AUTOMOTIVE PROTOCOL BUSES

CAN & LIN

AMIT

CONTROLLER AREA NETWORK

Controller Area Network:

Introduction To CAN:

- A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer.
- It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles to save on copper, but it can also be used in many other contexts. For each device, the data in a frame is transmitted sequentially but in such a way that if more than one device transmits at the same time, the highest priority device can continue while the others back off. Frames are received by all devices, including by the transmitting device.

AMIT

Controller Area Network:

Introduction To CAN:

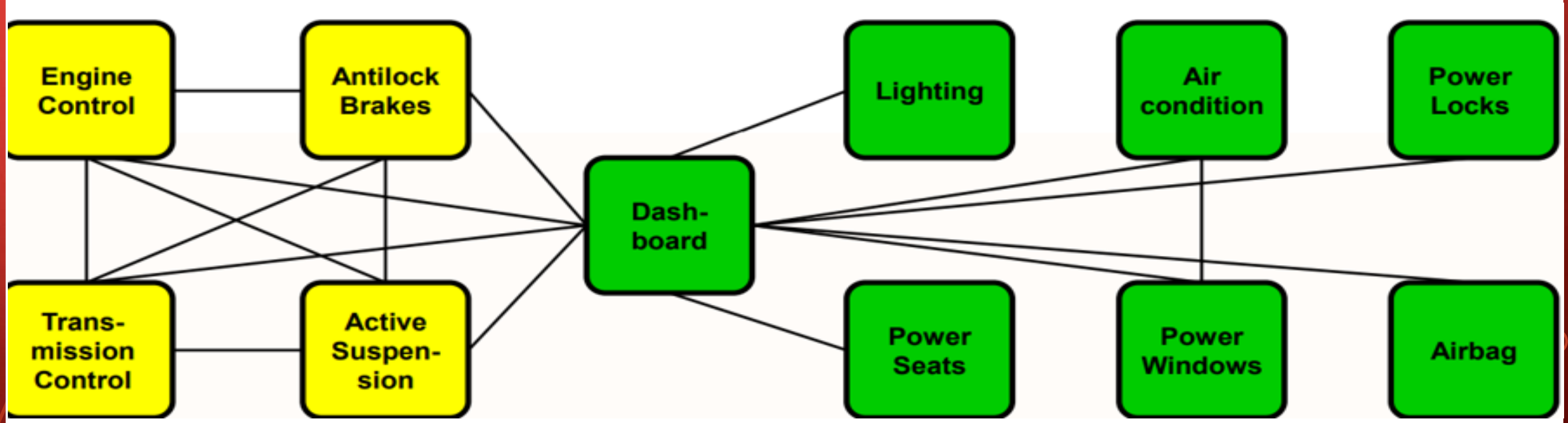
- Before CAN:
  - Before CAN was introduced in the automobiles, each electronic device was connected to other device using the wires (point to point wiring) which worked fine, when the functions in the system were limited.
  - Discrete interconnection of different systems (point to point wiring). Network cable with a length of up to several miles and many connectors was required!!
  - One of the major problems for automotive engineers was linking the ECUS of the different devices so that real time information can be exchanged.
- After widely using the microcontrollers inside cars, Bosch company needed to used a protocol to communicate between of them, but UART is a long distance but it is peer to peer, SPI is SMMS but it is a very short distance, IIC is MMMS but it is a short distance, so The Controller Area Network (CAN) is a serial bus communications protocol and Introduced by Robert Bosch in 1986 Developed for automotive applications.
- CAN is internationally standardized by ISO and SAE in 1986.

AMIT

Controller Area Network:

Description:

- Before CAN:



AMIT

Controller Area Network:

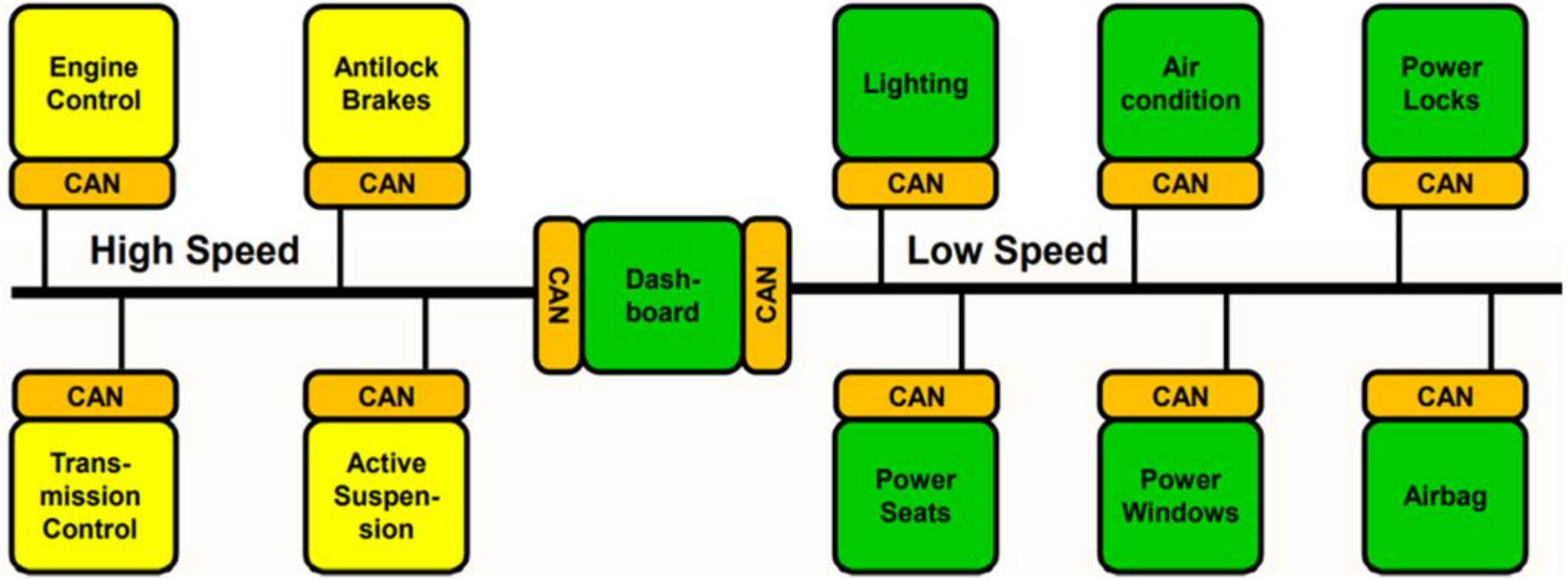
Description:

- CAN is a message oriented:
  - **UART** is a peer to peer protocol, so to start a protocol between the nodes such start the frame because there is no nodes into the network except the two nodes of UART.
  SPI is a SMMS protocol, and it can select the communicated slave by pull the SS pin of the slave to LOW.
  - **IIC** is address oriented, so just write the slave address on the data bus and the owned salve address will generate an acknowledge. The IIC can just communicate with one slave per time.
  - **Message oriented** means that the node will write a message on the data bus and any node exists into the network that is interested in this message can take its data at the same time, but IIC must send the data individually to every node using a repeated start condition and the data will be received sequentially not at the same time.

AMIT

Controller Area Network:

- After CAN:
  CAN Network Speed:
  - LOW Speed CAN: baud rates from 40Kbtis/s to 125Kbits/sec
  - High Speed CAN: baud rates from 40Kbtis/s to 1Mbits/sec, depending on Cable length.



Controller Area Network:

CAN Specification :

- Three Types of CAN Modules :
  - CAN 2.0A
    - 11-bit Identifier.
    - 1 Mbps.
    - Considers 29 bit ID as an error.
  - CAN 2.0B Passive.
    - Ignores 29 bit ID.
  - CAN 2.0B Active.
    - 125 kbps
    - Handles both 11 and 29 bit ID.

	Frame with 11 bit ID	Frame with 29 bit ID
V2.0B Active CAN	Tx/Rx OK	Tx/Rx OK
V2.0B Passive CAN	Tx/Rx OK	Tolerated
V2.0A CAN	Tx/Rx OK	Bus ERROR

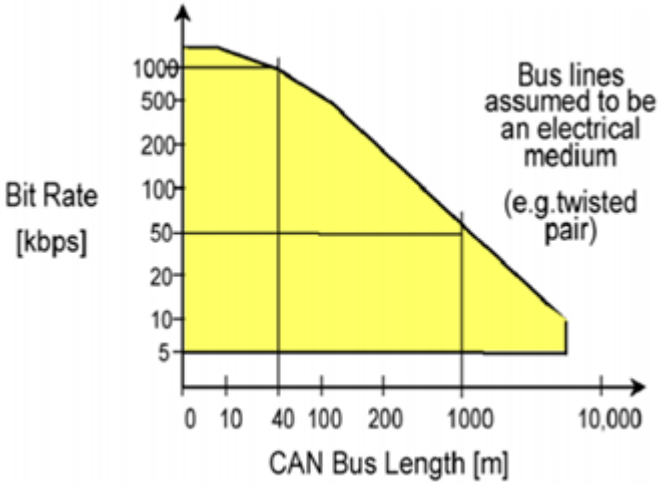
AMIT

Controller Area Network:

CAN Characteristic:

- The speed of CAN Bus differs according to the length of Bus, like:

Bit Rate	Bus Length
1M bit/sec	40 meters (131 feet)
500K bit/sec	100 meters (328 feet)
250K bit/sec	200 meters (656 feet)
125K bit/sec	500 meters (1640 feet)



Controller Area Network:

CAN Specification :

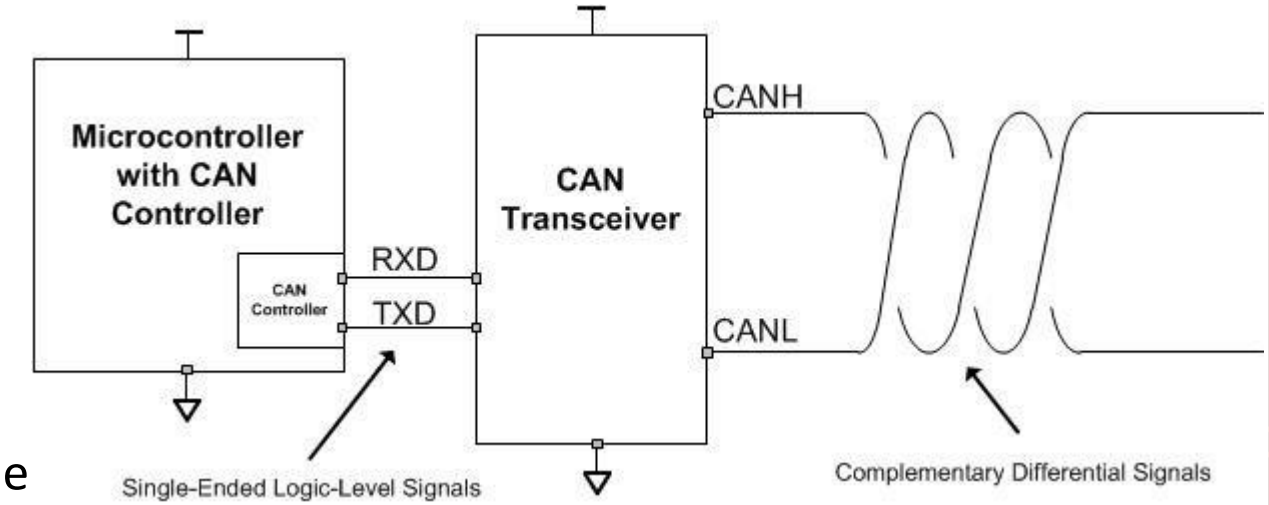
- General Specs of CAN Bus:
  - Wired.
  - Serial.
  - Half Duplex.
  - Asynchronous.
  - Multi-Master.
  - Open-Drain based.

AMIT

Controller Area Network:

CAN Characteristic:

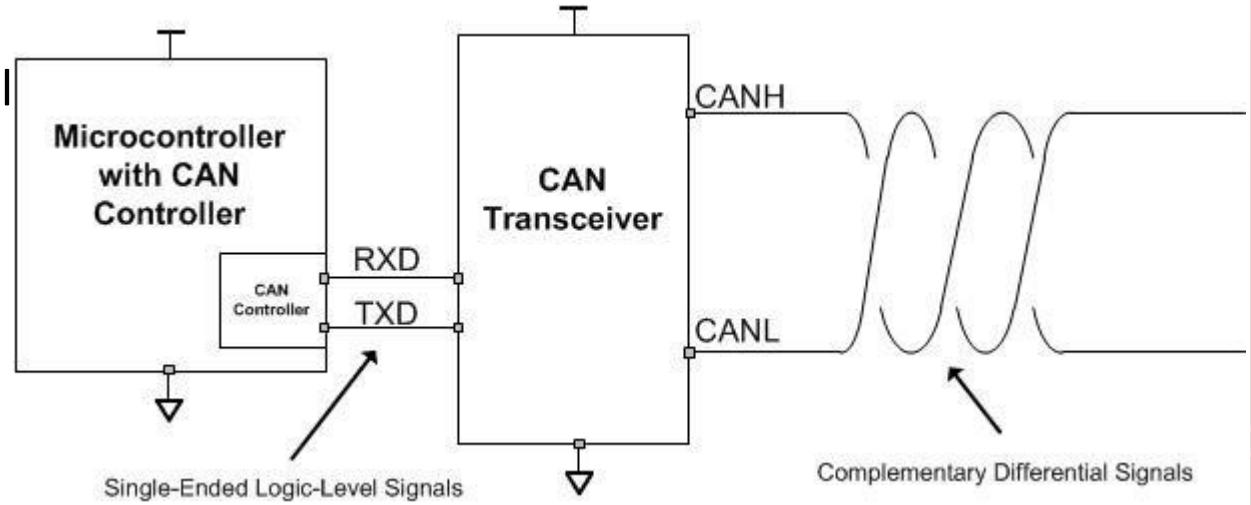
- CAN Protocol is a differential Protocol:
  It is the most important merit in this protocol, because CAN sends its data on two differential lines, Positive and negative lines “High line and Low Line”.
  These lines are connected to Pull-up Resistor.
  High line is always pulled to 12 volt.
  When Zero is sent, the low line will be pulled to 12 volt, so the differential between High line and Low line will be 0 volt and it will be considered as “**ZERO**”, and when one is sent, the low line will be pulled to 0 volt, so the differential between High line and Low line will be 12 volt and it will be considered as “**ONE**”.



Controller Area Network:

CAN Characteristic:

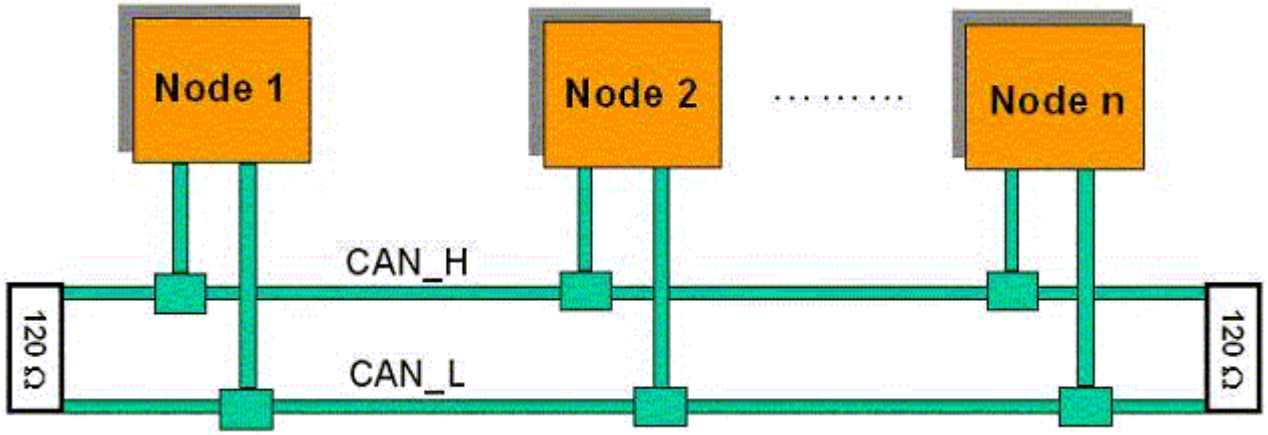
- CAN Protocol is a differential Protocol:
  The lines of can are twisted together so the noise on the bus will affect on the lines by same value of noise on the two lines, and because it is a differential protocol, the differential circuit will cancel any noise will affect on the bus because the data will be taken after substruction of High line from Low line.
  The High line is connected to 12v not 5 because the noise will reduced because of high power, and the power of several Microcontrollers have a different output levels like “3.3v, 5v and so on”, so using 12v will unify the level of lines.



Controller Area Network:

CAN Characteristic:

- CAN is an Open-Drain Protocol:
  1- Dominant bit is Zero like IIC protocol.
  2- Recessive bit is One like IIC protocol.
  3- Bus is high by default.
  4- It has a multi-master Arbitration

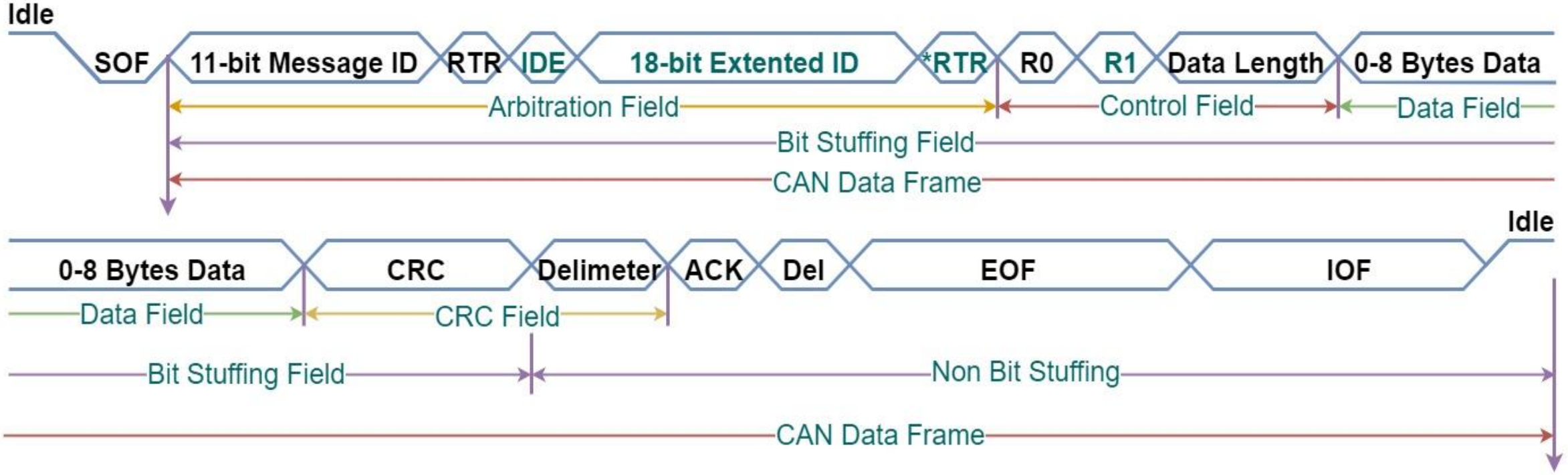


AMIT

Controller Area Network:

CAN Data Frame:

- CAN Frame Schematic:



Controller Area Network:

CAN Data Frame:

- CAN Frame:
  - Start Of Frame "SOF":
    This is a low bit to catch the bus.
  - Message ID:
    This is the ID of Message which the ECU wants to write or request its data.
    The message ID has only one owner, but it can have several listeners.
    Every CAN ECU has a Message-ID Table which contains the messages that owns and the messages to which listens, like the following example:

Message ID	Status
0x00	Owner
0x01	Listener
0x03	Listener
0x09	Owner
0x0A	Listener

Message ID	Status
0x00	Listener
0x01	Owner
0x03	Listener
0x09	Listener
0x0A	Listener

Message ID	Status
0x00	Listener
0x01	Listener
0x03	Owner
0x09	Listener
0x0A	Owner

Controller Area Network:

CAN Data Frame:

- CAN Frame:
  - Message ID:
    As the example shows, there is only one Owner for any message ID, the message represent the type of data, as an example: 0x00 is the message ID of the temperature of exhaust, so the ECU of this sensor owns this message, and the Throttle, Injection, Cooling system ECUs are interested in this into, so these ECUs will be listeners to 0x00 message ID, but as an example the ECUs of lighting system, car wiper system or car doors system are not interested in the temperature of exhaust.
    The Size of Message ID is **11 bits**.
  - Remote Transmission Request:
    The **'RTR'** This is a bit that indicates the type of frame "also it is called a Remote Request Substitution '**RRS**', so it will be written to **ZERO** to indicate that the owner will write its data, and it is called a Remote Transmission Request '**RTR**', so it will be written to **ONE** to indicate that the a listener of this ID wants the ID's Info".

Controller Area Network:

CAN Data Frame:

- CAN Frame:
  - ID Extension:
    It is one bit to make an ID extension “it is an optional bit if the CAN version supports the ID Extension”, which indicates that there is an extension of message ID.
  - Message ID Extended:
    They are additional 18 bits to message ID if the normal message ID is not enough, they are written if the IDE bit is written to ONE.
  - Remote Transmission Request:
    This bit is written again if the message ID has been extended.
  - Reserved Bit 0:
    It is a reserved bit to delay the bus before writing data.
  - Reserved Bit 1:
    It is a reserved bit to delay the bus and it is written only if the message ID has been extended.

Controller Area Network:

CAN Data Frame:

- CAN Frame:
  - Data Length:
    They are four bits to indicate how many bytes that will be transferred into this frame.
    Eight bytes can be transmitted into the same frame as a maximum to avoid bus starvation like IIC, so the data into can frame is limited to Eight bytes not infinite like IIC.
  - Cyclic Redundancy Check:
    They are fifteen bit for **CRC** that is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data, which is used to check on the transmitted data.
  - Cyclic Redundancy Check Delimiter:
    They are two bits to allow the receiver to calculate its own CRC and comparing it with the received CRC.

Controller Area Network:

CAN Data Frame:

- CAN Frame:
  - Acknowledge:
    It is a bit which is written by receivers to indicate that the data has been received successfully.
  - Acknowledge Delimiter:
    It is a bit to allow the transmitter to catch the bus again.
  - End of Frame:
    They are six high bits to indicate that the frame will be ended.
    Also, at their location, the error bits are written to indicate that an error has been occurred.
  - Inter Frame Space:
    They are three high bits to prevent any node into the CAN network to initiate any communication frame at this time.

Controller Area Network:

Types of CAN Frame:

- Data Frame:
  It is the frame that the owner of message ID writes its own data, there are two types of this frame:
  - Normal Frame: 11-bit Message ID.
  - Extended Frame: 29-bit Message ID.

- Request Frame:
  It is the frame that any listener of message ID can request the data of owner, there are also two types of this frame
  - Normal Frame: 11-bit Message ID.
  - Extended Frame: 29-bit Message ID.

Controller Area Network:

Types of CAN Frame:

- Error Frame:
  This frame has error sequence bit into the location of End of Frame bits, it indicates to an error has been occurred, there are two types of this frame:
  - Active Error Frame: it has six low bits at the location of EOF bits.
  - Passive Error Frame: it has six high bits at the location of EOF bits.
    To understand what are the meaning of active and passive, first we must mentioned that every CAN node into the network has two counter:
    - Transmission Error Counter "**TEC**".
    - Reception Error Counter "**REC**".

    If the receiver node generates an error on any frame, its REC will be increased by one, and the TEC of transmitter also.
    If the value of TEC or REC is lower than '127', this node will be in the active state that can generate an Active Error Frame, but if their values become higher than '127' and lower than '256', this node will be in the passive that can generate a Passive Error Frame and it will be a receiver only.

Controller Area Network:

Types of CAN Frame:

- Error Frame:
  - Active & Passive Error Frame:

    If the value of TEC or REC exceeds ‘256’, this node will be in the bus-off state that can never work again until resetting and maintenance the network.

    It is possible to return to the active state if the node was into the passive state, because every successful transmission or reception the counter will be decreased by one, but it is impossible to leave the bus-off state until resetting.

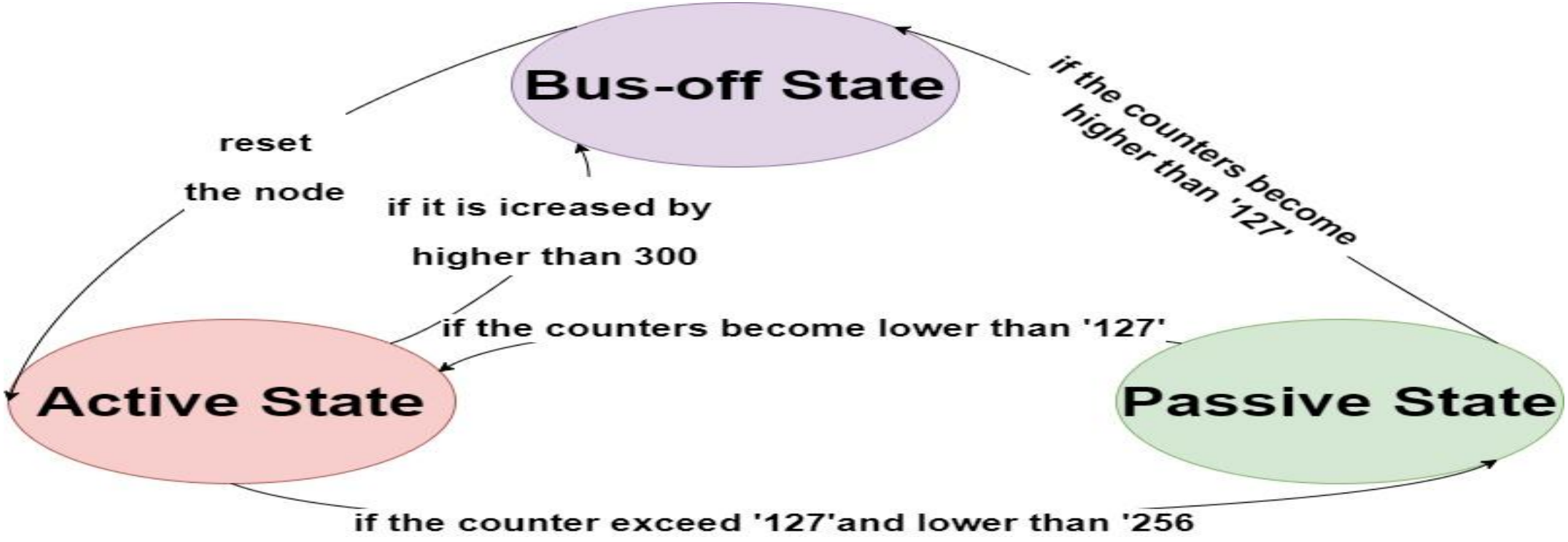
    In critical cases, the TEC and REC can be increased by any values as a long step, like increasing them by ‘20’ or ‘100’ or ‘10’ or any value according to the case, so as an example, if the counters are increased by ‘100’, it will be in the bus-off state an the third time when they become ‘300’.

Controller Area Network:

Types of CAN Frame:

- Error Frame:
  - Active & Passive Error Frame:

    Finite State Machine of states of Error frame types:



Controller Area Network:

Types of CAN Frame:

- Error Frame:
  - Overload Frame:
    it is a frame to indicate that a problem occurred on data while transmitting, it is six low bits that are written into the location of “IOF” bits.

    If a CAN node receives messages faster than it can process them, then an Overload Frame will be generated to provide extra time between successive Data or Remote frames. Similar to an Error Frame, the Overload Frame has two fields: an overload flag consisting of six dominant bits, and an overload delimiter consisting of eight recessive bits. Unlike error frames, error counters are not incremented.

Controller Area Network:

Bit Stuffing of CAN Frame:

- Bit Stuffing:
  It is a technique to avoid writing similar six bits through the whole frame except EOF or IOF.
  Bit Stuffing is that writing an inverted logic after similar five bits “means at the sixth bit” by hardware and clearing the sixth bit after similar five bits.
  We will declare it by examples.
  - The transmitted data such example 11-bit ID is “0b000 0000 1000”:
    it will be stuffed from the transmitter, like: “0b000 00**1**00 1000”, and the receiver automatically remove the next bit that come after five similar bits.

  - The transmitted data such example 11-bit ID is “0b111 1101 1010”:
    it will be stuffed from the transmitter, like: “0b111 11**0**01 1010”, and the receiver automatically remove the next bit that come after five similar bits.

LOCAL INTERCONNECT NETWORK

Local Interconnection Network:

LIN History:

- Lin stands for Local Interconnect Network.
- In 1996, Volvo and Volcano Communication Technologies (VCT) developed a UART based protocol for the Volvo S80 series, called Volcano Lite.
- This protocol was an integral part of the vehicle communication system as cost-effective alternate to CAN protocol.
- In 1997, Motorola joined Volvo and VCT in improving the Volcano Lite protocol to meet various requirements, such as self-synchronization of the slave without a need of crystal, and form an open standard that can support a wide range of semiconductor products.
- In December 1998 the car companies Audi, BMW, DaimlerChrysler and VW joined the activities and a workgroup was formed to set up the LIN communication protocol.

Local Interconnection Network:

LIN History:

- Begin of Q3/1999 the LIN Protocol Spec. and the LIN Configuration Language Spec. have been released (Revision 1.0).

- In November 2002 LIN 1.3 was released with changes mainly made in the physical layer., The latest version LIN 2.1 was released in 2006. With LIN 2.1 came some major changes and also some new features like diagnostics. The changes were mainly aimed at simplifying use of off-the-shelves slave nodes.

Local Interconnection Network:

LIN Specs:

- The LIN is a SCI/UART-based serial.
- The LIN is a single wire 12V bus connection.
- LIN Communication is a half-duplex protocols.

- Single master / multi slave concept.
- No needs for arbitration concept or circuit.
- LIN is also a broadcast type serial network.

- The number of Nodes is limited up to 16 slaves.
- Low baud rate, Low speed application.
- The maximum transmission speed is 20kps.

- Data checksum and error detection.
- Variable length of data frame (2, 4 and 8 byte).

Local Interconnection Network:

LIN Specs:

- An important feature of LIN is the self synchronization mechanism that allows the clock recovery by slave nodes without quartz or ceramics resonator. Only the master node will be using the oscillating device.
- LIN is a software protocol that needs a RTOS on all node into the network.
- No Node addressing, Message ID specifies contents and priority.
- Nodes can be added to the LIN network without requiring hardware or software changes in other slave nodes.

- No collision detection exists in LIN, therefore all messages are initiated by the master with at most one slave replying for a given message identifier.
- The main features of this protocol (compared to CAN) are low cost and low speed and used for short distance networks. Max 40m wire length.
- The LIN protocol is byte oriented according to the UART protocol, which means that data is sent one byte at a time. One byte field contains a start bit (dominant), 8 data bits and a stop bit (recessive), the data bits are sent LSB first.

Local Interconnection Network:

Applications on LIN:

- The master is typically a moderately powerful microcontroller, whereas the slaves can be less powerful, cheaper microcontrollers dedicated ASICS or sensors or actuators:

  - The LIN bus is having many applications in the automotive vehicles are
    - Steering wheel:
      - Cruise control
      - Radio
      - Mobile phone Wiper
      - Lights
    - Seats:
      - Seat position control.
      - Occupancy sensor.
      - Heating control (if installed).

Local Interconnection Network:

Applications on LIN:

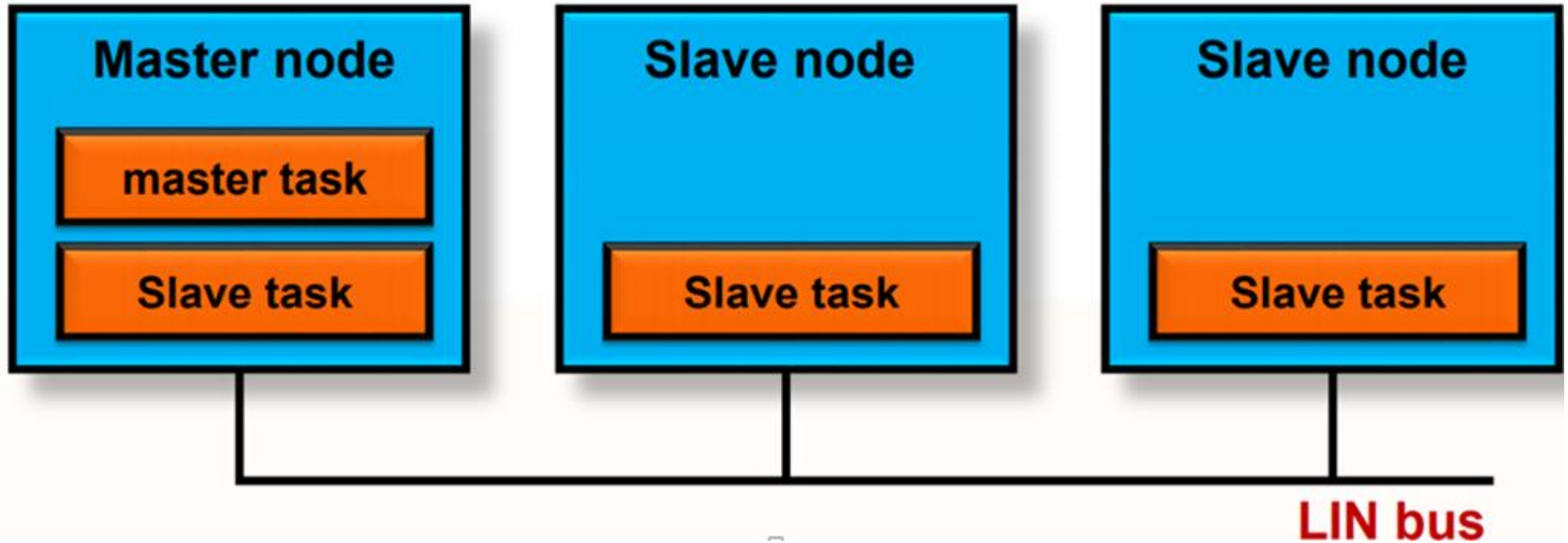
- The master is typically a moderately powerful microcontroller, whereas the slaves can be less powerful, cheaper microcontrollers dedicated ASICS or sensors or actuators:

  - The LIN bus is having many applications in the automotive vehicles are
    - Roof area:
      - Rain sensor - possibly interrogated every 10 – 20 ms.
      - Light sensor.
      - Light control Sunroof.
    - Door / window:
      - Mirror.
      - Window control.
      - Door lock.

Local Interconnection Network:

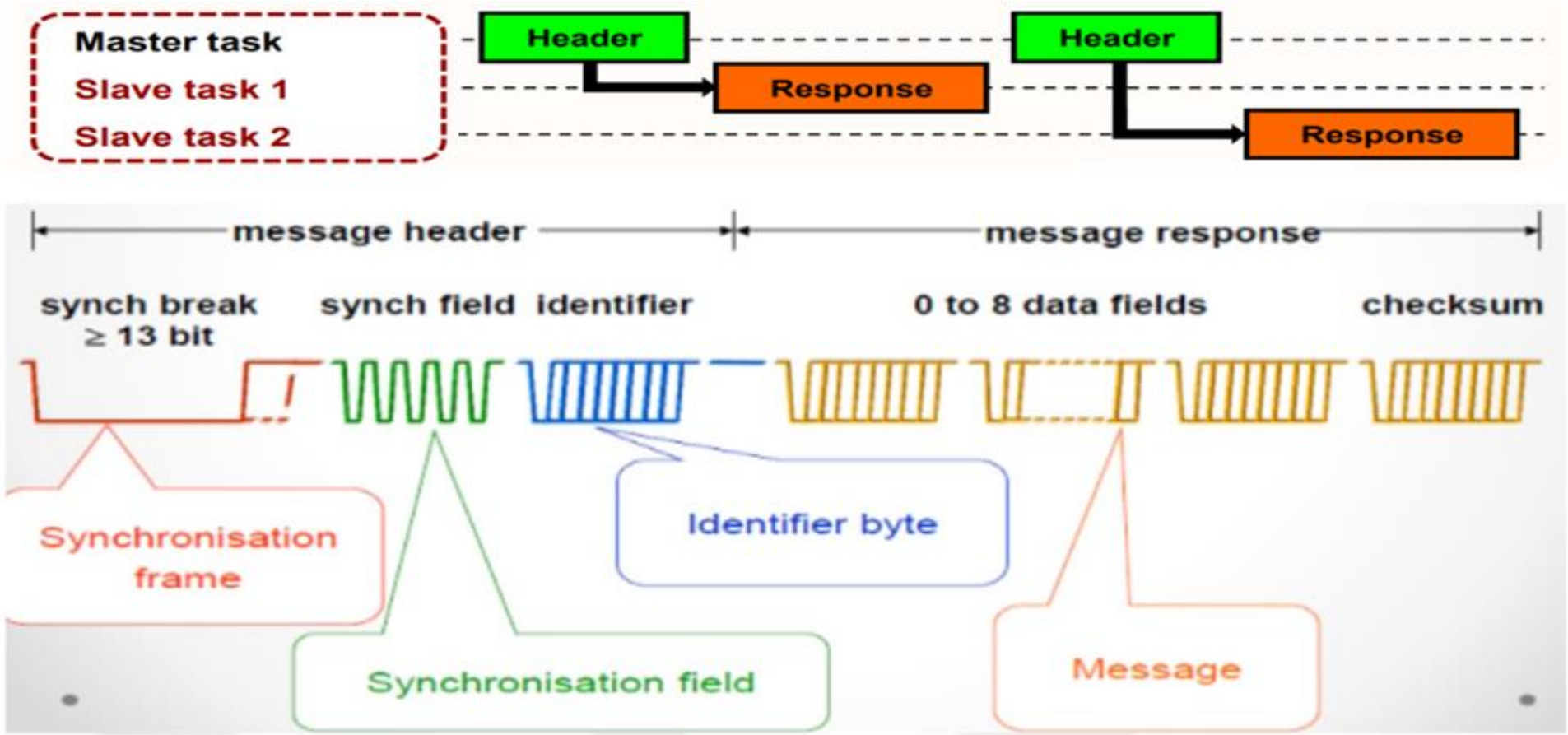
LIN Hardware Connection:

- The LIN is an open-drain protocol, so its bus will be pulled up to high.
- Every node into the network has a "**Slave Task**", but only the master node has the "**Master Task**"



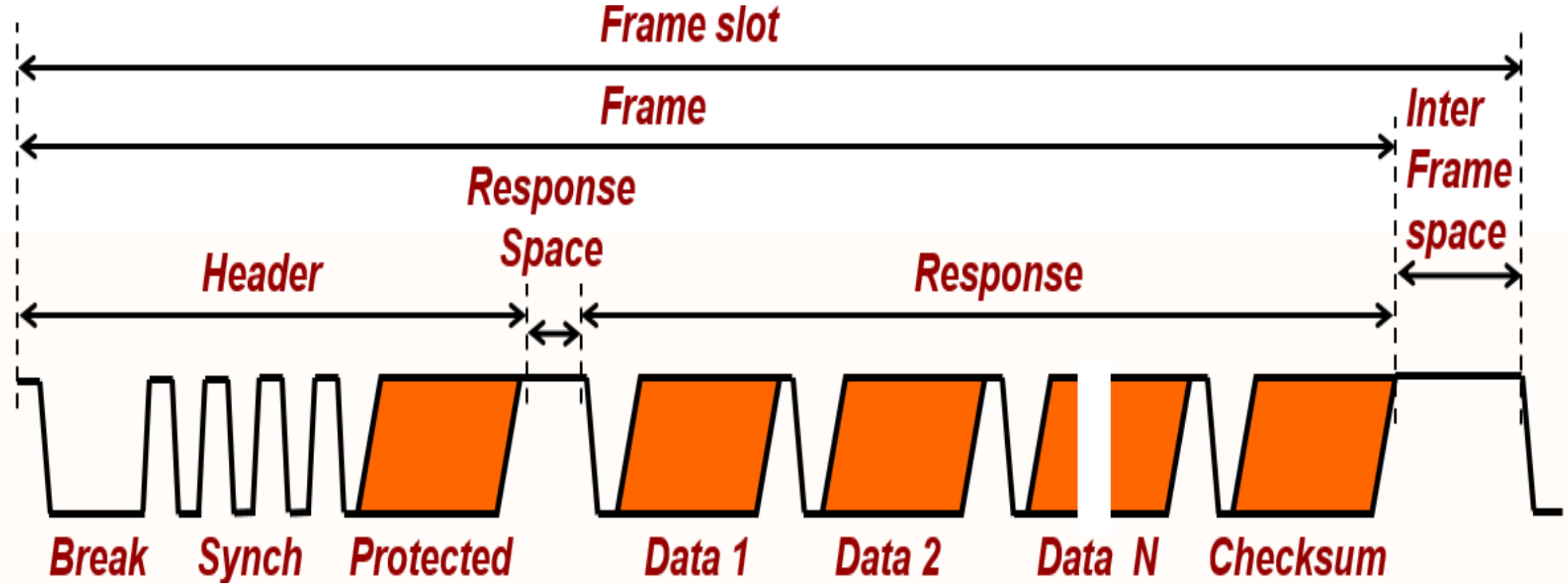
Local Interconnection Network:

LIN Data Frame:



Local Interconnection Network:

LIN Data Frame:



Local Interconnection Network:

LIN Data Frame:

- The basic unit of transfer on the LIN bus is the frame is divided in to a header and response:
  - The header frame:
    - Is always transmitted by master node.
    - It consists of three distinct fields: the break, the synchronization (sync), and identifier (ID).
  - The response frame:
    - Which is transmitted by a slave task .
    - It can reside in either the master node or a slave node It consists of a data payload and a checksum.

Local Interconnection Network:

LIN Data Frame:

- Header Frame consists of:
  - Break Field:
    - The break symbol is used to signal the beginning of a new frame.
    - A break is always generated by the master task and it shall be 13-18 bits of dominant value, including the start bit, followed by a break delimiter.
    - Synch break ends with a “break delimiter" which should be at least one recessive bit.

  - Synch Byte Field
    - Synch is a byte field with the data value 0x55.
    - A slave task shall always be able to detect the break/synch symbol sequence.
    - Synch byte is sent to decide the time between two falling edges and thereby determine the transmission rate.
    - The bit pattern is 0x55 (01010101, max number of edges).

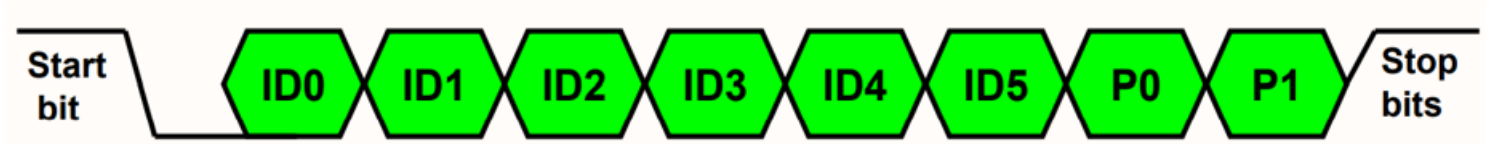
Local Interconnection Network:

LIN Data Frame:

- Header Frame consists of:
  - Protected Identifier:
    - The ID field is the final field transmitted by the master task in the header.
    - This field provides identification for each message on the network and ultimately determines which nodes in the network receive or respond to each transmission.
    - All slave tasks continually listen for ID fields, verify their parities, and determine if they are publishers or subscribers for this particular identifier.
    - Protected Identifier: the Identifier Field does not indicate the destination of the message but describes the contents of the message frame, Content Identifier "**6 bits**" and parity "**2 bits**"
      P1: P1 = ((b1 ^ b3) ^ (b4 ^ b5)).
      P0: P0 = ((b0 ^ b1) ^ (b2 ^ b4)).

Local Interconnection Network:

LIN Data Frame:



- Header Frame consists of:
  - Protected Identifier:
    - Contains information about sender and receiver and the number of bytes which is expected in the response.
    - Six bits are reserved for the identifier (ID).
    - Values in the range 0 to 63 can be used. The identifiers are split in four categories:
      - Values 0 to 59 (0x3b) are used for signal carrying frames
        - 60 (0x3c) and 61 (0x3d) are used to carry diagnostic data.
      - 62 (0x3e) is reserved for user defined extensions.
        - 63 (0x3f) is reserved for future protocol enhancements.

ID range		Frame length
0-31	0x00 – 0x1f	2
32-47	0x20 – 0x2f	4
48-63	0x30 – 0x3f	8

Local Interconnection Network:

LIN Data Frame:

- Response Frame consists of:
  - Data:
    - A frame caries between one and eight bytes of data .
    - A data byte is transmitted in a byte field .
    - The data bytes field is transmitted by the slave task in the response.
    - Every byte starts by start bit, and ends stop bit.
  - Check sum:
    - The checksum field is transmitted by the slave task in the response.
    - The LIN bus defines the use of one of two checksum algorithms to calculate the value in the eight-bit checksum field:
    - Classic checksum is calculated by summing the data bytes alone (V 1.3) .
    - enhanced checksum is calculated by summing the data bytes and the protected ID. (V2.0). The enhanced checksum is used for identifiers 0 to 59.

THANK YOU!