

See this training for *FREE* with *animations* and *computer voice* at:

<http://can-basics.canopen.nu>

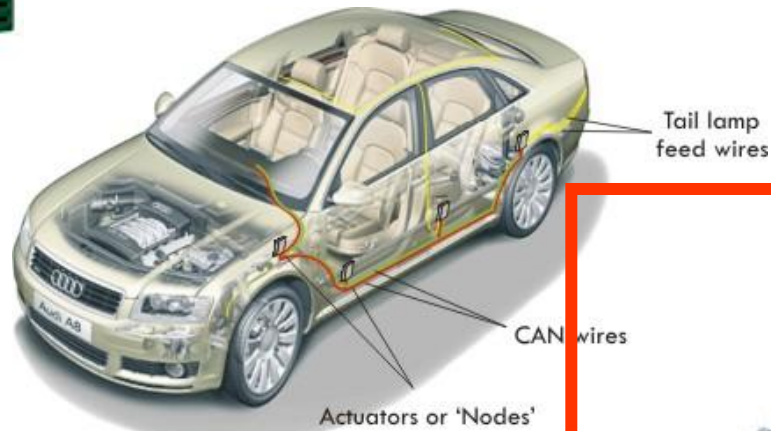
© Ulrik Hagström, 2009

ulrik@canopen.nu

# All uses CAN bus



Industrial  
automation

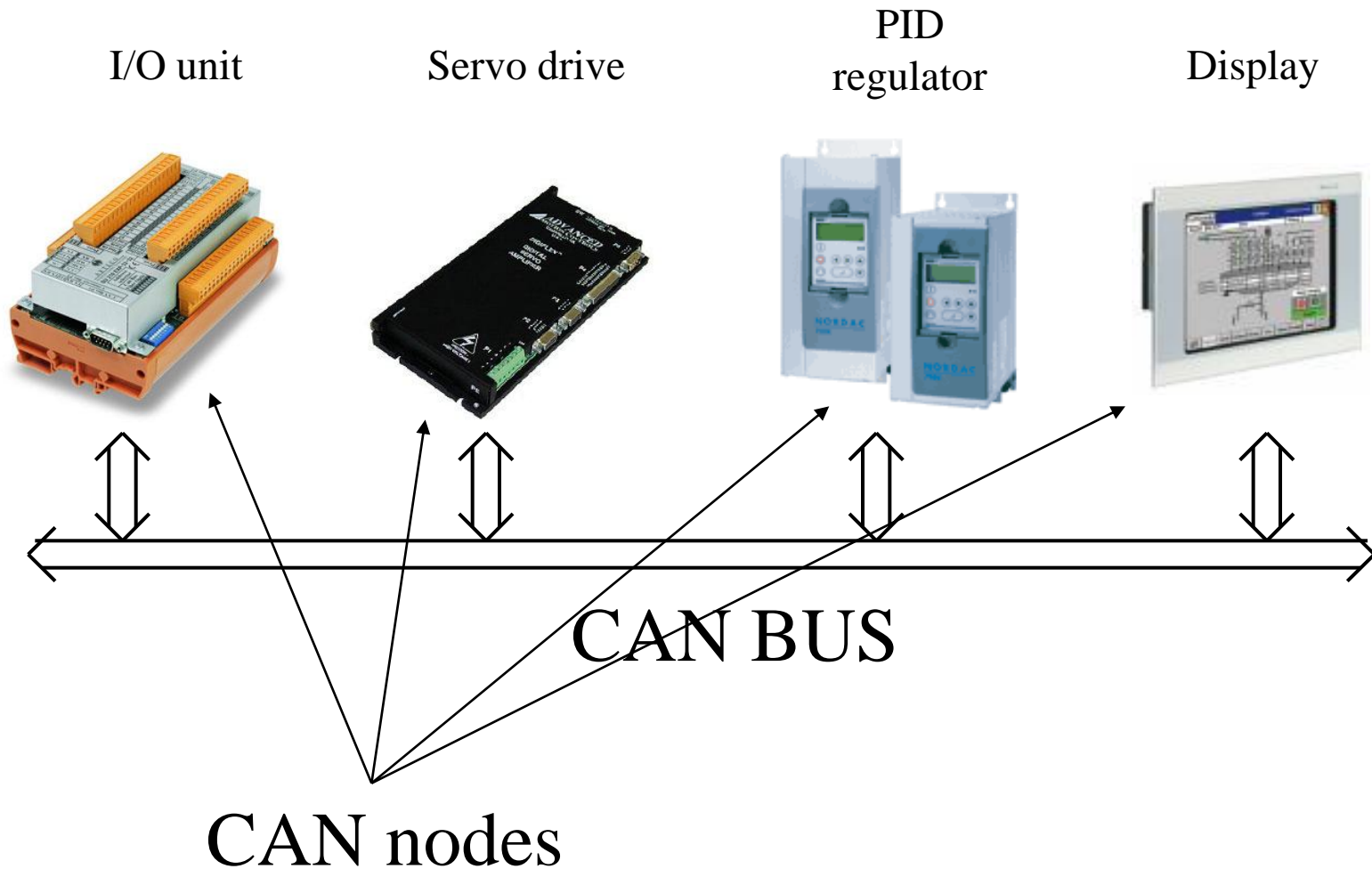


Automotive



Consumer products

# Typical CAN bus in industry



# CAN milestones

1986 - Robert Bosch GmbH  
requested by Mercedes.

1987 - The first CAN silicon  
fabricated in by Intel.

1988 - CAN available for  
everybody.

1991 - Mercedes S-model

1993 - ISO 11898  
specification.

# CAN offer

- Transmit/Receive a message

- Error handling

Standard CAN message

BROADCAST

“Babbling-idiot nodes” are silenced by setting them to “error passive” mode or even “bus off” automatically by the CAN chip if they keep destroying the data sent on the bus!

- Collision resolution handled with CSMA/CR with priority

If two nodes starts to transmit data at the same time the data sent with the highest priority wins.

(Two nodes sending messages with the same priority is not allowed!)

# Transmit / Receive

- CAN message contains:
  - priority
  - CRC checksum and other error protection data fields.
  - data
- No receive guarantee by consumer

# Error handling

- 5 error checks
- Error checks are done by all nodes.
- A CAN message is accepted by all nodes or no node.
- Automatic retransmission on error.

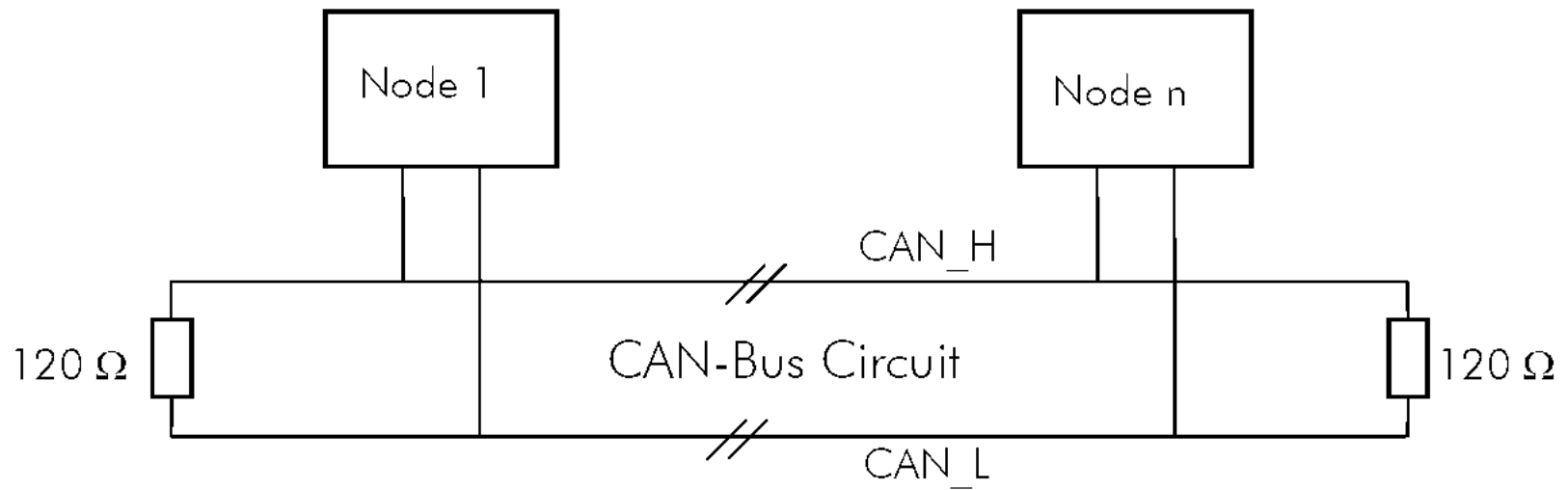
# Collision resolution

- Collisions never happens because:  
CSMA/CR = Carrier Sense Multiple Access  
Collision Resolution
- Collision Resolution is handled using priorities.

# Important characteristics

- ~ Max 110 nodes on one CAN bus  
(more in theory)
- 1 Mbps  $\Leftrightarrow$  40 m
- 5 Kbps  $\Leftrightarrow$  10 000 meters.

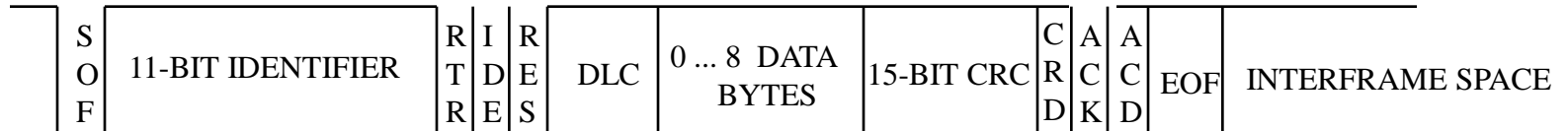
# CAN bus example



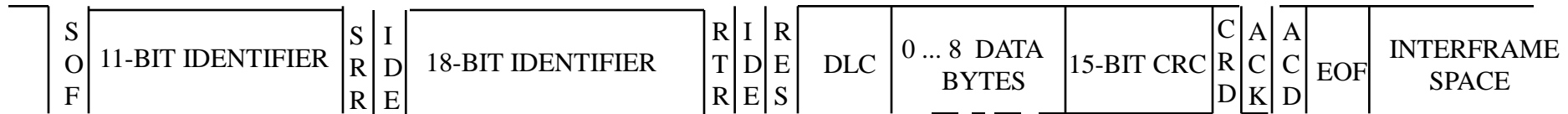
(the CAN bus is usually "twinned")

# CAN frame

## Standard CAN message



## Extended CAN



Number of data bytes	0	1	2	3	4	5	6	7	8
Minimum message length	44	52	60	68	76	84	92	100	108
Maximum message length	51	60	70	80	89	99	108	118	128
Bistuffing	(7)	(8)	(10)	(12)	(13)	(15)	(16)	(18)	(20)

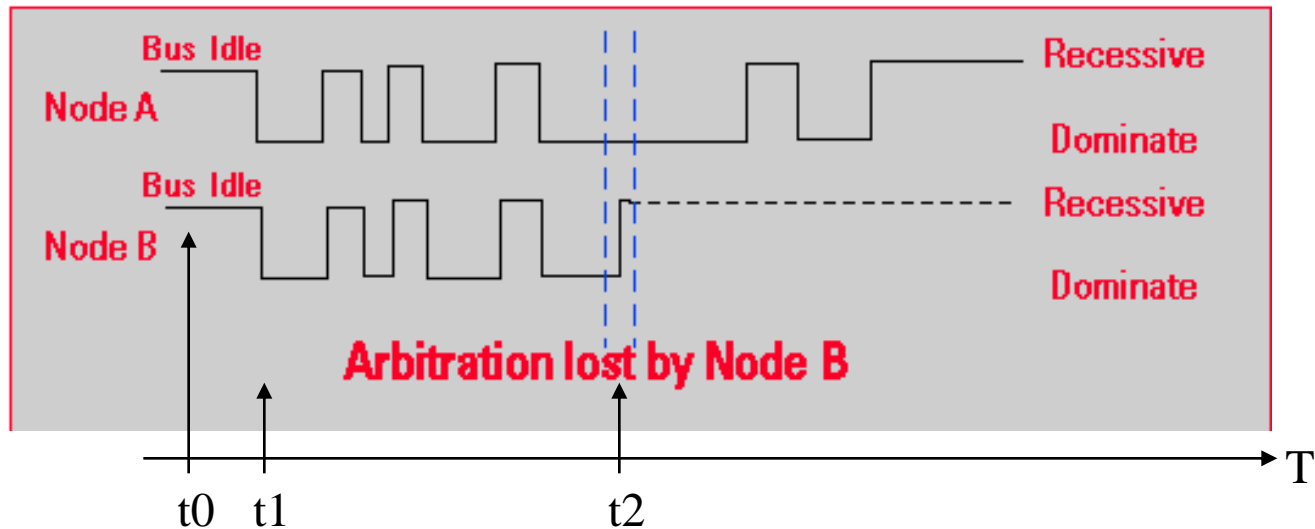
# Three types of CAN controllers

## Part A and Part B comp ability

There are three types of CAN controllers: Part A, Part B passive and Part B. They are able to handle the different parts of the standard as follows:

CAN chip type	Part A	Part B passive	Part B
11 bit identifier	OK.	OK.	OK.
29 bit identifier	ERROR!	Tolerated on the bus, but ignored.	OK.

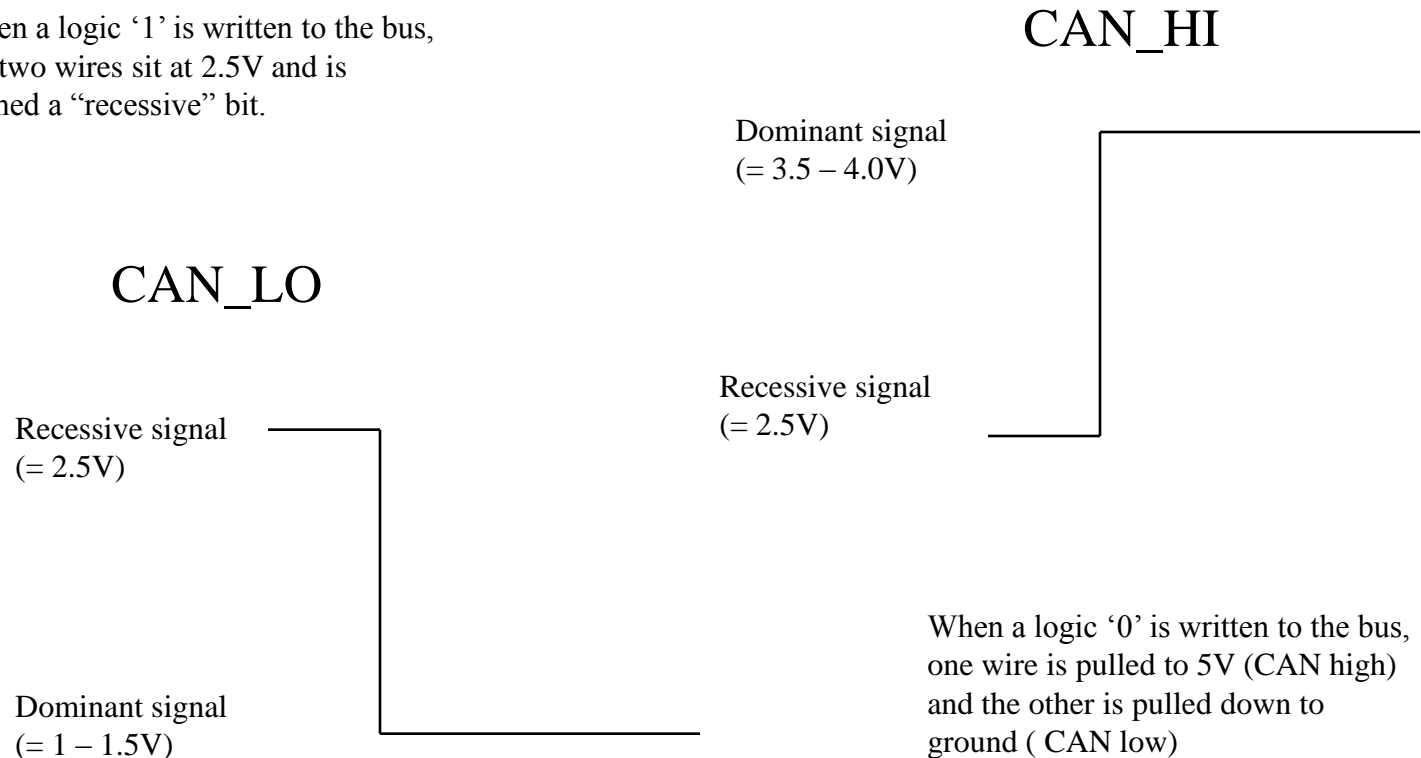
# Arbitration



- t0 Both "Node A" and "Node B" consider bus idle.
- t1 Both nodes start transmit "SOF" (Start of frame)
- t2 "Node A" transmits dominant bit and "Node B" recessive, and "Node A" wins the arbitration.

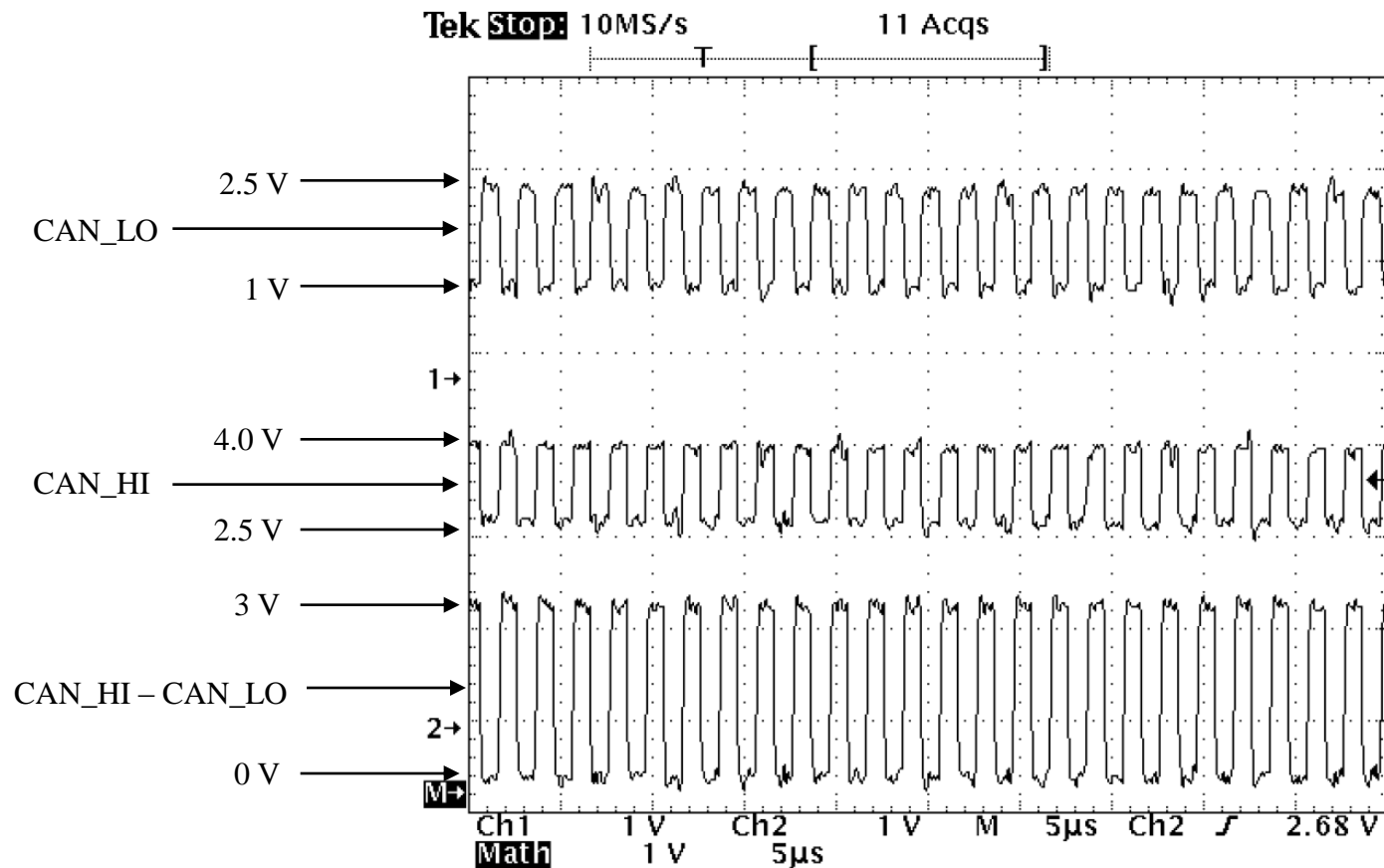
# CAN bit arbitration

When a logic '1' is written to the bus, the two wires sit at 2.5V and is termed a "recessive" bit.

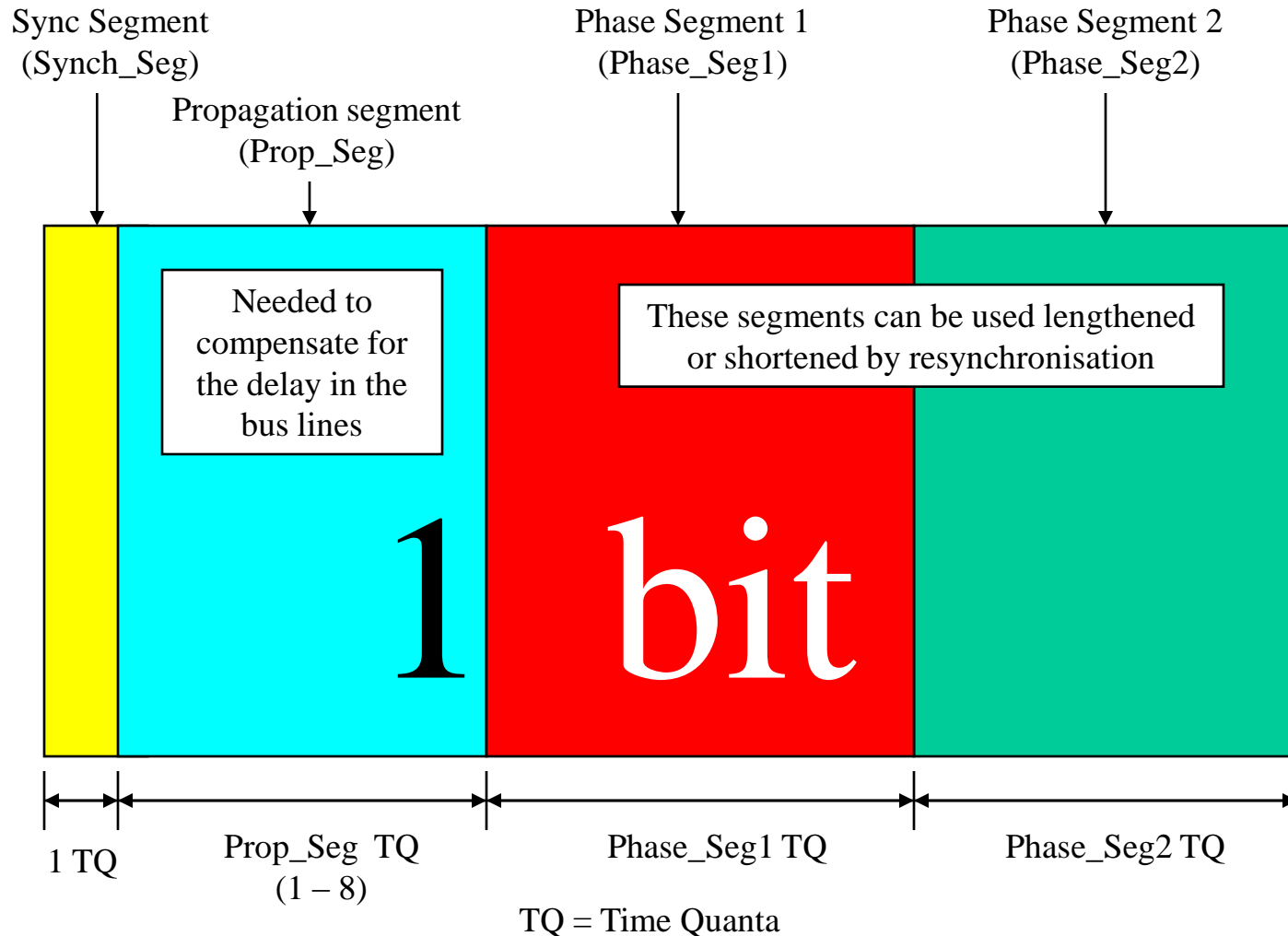


If both lines are at the same voltage, the signal is a recessive bit. If the CAN\_HI line is higher than the CAN\_LO line by 0.9V, the signal line is a dominant bit. If just one node is driving the bus to a logical 0 (=dominant bit), then the whole bus is in that state regardless of the number of nodes transmitting a logical 1.

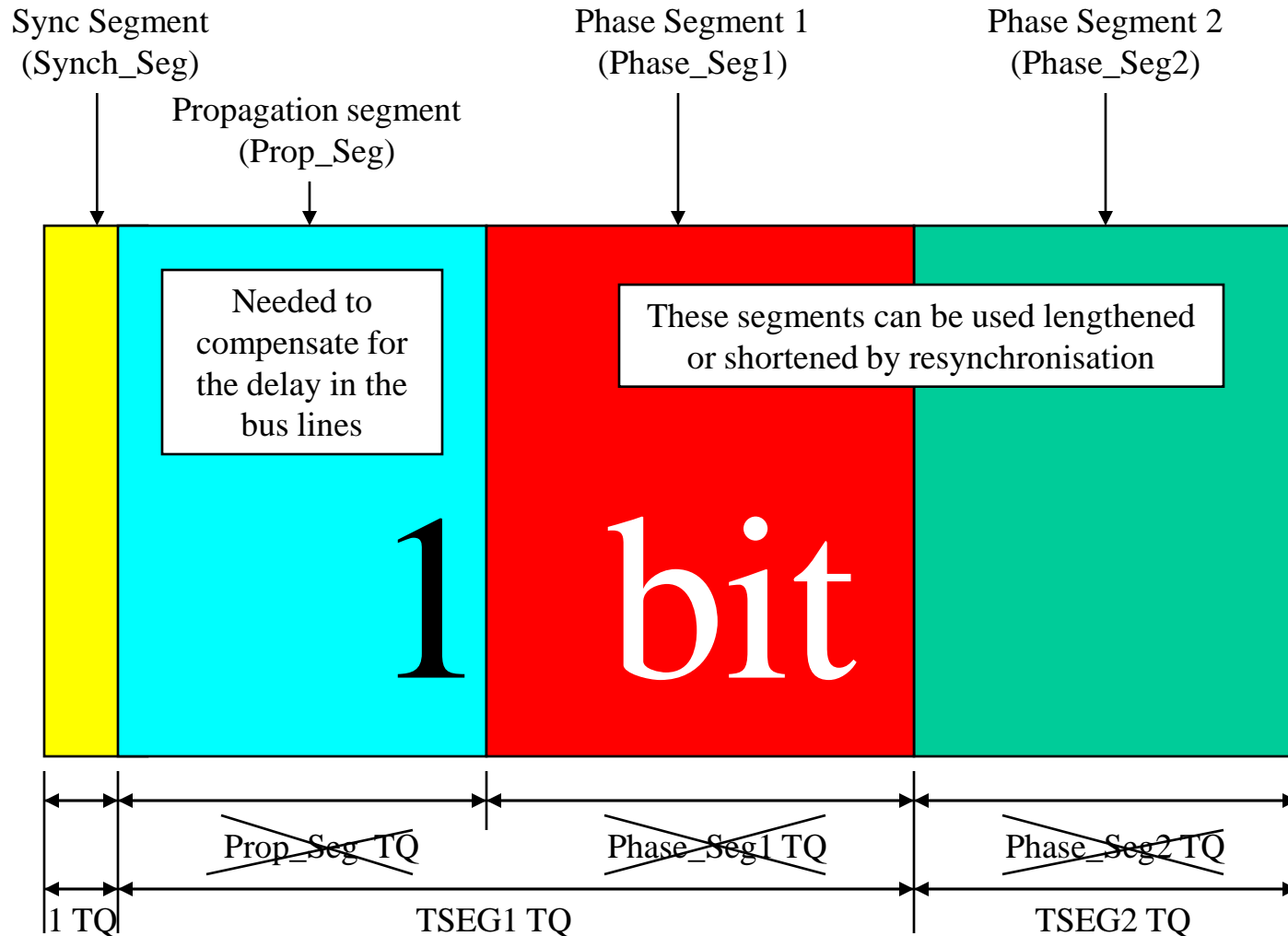
# CAN signals



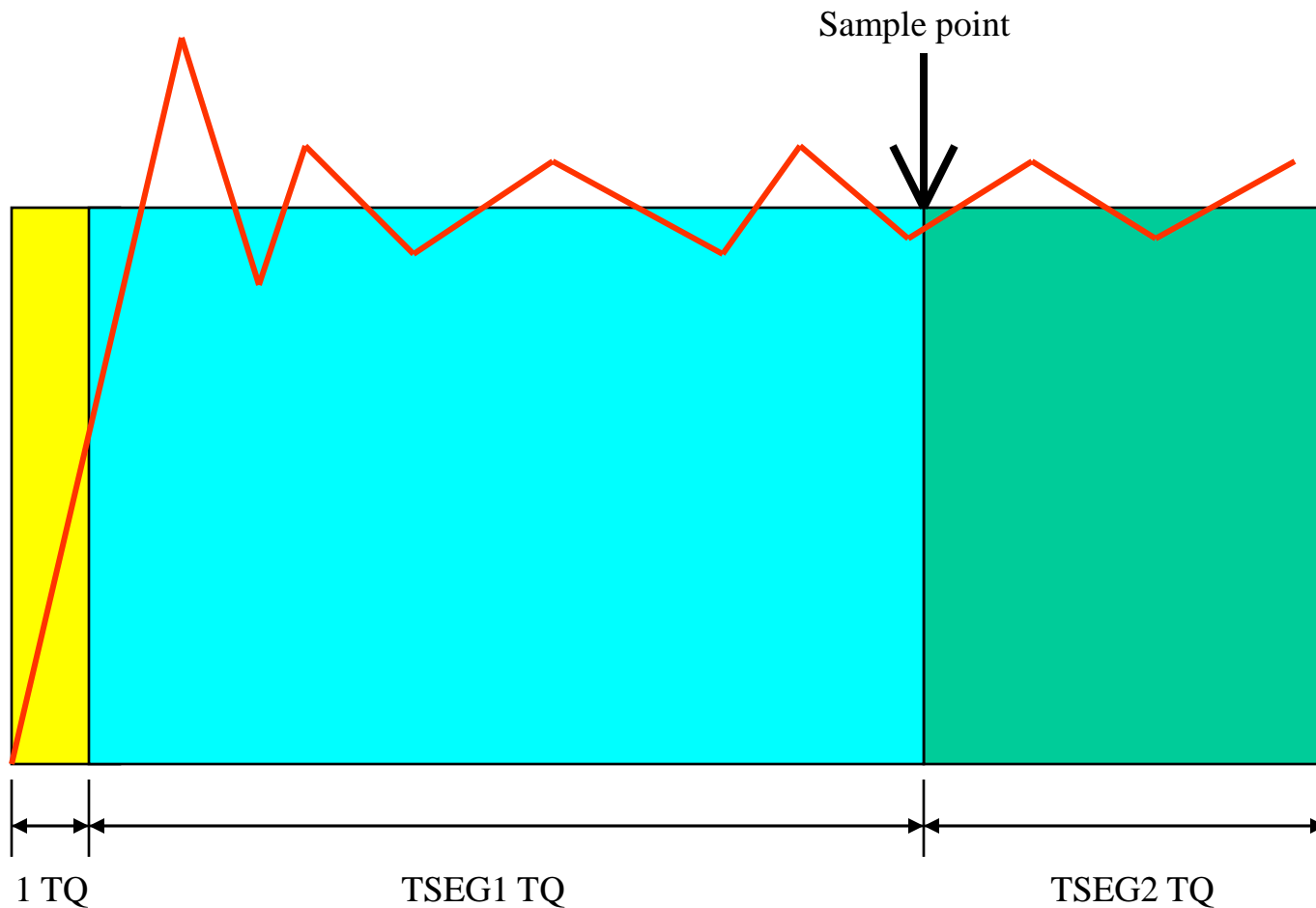
# 1 bit on the CAN bus (BOSCH)



# 1 bit on the CAN bus ("ISO")



# Sample point on 1 bit



# Bitrate settings

$$n = \text{SYNCHSEG} + \text{TSEG1} + \text{TSEG2}$$

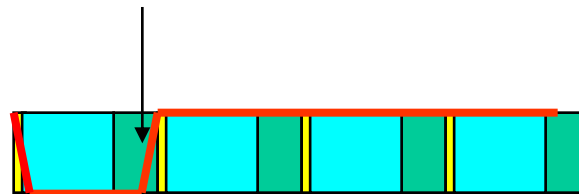
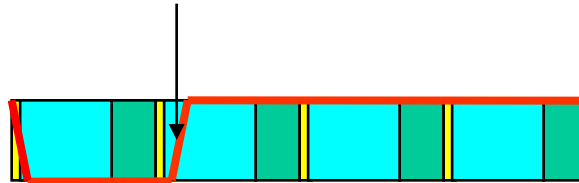
BRP = value of the Bit rate Prescaler  
(register in the CAN controller)

$$\text{Bitrate} = \frac{f_{\text{crystal}}}{2 * n * (\text{BRP} + 1)}$$

# Resync and SJW

Hard resynchronization

Resynchronization within a frame



# Typical settings...

Bit rate Bus length <sup>(1)</sup>	Nominal bit time $t_b$	Number of time quanta per bit	Length of time quantum $t_q$	Location of sample point
1 Mbit/s 25 m	1 $\mu$ s	8	125 ns	6 $t_q$ (750 ns)
800 kbit/s 50 m	1,25 $\mu$ s	10	125 ns	8 $t_q$ (1 $\mu$ s)
500 kbit/s 100 m	2 $\mu$ s	16	125 ns	14 $t_q$ (1,75 $\mu$ s)
250 kbit/s 250 m <sup>(2)</sup>	4 $\mu$ s	16	250 ns	14 $t_q$ (3,5 $\mu$ s)
125 kbit/s 500 m <sup>(2)</sup>	8 $\mu$ s	16	500 ns	14 $t_q$ (7 $\mu$ s)
50 kbit/s 1000 m <sup>(3)</sup>	20 $\mu$ s	16	1,25 $\mu$ s	14 $t_q$ (17,5 $\mu$ s)
20 kbit/s 2500 m <sup>(3)</sup>	50 $\mu$ s	16	3,125 $\mu$ s	14 $t_q$ (43,75 $\mu$ s)
10 kbit/s 5000 m <sup>(3)</sup>	100 $\mu$ s	16	6,25 $\mu$ s	14 $t_q$ (87,5 $\mu$ s)

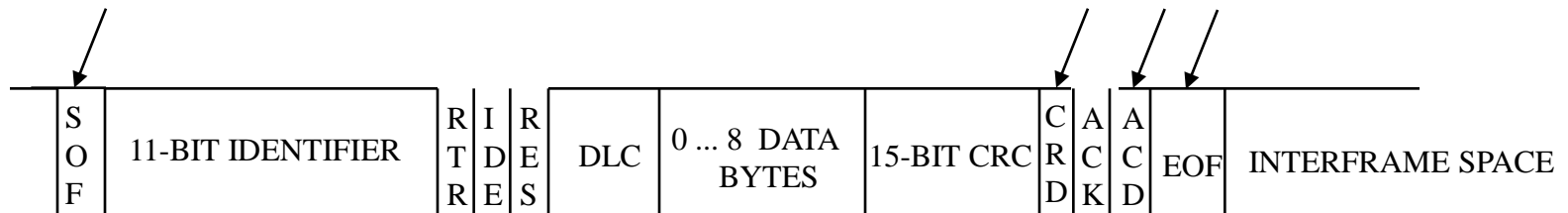
## The five error checks...

- **Bit monitoring.**
- **Bit stuffing.**
- **Frame check.**
- **Acknowledgement check.**
- **CRC check.**

→ Error Frame → Automatic retransmission

# Form- and biterror

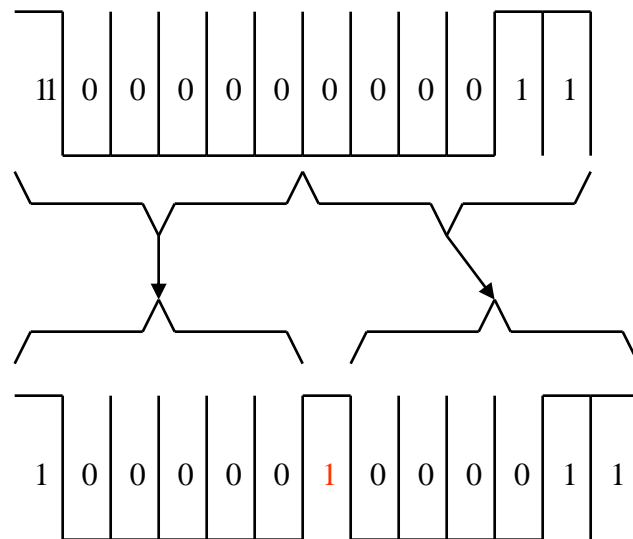
- Form error



- Bit error

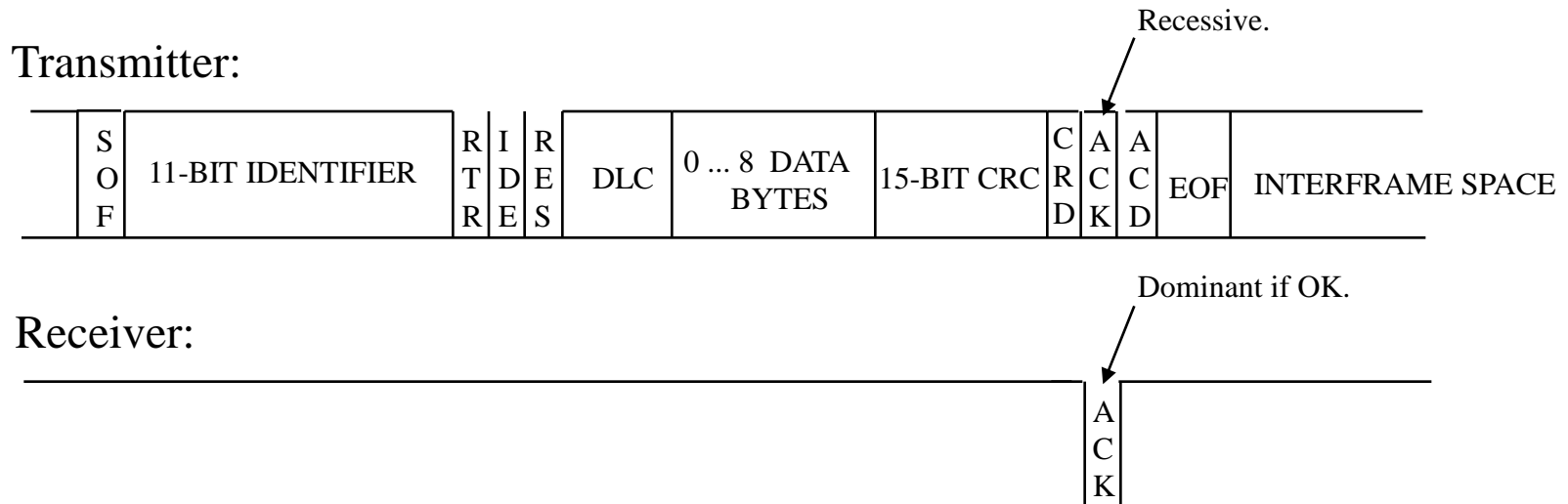
If a bit is written onto the bus and its complement is read back a “Bit error” is generated.

# Bit stuffing error

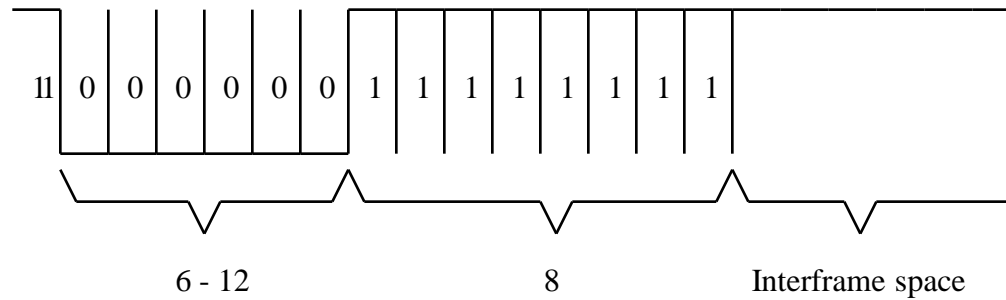


# Acknowledgement check

CRC check and the acknowledge slot ("Form error", "bit stuffing error", "CRC error", "Ack Error")



# CAN error frame



# CAN controller error modes

- **Error active**

Tx error counter  $\leq 127$  AND Rx error counter  $\leq 127$

- **Error passive**

(Tx error counter  $> 127$  OR Rx error counter  $> 127$ ) AND Tx error counter  $\leq 255$ .

- **Bus off**

(Tx error counter  $> 255$ )

# Rx error counter rules

- Receiver detects error (any): the Rx error counter will be increased by 1, except when the detected error was a bit error during the sending of an active error flag or an overload flag (=this specific node did not see the error that another node saw).
- Receiver detects a dominant bit as the first bit after sending an error flag: the Rx error counter will be increased by 8.
- If a receiver detects a bit error (“what was written was not read”) while sending an active error flag or an overload flag the Rx error counter is increased by 8.
- *After the successful reception of a message (reception without error up to the acknowledge slot and the successful sending of the acknowledge bit), Rx error counter is decreased by 1 if it was between 1 and 127. If Rx error counter was 0 it stays 0, and if it was greater than 127, it will be set to a value between 119 and 127.*

# Tx error counter rules

- When a transmitter sends an error flag, the Tx error counter is increased by 8. Important exception: If a node is the only one on the bus (or during start-up the only one that has become active), and it transmits a message, it will get an acknowledgement error, and will retransmit the message. This may lead to that node going to error passive mode – but it will not go bus off (=“oscillate”)
- If a transmitter detects a bit error while sending an active error flag or an overload flag, the Tx error counter is increased by 8.
- *After the successful transmission of a message (getting ack and no error until end of frame is finished) Tx error counter is decreased by 1 unless it was already 0.*