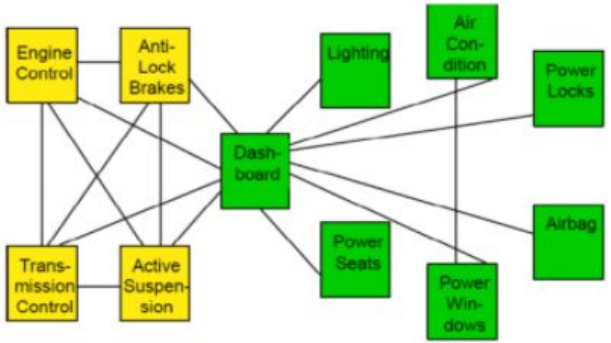
Automotive Industry

CAN Communication

Need for CAN

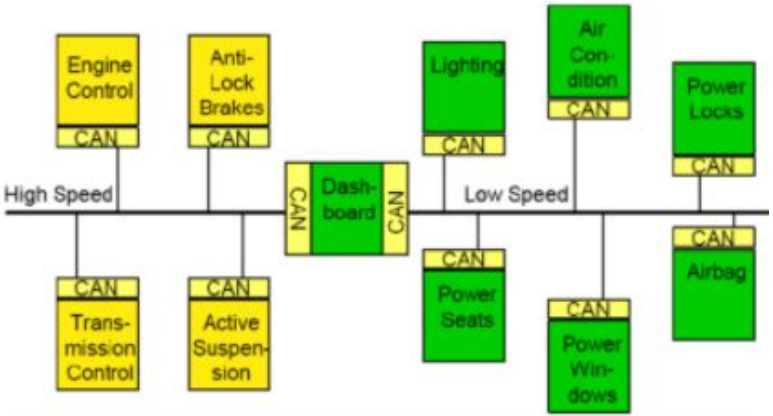
At first, independently operating electronic ECUs were sufficient to implement electronic functions. However, it was recognized early on that the coordination of electronic control units (ECUs) could enhance vehicle functionality immensely. Data exchange between the electronic control units was initially implemented conventionally, i.e. a physical communication channel was allocated to every signal to be transmitted.

However, intensive wiring effort enabled just limited data exchange. The only solution that seemed to offer a way out of this dilemma came from serial bit exchange of data via a single communication channel (bus). This led to the need to conceptualize a serial communication system tailored to the requirements of the automobile.



CAN History

At the beginning of the 1980s, Bosch began to develop such a serial communication system. It was given the name CAN (Controller Area Network). Even today, CAN is still performing useful services in motor vehicles in networking ECUs in the powertrain, chassis and convenience areas. Above all, CAN is characterized by very reliable data transmission that satisfies the real-time requirements of target usage areas.



In 1991, Bosch published their CAN 2.0 protocol which became a standard protocol and later adopted by the international Organization for Standardization (ISO) under (ISO 11898) in 1993. The CAN protocol is implemented in hardware. Many different CAN controllers have become available, which only differ in the way they handle the CAN messages.

ISO 11898

CAN technology has been standardized and described by four ISO documents. They describe two different CAN physical layers: the high-speed CAN physical layer and the low-speed CAN physical layer. They differ primarily in their definition of voltages and data transmission rates (data rates). The four ISO documents are:

ISO 11898-1
Covers the main CAN protocol, it's frames, Arbitration, error and all CAN rules in general.

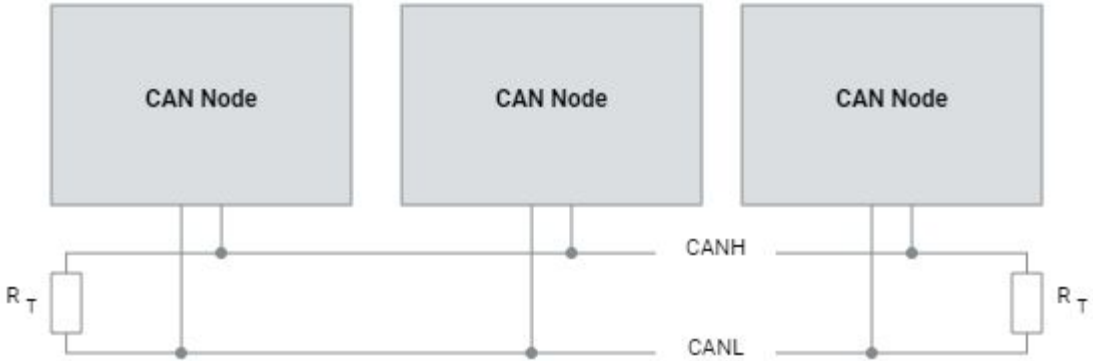
ISO 11898-2
Covers the high-speed CAN (1 Mb/s on CAN 2.0) and (5 Mb/s on CAN-FD)

ISO 11898-3
Covers the low-speed CAN (125 Kb/s)

ISO 11898-4
It is an extension of the main CAN that adds a time triggered communication option for CAN-based networks.

CAN Network

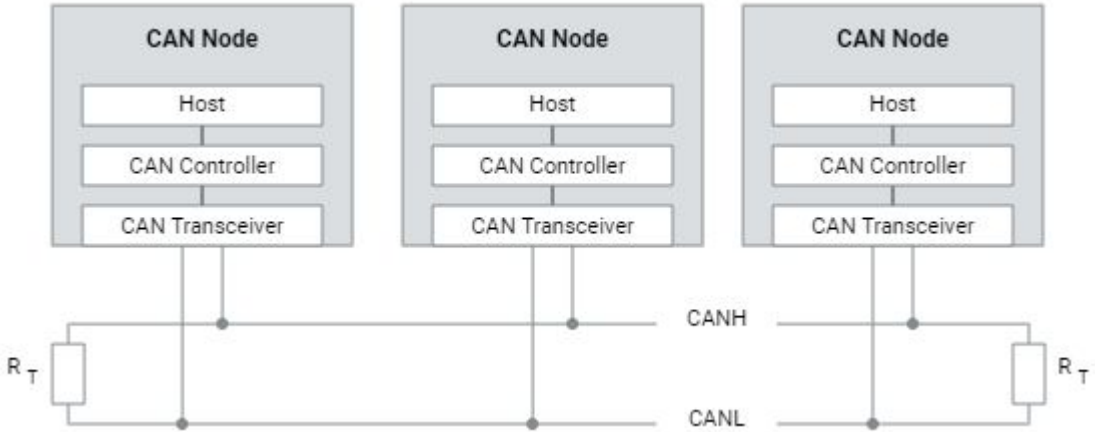
A CAN network consists of a number of CAN nodes which are linked via a physical transmission medium (CAN bus). In practice, the CAN network is usually based on a line topology with a linear bus to which a number of electronic control units are each connected via a CAN interface.



A twisted two-wire line is the physical transmission medium used. A CAN transceiver always has two bus pins: one for the CAN high line (CANH) and one for the CAN low line (CANL). Physical signal transmission in a CAN network is based on transmission of differential voltages (differential signal transmission). This effectively eliminates the negative effects of interference voltages induced by motors, ignition systems and switch contacts.

CAN Node

An electronic control unit (ECU) that wants to participate in CAN communication requires a CAN interface. This comprises a CAN controller and a CAN transceiver. The CAN controller fulfills communication functions prescribed by the CAN protocol, which relieves the host considerably.

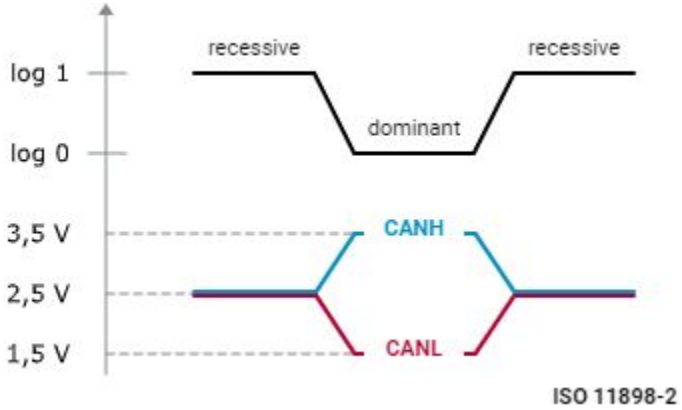


The CAN transceiver connects the CAN controller to the physical transmission medium. Usually, the two components are electrically isolated by optical or magnetic decoupling, so that although overvoltages on the CAN bus may destroy the CAN transceiver, the CAN controller and the underlying host are preserved.

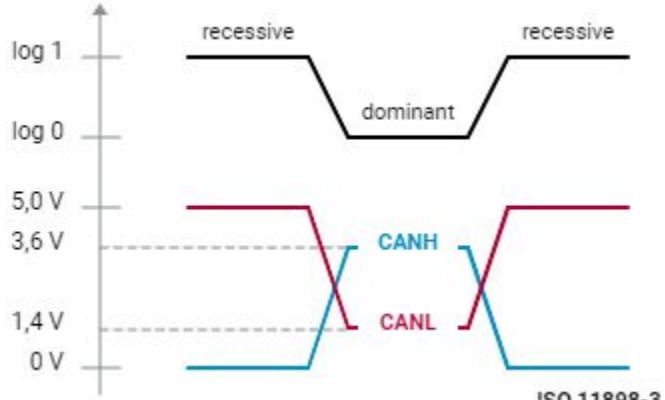
Low Speed and High Speed

Typically, a distinction is made between high-speed CAN transceivers and low-speed CAN transceivers. High-speed CAN transceivers support data rates up to 1 Mbit/s. Low-speed CAN transceivers only support data rates up to 125 kbit/s.

ISO 11898-2 assigns logical “1” to a typical differential voltage of 0 Volt. The logical “0” is assigned with a typical differential voltage of 2 Volt. ISO 11898-3 assigns a typical differential voltage of 5 Volt to logical “1”, and a typical differential voltage of 2 Volt corresponds to logical “0”.



ISO 11898-2

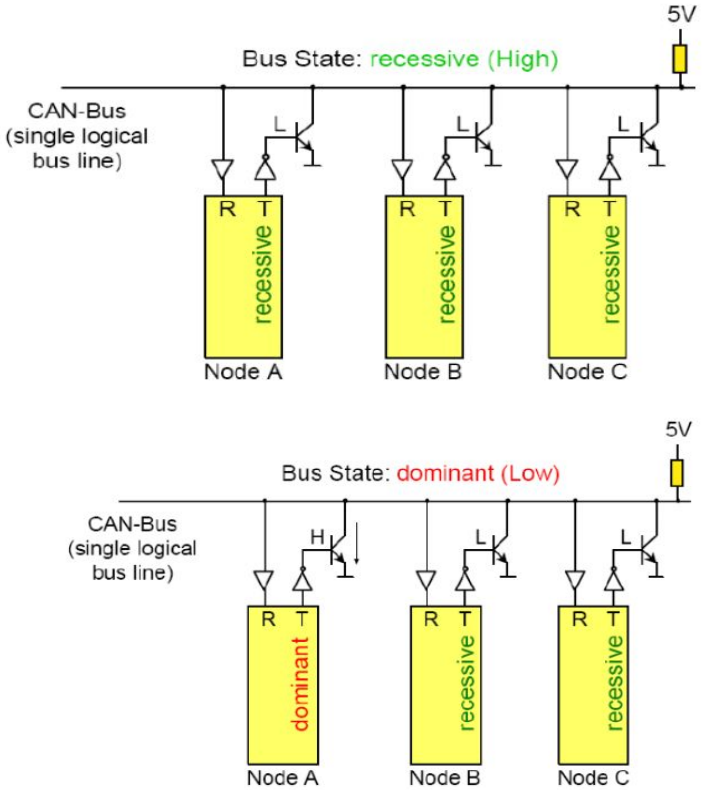


ISO 11898-3

CAN Bus Logic

A basic prerequisite for smooth communication in a CAN network — especially for bus access, fault indication and acknowledgement — is clear distinctions between dominant and recessive bus levels. The dominant bus level corresponds to logical “0”. The recessive bus level corresponds to logical “1”.

The dominant bus level overwrites the recessive bus level. When different CAN nodes send dominant and recessive bus levels simultaneously, the CAN bus assumes the dominant bus level. The recessive bus level only occurs if all CAN nodes send recessive levels. In terms of logic, such behavior is AND-logic. Physically, AND-logic is implemented by a so-called open collector circuit.



CAN General Specifications

- **Differential Communication System**
  Physical signal transmission in a CAN network is based on transmission of differential voltages

- **Serial Protocol**
  Data Sent bit by bit in series

- **Half Duplex Communication**
  Only one node can transmit data on the bus at certain time

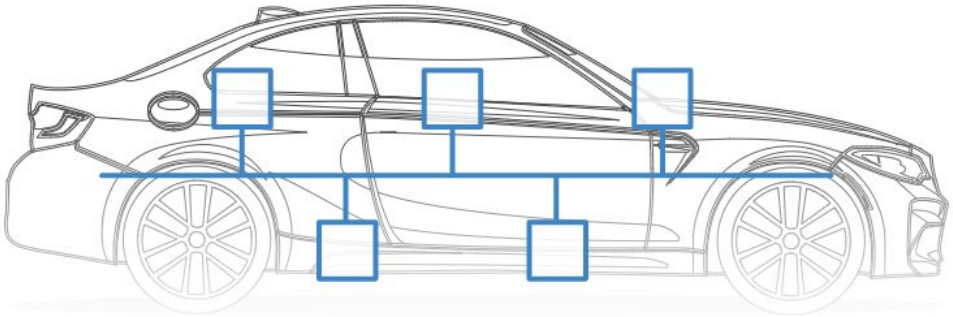
- **Multi Master No Slave**
  All Nodes can send on the bus.

- **Asynchronous Protocol**
  No shared clock nor synchronization bits

- The maximum data rate is 1 Mbit/s. A maximum network extension of about 40 meters is allowed. The standards recommends the maximum number of CAN nodes as 32.

Communication Principle

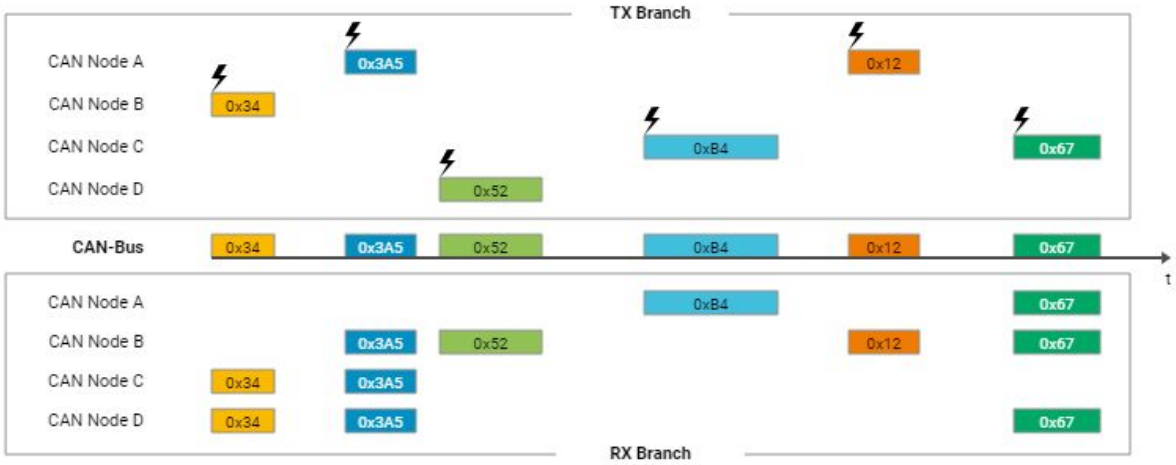
CAN network is based on a combination of multi-master architecture and line topology: essentially each CAN node is authorized to place CAN messages on the bus in a CAN network. The transmission of CAN messages does not follow any predetermined time sequence, rather it is event-driven.



A method of receiver-selective addressing is used in a CAN network to prevent dependencies between bus nodes and thereby increase configuration flexibility: Every CAN message is available for every CAN node to receive (broadcasting). A prerequisite is that it must be possible to recognize each CAN message by a message identifier (ID) and node-specific filtering. Although this increases overhead, it allows integration of additional CAN nodes without requiring modification of the CAN network.

Communication Principle

Data Frame	CAN Node A	CAN Node B	CAN Node C	CAN Node D
ID = 0x12	Sender	Receiver		
ID = 0x34		Sender	Receiver	Receiver
ID = 0x52		Receiver		Sender
ID = 0x67	Receiver	Receiver	Sender	Receiver
ID = 0xB4	Receiver		Sender	
ID = 0x3A5	Sender	Receiver	Receiver	Receiver



CAN Data Frame

Standard Data Frame

Data frames assume a predominant role in a CAN network: They serve to transmit user data. A data frame is made up of many different components. Each individual component carries out an important task during transmission. Tasks to be performed are: Initiate and maintain synchronization between communication partners, establish the communication relationships defined in the communication matrix and transmit and protect the user data.

CAN Data Frame

SOF

Transmission of a data frame begins with the start bit (Start of Frame — SOF). It is transmitted by the sender as a dominant level which produces a signal edge from the previous recessive (bus idle) level which is used to synchronize the entire network. In order for the receivers not to lose synchronism to the sender during transmission of the frame, they compare all recessive-to-dominant signal edges with their preset bit timing. In case of deviation, receivers re-synchronize by the amount of the relevant phase error (re-synchronization).

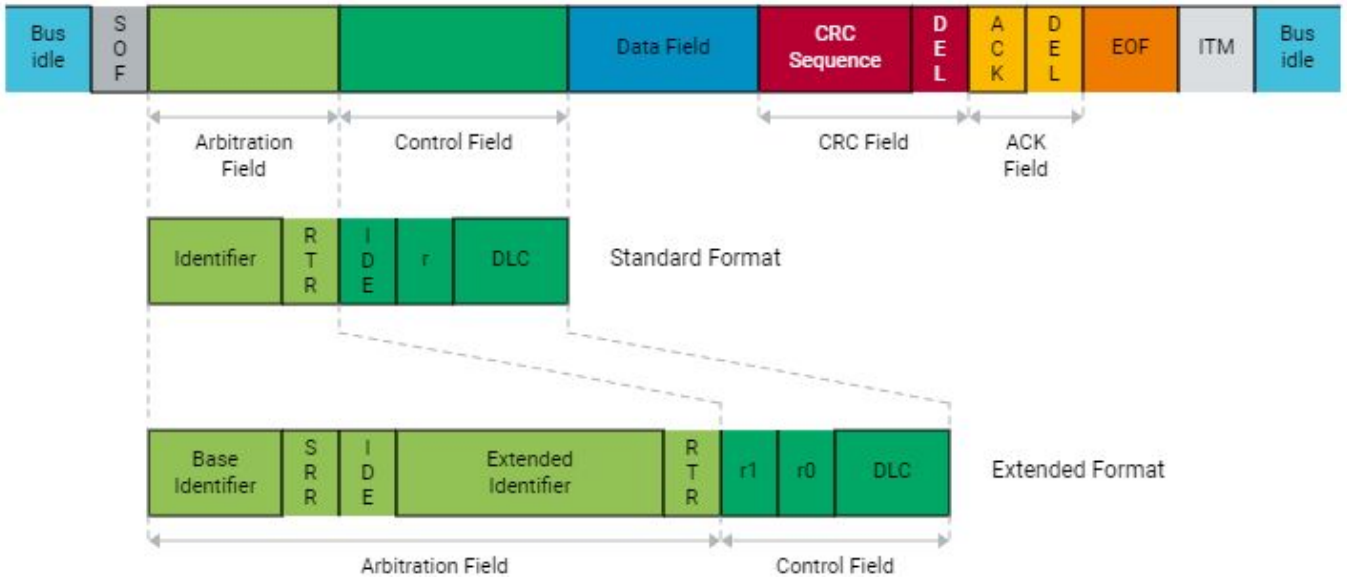
ID and RTR

Following the SOF is the identifier (ID). This sets the priority of the data frame, and together with the acceptance filtering it provides for sender-receiver relations in the CAN network that are defined in the communication matrix. Next comes the RTR bit (Remote Transmission Request). It is used by the sender to inform receivers of the frame type (data frame or remote frame). A dominant RTR bit indicates a data frame.

CAN Data Frame

IDE

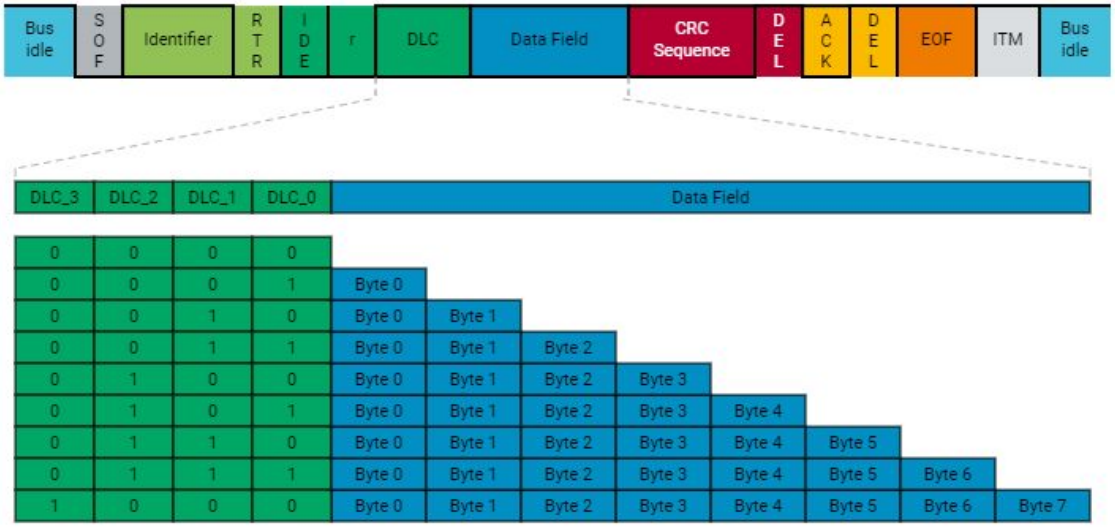
The IDE bit (Identifier Extension bit) which follows serves to distinguish between standard format and extended format. In standard format the identifier has 11 bits, and in extended format 29 bits.



CAN Data Frame

DLC

The DLC (Data Length Code) communicates the number of payload bytes to the receivers. The payload bytes are transported in the data field. A maximum of eight bytes can be transported in one data frame.



CAN Data Frame

CRC and ACK

The payload is protected by a checksum using a cyclic redundancy check (CRC) which is ended by a delimiter bit. Based on the results of the CRC, the receivers acknowledge positively or negatively in the ACK slot (acknowledgement) which also is followed by a delimiter bit.

EOF

After this the transmission of a data frame is terminated by seven recessive bits (End Of Frame — EOF).

Reserved Bits

In classical CAN, reserved bits are dominant bits used for frame check.

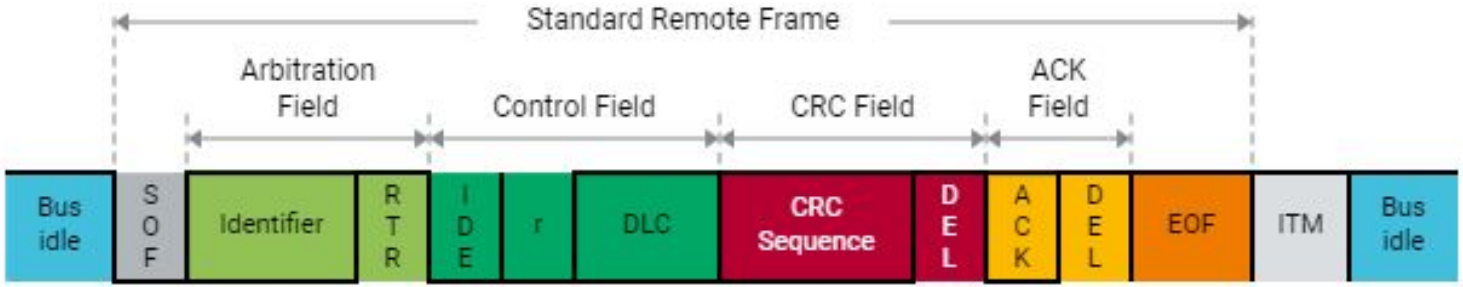
Inter Frame Space IFS

Used to separate consecutive frames and consists of 3 recessive bits

CAN Remote Frame

Besides the data frame used to transport data, there is the remote frame — a frame type used to request data, i.e. data frames, from any CAN node. Except for the lack of a data field, the layout of a remote frame identical to that of a data frame. Data and remote frames are differentiated by the RTR bit (Remote Transmission Request). In the case of a data frame, the RTR bit is sent as dominant. A remote frame is identified by a recessive RTR bit.

Standard Remote Frame



Addressing

Communication in the CAN network is based on content-related addressing. It is not the CAN nodes that have identifiers, but rather the data and remote frames are identified (identifier — ID). So, all CAN messages can be received by every CAN node (broadcasting). Each receiver is independently responsible for selecting CAN messages. Such receiver-selective addressing is very flexible, but it requires that each receiver filters the received CAN messages (acceptance filtering).

Each can node stores a database called CAN DB. This database identify the owner of each message and the receivers of it. Only one owner per message ID is allowed. So that only one node can send a certain message.

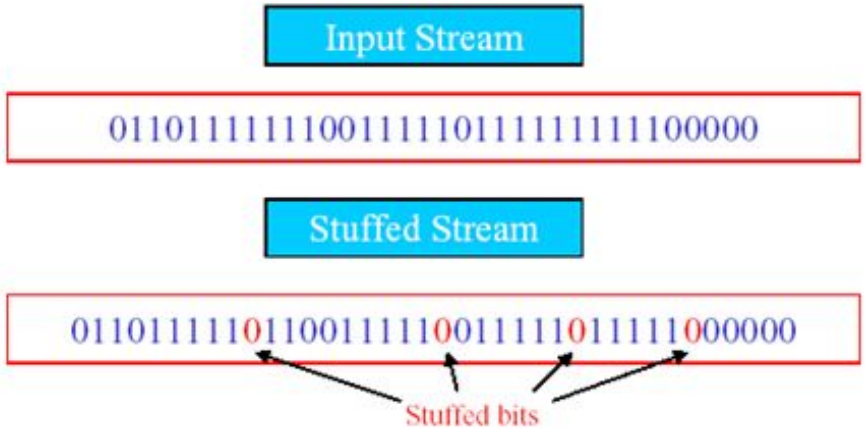
However, the message can be received by many nodes. Once a node send a message, all other nodes receive the message and check their DB, if the received ID matches a record in the database then the node continue listening to the frame. Otherwise, the node would neglect the frame.

Bit Stuffing

Senders must transmit a complementary bit at the latest after transmitting five homogeneous bits; a stuff bit is added even if a complementary bit followed the five homogeneous bits anyway.

This technique is used for frame synchronization and to ensure the frame is correctly sent.

Bit stuffing begins with transmission of the SOF and ends with transmission of the last bit of the CRC sequence



Questions … ?

1-What is the worst case the theoretical number of stuff bits … ?

A- 20
B- 22
C- 24

1-What is longest possible data frame in standard format … ?

A- 129
B- 132
C- 139

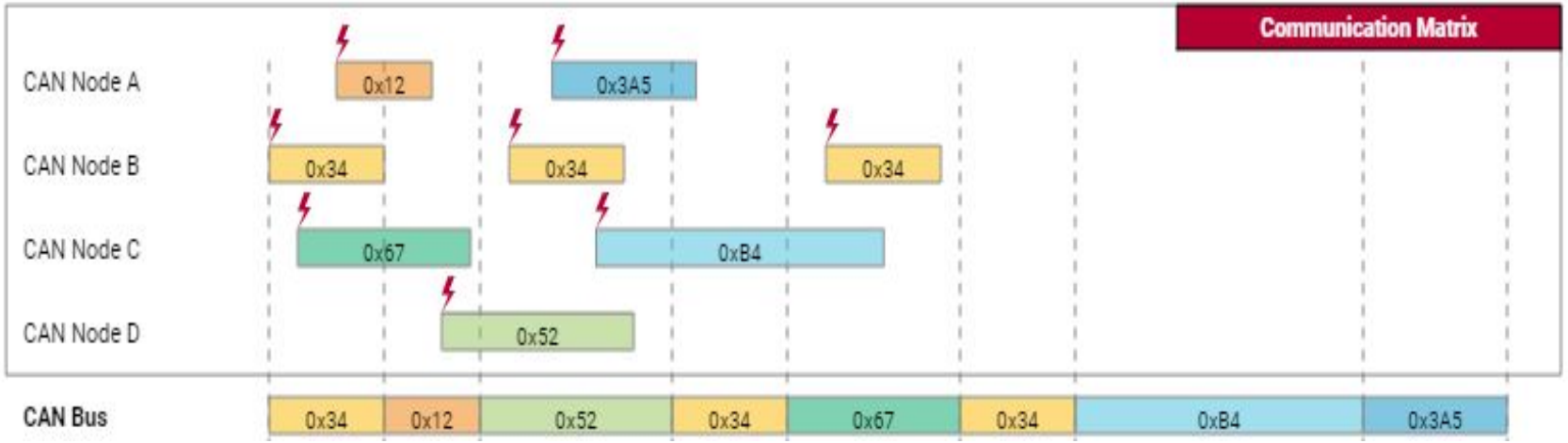
Bus Arbitration

CAN defines a multi-master architecture to assure high availability and event-driven data transmission. Each node in the CAN network has the right to access the CAN bus without requiring permission and without prior coordination with other CAN nodes. Although bus access based on an event-driven approach enables very quick reactions to events, there is the inherent risk that several CAN nodes might want to access the CAN bus at the same time, which would lead to undesirable overlaps of data on the CAN bus.

In case of simultaneous bus access, a bitwise bus arbitration ensures that the highest priority CAN message among the CAN nodes prevails. In principle, the higher the priority of a CAN message the sooner it can be transmitted on the CAN bus. In case of poor system design, low priority CAN messages even run the risk of never being transmitted.

All CAN nodes wishing to send place their identifier of the CAN message bitwise onto the CAN bus, from most significant to least significant bit. In this process, the wired-AND bus logic upon which the CAN network is based ensures that a clear and distinct bus level results on the bus.

Bus Arbitration



Error Detection

To detect corrupted messages, the CAN protocol defines five mechanisms: bit monitoring, Form Check, Stuff Check, ACK Check and Cyclic Redundancy Check.

Bit monitoring (Sender Task)

The sender compares the sent bit level with the actual bus level. A bit error exists if the sender detects a discrepancy between the two levels. The sender then stop sending data and convert to be a listener.

Form check (Receiver Task)

The form check serves to check the format of a CAN message. Each CAN message always exhibits the same bit sequences at certain positions. They are the CRC delimiter, ACK delimiter and EOF. Senders always transmit these message components recessively. A format error exists if a receiver detects a dominant bus level within one of these message components in the Form Check.

Error Detection

CRC Check (Receiver Task)

In the cyclic redundancy check (CRC) the polynomial R(x) associated with the arriving data or remote frame should equal a multiple of the generator polynomial G(x) specified by ISO 11898-1. If this is not the case (CRC error), then the data or remote frame was corrupted during its transmission.

ACK check (Sender Task)

The acknowledgement mechanism defined in the CAN protocol specifies that all receivers must acknowledge every arriving CAN message right after the cyclic redundancy check. A single positive acknowledgement is sufficient to signal to the sender that at least one receiver received the CAN message it transmitted correctly. If not a single positive acknowledgement arrives at the sender, then an acknowledgement error has occurred (ACK error).

Stuff check (Receiver Task)

The stuff check serves to check the bit stream. The CAN protocol specifies that the sender must transmit a complementary bit after five homogeneous bits — for synchronization purposes. There is a stuffing error if more than five homogeneous contiguous bits are received.

Error Reporting

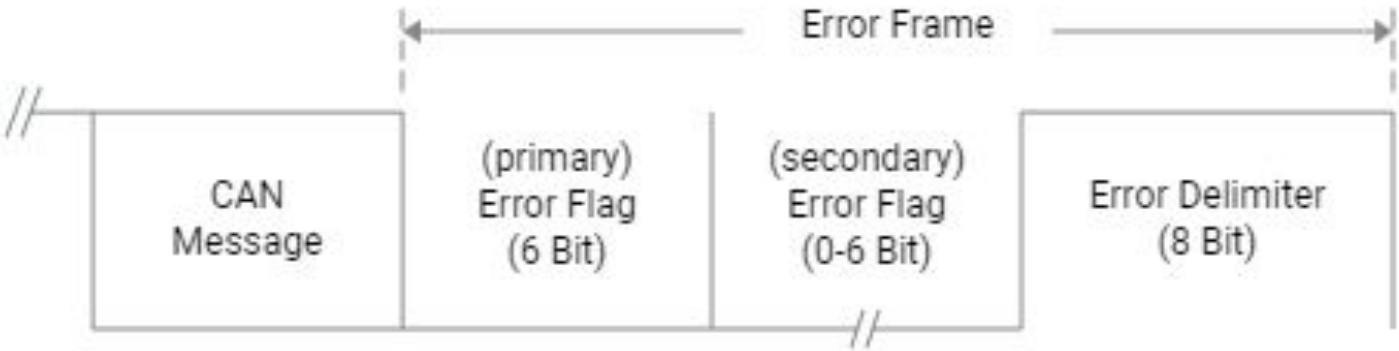
The CAN protocol prescribes — for reasons of network-wide data consistency — that if the error-detecting CAN node is experiencing a local disturbance, it must inform all CAN nodes connected to the CAN network. The error-detecting CAN node transmits an error signal (error flag) for this purpose, which is made up of six dominant bits. This is an intentional violation of the bit stuffing rule, and it generates a bit stuffing error.

Transmission of an error flag ensures that all other CAN nodes will also transmit an error flag (secondary error flag) and thereby also terminate the regular data transmission just like the sender of the primary error flag. Depending on the situation, the primary and secondary error flags might overlap.

Transmission of an error flag is always terminated by an error delimiter. This consists of eight recessive bits. The error delimiter replaces the ACK delimiter and the EOF of a regular message transmission, so that together with the obligatory transmission pause (ITM — Intermission) on the CAN bus, this results in eleven recessive bits (bus-idle identifier).

Error Reporting

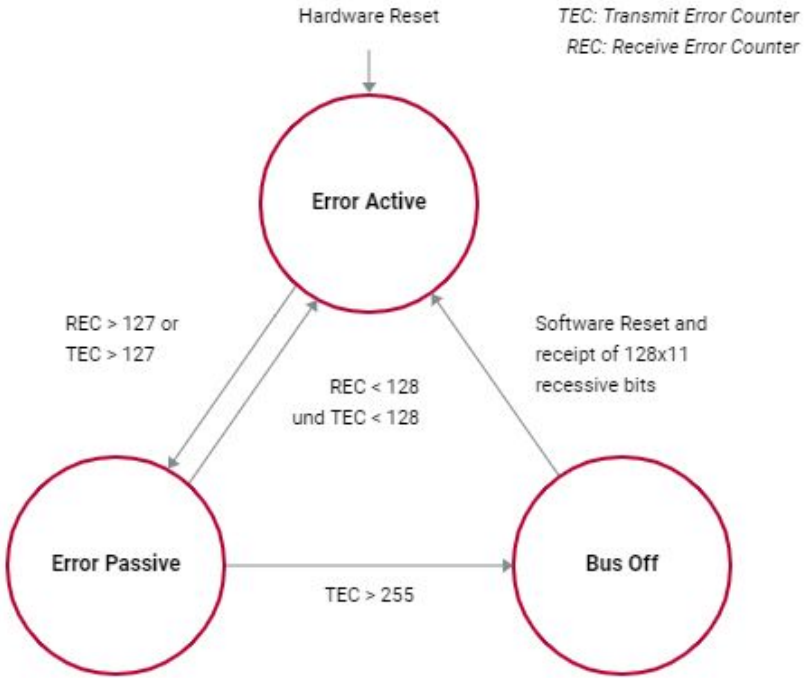
Error Signaling



Error Tracking

To assure network-wide data consistency, each node in a CAN network has the right to terminate any CAN message interpreted as faulty.

Consequently, each CAN controller has a TEC (Transmit Error Counter) and a REC (Receive Error Counter). In case of successful transmission of a data or remote frame, the relevant error counter is decremented (TEC=TEC-1; REC=REC-1). Detection and subsequent transmission of a error flag causes the relevant error counter to be incremented according to certain rules. For the sender the following rule applies: TEC=TEC+8. Error-detecting receivers initially increment their REC by one unit (REC=REC+1). For the error causing receiver: REC=REC+8.



Error Tracking

Active Error

Depending on the specific error count, a CAN controller handles switching of the error state. After the start, a CAN controller assumes the normal state Error Active. In this state, the CAN controller sends six dominant bits (active error flag) after detecting an error. When a limit is exceeded (TEC>127; REC>127), the CAN controllers switch over to the “Error Passive” state.

Passive Error

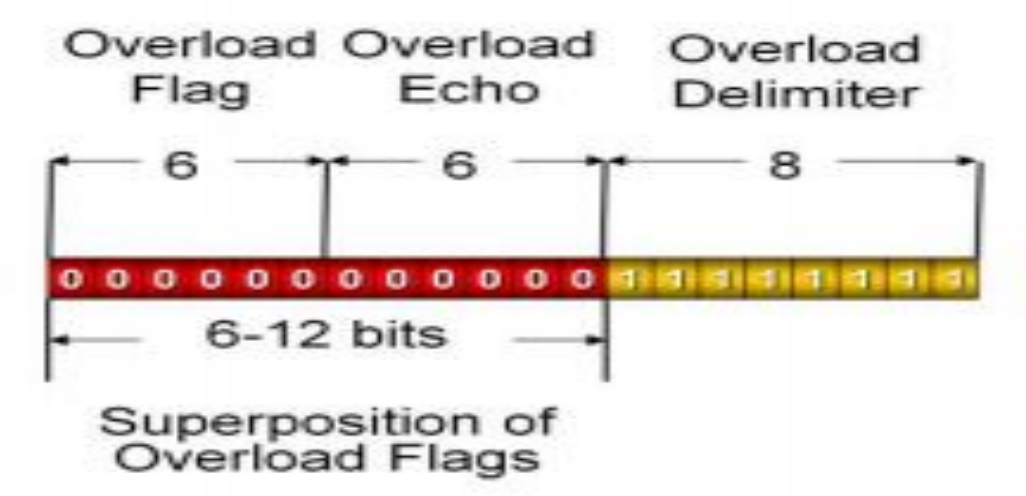
CAN controllers in the Error Passive state can only indicate a detected error by sending six homogeneous recessive bits. This prevents the error-detecting receivers from globalizing detected errors. In addition, when sending two consecutive data or remote frames, CAN controllers that are in the “Error Passive” state must wait the “Suspend Transmission Time” (8 bits).

Bus Off

If a CAN controller fails or if there are extreme accumulations of errors, a state transition is made to the Bus Off state. The CAN controller disconnects from the CAN bus. The Bus-Off state can only be exited by intervention of the host (with a mandatory waiting time of 128 x 11 bits) or by a hardware reset.

Overload Frame

It is very similar to Error Frame in format and it is transmitted by any node to buy some time if it is too busy so that it can complete its pending task.



www.imtschool.com

ww.facebook.com/imaketechnologyschool/