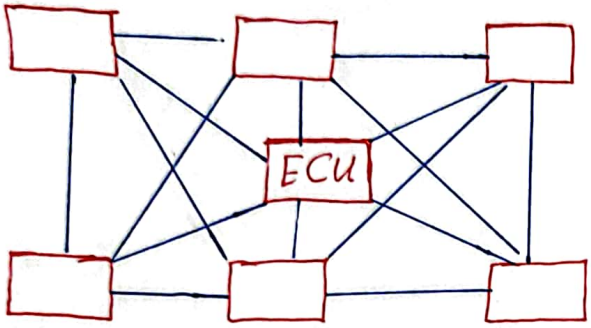
Automotive protocols
Controller Area Network
(CAN protocol)

→ Why do we need CAN Bus ?

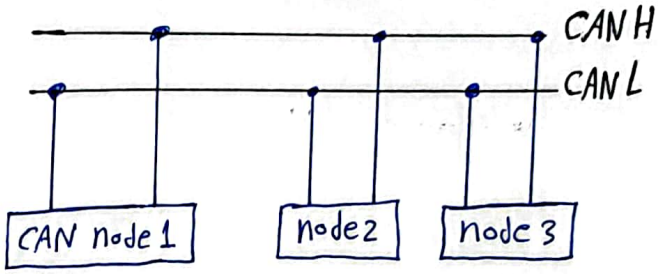
- Before the invention of the CAN protocol, a large number of wires were used to connect different ECUs with each other
- However, after the invention of the CAN protocol, only two wires were needed to replace all these wires.

*wiring before CAN protocol



→ huge number of wires

*wiring after CAN protocol which replaces huge number of wires with only a two-wire bus



→ What is CAN protocol ?

- the Controller Area Network protocol is a widely used communication protocol originally developed for the automotive industry. However, its applications have expanded to other industries such as industrial automation, and medical equipment.
- it becomes international standard (ISO11898) in 1993
- in CAN, any device on the network, referred to as a node
- any node can initiate communication.

→ Features :-

→ serial protocol

→ multi-master protocol : each node can send or receive data at any time

→ Broadcasting protocol : when a node sends data, it is received by all other nodes.

→ Built-in error detection and error handling mechanism.

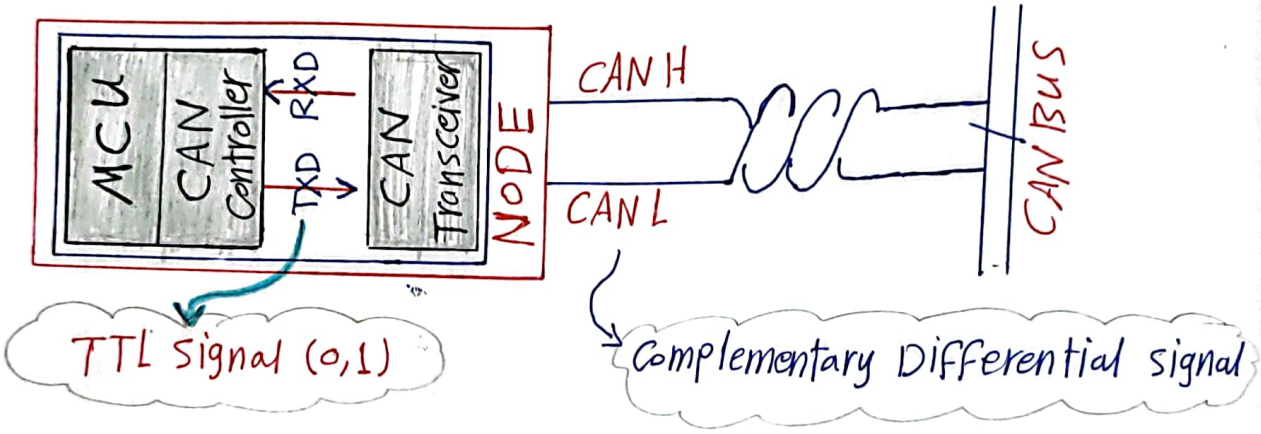
→ easy connection and disconnection of any CAN node from the bus.

→ message-based protocol : Data is sent within a message "data frame" with a unique identifier (ID)

→ supports periodic messages (e.g., 10ms, 20ms...) or message triggering (event-based or data change)

→ Retransmission of faulty messages.
→ Asynchrounous protocol ~~-~~ - Half-duplex .
→ medium : wired → "twisted pair" lines for data (CANH - CANL), having 120Ω impedance connected at each end
→ Low cost - Robustness - Flexibility
→ Scalability : supports a large number of devices on a network, facilitating easy addition or removal of devices as needed.
→ Data speed from 20 Kbit/sec up to 1 Mbit/sec at High speed CAN.

* CAN Node structure :-



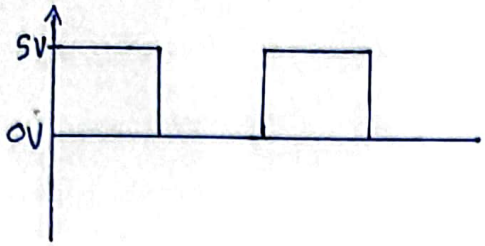
→ the CAN node consists of :
1- microcontroller unit : decides how to deal with received data or data to be transmitted
2- CAN controller : this unit applies acceptance filtering as well as it receives whole data until a complete message is received
3- CAN transceiver : this unit converts transmit signal from CAN controller level to CAN-Bus level and vice-versa.
"Differential signal"

→ CAN Bus is made of twisted pair cable :.
→ to reduce the cross talk "electromagnetic interference"

→ one of the most important characteristics of the CAN protocol is its robust bus due to its high immunity to electrical interference.
→ this achieved through sending data via a differential signal, which means that data is sent one bit at a time through two complementary signals called CAN High and CAN low

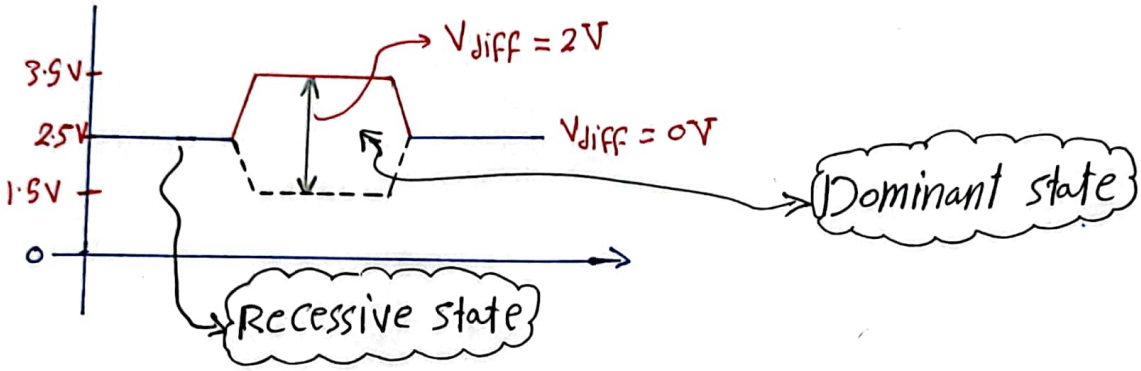
* Bus signals : "TTL signals Vs. differential signals".

[1] TTL :



→ it has → Low logic level = 0 V
→ High logic level = 5 V

[2] Differential signals :-



* recessive state = logical 1
CAN H = 2.5 V
CAN L = 2.5 V
CAN Diff = 0 V

* dominant state = Logical 0
CAN H = 3.5 V
CAN L = 1.5 V
CAN Diff = 2 V

$V_{diff} = CANH - CANL$

→ the dominant state : occurs when a logic low level is applied to the transmit input pin (TXD)

→ the recessive state : corresponds to a logic high level on the transmit input pin of the transceiver.

→ if (V_{diff} < 0.5 Volt) : the bus is considered to be in a recessive state

→ if (V_{diff} > 0.9 Volt) : the bus is considered to be in a dominant state

→ if (0.5 V < V_{diff} < 0.9 V) : the bus is considered to be in a undefined state

* noise immunity of CAN bus :-

1- Differential signal : it allows for the cancellation of both internal and external noise.

→ external noise refers to disturbances that can affect the bus from outside sources.

→ However, since we utilize the differential signal, which takes the difference between the CAN-L and CAN-H wires, the noise is effectively canceled out.

→ for example, if CAN-H = 3, CAN-L = 2 and the noise = 1, the resulting calculation would be (1+3) − (1+2) = 1, indicating successful cancellation of the noise

2- Twisted-Pair Lines :

→ two wires are twisted together along their length.

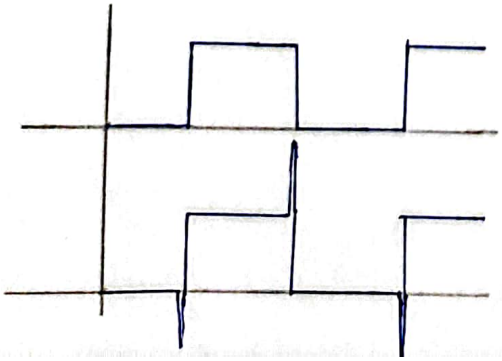
→ the twisting of the wire helps in reducing the magnetic interference between them. As the wires are twisted, the magnetic fields they generate interact with each other, resulting in canceling out the magnetic interference.

3- the pull-up resistor (termination resistor) :

→ the pull-up resistor, with a value 120 Ω, is connected between the CAN-L and CAN-H

→ During voltage transitions between High and Low states within a high speed CAN protocol, Transient phenomena like voltage spikes can occur.

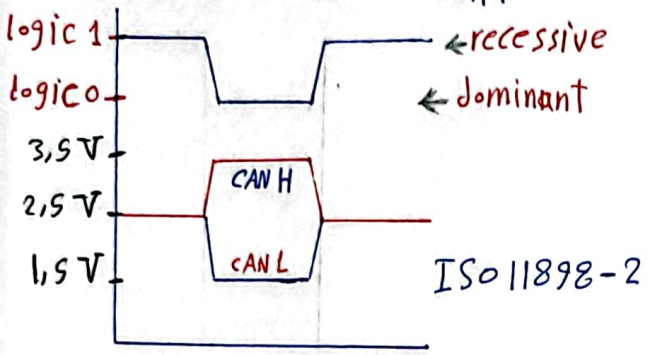
→ when there is a spike in voltage during these transitions, the termination resistor is used to absorb the energy from these reflections, ensuring smooth voltage transitions and maintaining signal integrity.



Idle state

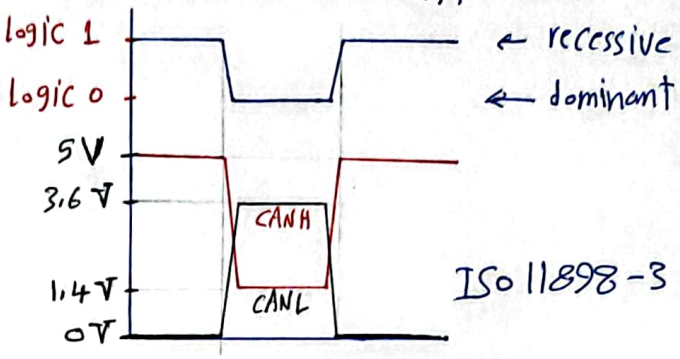
transient phenomena

* High speed CAN bus (ISO 11898-2)
  → up to 1 Mb/s
  → up to 5 Mb/s → in CAN-FD
* ISO 11898-2 assigns:
  logical "1" to → V_{diff} = 0 V
  logical "0" to → V_{diff} = 2 V



* low speed CAN bus (ISO 11898-3)
  → up to 125 Kb/s

* ISO 11898-3 assigns:
  logical "1" to → V_{diff} = 5 V
  logical "0" to → V_{diff} = 2 V



* Bus arbitration :-
  → if two messages are simultaneously sent over the CAN Bus, the bus takes the "wired logical AND connection"
  → Recessive state will only exist on the bus if all transceivers connected to the bus are transmitting a recessive state.
  → Dominant state will be on the bus if one of the CAN nodes wrote a dominant state.
  → Recessive state is weakly biased, while the dominant state is strongly biased

node 1	0	1	0	1	0	1	0	1
node 2	0	0	1	1	0	0	1	1
node 3	0	0	0	0	1	1	1	1
CAN Bus	0	0	0	0	0	0	0	1

→ Standard CAN (CAN 2.0A) Vs. Extended CAN (CAN 2.0B)
  • CAN protocol is message based protocol → each message has a unique identifier.
  → Both standard and Extended CAN message frame can coexist on the same CAN bus.
  → the controller supports Extended CAN, can also support standard CAN
  → the controller supports standard CAN only, can't support Extended CAN

→ the standard CAN :

- standard CAN message frame uses an 11-bit message identifier
- Version → CAN2.0A
- the most one used as it:
  - less silicon overhead - Greater message throughput
  - improved latency times.
- this allows up to (2^11) messages = 2048 identifier.
- the lowest 16 messages IDs are reserved (ID=2032 : ID=2047).
- the standard CAN will always have priority over the same extended CAN with identical 11-bit "base ID"

→ the Extended CAN:

- Extended CAN message frame uses a 29-bit message identifier
- this allows up to (2^29) messages
- this 29-bit → 11-bit base id + 18-bit extended id
- more silicon cost and less efficient use of bus - requres more bus bandwidth

	Standard	Speed	identifier
Low speed CAN	ISO11519	125 kbps	11-bit
CAN2.0A	ISO11898:1993	1Mbps	11-bit
CAN2.0B	ISO 11898:1995	1Mbps	29-bit

---

* principle of Bus Access:.

- Bus access for all nodes:.
  - in a CAN bus, devices connected to the network can independently access the bus without requiring permission or coordination from other devices.
  - this design, known as a multi-master architecture, ensures high availability and enables quick responses to events.
  - the bus access based on an event-driven approach, which allows devices to rapidly transmit data when needed, facilitating quick communication.
  - However, the simultaneous access of multiple devices can lead to data overlap or conflicts, causing problems in data transmission.
- Collision Avoidance mechanism:
  - the CAN bus employs a collision avoidance mechanism to prevent data overlaps and conflicts
  - the ISO 11989-1 standard defines the rules for bus access management
  - the system uses a method called carrier sense multiple access with collision avoidance (CSMA/CA) to ensure efficient bus access.

- devices check if the bus is free before sending data, and if it is already in use, they wait for their turn to access the bus, avoiding collisions.

- Bitwise bus arbitration :-
  - in case of simultaneous bus access, the CSMA/CA method based on 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.

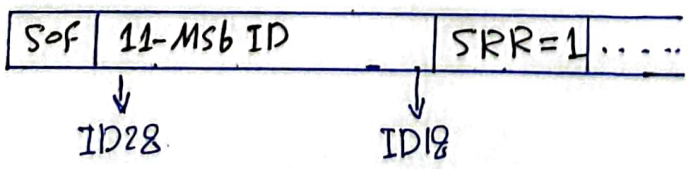
* unique bus level :-
  - after network-wide synchronization, all CAN nodes wishing to send place their identifier of the CAN message bitwise onto the CAN bus, from most significant to least significant bit.
  - in this process, the "wired-AND" bus logic upon which the CAN network is based ensures that a clear and distinct bus level results on the bus.

* Arbitration logic:
  - the arbitration logic determines whether a CAN node continue sending its message or must stop
  - it uses bitwise comparison of message identifier to determine the node with the lowest identifier (highest priority) that can transmit its message on the bus.
  - other nodes with higher identifier switch to a receiving state and attempt to access the bus later.

* one sender at a time :-
  - at the end of the arbitration phase, the CAN node transmitting the CAN message with the lowest ID gets authorization to send

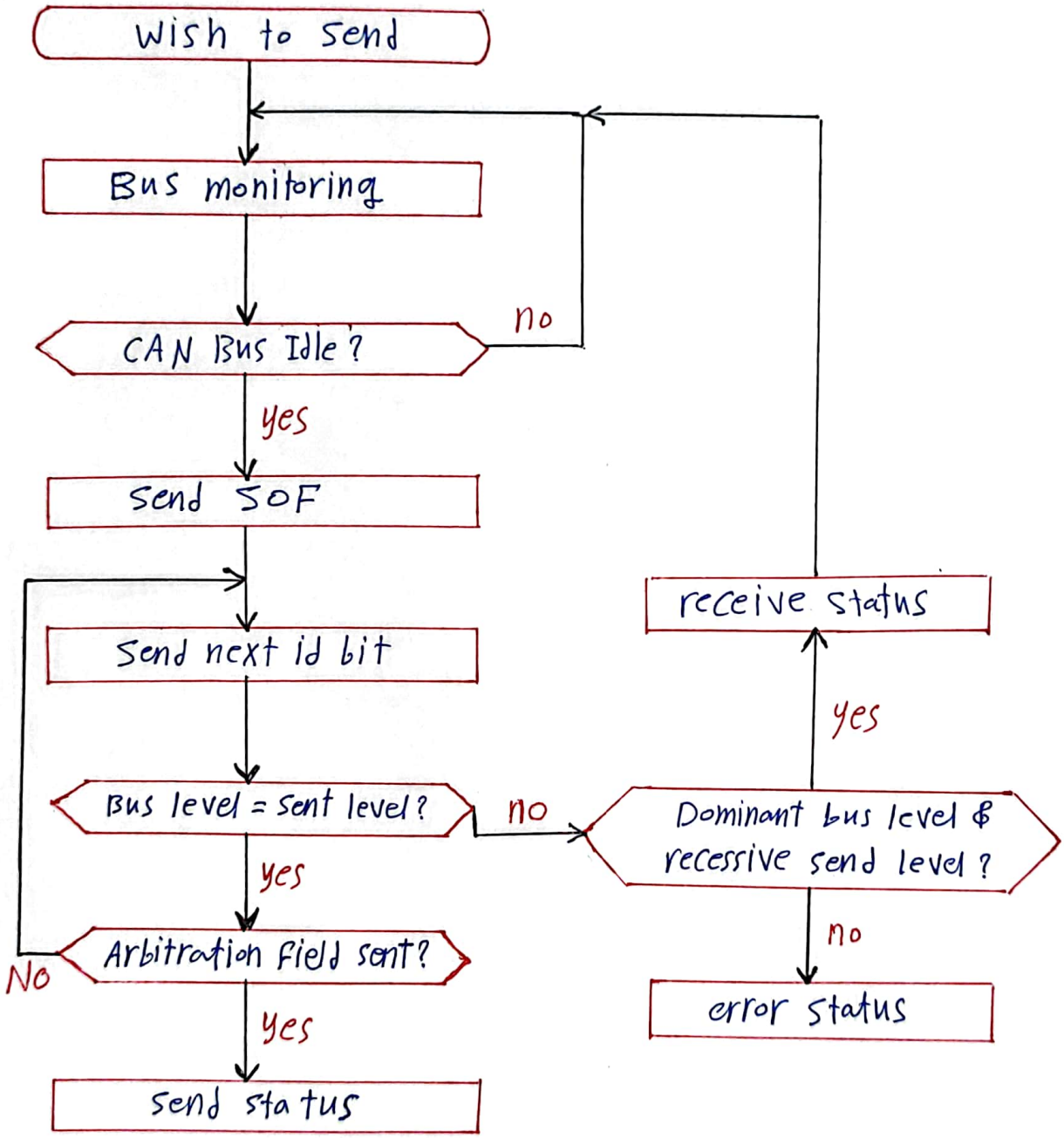
* standard frame arbitration field

* extended frame arbitration field



→ standard CAN priority always > extended CAN priority

* Bus Access procedure :-



* CAN ID prioritization :-

→ the prioritization of CAN message is determined by their identifier, with lower ID values indicating higher priority

→ the priorities of the CAN message are encoded via the identifier which is transmitted bitwise from the MSB to the LSB

→ the wired-AND bus logic and arbitration logic work together to ensure this prioritization.

*arbitration logic

CAN node/ send level	CAN Bus level	Reaction of CAN node
0	0	Routing
0	1	Error: stop
1	0	stop: receiving
1	1	Routing

0 ← identifier of CAN message → 2047

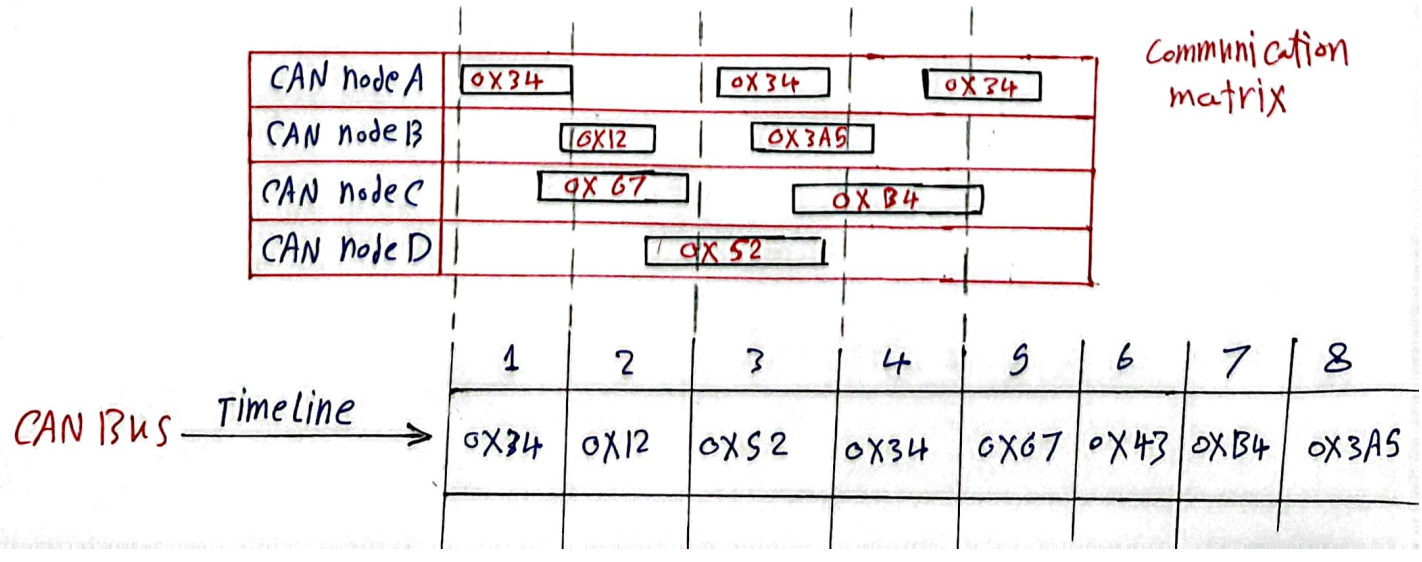
Highest priority ← priority of CAN message → lowest priority

* Busload and real-time behavior :-

→ if the bus load is not too high, this type of random, non destructive and priority-controlled bus access provides for fair and very quick bus access.

→ Nonetheless, it must be taken into account that increasing bus load primarily causes delays in lower-priority CAN message to grow.

→ this could impair the real-time capability of the CAN communication system



* CAN message types: types of Frames.

1- Data frame:
→ if a CAN node needs to send data to another node, it uses the CAN data frame

2- Remote frame:
→ the same structure of data frame but without data
→ it is used by one node to request data from other nodes

3- Error frame:
→ if a transmitting or receiving node detects an error, it will immediately abort the transmission and broadcast an error frame

4- overload frame:
→ if a CAN node receives messages faster than it can process them, then an overload frame will be generated to provide extra delay between successive data or remote frames.

1- Standard Data Frame:-
→ the standard CAN frame with 11-bits identifiers (CAN2.0A), which is the type used in most cars.

SOF	12-bits arbitration field (ID+RTR)	6-bits control field	0:8 bytes data field	16-bits CRC field	2-bits Ack field	2-bits EOF

1- SOF (start of frame):
→ indicating the begining of the frame
→ the sender transmits the SOF as a dominant level, causing a signal edge from the previous recessive level (bus idle).
→ this signal edge is used to synchronize the entire network.
→ receivers compare all recessive-to-dominant signal edges with their preset bit timing to maintain synchronism with the sender.
→ if there is any deviation, receivers re-synchronize by adjusting for the relevant phase error
→ the SOF is a high-to-low transition, indicating the start of the frame and serving as a synchronization signal for nodes on the bus.

2- Arbitration field (ID → identifier and RTR → remote transmission request)

1- CAN ID :-

→ the CAN ID consists of 11-bits, identifying the message priority

→ the lower ID number, the higher the priority

→ the most significant 7-bits of the ID (ID10 :ID4) field must not be all recessive to avoid losing synchronization.

2- RTR bit :-

→ it is used by the sender to inform receivers of the frame type.

- a dominant bit means a data frame
- a recessive bit means a remote frame

→ Data frames have higher priority than remote frames.

3- Control field :-

→ the control field is divided into three parts :.

1- the Identifier Extension (IDE) field, which is

- a dominant bit → standard data frame 11-bit
- a recessive bit → Extended data frame 29-bit

2- A dominant bit reserved for future usage (Ro)

3- the Data Length Code (DLC) field, which consists of four bits determining the number of bytes to be transmitted.

4- the data field :-

→ it consists of 0:8 bytes and represents the data payload to be transmitted

5- the CRC field (cycle redundancy check) :-

→ it consists of 15-bit checksum used for error detection, followed by a delimiter (DEL) → a recessive bit separator.

6- the ACK field :-

→ it consists of an ack bit and a delimiter bit.

→ the ACK bit confirms a successful CRC

→ the sending and receiving nodes work collaboratively on the ACK bit. So the sending node writes the acknowledgment bit as a recessive bit. then the receiving node overwrites it with dominant bit, if CRC succeeds.

→ the delimiter bit is also a recessive bit separator.

7- the End of Frame (EOF):-

→ it is a 7- recessive bits frame that indicates the end of the frame and disable bit stuffing.

→ Bit stuffing is a technique employed in data transmission for synchronization between the sender and receiver.

→ when transmitting data, the sender identifies six consecutive bits of the same polarity in the bit stream.

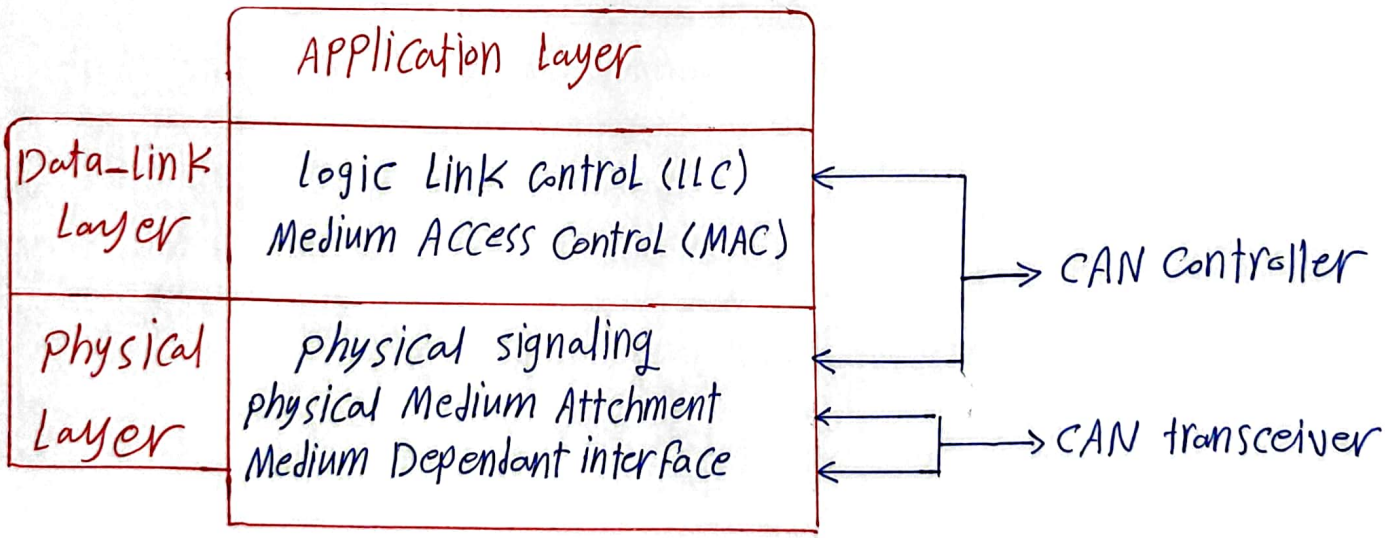
→ to prevent synchronization loss, the sender automatically inserts an additional bit of the opposite polarity after the fifth bit.

→ for example, if the sender encounters six consecutive ones, it inserts a zero after the fifth one, resulting in a pattern of 01 1111 in the transmitted bit stream.

→ the receiver recognize this pattern and removes the extra "0", restoring the original data stream.

---

* Layered ISO 11898 standard architecture:



1- the physical Layer:

→ defines how signals are actually transmitted, Bit timing, Bit encoding, and synchronization.

2- the LLC sub Layer:-

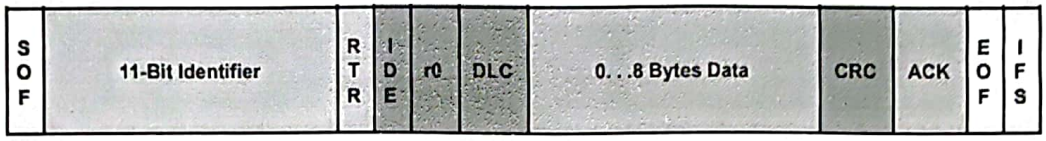
→ concerned with message filtering, overload notification and recovery management.

3- the MAC sublayer: responsible for message framing, arbitration, acknowledgment, error detection and signaling

➔ Standard CAN vs. Extended CAN

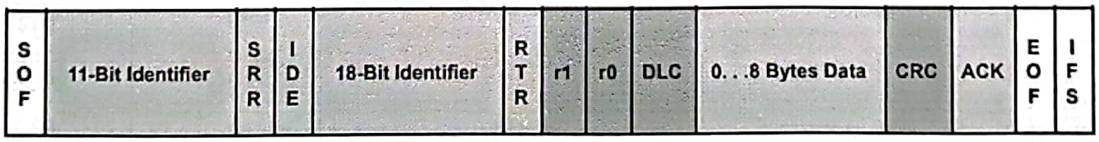
Frame Format for Standard CAN

Field name	Sub-field	Length(bits)	purpose
Start of frame	SOF	1	Indicates the start of frame on CAN Bus(must be dominant (0))
Arbitration field	Identifier	11	Decides the message priority(Arbitration) on CAN Bus
	RTR	1	Differentiate between Remote Frame Or Request Frame
Control Field	IDE	1	Tells about frame format : Standard(0) or Extended(1)
	R0	1	Reserved must be dominant
	DLC	4	Data Length on the CAN bus
Data Field	D0-D8	64	Data
CRC Field	CRC	15	CRC
	CRC Delimiter	1	Must be Recessive
ACK Field	ACK	1	Acknowledge by receiving node
	ACK Delimiter	1	Must be Recessive
End Of Frame	EOF	7	Indicates end of current frame (Must be Recessive)



Frame Format for Extended CAN

Field name	Sub-field	Length(bits)	purpose
Start of frame	SOF	1	Indicates the start of frame on CAN Bus(must be dominant (0))
Arbitration field	Identifier	11	Decides the message priority(Arbitration) on CAN Bus
	SRR	1	Always Recessive
	IDE	1	Tells about frame format : Standard(0) or Extended(1)
	Identifier	18	Decides the message priority(Arbitration) on CAN Bus(Extended)
	RTR	1	Tells about frame format : Standard(0) or Extended(1)
Control Field	R0	1	Reserved must be dominant
	R1	1	Reserved must be dominant
	DLC	4	Data Length on the CAN bus
Data Field	D0-D8	64	Data
CRC Field	CRC	15	CRC
	CRC Delimiter	1	Must be Recessive
ACK Field	ACK	1	Acknowledge by receiving node
	ACK Delimiter	1	Must be Recessive
End Of Frame	EOF	7	Indicates end of current frame (Must be Recessive)



* Standard Data Frame Vs. extended Data frame:-

→ the major difference between the two format is the arbitration field:-

- the standard frame has an 11-bit "ID" in the arbitration field allows up to 2048 unique messages.
- the extended frame has a 29 bit "ID" that allows up over 536 million unique messages.

→ Both the standard and extended data frames can coexist on the same CAN bus.

- the controller that supports Extended CAN, can also support standard CAN.
- the controller that supports standard CAN only, can't support extended CAN.

→ the standard 11-bit frame of data will always have priority over the some extended 29-bit frame with identical 11-bit base ID

---

* Remote Frame:-

→ it is used by one node to request data from other nodes.

→ RTR bit in this frame is recessive

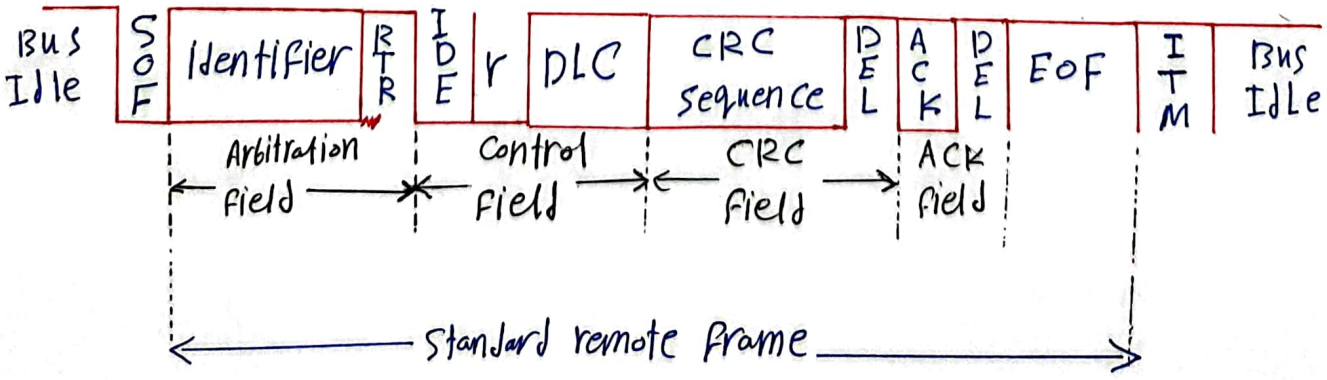
→ no data field is required

→ except for the lack of a data field, the layout of a remote frame identical to that of a data frame

→ the Remote frame and the corresponding data frame are named by the same identifier

→ these frames are hardly ever used in automotive applications, since data transmission is not based on requesting, rather it is primarily based on the self-initiative of information producers.

→ remote frames may be transmitted in either standard or extended format.



* Some concepts related to Remote Frame :-

* Two frames and one-identifier :-

- Remote frames can be defined for all data frames in the CAN network.
- the identifier of a remote frame should match the identifier of the associated data frame
- when a remote frame is received, the responsible ECU generates and sends the corresponding data frame in response.

* remote frame and response :.

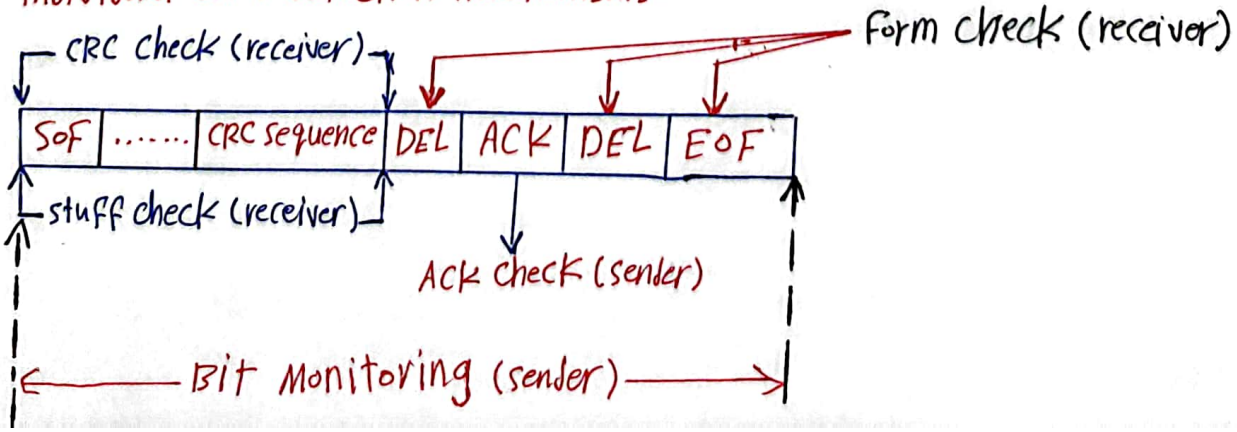
- in the case of a CAN controller with object storage (Full CAN → Buffers) the CAN controller automatically responds to a remote frame without letting the host know about the remote frame
- CAN controllers without object storage need to inform the host about the remote frame to initiate a response.
- in the ideal case, the request by remote frame immediately leads to a response with the relevant data frame.
- However, it may occur that CAN messages with higher priority may be inserted between request and response.

logical Error Detection

→ there are five mechanisms to detect errors in CAN frame :

1- Bit Monitoring (performed by the sender)
2- stuff check (performed by the receiver)
3- Form "format" check (performed by the receiver)
4- Cycle Redundancy check (CRC) (performed by the receiver)
5- ACK check (performed by the sender)

→ this figure shows which fields of a data or remote frame are affected by the individual error detection mechanisms.



1- Bit Moitoring (performed by the sender)

→ in the CAN bus, when any node transmits a frame, all other nodes, including the transmitter itself, act as listeners.

→ the transmitter also reads back the data it has transmitted from the CAN bus to ensure that the transmitted data remains intact (valid).

→ if the read bit is different from the bit transmitted by the sending node, it is considered a transmission error, known as a Bit-Error.

→ only the sending node can detect and identify bit errors.

→ Bit Monitoring ensures that all global errors (affecting the entire network) and all local errors (occurring at the sender) are detected.

2- Stuff Check (performed by the receiver)

→ the "stuff check" performed by the receiver to ensure the integrity of the bit stream

→ the sender in CAN protocol transmits a complementary bit after five homogeneous bits for synchronization purposes.

→ if more than five homogeneous contiguous bits are received, it indicates a stuffing error.

→ Bit stuffing is the process of inserting bits in the bit stuffing area, while de-stuffing is the removal of these inserted bits on the receiver side

→ Synchronization Concept:

- a basic [redacted] prerequisite for correct data transmission is synchronized communication partners
- the dominant-to-recessive signal edge of the start bit (SOF) establishes synchronization for a CAN message
- afterwards, a resynchronization mechanism is used to maintain synchronism up to the end of the message transmission

→ Resynchronization Concept:

- resynchronization is based on evaluation of recessive-to-dominant signal edges.
- the bit stuffing mechanism ensures the presence of such signal edges.
- according to ISO 11898-1, senders must insert a complementary bit after transmitting five consecutive homogeneous bits, even if a complementary bit was already present.

→ Bit stuffing area :-

- Bit stuffing begins with the transmission of the SoF and continues until the last bit of the CRC sequence.
- in the worst-case scenario, when transmitting a data frame in standard format with an 8-byte data field, the theoritical maximum number of stuff bits would be 24
- thus, the longest possible data frame in standard format would be consist of 132 bits.

3- Form check (performed by the receiver) :.

→ it is performed to validate the format of a CAN message.

→ certain bit sequences are expected at specific positions in every CAN message

→ the receiver checks the following bits during the form check :

- CRC DEL - ACK DEL - EOF

→ these message components are always transmitted recessively by the senders

→ if a receiver detects a dominant bus level within any of these message components during the form check, it indicates a format error.

4- CRC check (performed by the receiver) :-

1- Polynomial matching: the receiver applies the CRC algorithm to the arriving data or remote frame by using a polynomial known as R(X).

→ this polynomial should be divisible by the generator polynomial G(X). if the reminder is zero it indicates that the received data or remote frame is likely error-free.

→ How ever, if the remainder is non zero, it signifies a CRC error, indicating that the data or remote frame has been corrupted during transmission

→ the generator polynomial G(X) specified by the ISO 11898-1 standard

5- Ack check (performed by the sender):-

→ according to the CAN protocol, the sender must receive an acknowledgement from all receivers for every transmitted CAN message immediately after the CRC field.

→ a single positive ack is considered satisfactory, indicating that at least one receiver received the transmitted CAN message correctly.

→ if the sender does not receive any positive ACK from the receivers, it signifies an ack error. this indicates that none of the receivers received the CAN message correctly.

Logical Error Handling

→ error handling mechanism is responsible for:-

1- identifying and rejecting faulty (erroneous) messages, allowing the sender to re-transmit the message

2- identifying CAN nodes that consistently transmit erroneous messages.

→ for reasons of network-wide data consistency, if the error detecting node detects a local disturbance, it must inform all connected CAN nodes for network-wide data consistency.

→ to fulfill this purpose, the error-detecting CAN node transmits an error signal known as Error flag (error frame)

→ the error flag consists of six dominant bits, which violates the bit stuffing rule and results in a "Bit stuffing error"

→ this error is visible to all CAN nodes on the network

→ thus, the raising of error flags can be seen as a way of "globalizing" the detection of an error, ensuring that every CAN node is informed.

→ the error frame is used to indicate errors detected during communication.

→ when a CAN node detects an error during the transmission of a CAN message, it terminates the ongoing erroneous data transmission and issues an error frame.

→ upon detecting an error, the CAN node immediately transmits a sequence of 6-bits of the same logic level, known as raising an Active Error Flag

→ the error frame layout consists of just two parts:
- the error flag
- the error delimiter

→ When the error flag is transmitted, all other CAN nodes also send an error flag "secondary error flag", leading to the termination of regular data transmission

→ note that the other CAN nodes will see the active error flag as a Bit stuffing error

→ in response they also raise an active error flag.

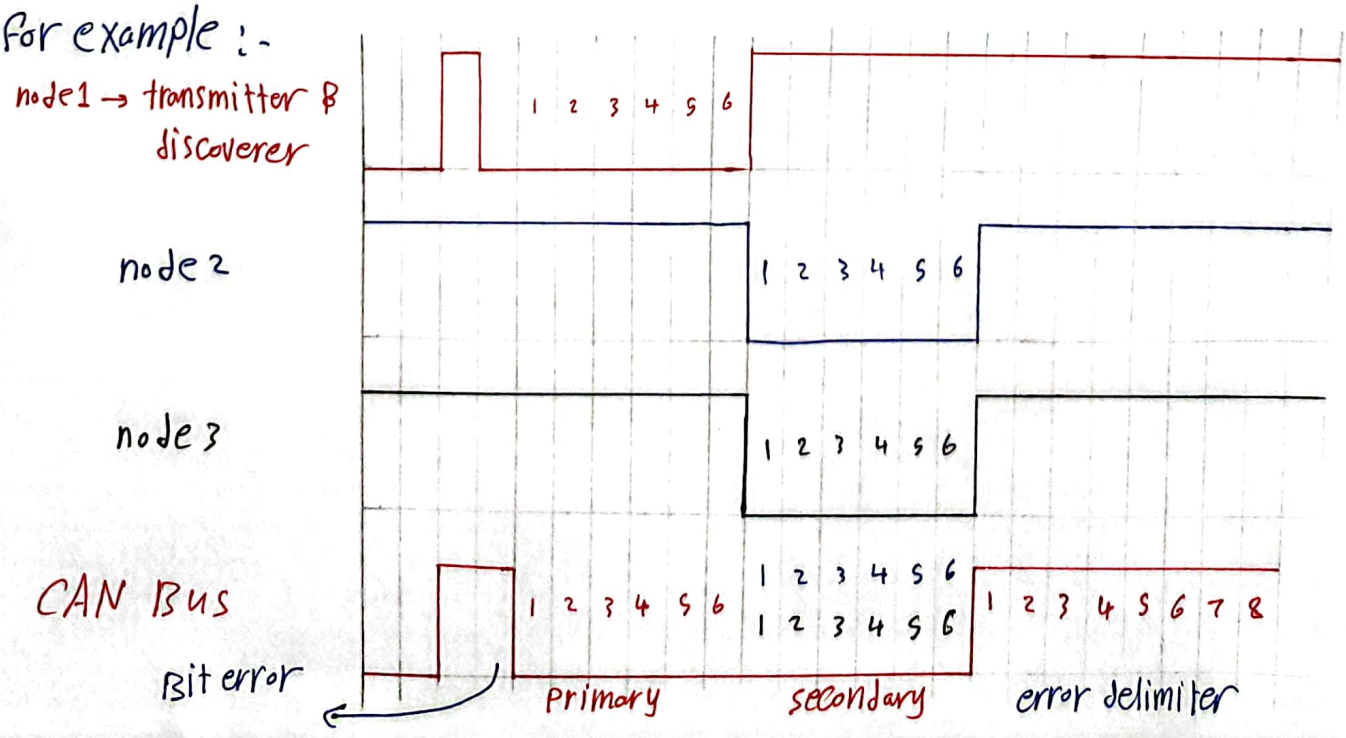
→ Transmission of an error flag always concludes with an error delimiter consisting of eight recessive bits.

→ the error delimiter replaces the ACK delimiter and the EOF of a regular message transmission. resulting in eleven recessive bits during the obligatory (ولجباري) transmission pause (ITM (intermission)) on the CAN bus.
→ the eleven recessive bits → bus idle identifier.

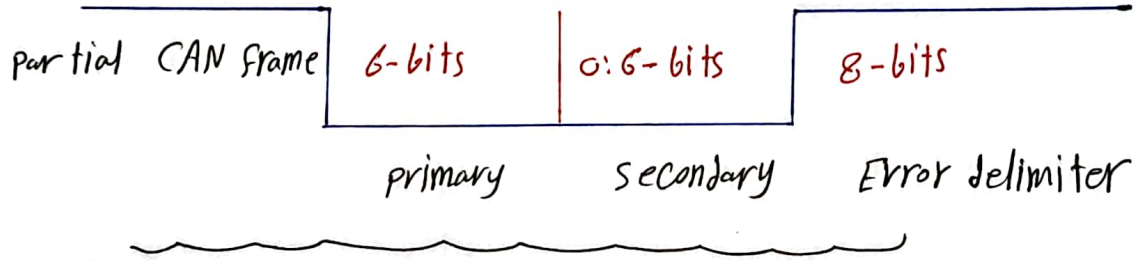
→ the initial error flag sent by the discovering node is called the primary active error flag. the error flags sent by the following reacting nodes are known as the secondary active error flags.

- all CAN nodes simultaneously discover an error in a CAN message and raise their error flags at the same time after primary active error flag.
- the error flags overlap, resulting in a total sequence of dominant bits lasting for 12 bits

• for example :-



→ CAN node1: transmits a dominant bit but reads it as recessive, discovering a bit error
→ it immediately transmits a sequence of 6-dominant bits
→ other nodes discover the bit stuffing error after reading the full 6-bits and raise their error flags.
→ this result in a subsequent sequence of 6 dominant bits, totaling 12 bits
→ depending on the situation the primary and secondary error flags might overlap.
→ the result is that the dominant bit sequence from raised error flags may be 6-to-12 bits long
→ this sequence is always terminated by a sequence of 8 recessive bit, marking the end of the error frame.



* How does CAN error handling work?
→ Error handling is an integral part (built-in) of the CAN standard and is implemented in every CAN controller.
→ every CAN node handles fault identification and confinement identically.
→ if the transmission is successful without errors, the message is sent
→ However, what if errors are caused by a systematic malfunction in a transmitting node?
- this could lead to an endless loop of sending and destroying the same message, effectively jamming (تشويش) the CAN bus.
- to address this, CAN node states and error counters plays a crucial role.

→ When a CAN node restarts, it enters the error active state.

* CAN node states :-

1- Error Active: "6 consecutive dominant bits"
   → this is the default state of every CAN node.
   → in this state, the node can transmit data and raise "active error flag" when errors are detected.

2- Error Passive: "6 consecutive recessive bits"
   → in this state, the CAN node is still able to transmit data, but it now raises "passive error flag" when errors are detected.

3- Bus-off :
   → in this state, the CAN node disconnects itself from the CAN Bus and can no longer transmit data or raise error flags.

* Error Counters:
   → every CAN node keeps track of its own state and adjusts its behavior based on the value of its error counter
   → there are two types of error counters:
      1- Transmit Error Counter (TEC): this counter tracks the number of transmit errors encountered by the CAN node.
      2- Receive Error Counter (REC): this counter tracks the number of receive errors encountered by the CAN node
   → By monitoring these error counters, each CAN node determine its current state and respond accordingly.

→ the CAN node is in error active mode if:
(TEC ≤ 127 OR REC ≤ 127)

→ in passive mode if: (TEC > 127 or REC > 127) & TEC ≤ 255

→ in Bus-off if : TEC > 255

→ once the CAN controller has entered bus-off state. it must be reset by the host microcontroller in order to be able to continue operation. in addition, this is only allowed after the reception of 128 occurrences of 11 consecutive recessive bits.

* the counters are updated as follows:-

1- Receiver:-

→ if a receiver detects an error, the Receiver Error Counter (REC) will be increased by 1

→ if a receiver detects a Bit Error while sending an active error flag or an overload flag the REC is increased by 8

→ if a receiver detects a dominant bit as the first bit after sending an error flag the REC will be increased by 8

→ after successfully receiving a message, the REC decreases by 1 if it between 1 and 127.

→ if the REC was '0', it stays '0', and if it was greater than 127, it is set to a value between 119 and 127.

2-Transmitter:-

→ When a transmitter sends an Error Flag the Transmit Error Count (TEC) is increased by 8.

→ Exceptions:

- Exception 1: if the transmitter is in an Error-Passive state and detects an Acknowledgment error because it fails to detect a dominant ACK bit, and at the same time it doesn't detect a dominant bit while sending its passive error flag because of detecting an ACK error, then the TEC will not be increased.
- Exception 2: if the transmitter sends an error flag because a stuff error occurred during arbitration whereby the stuff bit is located before the RTR bit, and should have been recessive, and has been sent as recessive but monitored as dominant. then the TEC will not be increased.

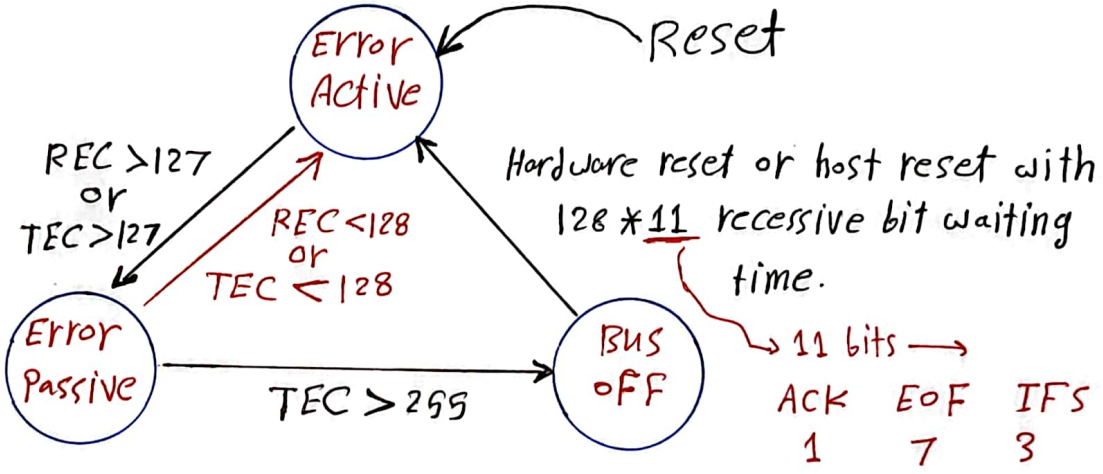
→ if a transmitter detects a bit error while sending an error flag, the TEC increases by 8

→ any node accepts up to 7 consecutive dominant bit after sending an active or passive error flag or an overload flag. After detecting the 14th consecutive dominant bit (in the case of an active error flag or an overload flag), or after detecting the 8th

consecutive dominant bits following a passive error flag, and after each sequence of additional 8 consecutive dominant bits every transmitter increases its TEC by 8 and every receiver increases its REC by 8.

→ After successfully transmission of a message, the TEC decreases by '1', unless it was already '0'.

→ if a node is the only one on the bus and it transmits a message, it may receive an acknowledgement error. this can lead to the node entering an error passive mode but not becoming bus off (due to EXception 1).



overload Frame and Inter Frame space

→ the overload frame is sent by a busy node to request extra delay between two data or remote frames, meaning that the over load frame can only occur between data or remote frame transmissions.

→ the overload frame consists of two fields:

1- the overload flag, which consists of 6 consecutive dominant bits.

2- the overload delimiter, which consists of 8 consecutive recessive bits.

→ the over-load frame is identical to the active error frame. the difference is that:

- the overload frame will not cause a retransmission of the last destroyed message and doesn't increase the error counter
- the over load frame only occurs at the distance between consecutive frames which is the (ITM/IFS), a minimum of 3 recessive bit times (ITM: inter frame space)

* Inter Frame space

→ this is a form of separation between 2 frames for data and remote frames. there are two types of interframe spaces:-

1- the normal interframe space, which consists of intermission (ITM) and Bus idle. intermission has 3 recessive bits and bus idle is also recessive bits that signal that nodes can compete for the bus now

2- for error passive nodes after the intermission field, 8- recessive bits of suspend transmission is sent before Bus idle is sent.

→ there is no interframe space between Error frames and overload frames.

→ the IFS is also used to let the controller move a correctly received frame to the message buffer area.