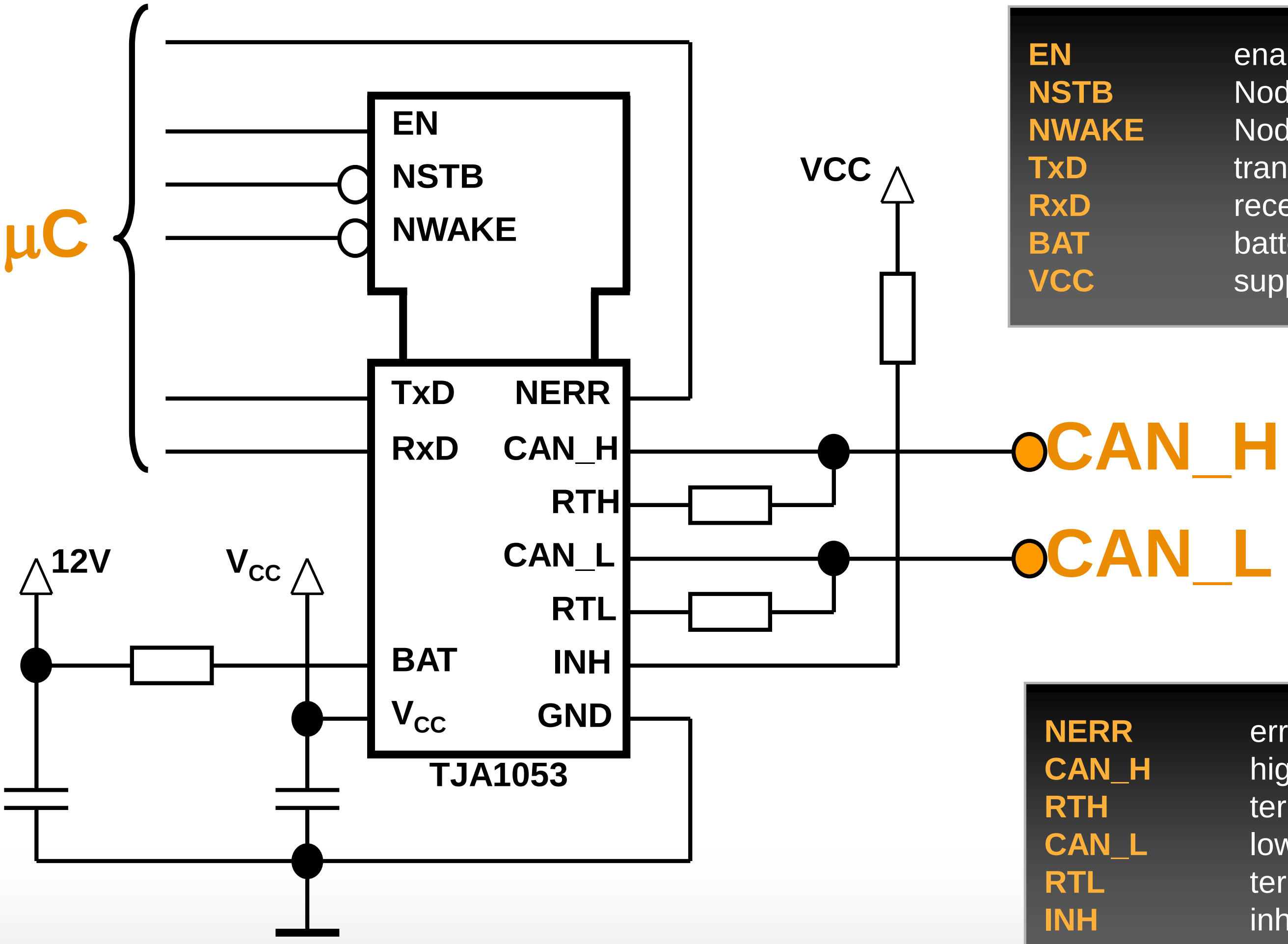
CAN

- Transmission rate up to 1 MBit/s needs bus driver circuits (transceiver) according to ISO/DIS 11898

- CAN High Speed Transmission (ISO/DIS 11898)
  - Transmission rate 125 Kbit/s up to 1 MBit/s
  - Max. bus length depend on transmission rate (e.g. 1 MBit/s max. 40 m bus length)
    - Up to 30 nodes



Physical Layer – Transceiver Fault Tolerant

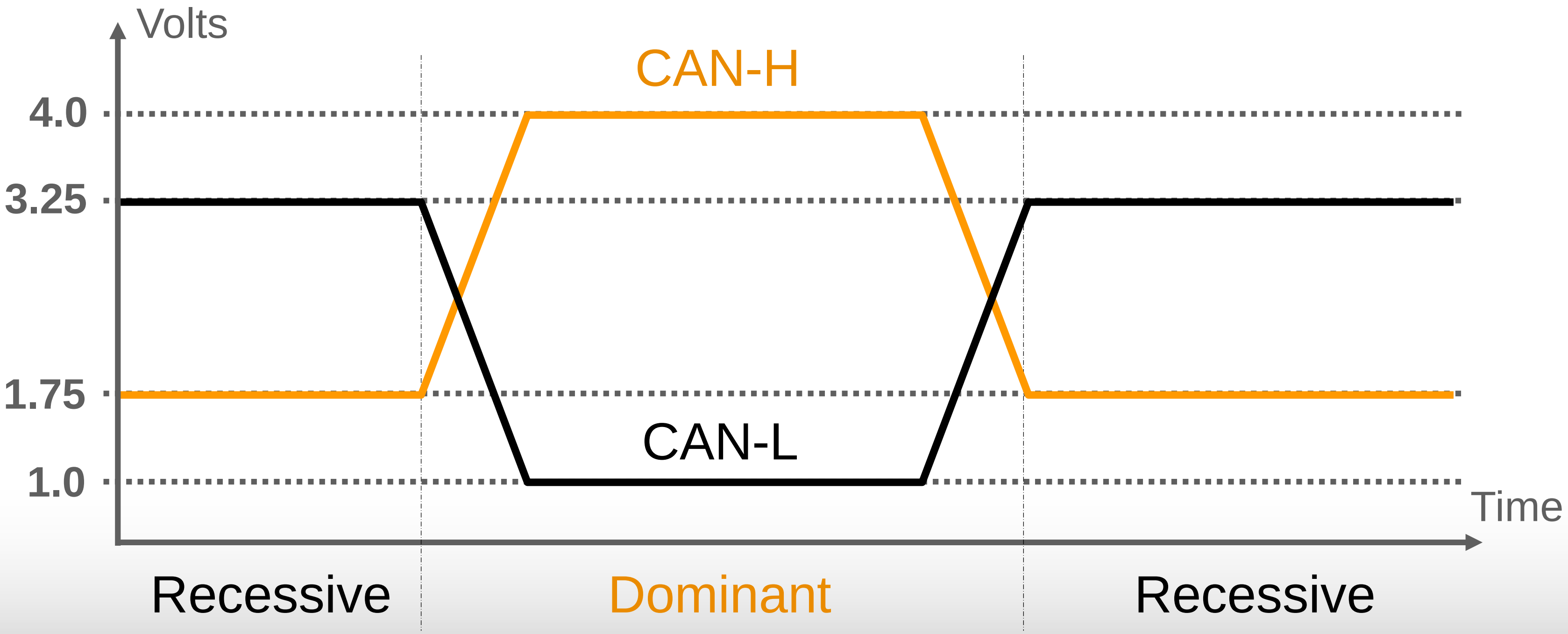


EN	enable input signal
NSTB	Node standby input signal
NWAKE	Node wake input signal
TxD	transmit data input
RxD	receive data output
BAT	battery voltage
VCC	supply voltage

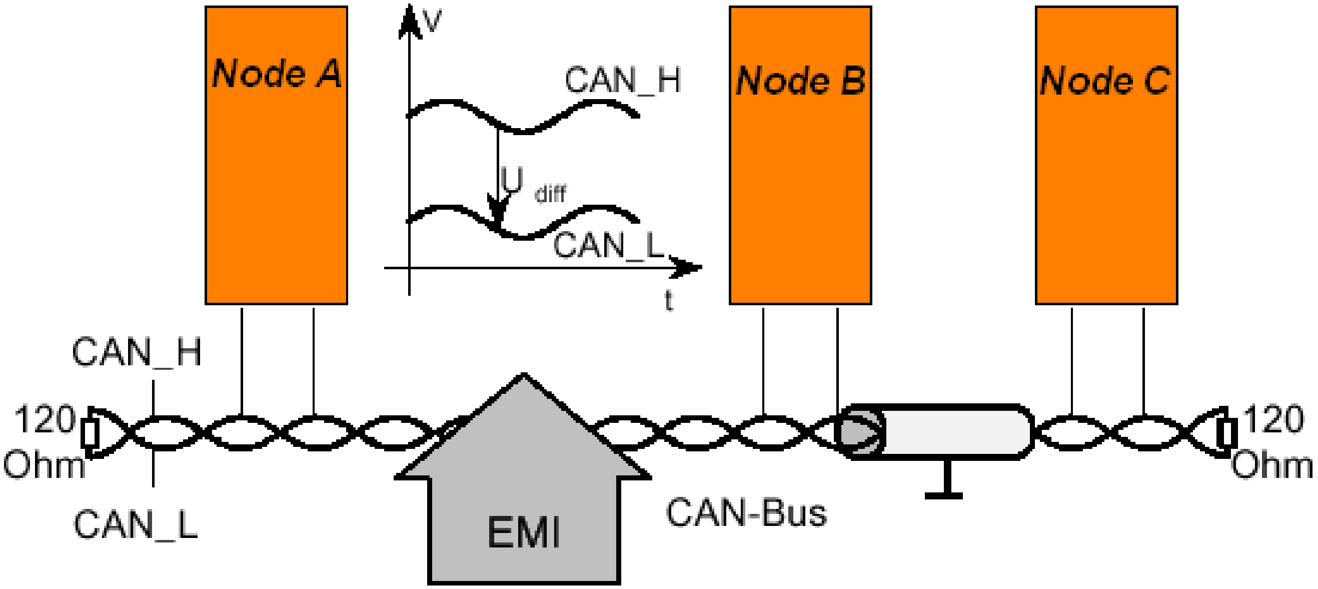
NERR	error output pin
CAN_H	high voltage bus line
RTH	termination resistor
CAN_L	low voltage bus line
RTL	termination resistor
INH	inhibit output
GND	ground

Physical Layer – Physical Aspects

- **CAN Low Speed Transmission** (ISO / DIS 11519-2)
  - ➔ Transmission rate 10 KBit/s up to 125 KBit/s
  - ➔ max. bus length depend on the distributed capacity of the line
  - ➔ up to 20 nodes



Physical Layer – CAN and EMI

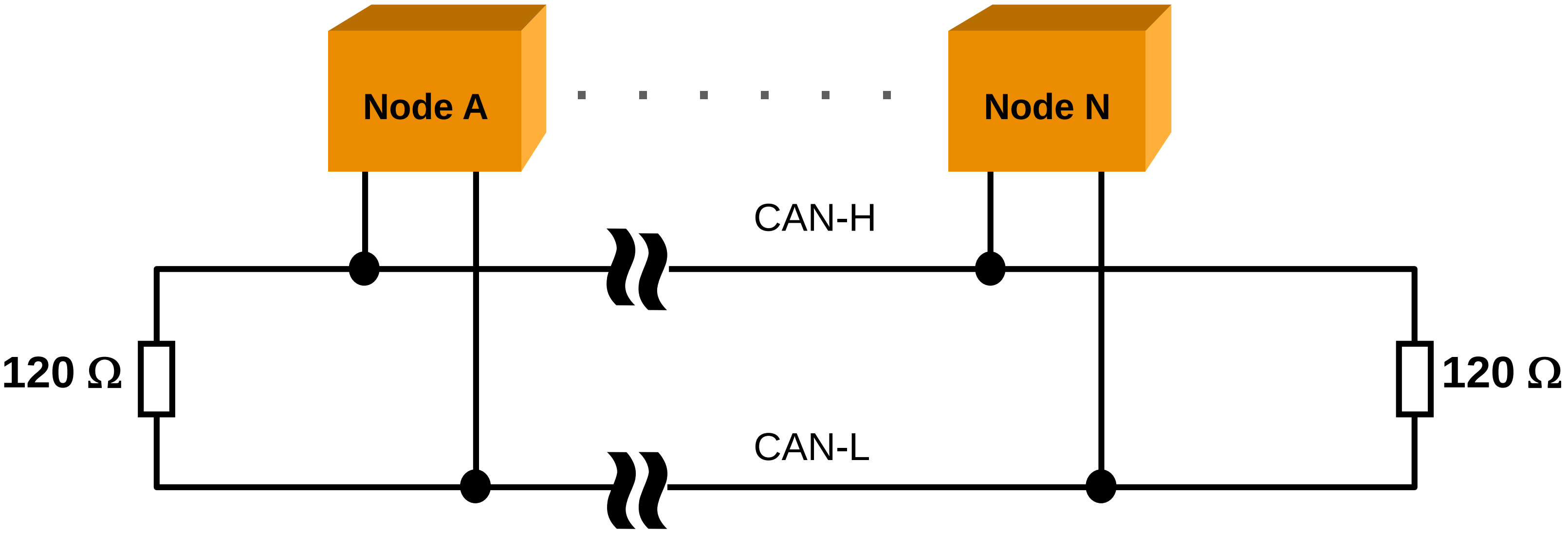


- Due to the differential nature of transmission CAN is insensitive to electromagnetic interference, because both bus lines are affected in the same way which leaves the differential signal unaffected

- To reduce the sensitivity against electromagnetic interference even more, the bus lines can additionally be shielded. This also reduces the electromagnetic emission of the bus itself, especially at high baudrates.

Physical Layer – DW bus line Termination

CAN



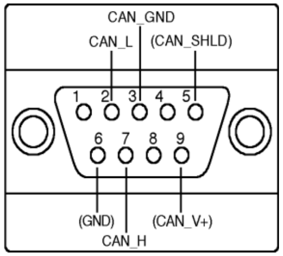
Physical Layer

CAN

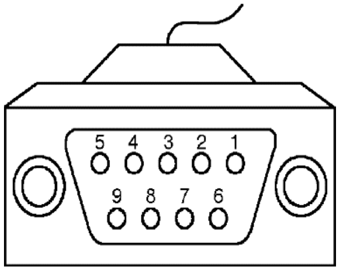
Connectors for CAN

- The CAN connector is defined according to the specifications ISO/DIS 11898 and CiA/DS 102-1 (defines pinning). The optional Pins 6 and 9 may be used to feed a NiPC (Networked industrial Process Control) unit.

male



female

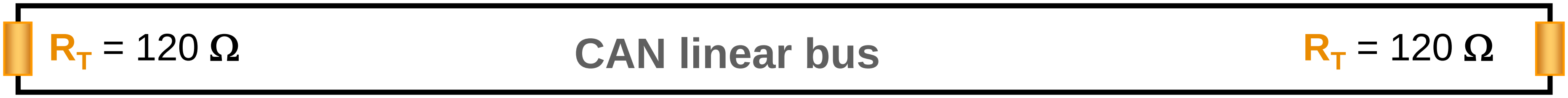


Pin	Signal	Description
1	-	Reserved
2	CAN_L	CAN_L bus line dominant low
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	Optional CAN Shield
6	(GND)	Optional Ground
7	CAN_H	CAN_H bus line dominant high
8	-	Reserved
9	(CAN_V+)	Optional CAN external pos. supply

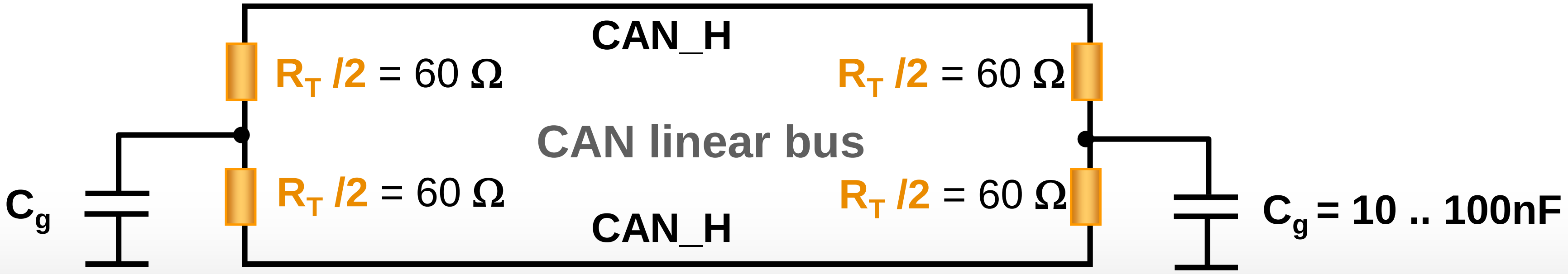
Physical Layer – Terminator Resistor

- The CAN bus cable needs at the ends a termination resistor R_T to provide EMC characteristics without corrupting the DC characteristics:

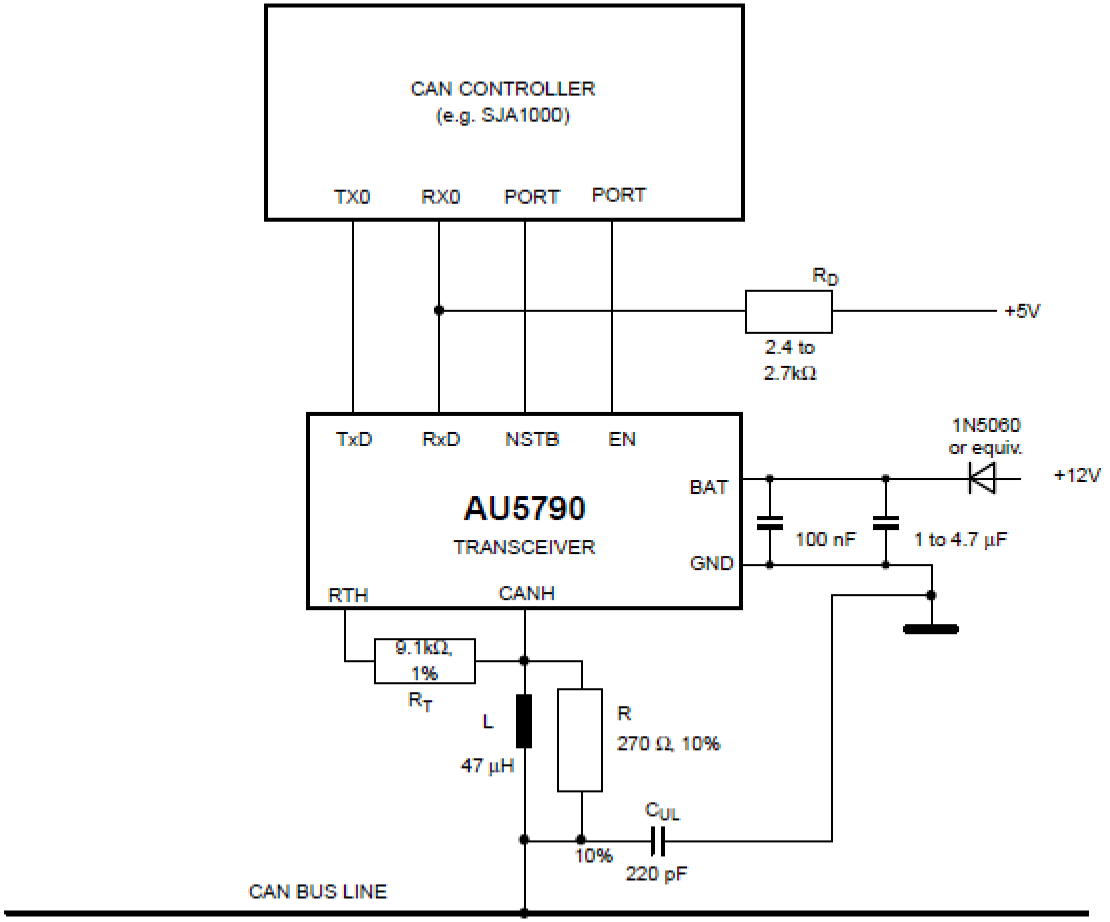
- For basic termination at a serial cable use for termination resistor R_T = 120 Ω with linear CAN bus lines.



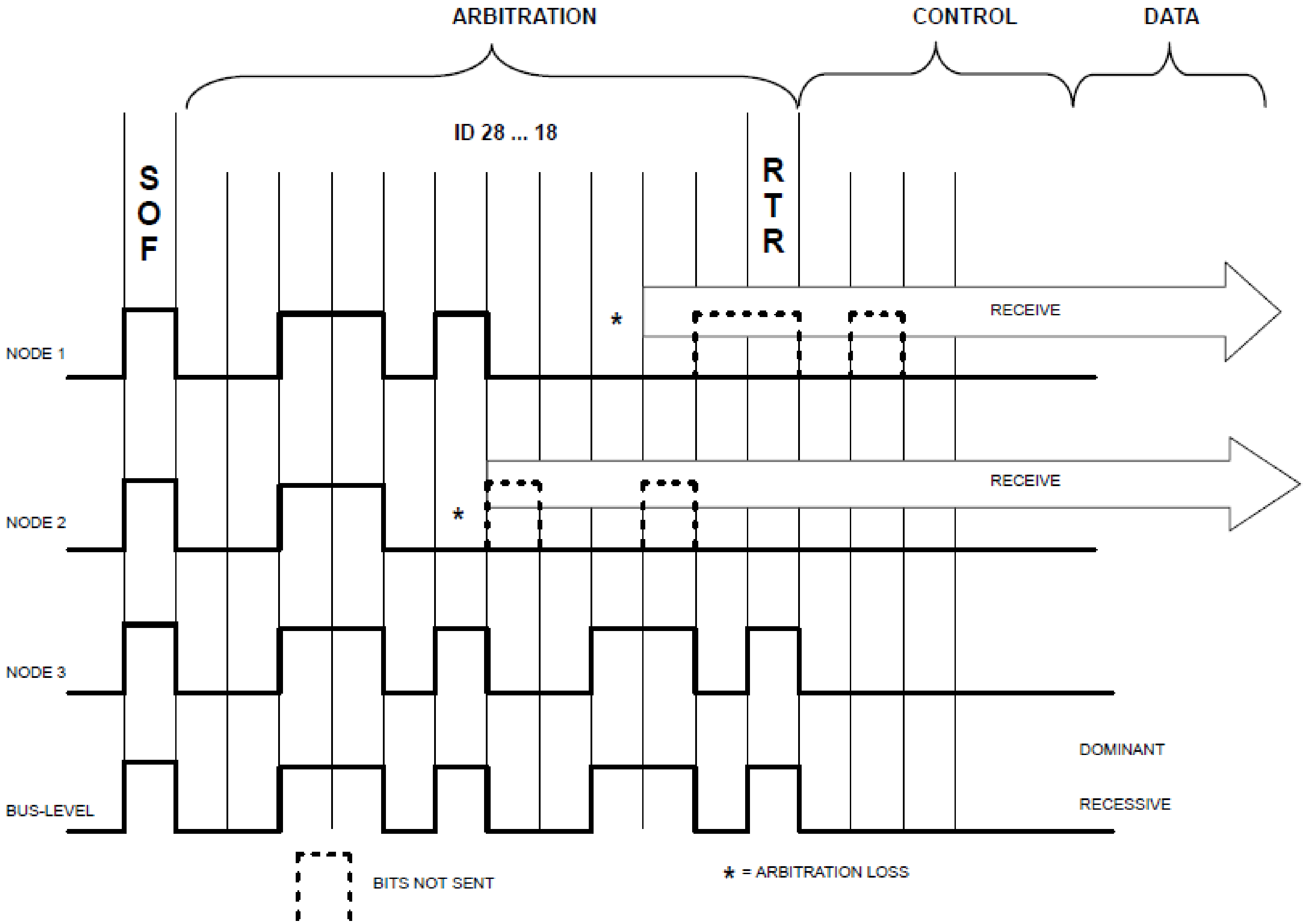
- For a split termination concept use for termination resistor $R_T/2$ = 60 Ω with a star architecture.



Physical Layer – SW CAN



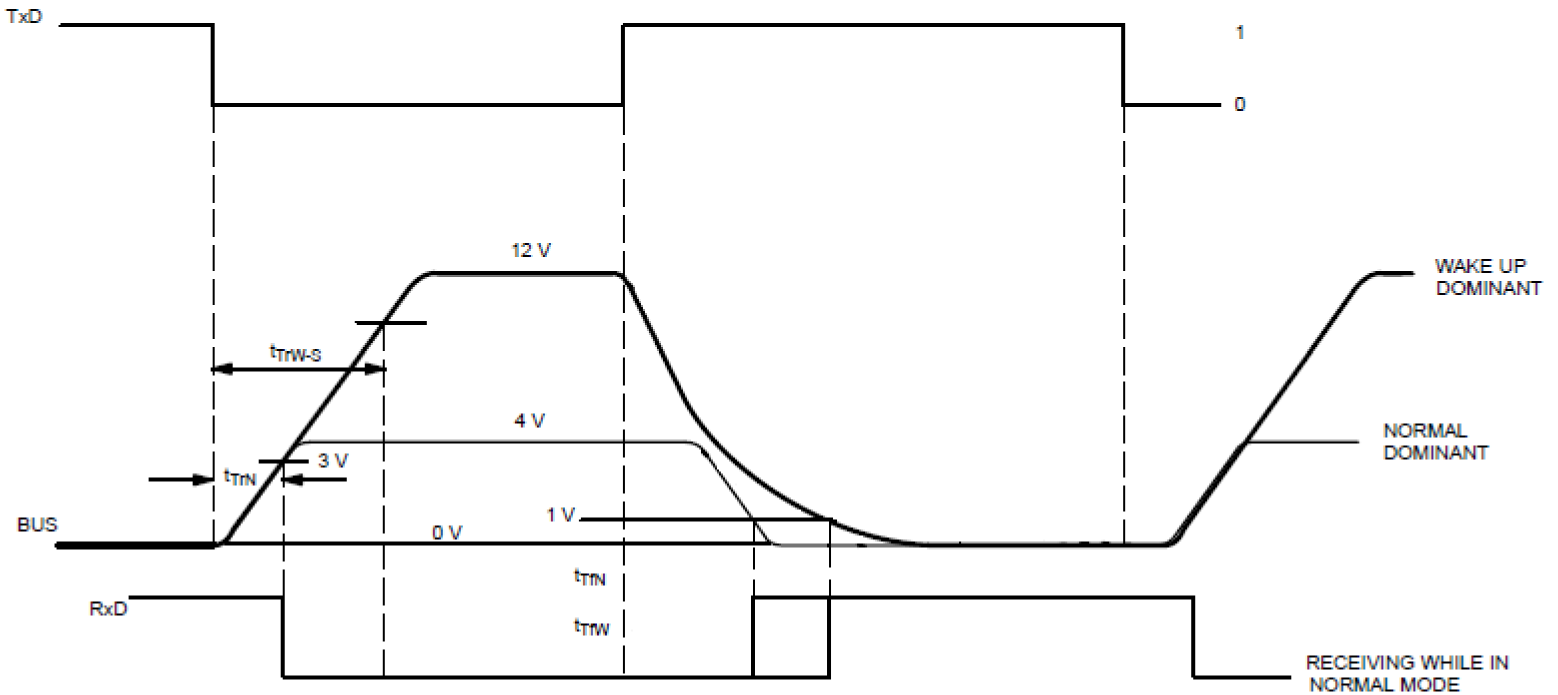
Arbitration – SW CAN



Abraham Tezmol, P ES FSD GDL

WK Up and Bus Signal Levels – SW CAN

CAN



TxD
1
0
12 V
WAKE UP
DOMINANT
t_{TrW-S}
4 V
NORMAL
DOMINANT
3 V
t_{TrN}
BUS
0 V
1 V
t_{TfN}
RxD
t_{TfW}
RECEIVING WHILE IN
NORMAL MODE