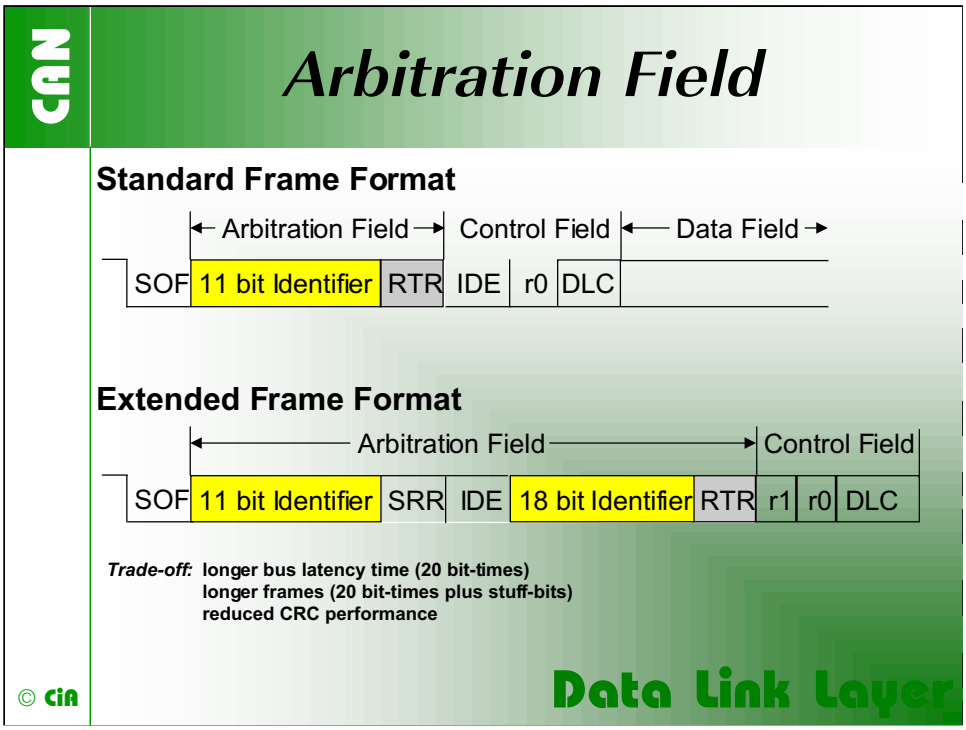
CAN

Controller Area Network

- CAN data link layer
- CAN implementation
- CAN high-speed physical layer

© CiA

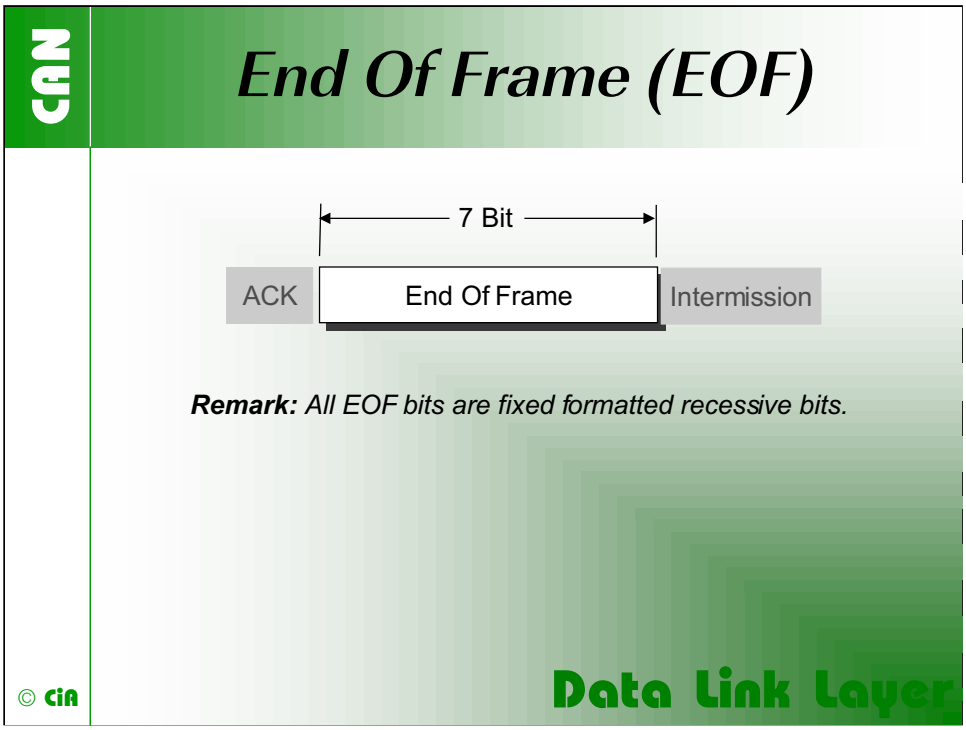
Data Link Layer



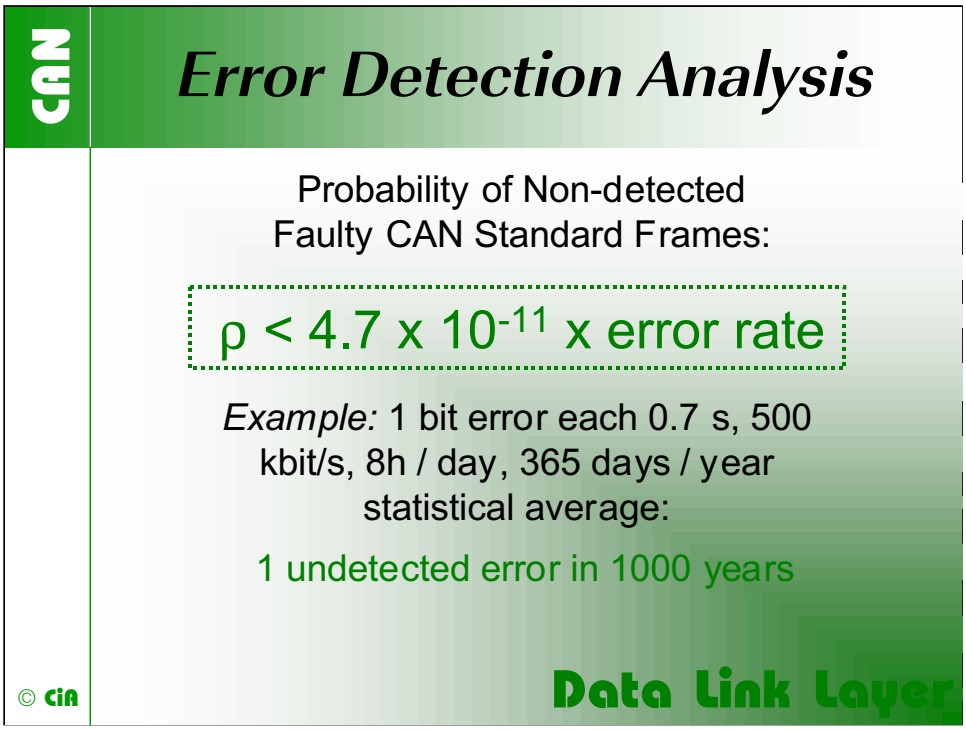
The Identifier´s length in the Standard Format is 11 bit and corresponds to the Base ID in Extended Format. The Identifier is followed by the RTR bit. In Data Frames the RTR bit has to be dominant. Within a Remote Frame the RTR bit has to be recessive. The Base ID is followed by the IDE (Identifier Extension) bit transmitted dominant in the Standard Format (within the Control Field) and recessive in the Extended Format. So the Standard Frame prevails the Extended Frame in case of collision.

The Extended Format comprises two sections: Base ID with 11 bit and the Extended ID with 18 bit. The SRR (Substitute Remote Request) bit substitutes the RTR bit and is transmitted recessive. If the SRR is corrupted and transmitted dominant, the receivers will ignore this. But the value is not ignored for bit stuffing and arbitration. Since the SRR bit is received before the IDE bit, a receiver cannot decide instantly whether it receives a RTR or a SRR bit.

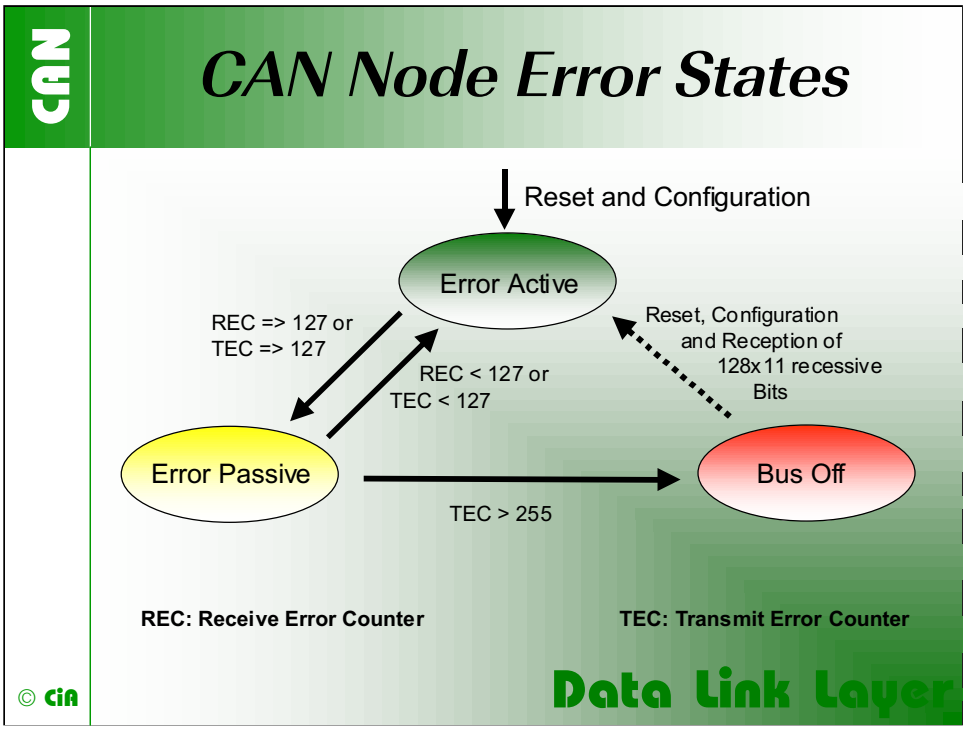
The format of the Control Field is similar for Standard Format and Extended Format. The Control Field in Standard Format includes the Data Length Code (DLC), the IDE bit, which is transmitted dominant and the reserved bit r0 also transmitted dominant. The Control Field in Extended Format includes the DLC and two the reserved bits r1 and r0. The reserved bits have to be sent dominant, but receivers accept dominant and recessive bits in all combinations.



Each Data and Remote Frame is delimited by a flag sequence of seven recessive bits. This EOF was introduced because an Error Frame caused by a global CRC failure should be transmitted within the length of Data or Remote Frame.

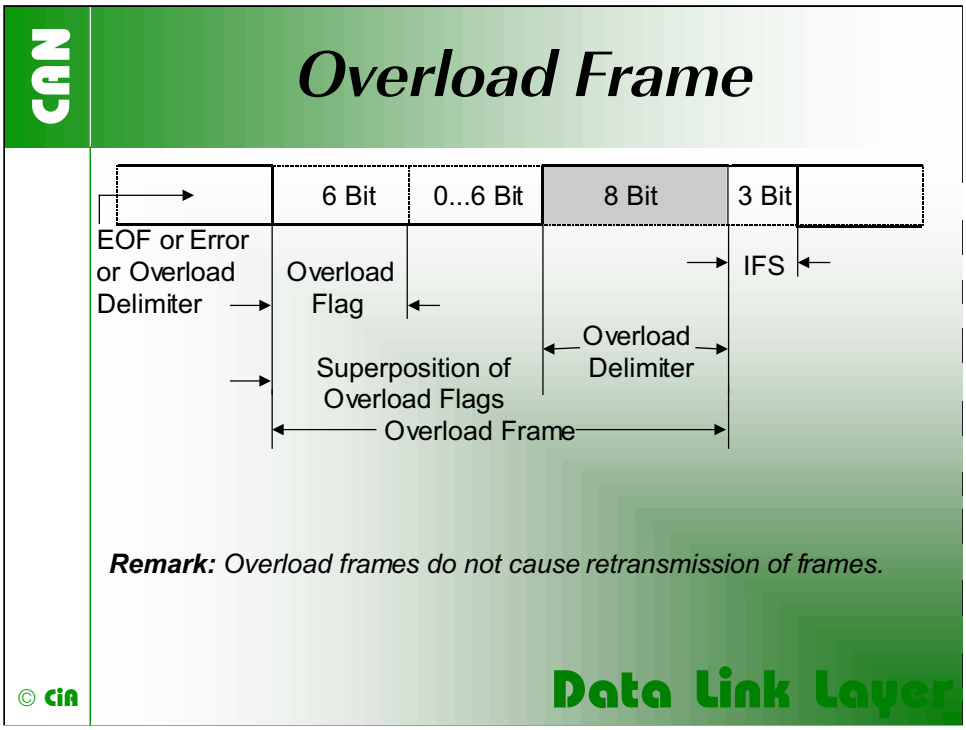


The probability of non-detected faulty messages is subject of several research projects, which are reported in the literature especially on the international CAN Conferences. This probability depends on several parameters such as the average corrupted message rate, which is about 10^{-3} for standard cables. In general, the probability of non-detected faulty messages is higher for Extended Frames as for Standard Frames.



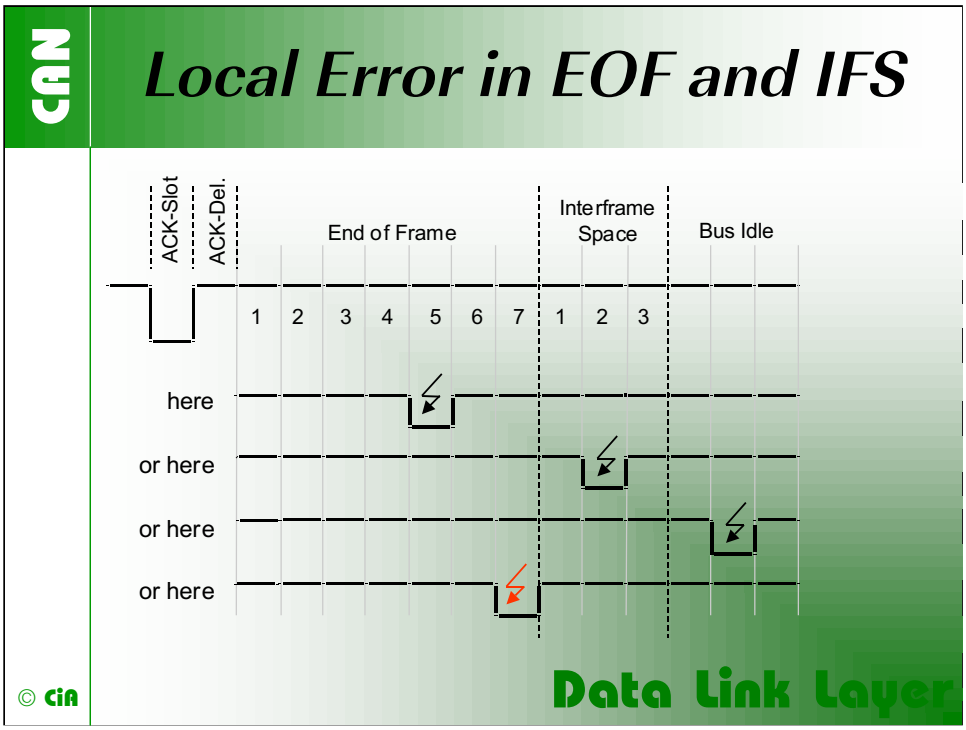
To distinguish between temporary and permanent failures every CAN controller has two Error Counters: The REC (Receive Error Counter) and the TEC (Transmit Error Counter). The counters are incremented upon detected errors respectively are decremented upon correct transmissions or receptions. Depending on the counter values the state of the node is changed: The initial state of a CAN controller is Error Active that means the controller can send active Error Flags. The controller gets in the Error Passive state if there is an accumulation of errors.

On CAN controller failure or an extreme accumulation of errors there is a state transition to Bus Off. The controller is disconnected from the bus by setting it in a state of high-resistance. The Bus Off state should only be left by a software reset. After software reset the CAN controller has to wait for 128 x 11 recessive bits to transmit a frame. This is because other nodes may pending transmission requests. It is recommended not to start an hardware reset because the wait time rule will not be followed then.



An Overload Frame can be generated by a node if due to internal conditions the node is not yet able to start reception of the next message or if during Intermission one of the first two bits is dominant. Another overload condition is that a message is valid for receivers, even when the last bit of EOF is received as dominant. Therefore, this dominant bit is not regarded as an error. On the other hand, the fixed-form bit filed EOF contains an illegal bit and the receiver of the dominant bit may have lost synchronization, which requires a reaction. The Reference CAN Model follows the example of a dominant bit as the last bit of Error or Overload Delimiter which are responded with an Overload Frame.

With an Overload Frame the transmitter is requested to delay the start of the next transmission. The Overload Frame is identical to an Active Error Frame. The only difference is that an Overload Frame does not increase the error counters (see error confinement) and does not causes a retransmission of a frame. Every node may transmit consecutively only 2 Overload Frames.



If one of the EOF (End of Frame) bits 1 to 6 are detected locally as dominant bits the node will send an Error Flag to globalize this failure.

The CAN specification reads as follows:

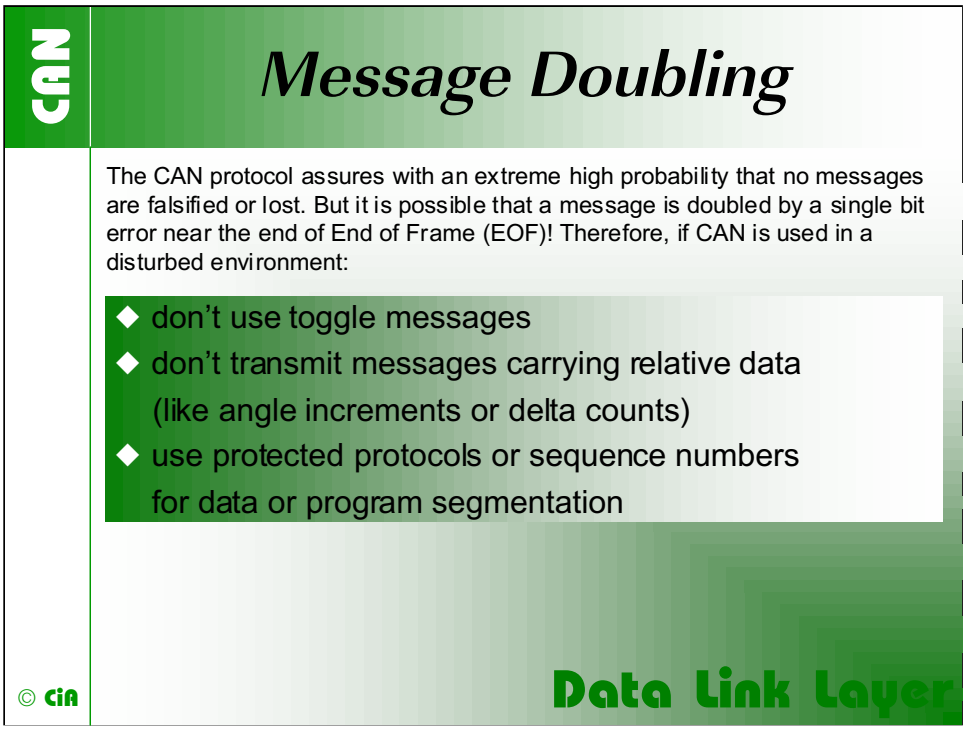
The point of time at which a message is taken to be valid is different for the transmitter and the receivers of the message.

• Transmitter:The message is valid for the transmitter, if there is no error until the end of End of Frame. If a message is corrupted, retransmission will follow automatically.

• Receiver: The message is valid for the receiver, if there is no error until the last but one bit of End of Frame.

A Receiver, which has sampled a dominant value during the 7th EOF bit do not regard this as an error. On the other hand, the fixed-form bit contains an illegal bit and the Receiver may have lost synchronization, therefore an Overload Frame is transmitted.

As the Receiver can inform the sender at the earliest during the next bit time, it's obvious that the Transmitter must wait one additional bit time for his message validation. This is independent of which bit is defined as the validation bit. It does not help to introduce (one or more) additional bits at the end of the message frame.

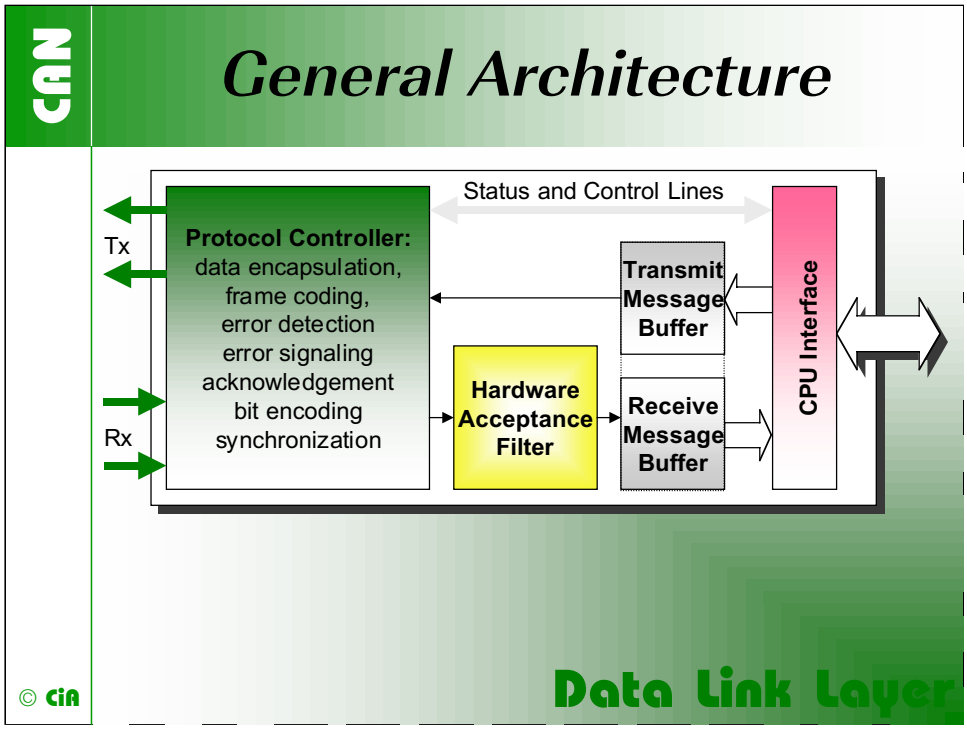


If a Transmitter samples a dominant bus level during the last bit of EOF it must retransmit that message. The dominant bus level can result from:

1. a receivers Error Flag reporting a local error in the last but one bit of EOF
2. a disturbance of the last bit of EOF

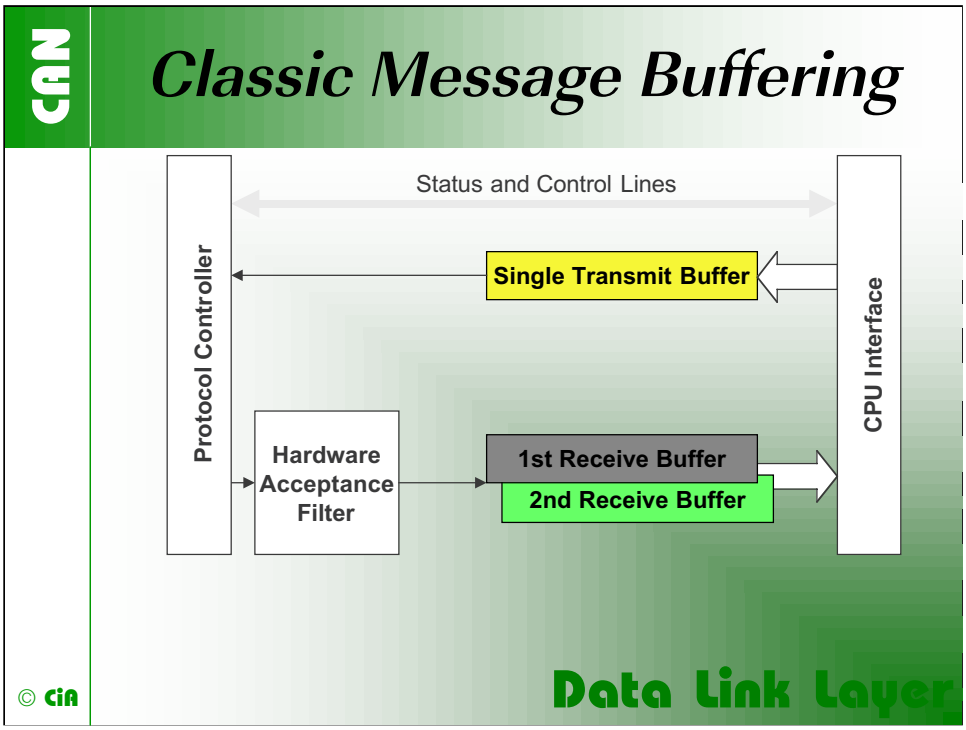
In case 1 it's likely that there are other receivers which have not been affected by the local error and therefore have already accepted the first message.

In case 2 all receivers have already accepted the first message. After the retransmission they have got the same message twice.

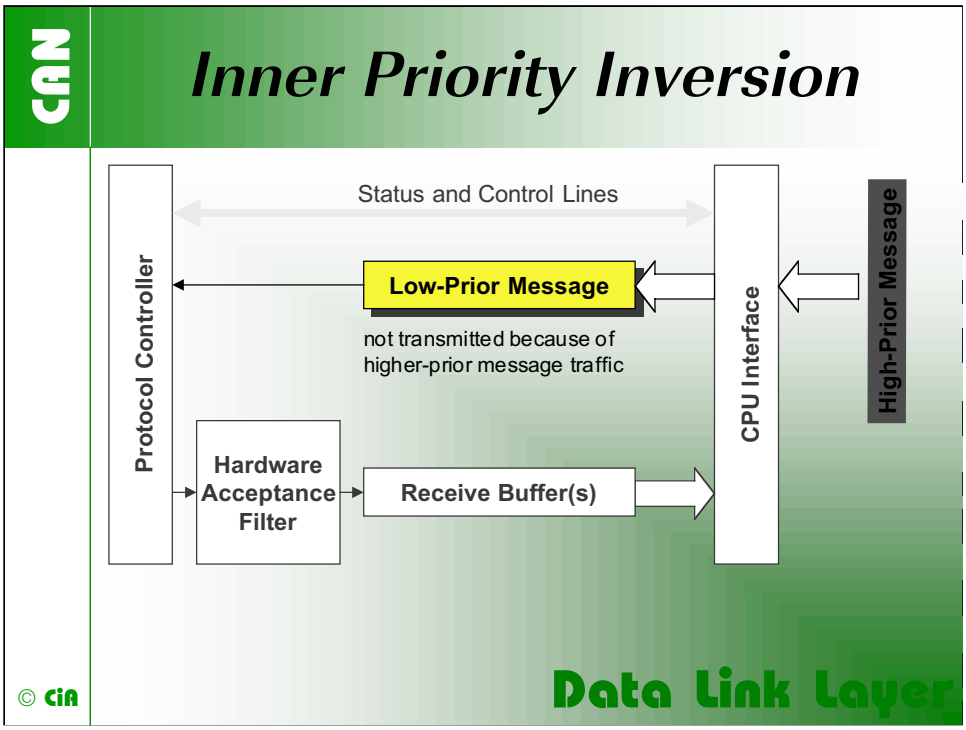


Several companies have implemented in silicon the international standardized Controller Area Network (ISO 11898). The layered data link layer provides basic communication services such as transmission of data and requesting of data. The CAN protocol also handles the detection of failures as well as the error indication. All existing implementations use the same protocol controller, which has to be compliant with the C or VHDL reference model from Bosch. The implementation differ in the acceptance filtering, the frame storage capabilities, and in additional, nice-to-have features.

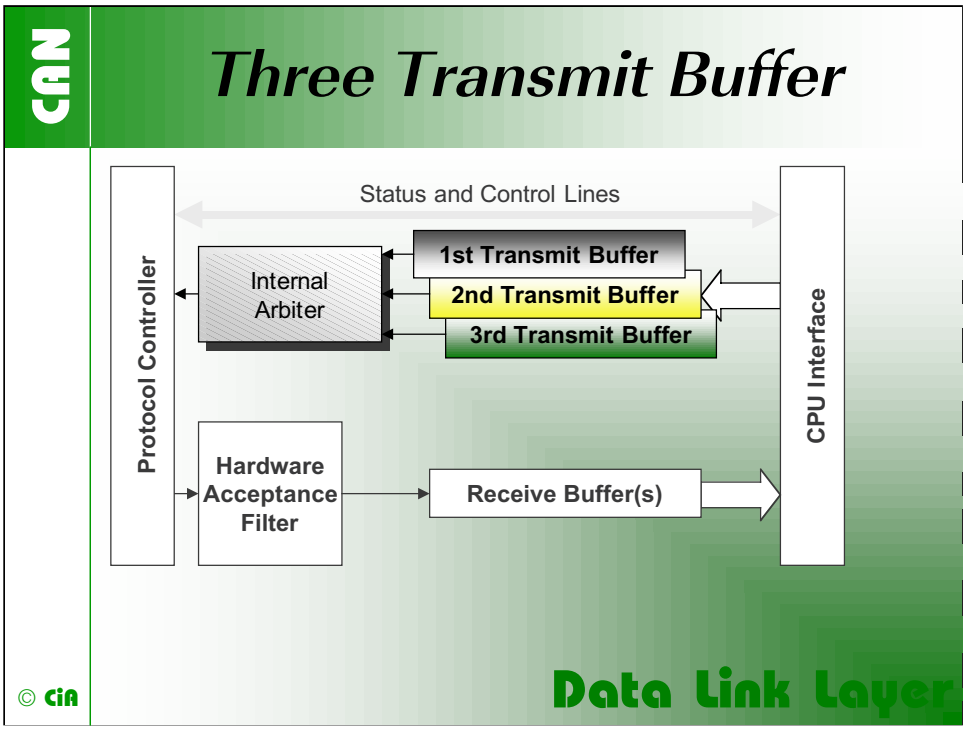
The hardware acceptance filter unburdens the microcontroller from filtering all messages. In addition, the CAN controller implements message buffers for CAN data frames to be transmitted and for those which are received. The acceptance filter and the message buffers as well as the CPU interface are implementation-specific. They make the difference between the CAN implementations.



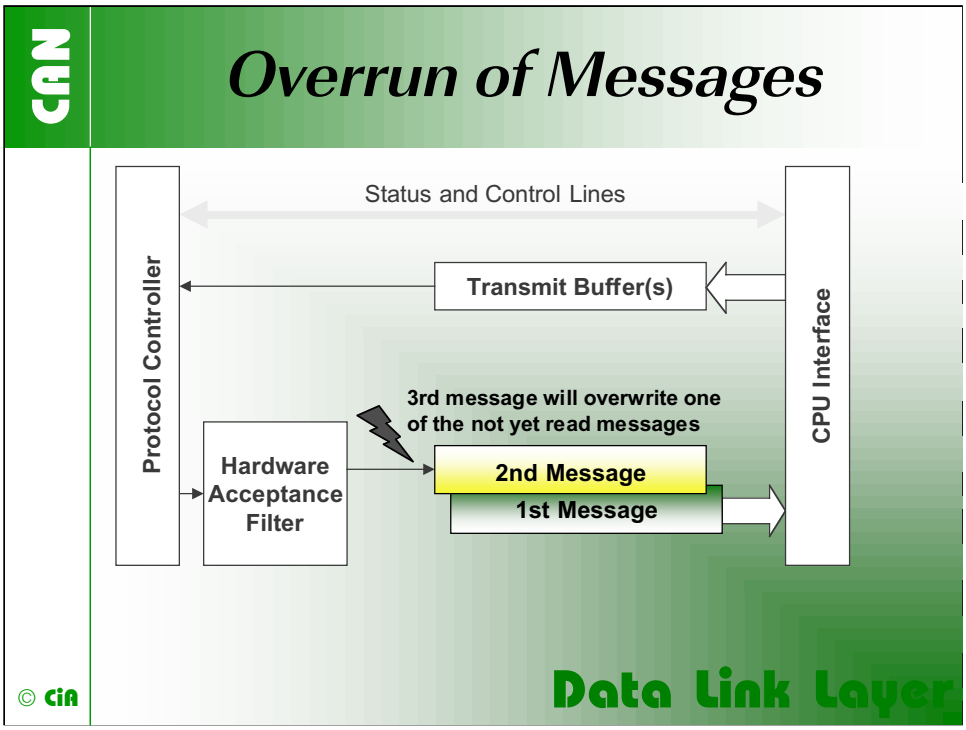
CAN controllers with intermediate buffers (formerly called BasicCAN chips) have the logic necessary to create and verify the bitstream according to the CAN protocol implemented as hardware. The simplest CAN controllers with intermediate buffer have only two reception and one transmission buffers.



If only a single transmit buffer is used inner priority inversion may occur. Because of low priority a message stored in the buffer waits until the ”traffic on the bus calms down”. During the waiting time this message could prevent a message of higher priority generated by the same microcontroller from being transmitted over the bus.

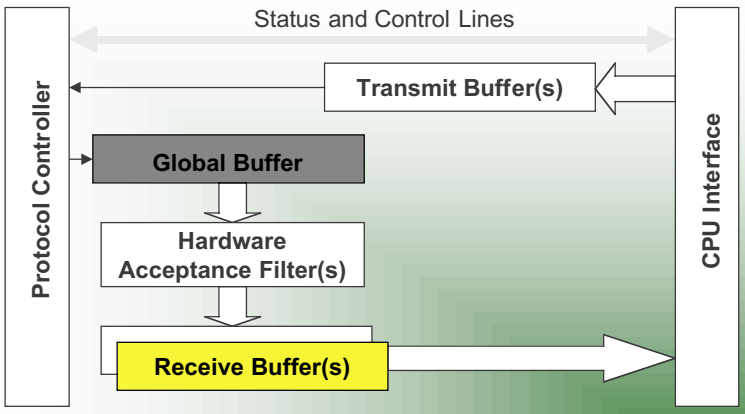


To overcome the inner priority inversion problem some of today's CAN controllers with an intermediate buffer provide more than one transmit buffer.



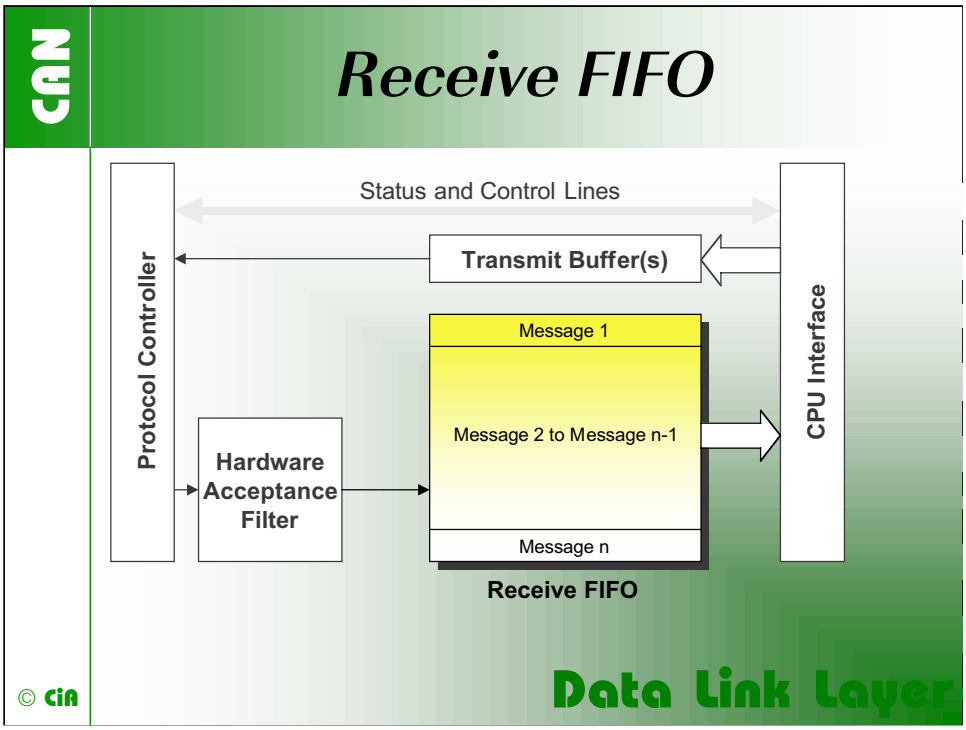
Implementing only a few reception buffers may cause the loss of data frames if the microcontroller is not fast enough to handle the interrupt requests produced by received messages.

Global Receive Buffer



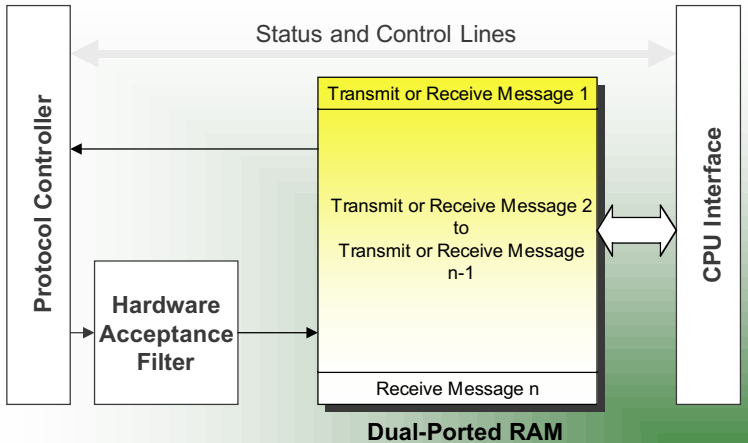
© CiA

Data Link Layer

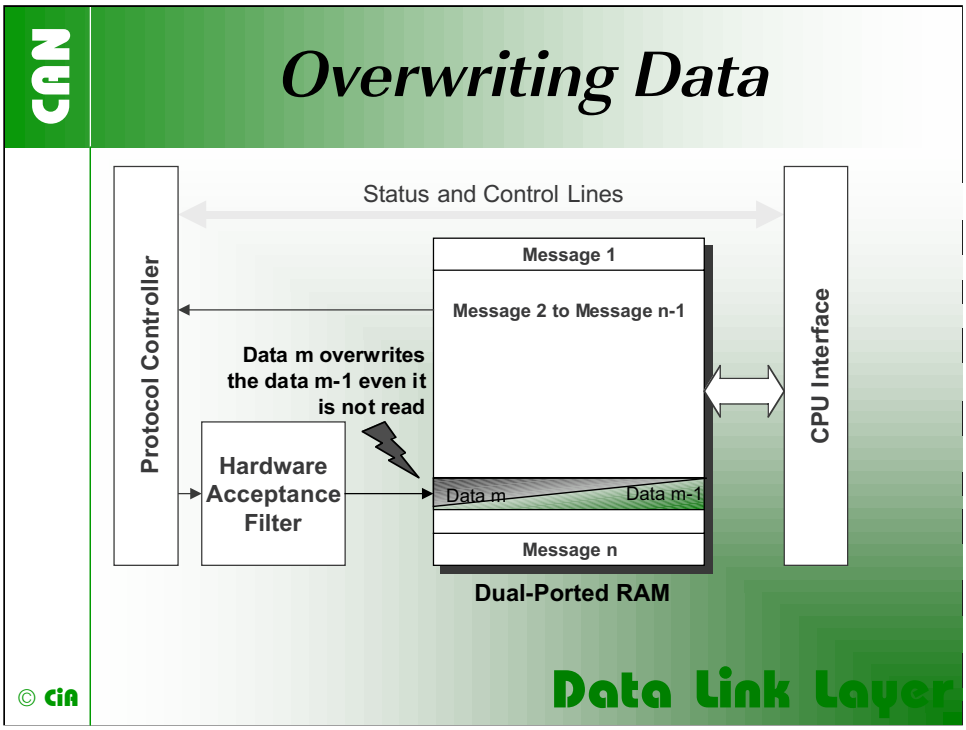


Modern CAN controllers with intermediate message buffering capabilities use deeper FIFO buffers, e.g. 64-byte. In the case of a data overrun condition, a CAN controller of this kind can optionally generate an overrun interrupt.

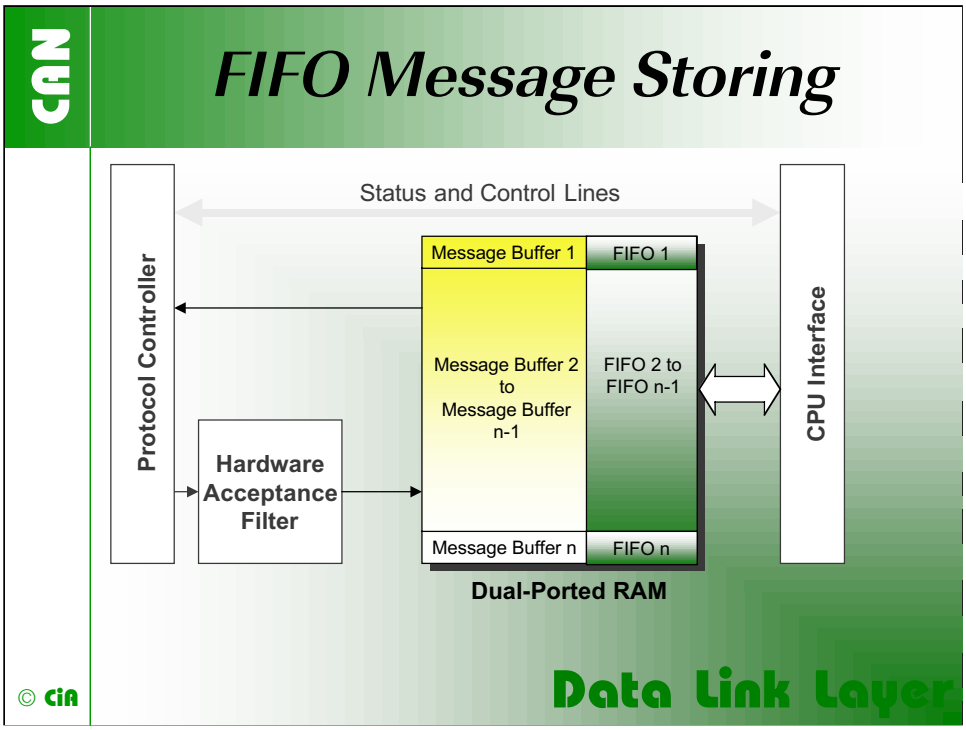
Classic Message Storing



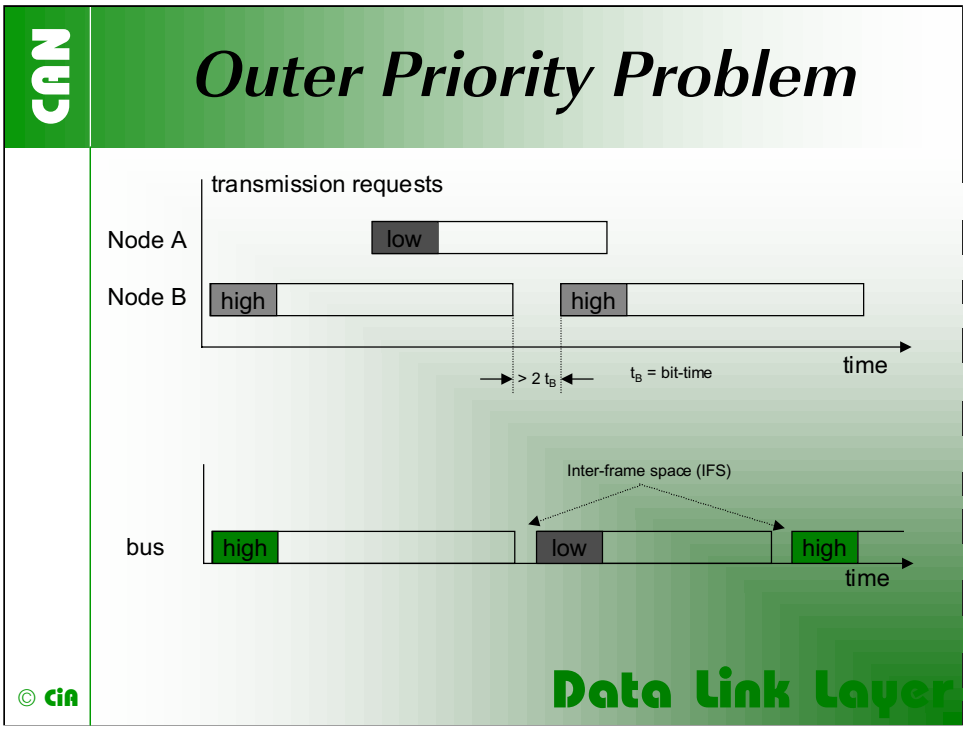
Data Link Layer



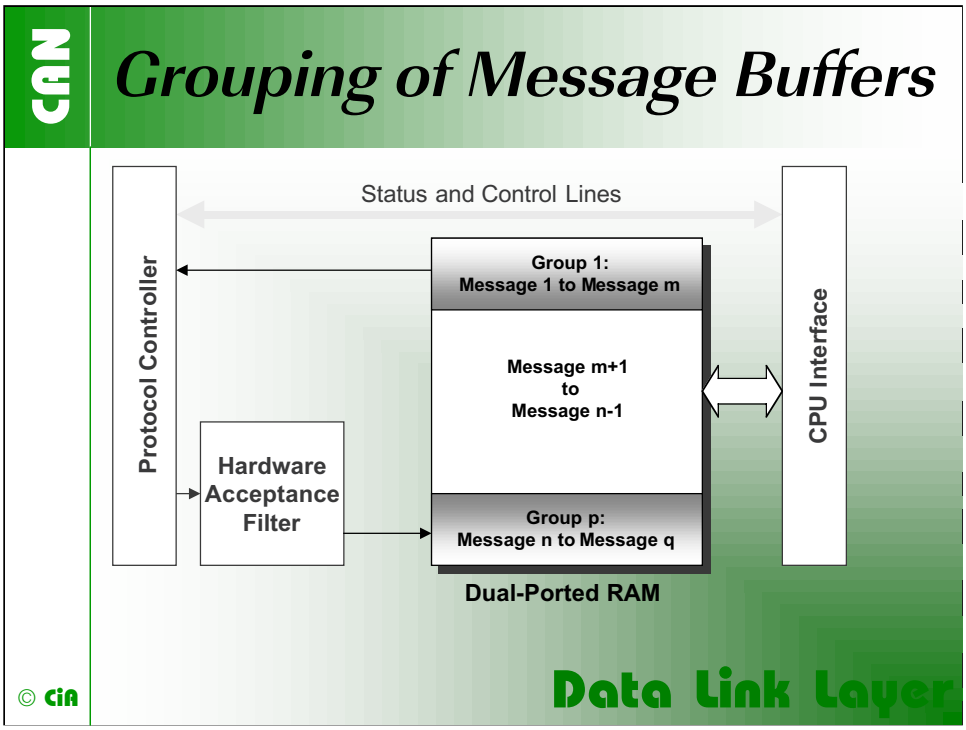
No message will be lost, but a newer one may overwrite the previous data. This dual-ported memory is practically restricted to a limited number of objects.
Implementations are available today for between 16 and 64 object memories.



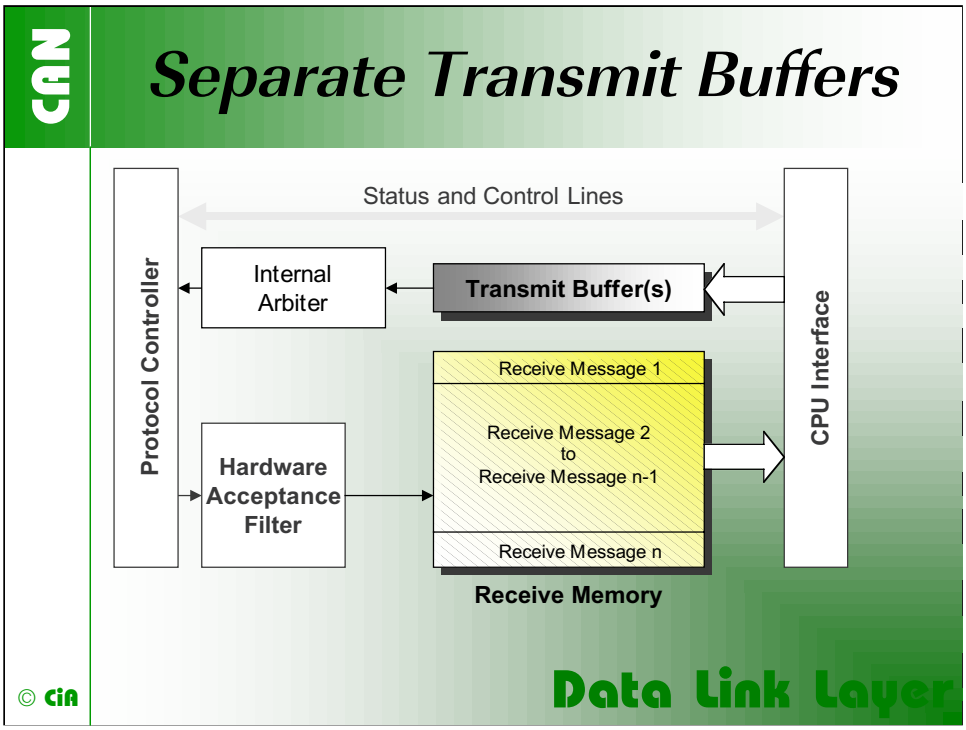
CAN controllers with object storage (formerly called FullCAN) function like CAN controllers with intermediate buffers, but also administer certain objects. The interface to the following microcontroller corresponds to a dual-ported RAM. Messages received are stored in the defined memory area, and only will be updated if a new message with the very same identifier is received.



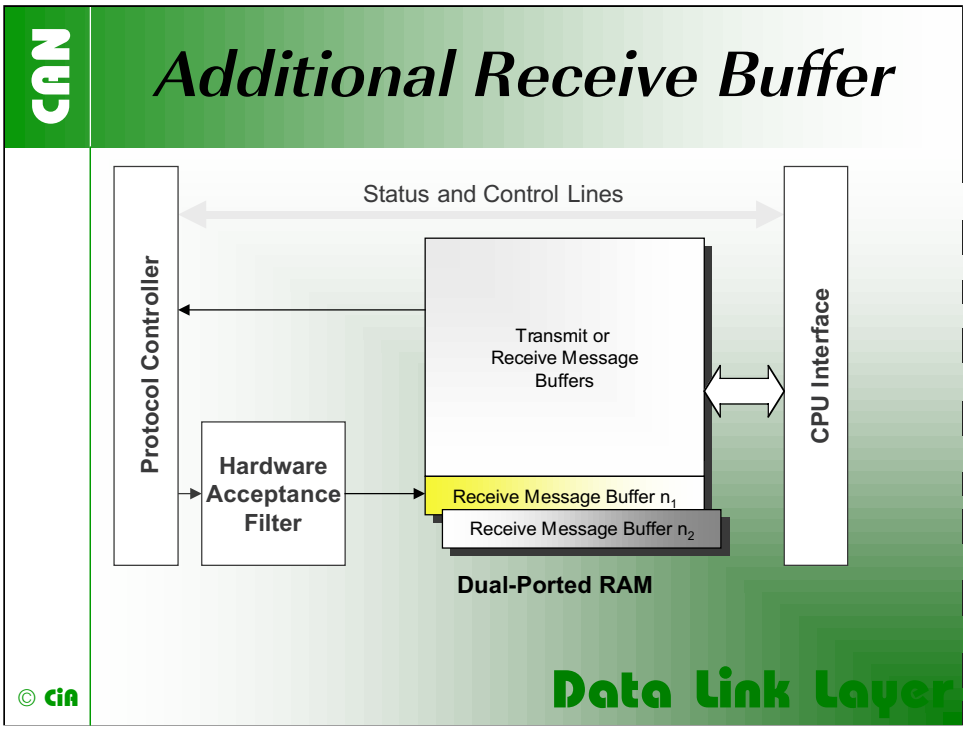
The problem of outer priority inversion may occur in some CAN implementations. Let us assume that a CAN node wishes to transmit a package of consecutive messages with high priority, which are stored in different message buffers. If the interframe space between these messages on the CAN network is longer than the minimum space defined by the CAN standard, a second node is able to start the transmission of a lower prior message. The minimum interframe space is determined by the Intermission field, which consists of 3 recessive bits. A message, pending during the transmission of another message, is started during the Bus Idle period, at the earliest in the bit following the Intermission field. The exception is that a node with a waiting transmission message will interpret a dominant bit at the third bit of Intermission as Start-of-Frame bit and starts transmission with the first identifier bit without first transmitting an SOF bit. The internal processing time of a CAN module has to be short enough to send out consecutive messages with the minimum interframe space to avoid the outer priority inversion under all the scenarios mentioned.



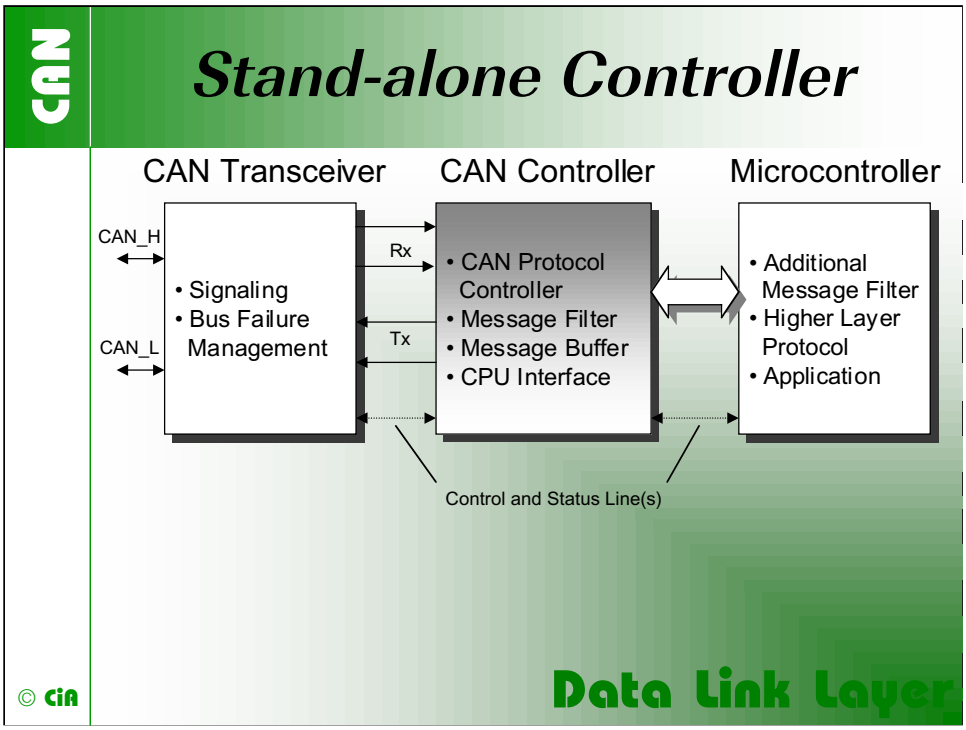
To optimize transfer of segmented data with the same or consecutive identifiers some CAN implementations support the grouping of message buffers.



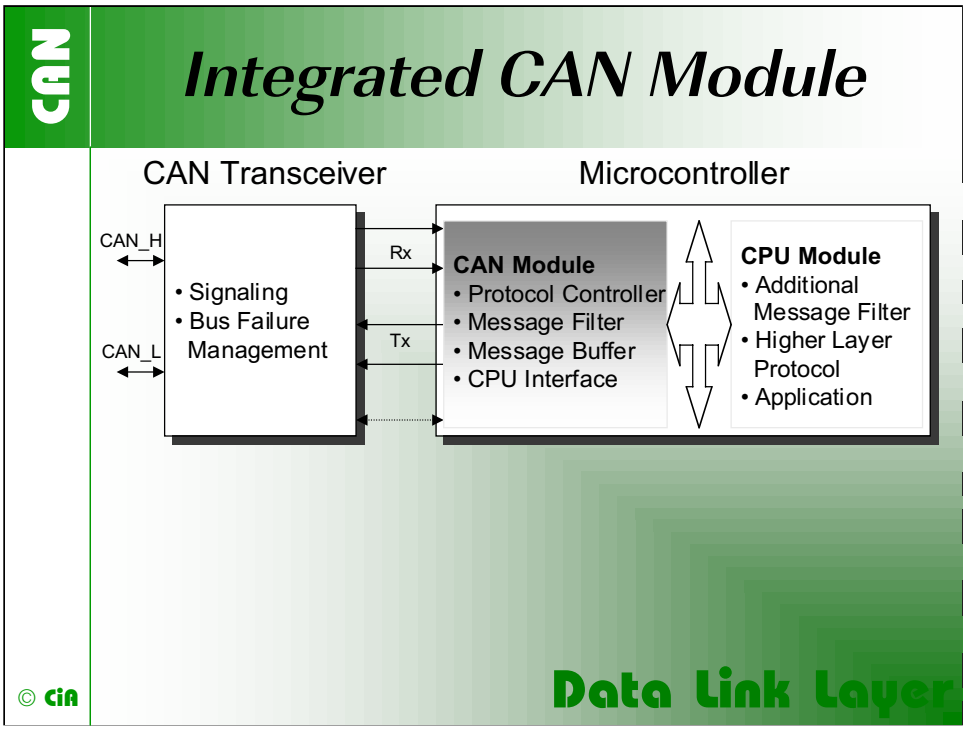
For nodes requiring to transmit the complete range of objects, some CAN implementations provide a separate transmit buffer capability without loosing the benefits of the advanced acceptance filtering.



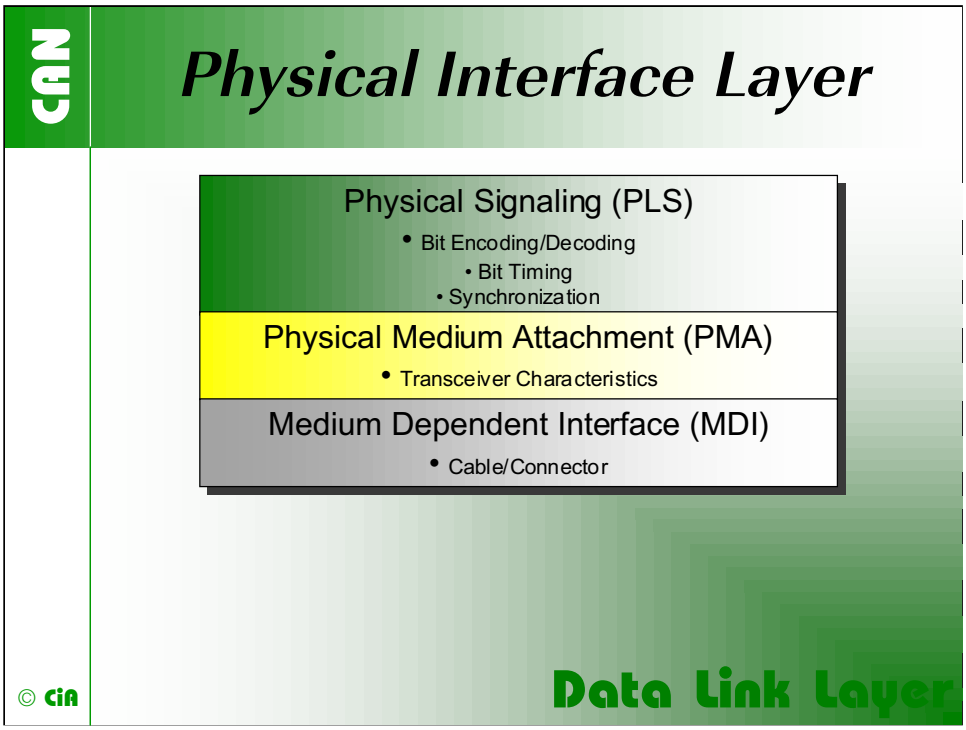
Most of the new CAN implementations will combine different message buffering features. Some of the CAN implementations with classic storing message capability provide an additional receive-only message buffer. This buffer is able to store two messages, so that the probability of message lost is decreased. In some applications, you can select if te buffer n_1 or buffer n_2 will be overwritten in case both buffers have not been read by the host controller.



There are some trade-offs between stand-alone and integrated CAN peripherals. The integrated CAN peripheral is cheaper not only because of the chip price itself, the design of the printed circuits boards is easier, and space requirements and costs are lower. Although the software development costs are about the same for both solutions, software reusability may differ. Stand-alone CAN chips are designed to interface different CPUs allowing the software developed for one system to be reused in another system, even if the CPU is different. Software developed for an integrated CAN peripheral may not function on another CPU with on-chip CAN.

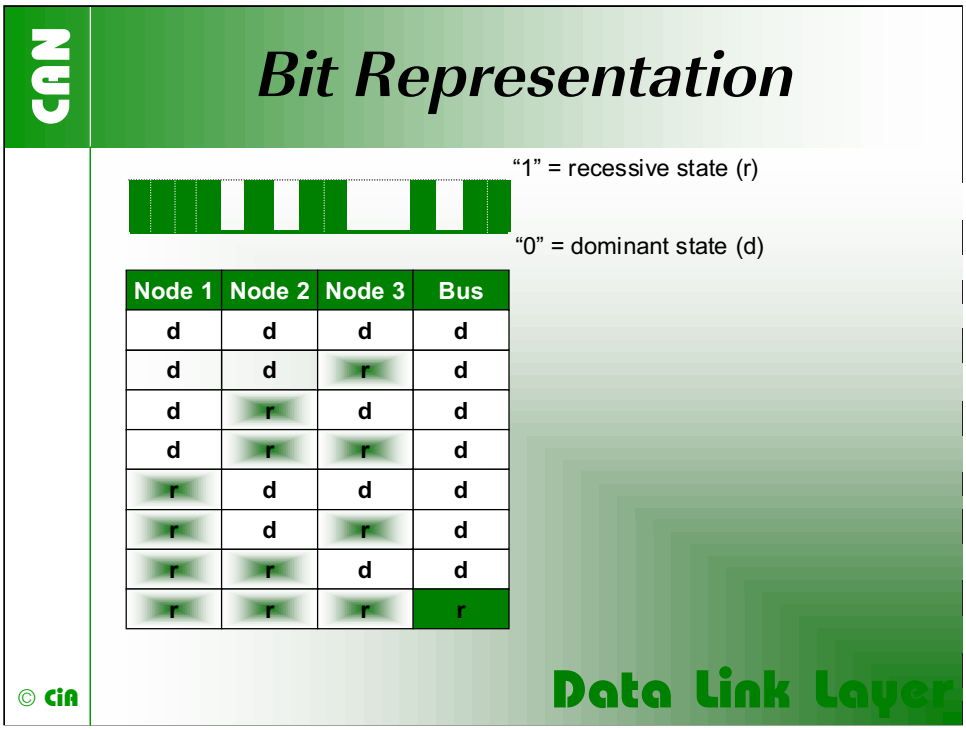


Integrated CAN peripherals cause a lower CPU load than do stand-alone controllers. The most critical factor is the amount of time required to read/write to the CAN peripheral. In the case of an on-chip CAN controller, the CAN registers are addressed using the internal address/data bus designed for high-speed access. In the case of a stand-alone CAN chip the CPU uses its external address/data bus or a serial communications link, which of course is slower. The CPU load on an on-chip CAN is approximately one-half of a stand-alone CAN chip. A CAN node using an integrated CAN peripheral has a reliability advantage over a system using a stand-alone CAN chip because of its smaller form factor. CAN chips are shipped in large numbers and the combined price of a CPU and a stand-alone CAN chip is still competitive. However, as CAN gains more market acceptance, CPUs with on-chip CAN will be the solution of choice.

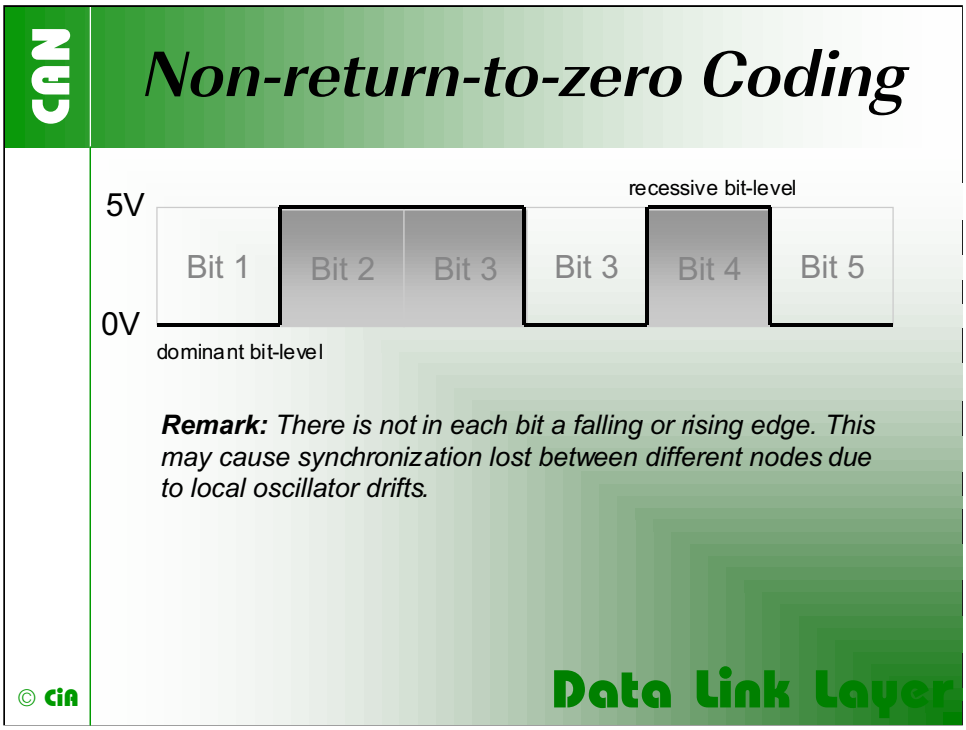


The CAN physical layer can be divided in three sub-layers. The PLS layer is implement in the CAN controller chips. The PMA layer describes the transceiver characteristics. The MDI layer specifies the cable and connector characteristics.

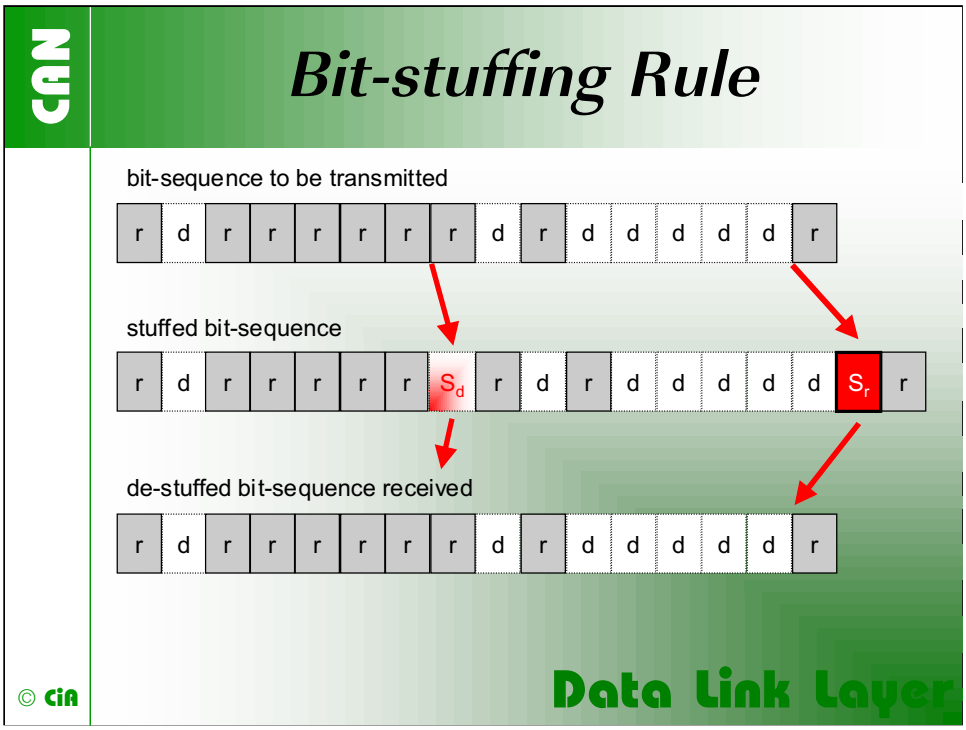
The PMA and MDI layers are subject of different international, national and industry standards as well as proprietary specifications. Most common is the ISO 11898 standard specifying a high-speed transceiver for CAN-based networks.



The MAC (Medium Access Control) layer of the CAN protocol defines the non-destructive bit-wise arbitration, so that the message with highest prior identifier will get the bus. Therefore, any CAN physical layer has to support the representation of a recessive and a dominant state on the transmission medium. The transmission shall be in the recessive state if no bus node transmit a dominant bit. If one or multiple bus nodes transmit a dominant bit, then the transmission medium shall enter the dominant state, thus overwriting the recessive state.

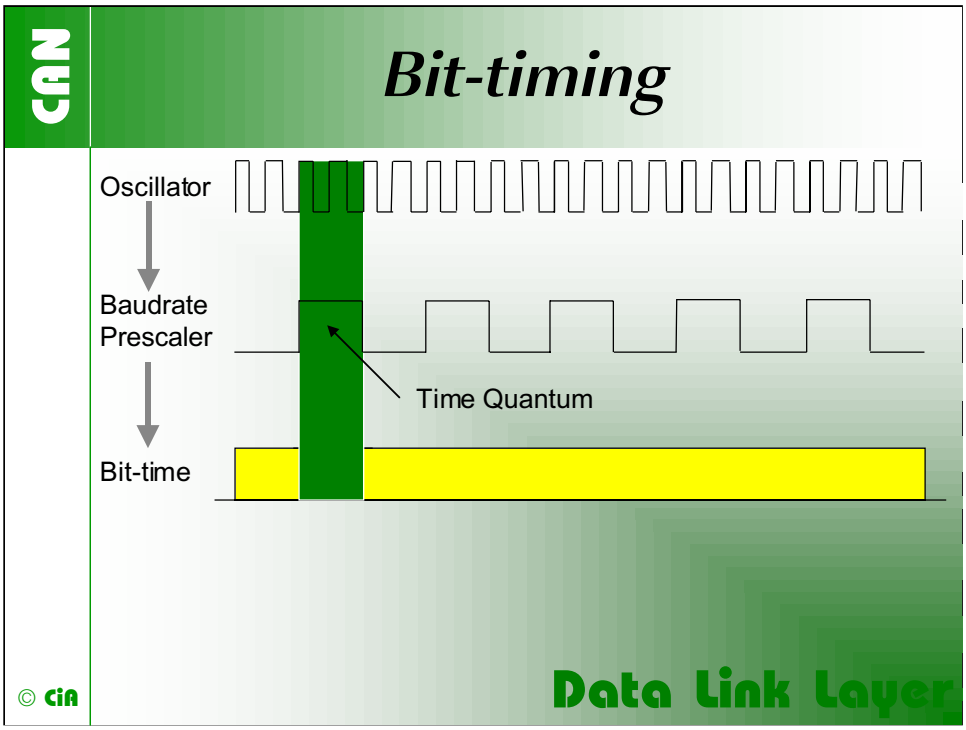


The bit stream in a CAN message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during the total bit time the generated bit level is either ‘dominant’ or ‘recessive’. The alternative method, the Manchester coding, requires in each bit a falling or rising edge, which leads to higher frequency.

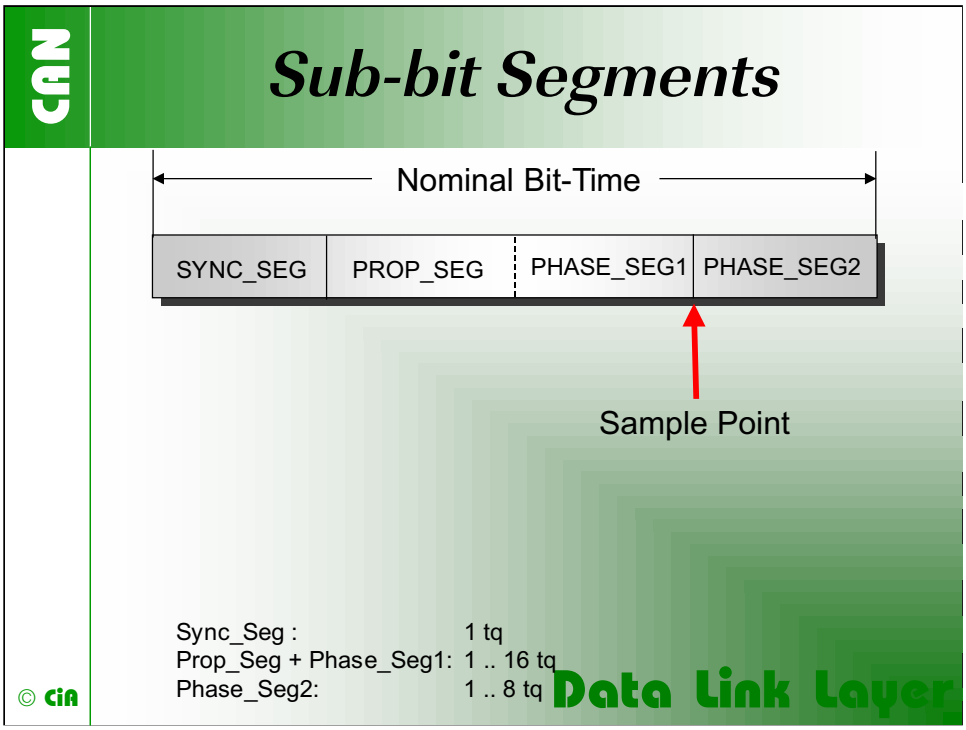


One characteristic of Non-Return-to-Zero code is that the signal provides no edges that can be used for resynchronization if transmitting a large number of consecutive bits with the same polarity. Therefore bit-stuffing is used to ensure synchronization of all bus nodes. This means that during the transmission of a message, a maximum of five consecutive bits may have the same polarity.

The bit-stuff area in a CAN frame includes the SOF, Arbitration field, Control field, Data field and CRC field.



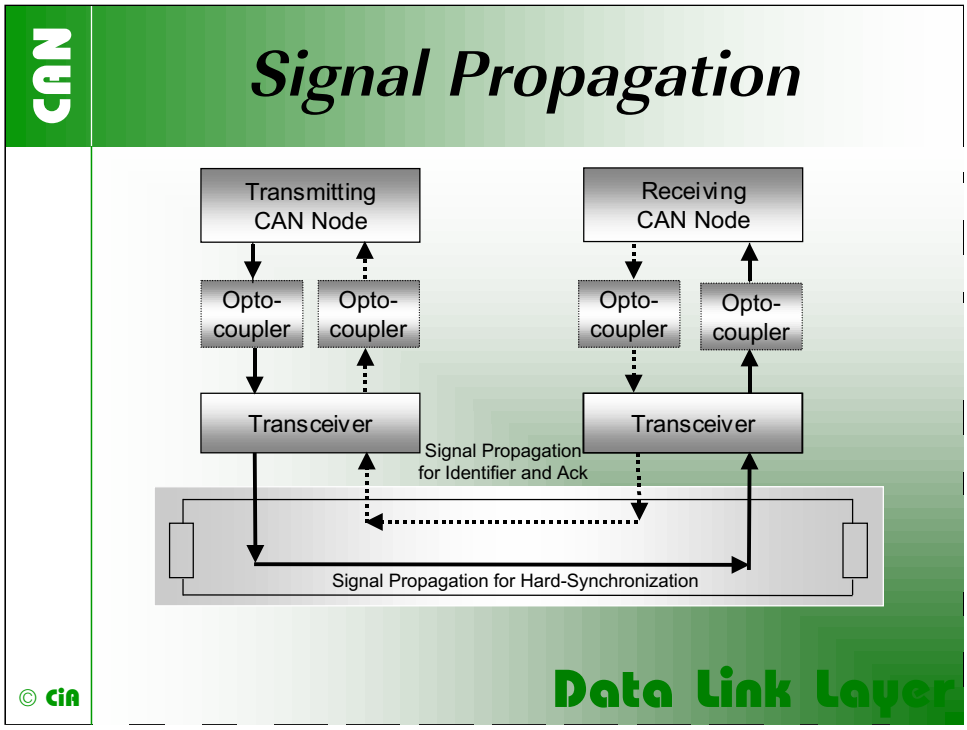
One bit time is specified as four non-overlapping time segments (see next slide). Each segment is constructed from an integer multiple of the Time Quantum (tq). The Time Quantum is the smallest discrete timing resolution used by a CAN node. Its length is generated by a programmable divide of the CAN node's oscillator frequency. There is a minimum of 8 and a maximum of 25 Time Quanta per bit. The bit time is selected by programming the width of the Time Quantum and the number of Time Quanta in the various segments. This has to be done in the CAN controllers.



Basically the CAN bit period can be subdivided into four time segments. Each time segment consists of a number of Time Quanta.

• SYNC_SEG is 1 Time Quantum long. It is used to synchronize the various bus nodes.
• PROP_SEG is programmable to be 1, 2,... 8 Time Quanta long. It is used to compensate for signal delays across the network.
• PHASE_SEG1 is programmable to be 1,2, ... 8 Time Quanta long. It is used to compensate for edge phase errors and may be lengthened during resynchronization.
• PHASE_SEG2 is the maximum of PHASE_SEG1 and the Information Processing Time long. It is also used to compensate edge phase errors and may be shortened during resynchronization.
• Information Processing Time is less than or equal to 2 Time Quanta long.
• The total number of Time Quanta has to be from 8 to 25.

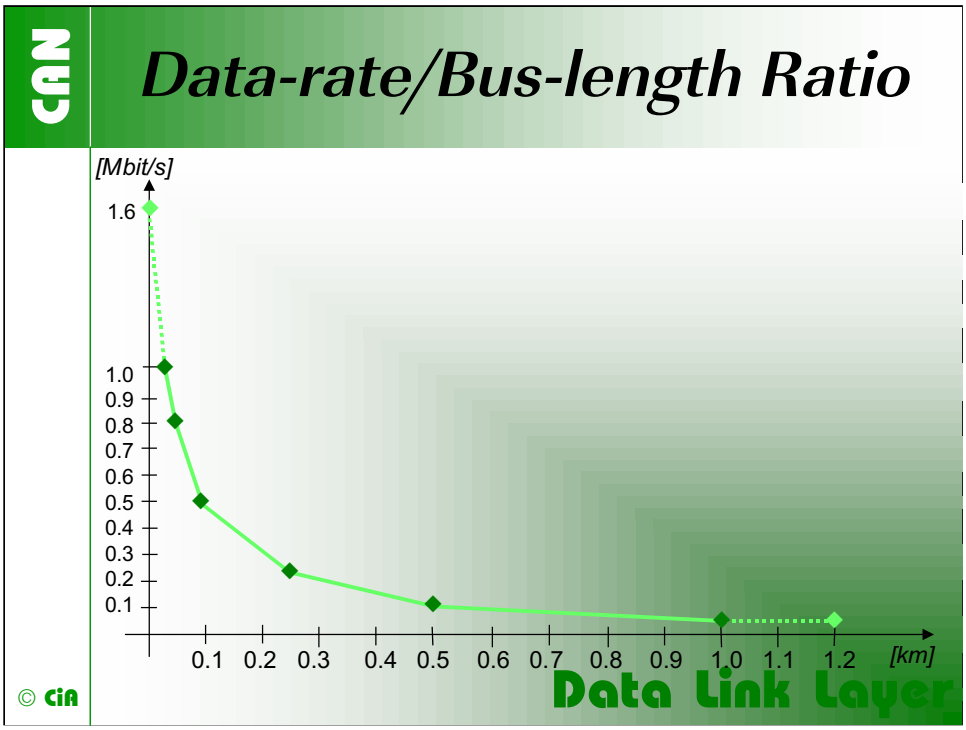
Programming of the Sample Point allows optimizing the Bit Timing: A late sampling for example allows a maximum bus length; an early sampling allows slower rising and falling edges.



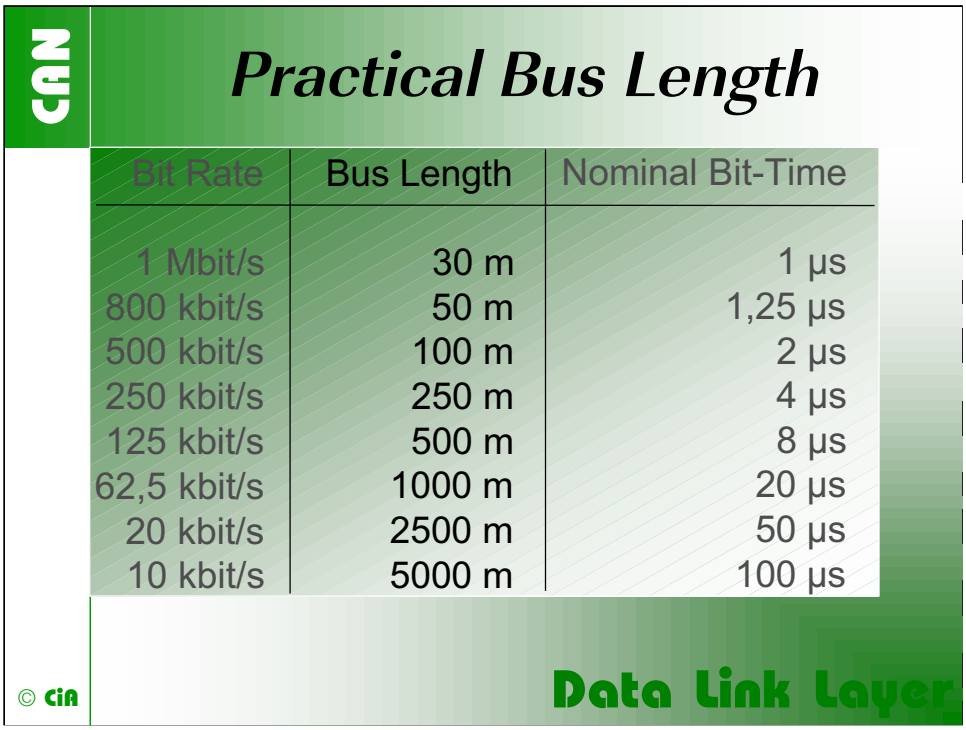
It is necessary to compensate for signal propagation delays on the bus line and through the electronic interface circuits of the bus nodes. The sum of the propagation delay times of controller, optional galvanic isolation, transceiver and bus line has to be less than the length of the Propagation Time Segment (Prop_Seg) within one Bit.

You have to add up the following delays depending on the selected components: CAN controller (50 ns to 62 ns), optocoupler (40 ns to 140 ns), transceiver (120 ns to 250 ns), and cable (about 5 ns/m). These delays have to be considered twice, because after hard synchronization the most far away node is expect switching edges with delay of the propagation time, and the bit of the transmitter has to wait another propagation time to guarantee that the identifier bit or the Acknowledge slot bit of the Receiver is valid. Using ISO 11898 compliant transceiver and high-speed optocoupler you can reach a maximum bus length of 9 meters at 1 Mbit/s.

$$t_{propagation} = 2\,(t_{cable} + t_{controller} + t_{optocoupler} + t_{transceiver})$$



At bit rates lower than 1 Mbit/s the bus length may be lengthened significantly. A data rate of 50 kbit/s allows a bus length of 1 km. ISO 11898 compliant transceivers specify max. bus length of about 1 km. But it is allowed to use bridge-devices or repeaters to increase the allowed distance between ISO 11898 compliant nodes to more than 1 km.



The maximum achievable bus line length in a CAN network is determined essentially by the following physical effects:

• the loop delays of the connected bus nodes and the delay of the bus lines

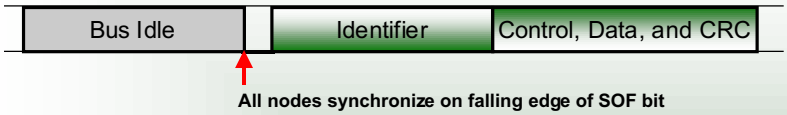
• the differences in bit time quantum length due to the relative oscillator tolerance between nodes

• the signal amplitude drop due to the series resistance of the bus cable and the input resistance of bus nodes

The shown practical bus length can be reached with ISO 11898 compliant transceivers and standard bus line cables. Note, there are no optocouplers considered.

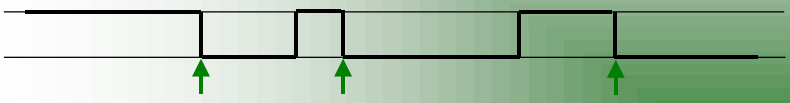
Bit Synchronization

- Hard synchronization at SOF, last bit of IFS, and during suspend transmission



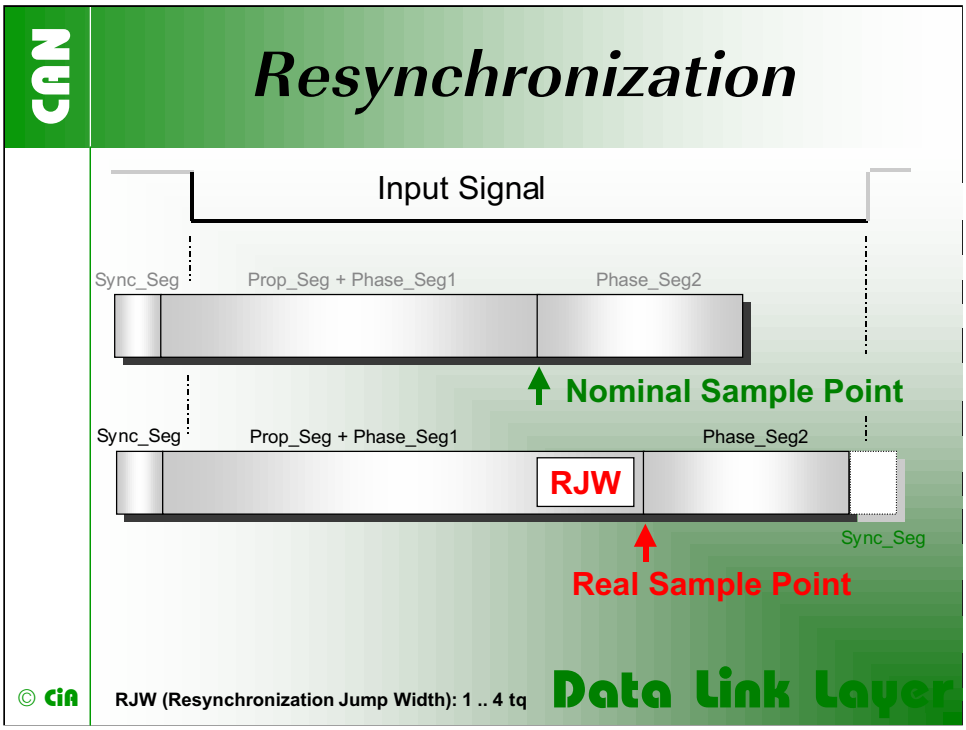
All nodes synchronize on falling edge of SOF bit

- Re-synchronization on each recessive-to-dominant edge



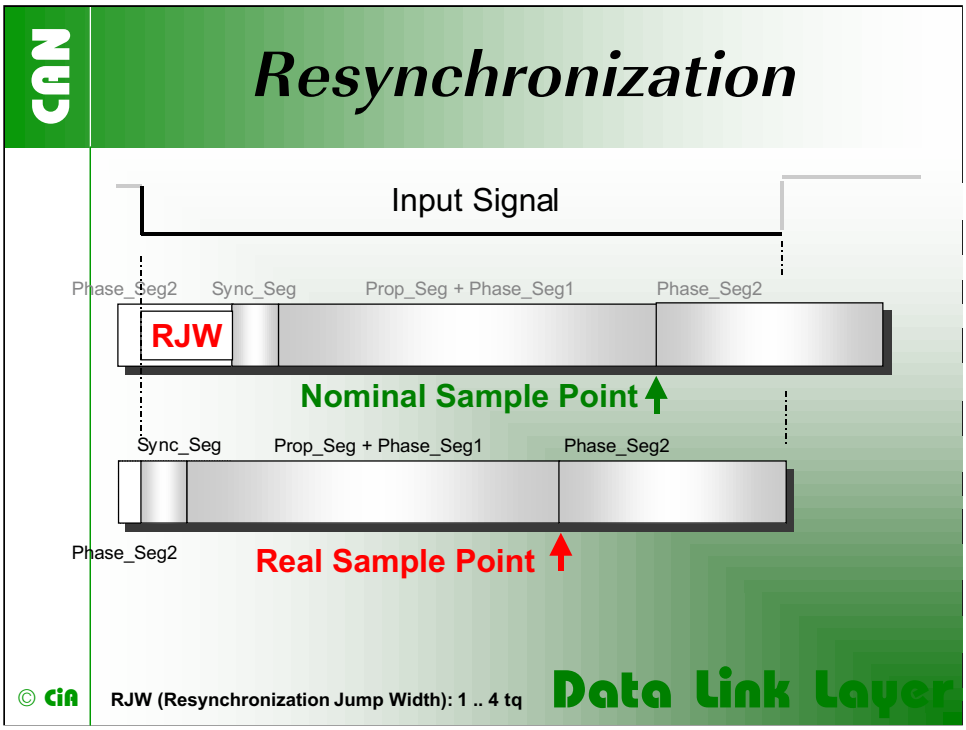
All nodes perform automatically re-synchronization

Data Link Layer



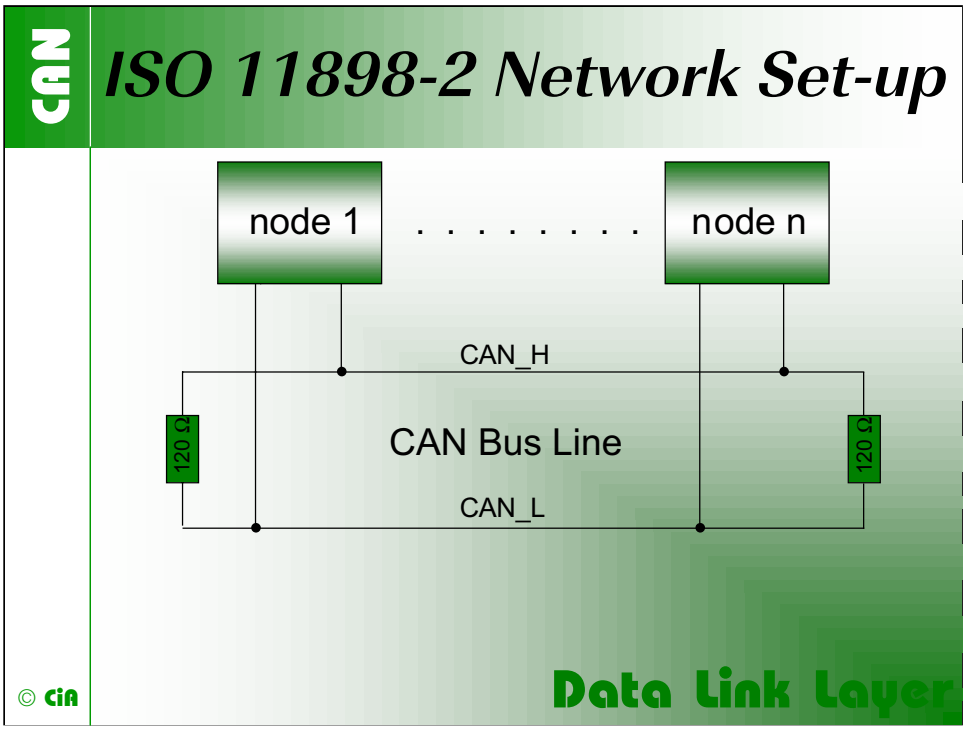
A CAN network consists of several nodes, each clocked with its individual oscillator. Because of this, phase shifts can occur in different nodes. Each CAN controller provides a resynchronization (soft synchronization) mechanism to compensate phase shifts while receiving a CAN frame.

An edge is expected in the Sync_Seg. In the case of a slower transmitter meaning the edge is detected in the Prop_Seg, the receiver lengthens the Phase_Seg1 with a maximum of the programmed value of the Resynchronization Jump Width (RJW = 1 .. 4 tq).

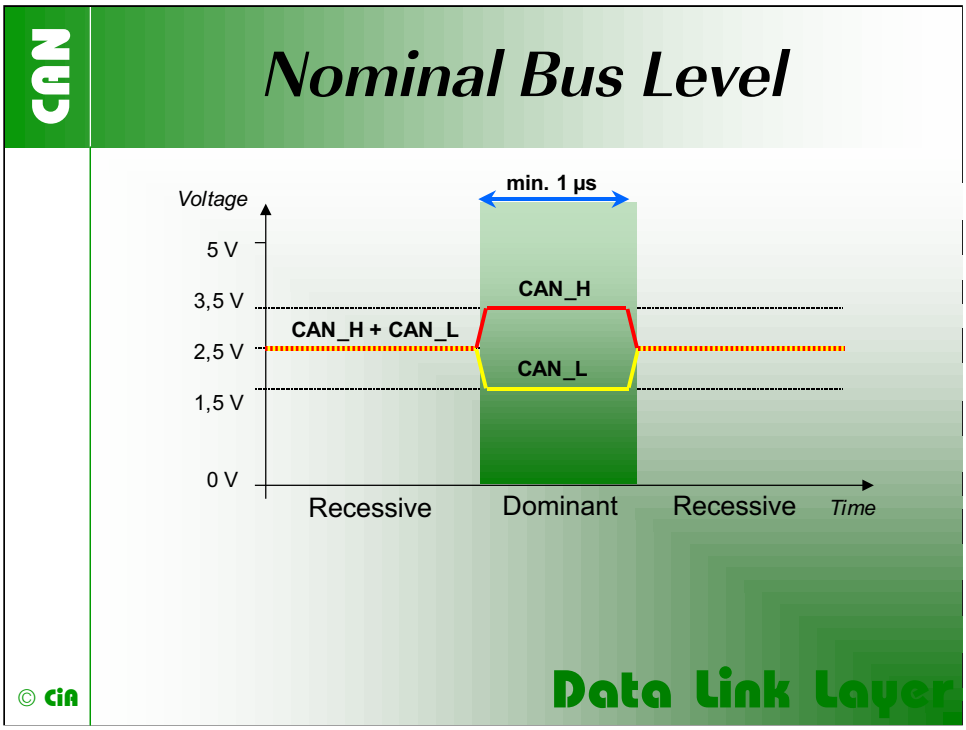


In the case of a faster transmitter meaning the edge is detected in the previous Phase_Seg2, the receiver shortens the Phase_Seg2 with a maximum of the programmed value of the Resynchronization Jump Width (RJW = 1 .. 4 tq).

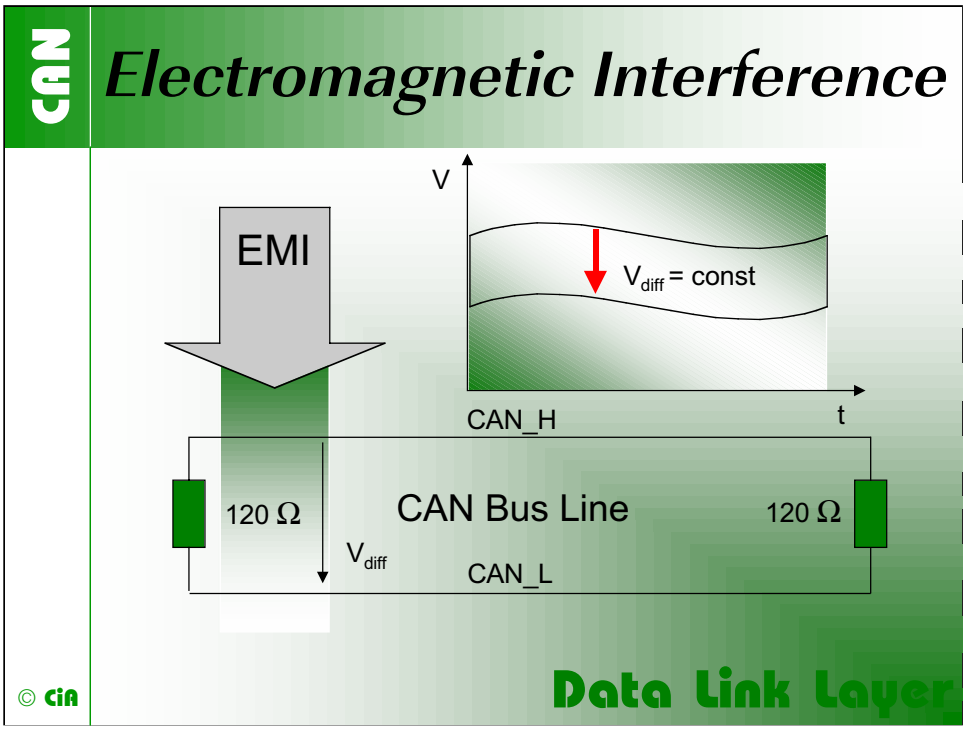
There is only one resynchronization allowed within one bit time.



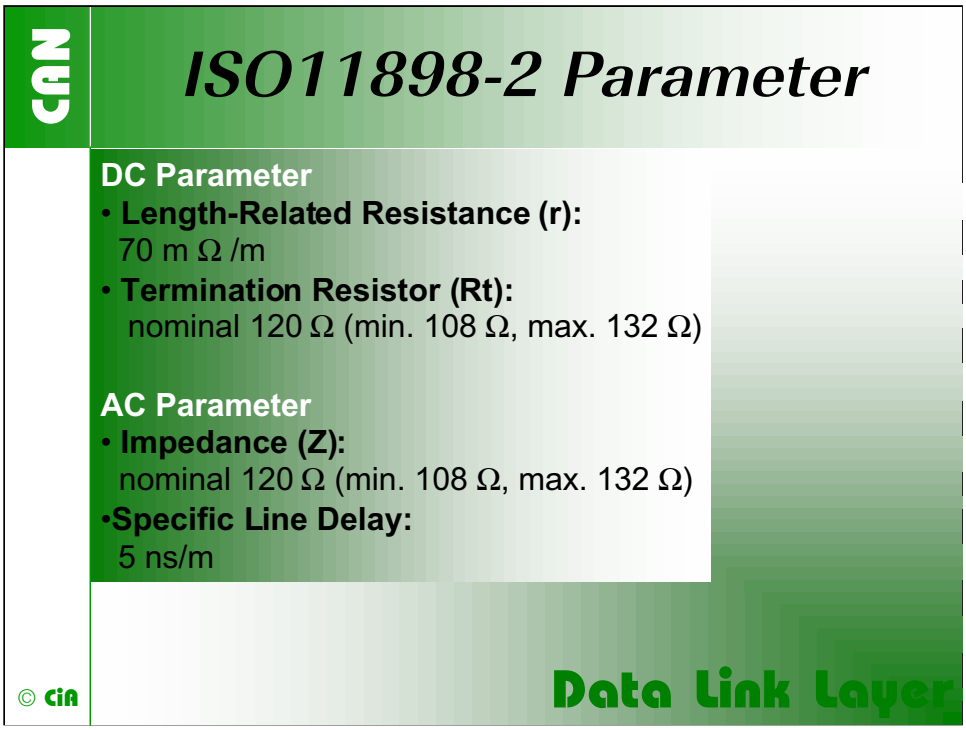
The ISO 11898-2 standard assumes the network wiring technology to be close to a single line structure in order to minimize reflection effects on the bus line. The bus lines have to be terminated by resistors at both ends.



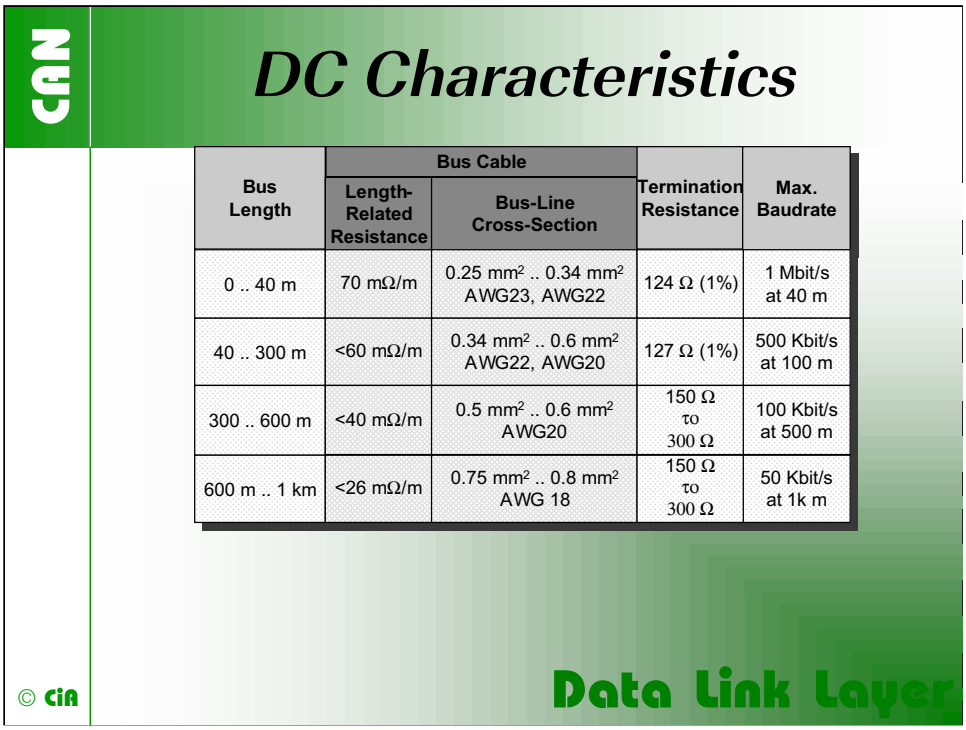
The bus nodes shall detect a recessive bus condition if the voltage of CAN_H is not higher than the voltage of CAN_L plus 0.5 V. If the voltage of CAN_H is at least 0.9 V higher than CAN_L, then a dominant bus condition shall be detected. The nominal voltage in the dominant state is 3.5 V for the CAN_H line and 1.5 V for the CAN_L line.



Due to the differential nature of the transmission signal CAN is insensitive to electromagnetic interference, because both bus lines are affected in the same way which leaves the differential signal unaffected (Vdiff = constant).

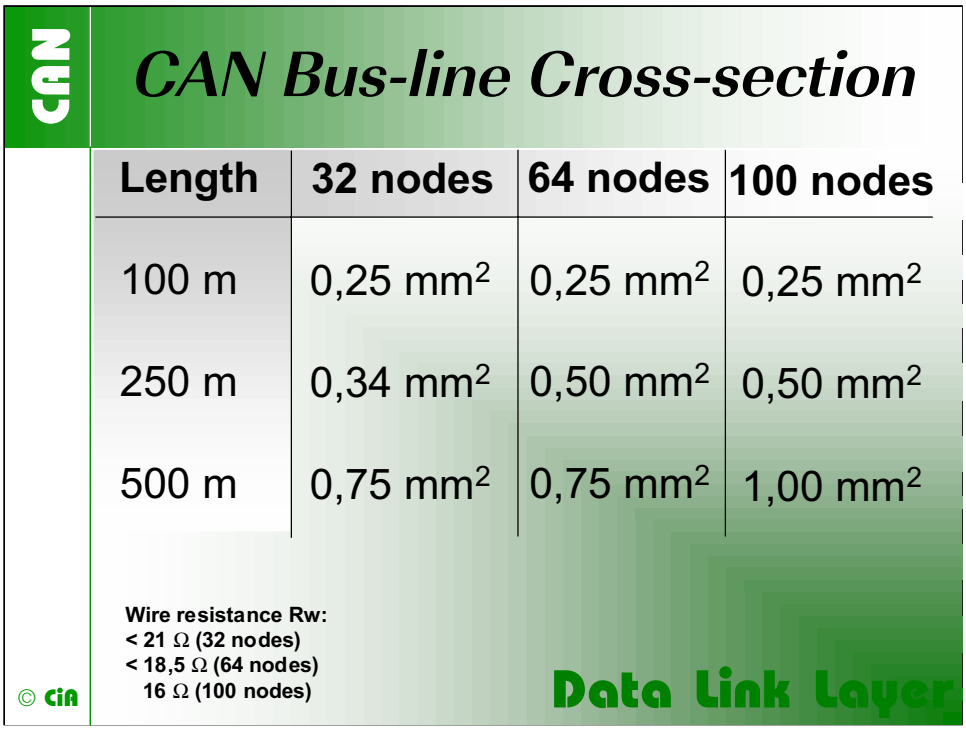


According to the ISO 11898-2 standard, cables to be chosen for CAN bus lines should have a nominal impedance of 120 Ohm, and a specific line delay of nominal 5 ns/m. Line termination has to be provided through termination resistors of 120 Ohm located at both ends of the line. The length related resistance should have 70 mOhm/m. All these mentioned AC and DC parameters are suitable for a 1 Mbit/s transmission rate.



These recommended DC parameters for bus line cables are suitable for ISO 11898-2 based networks. To minimize the voltage drop on long distances the termination resistor should be higher than in the ISO 11898-2 standard.

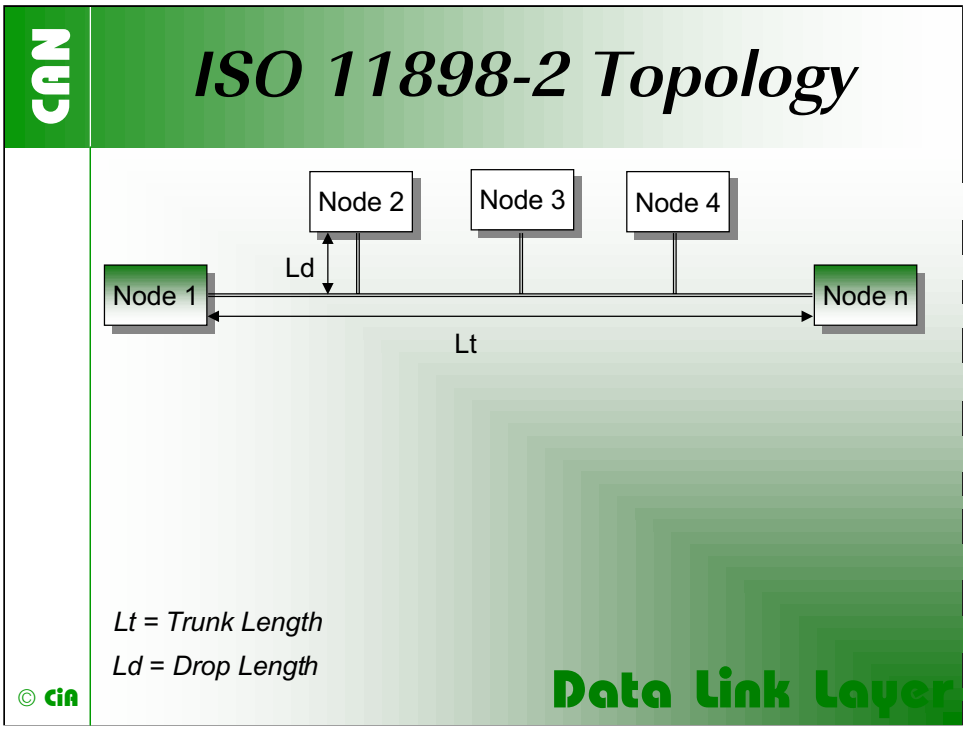
The system integrator has to consider the DC parameters for the connectors as well. To calculate the voltage drop, he has to add for each node with 9-pin D-Sub connector about 5 m Ω to 20 m Ω to the total transmission resistance.



The table provides a first indication on which kind of wire cross section should be considered for the signal pair of the bus trunk cable (Philips Application Note AN96116 for the PCA82C250/1 CAN Transceiver). The table bases on the following assumptions:

- 32 nodes: Rw < 21 Ω
- 64 nodes: Rw < 18.5 Ω,
- 100 nodes: Rw < 16 Ω.

Ground potential shifts should not lead to a fall of voltage of more than 2 V.



The wiring-topology of a CAN-network should be as close as possible to a single line structure in order to avoid cable-reflected waves. Essentially it depends on the bit timing parameters, the trunk cable length Lt and the drop cable length Ld whether reflections will be tolerated. In practice short stubs Ld are necessary to connect devices to the bus line successfully. They should be as short as possible, especially at high bit rates. At 1 Mbit/s the length of the cable stubs should not exceed 0.3 m.

Cable Drop Length

Rules of thumb for the maximum length of a unterminated cable drop Ld and for for the cumulative drop length Ldi:

$$Ld < t_{PROPSEG} / (50 * t_P)$$

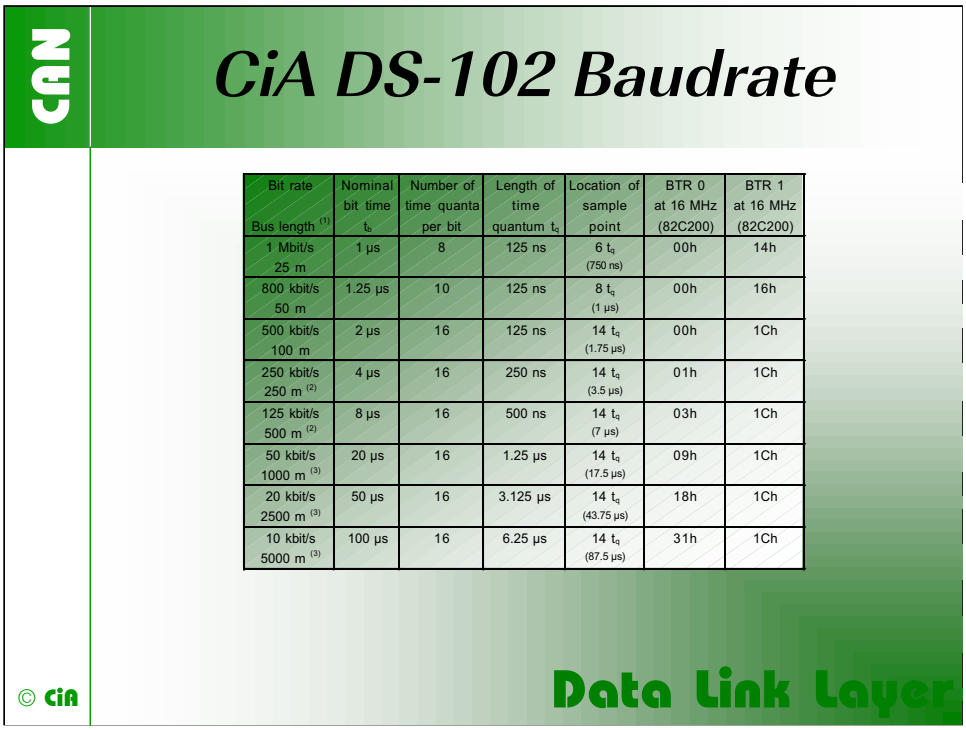
$$\sum_{i=1}^{n} Ldi < t_{PROPSEG} / (10 * t_P)$$

$t_{PROPSEG}$: length of the propagation segment of the bit period
t_P : specific line delay per length unit

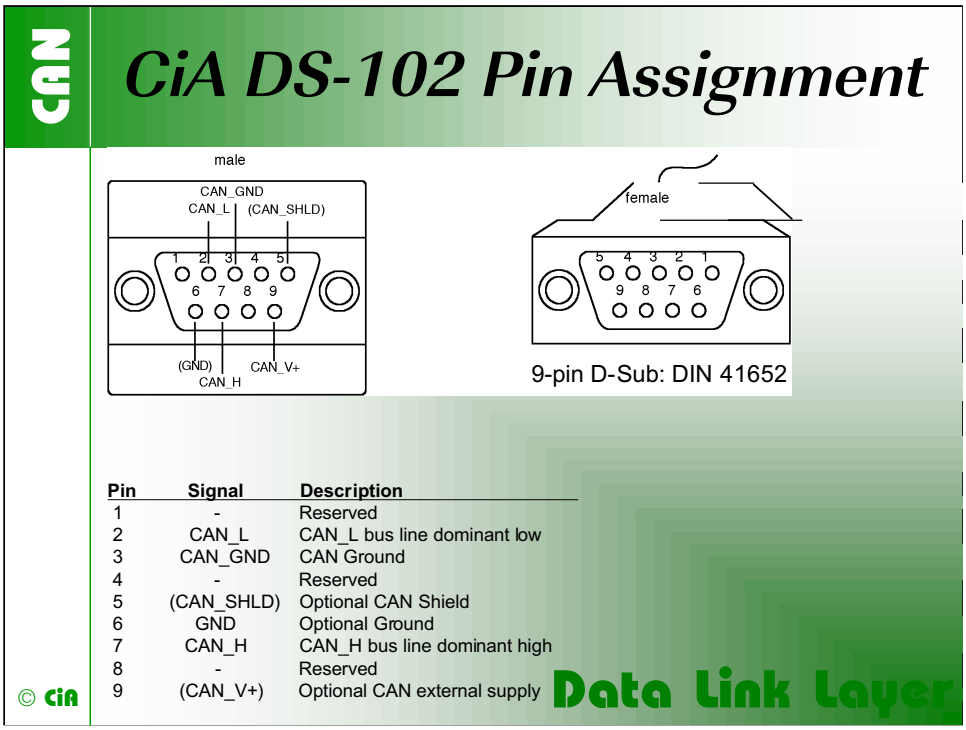
Example: bit rate = 500 kbit/s: $t_{PROPSEG} = 12 * 125ns = 1500$ ns; $t_P = 5$ ns/m

$Ld < 1500$ ns / (50 * 5 ns/m) = 6 m; $\sum_{i=1}^{n} Ldi < 1500$ ns /(10 * 5 ns/m) = 30 m

Data Link Layer



The CAN in Automation (CiA) international users and manufacturers group has recommended some baud rates to be used in general purpose CAN networks as well as the maximum bus length for a given baud rate. In addition, the bit-timing is recommended, so that nodes from different manufacturers can be connected to one CAN network without calculating the bit-timing parameters.



The CiA DS-102 standard includes a pin assignment for 9-pole Sub-D connectors for the connection of nodes to the CAN bus lines. This pin assignment is also used by some higher-layer protocol specifications (e.g. CANopen, Smart Distributed System).