ARM-Based Microcontrollers

Automotive Bus Technologies

By : Ahmed Magdy

6/8/2019

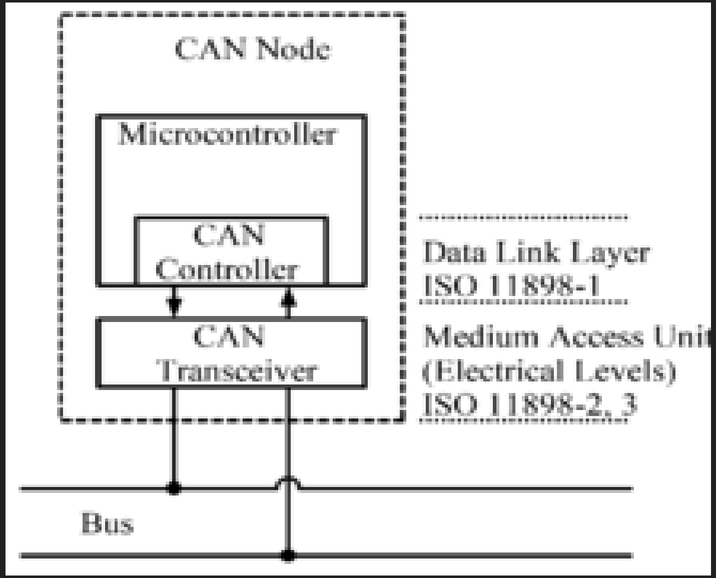
AGENDA

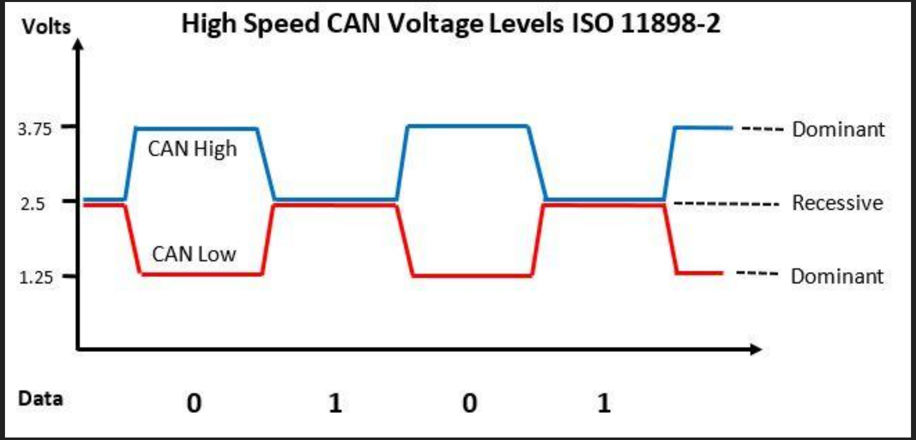
- CAN standards
- How does CAN work?
- CAN Message formats
- CAN Error detection and fault confinement
- Bit Timing
- CAN Driver in Tivaware Library
- CAN vs CAN-FD
- WHAT IS LIN BUS?
- LIN BUS VS CAN BUS
- HOW DOES LIN BUS WORK?

CAN standards

- The original specification is the Bosch specification. Version 2.0 of this specification is divided into two parts:
  - Standard CAN (Version 2.0A). Uses 11 bit identifiers.
  - Extended CAN (Version 2.0B). Uses 29 bit identifiers.

- CAN Hardware Node:
  1. **Central processing unit:** The host processor decides what the received messages mean and what messages it wants to transmit.
  2. **CAN controller:** Receiving: the CAN controller stores the received serial bits from the bus, Sending: the host processor sends the transmit message(s) to a CAN controller
  3. **Transceiver:** it converts the data stream from CANbus levels to levels that the CAN controller uses(Dominant Bit = 0 bit , Recessive= 1 bit)

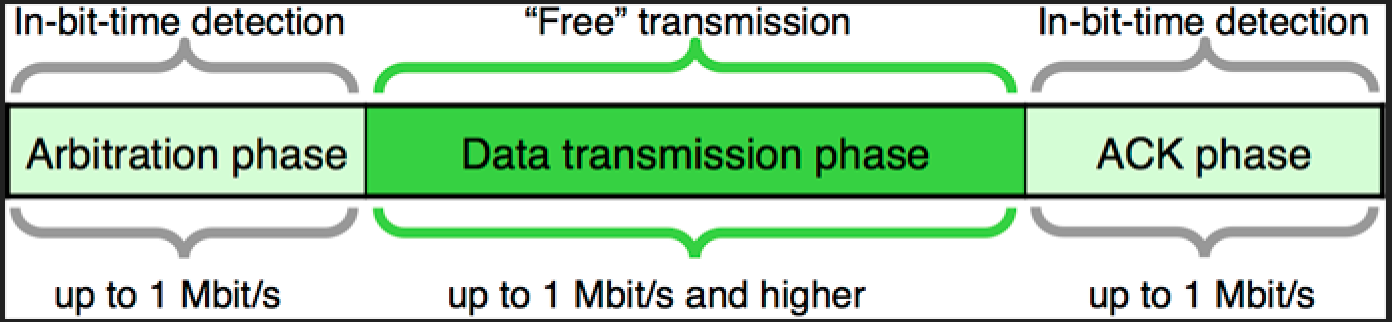

CAN Node
Microcontroller
CAN Controller
CAN Transceiver
Data Link Layer ISO 11898-1
Medium Access Unit (Electrical Levels) ISO 11898-2, 3
Bus



CAN standards

- **Bit Rates:**

  Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances (e.g.,500 m at 125 kbit/s).



How does CAN work?

- Data messages transmitted from any node on a CAN bus do not contain addresses of either the transmitting node, or of any intended receiving node.

- Instead, the content of the message is labelled by an identifier that is unique throughout the network. All other nodes on the network receive the message and each performs an acceptance test on the identifier to determine if the message, and thus its content, is relevant to that particular node.

- If the message is relevant, it will be processed; otherwise it is ignored.

How does CAN work?

- **Arbitration:**

  If two nodes want to send data at same time, who take the bus for himself and send? Who will dominate ?

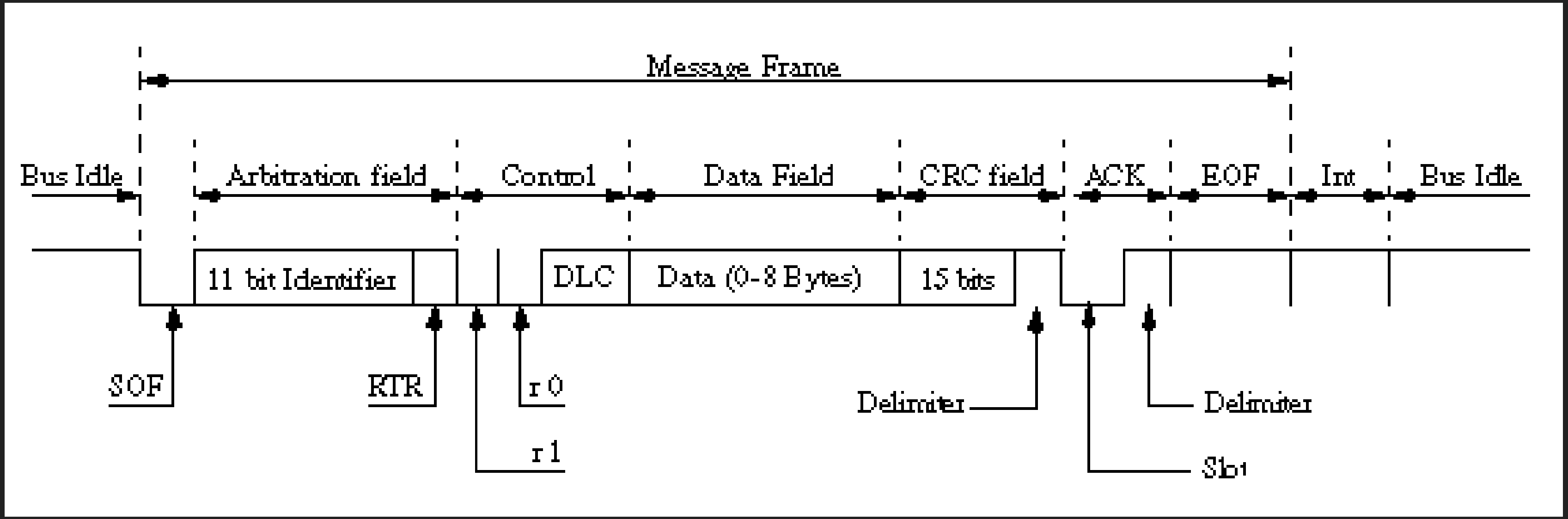
  Each CAN message has an identifier which is 11 bits (CAN specification part A) or 29 bits (part B). This identifier is the principle part of the CAN arbitration field, which is located in the beginning of each CAN message. The identifier identifies the type of message, but is also the message priority.

  The higher priority message is guaranteed to gain bus access as if it were the only message being transmitted. Lower priority messages are automatically re-transmitted in the next bus cycle, or in a subsequent bus cycle if there are still other, higher priority messages waiting to be sent.

  The message has more zeros in ID will dominate !! (Check the CAN Frame in the coming slides to know the ID)

CAN Message formats

- **1) 2.0A Format: (**11 bit message identifier)



CAN Message formats

- **1) 2.0A Format: (**11 bit message identifier)

  - - **A Start of Frame (SOF) field**. This is a dominant (logic 0) bit that indicates the beginning of a message frame.
  - - **An Arbitration field,** containing an 11 bit message identifier
  - **Remote Transmission Request (RTR) bit.** A dominant (logic 0), RTR bit indicates that the message is a Data Frame. A recessive (logic 1) value indicates that the message is a Remote Transmission Request (otherwise known as Remote Frame.) A Remote Frame is a request by one node for data from some other node on the bus. Remote Frames do not contain a Data Field.
  - ***A Control Field containing six bits:***

    two dominant bits (r0 and r1) that are reserved for future use, and

    a four bit Data Length Code (DLC). The DLC indicates the number of bytes in the Data Field that follows

    - A Data Field, containing from zero to eight bytes.

CAN Message formats

- **1) 2.0A Format: (**11 bit message identifier)
  - ***A Control Field containing six bits:***

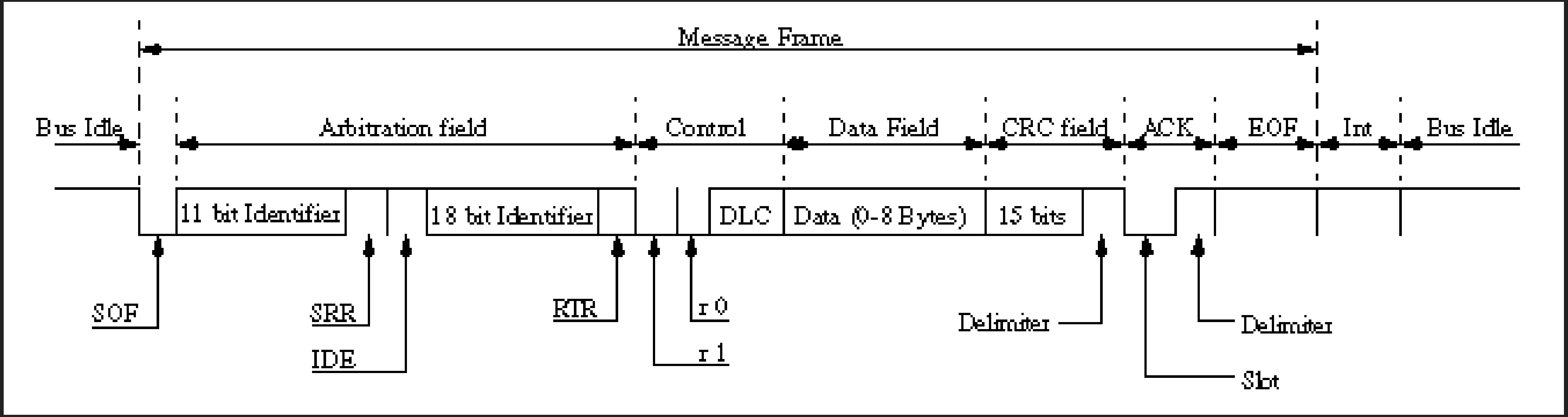
    - The CRC field, containing a fifteen bit cyclic redundancy check code and a recessive delimiter bit

    - The ACKnowledge field, consisting of two bits. The first is the Slot bit which is transmitted as a recessive bit, but is subsequently over written by dominant bits transmitted from all other nodes that successfully receive the message. The second bit is a recessive delimiter bit

    - The End of Frame field, consisting of seven recessive bits.

CAN Message formats

- **2) 2.0B Format:** (29 bit identifier)



CAN Message formats

- **2) 2.0B Format:** (29 bit identifier)
  - **The differences are:**
    - - In Version 2.0B the Arbitration field contains two identifier bit fields. The first (the base ID) is eleven (11) bits long for compatibility with Version 2.0A. The second field (the ID extension) is eighteen (18) bits long, to give a total length of twenty nine (29) bits.
    - - The distinction between the two formats is made using an Identifier Extension (IDE) bit.
    - - A Substitute Remote Request (SRR) bit is included in the Arbitration Field. The SRR bit is always transmitted as a recessive bit to ensure that, in the case of arbitration between a Standard Data Frame and an Extended Data Frame, the Standard Data Frame will always have priority if both messages have the same base (11 bit) identifier.
    - All other fields in a 2.0B Message Frame are identical to those in the Standard format.

CAN Error detection and fault confinement

- The built in error detection of the controllers together with the error signaling make sure that the information is correct and consistent:
  1. **The CAN error process**
  2. **The error is detected by the a CAN controller (a transmitter or a receiver).**
  3. **An error frame is immediately transmitted.**
  4. **The message is cancelled at all nodes (exceptions exist - see CAN controller error modes).**
  5. **The status of the CAN controllers are updated (see CAN controller error modes).**
  6. **The message is re-transmitted. If several controllers have messages to send, normal arbitration is used.**

CAN Error detection and fault confinement

- **Error detection:**
  1. Bit stuffing error - normally a transmitting node inserts a high after five consecutive low bits(and a low after five consecutive high).
  2. Bit error: If it detects a different bit value on the bus than it sent (i.e. Microcontroller send a bit and listen to the bus and found another bit)

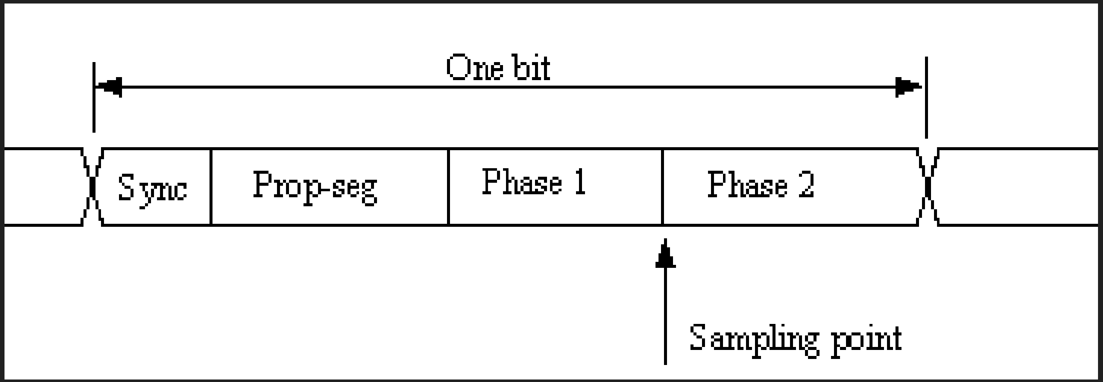
- **Message errors:**
  1. Checksum error - each receiving node checks CAN messages for checksum errors.
  2. Frame error - There are certain predefined bit values that must be transmitted at certain points within any CAN Message Frame. If a receiver detects an invalid bit in one of these positions a Form Error (sometimes also known as a Format Error) will be flagged.
  3. Acknowledgement Error - If a transmitter determines that a message has not been ACKnowledged then an ACK Error is flagged.

CAN Error detection and fault confinement

- **CAN controller Error modes**
  1. **Error active** - the normal operating mode for a controller. Messages can be received and transmitted. On detecting an error an active error flag is sent when the error counter is below 128.
  2. **Error passive** - a mode entered when the controller has frequent problems transmitting or receiving messages. Messages can be received and transmitted. On detecting an error while receiving, a passive error flag is sent when the error counter is greater than or equal 128.
  3. **Bus off** - entered if the controller has serious problems with transmitting messages. No messages can be received or transmitted until the CAN controller is reset by the host microcontroller or processor.

Bit Timing

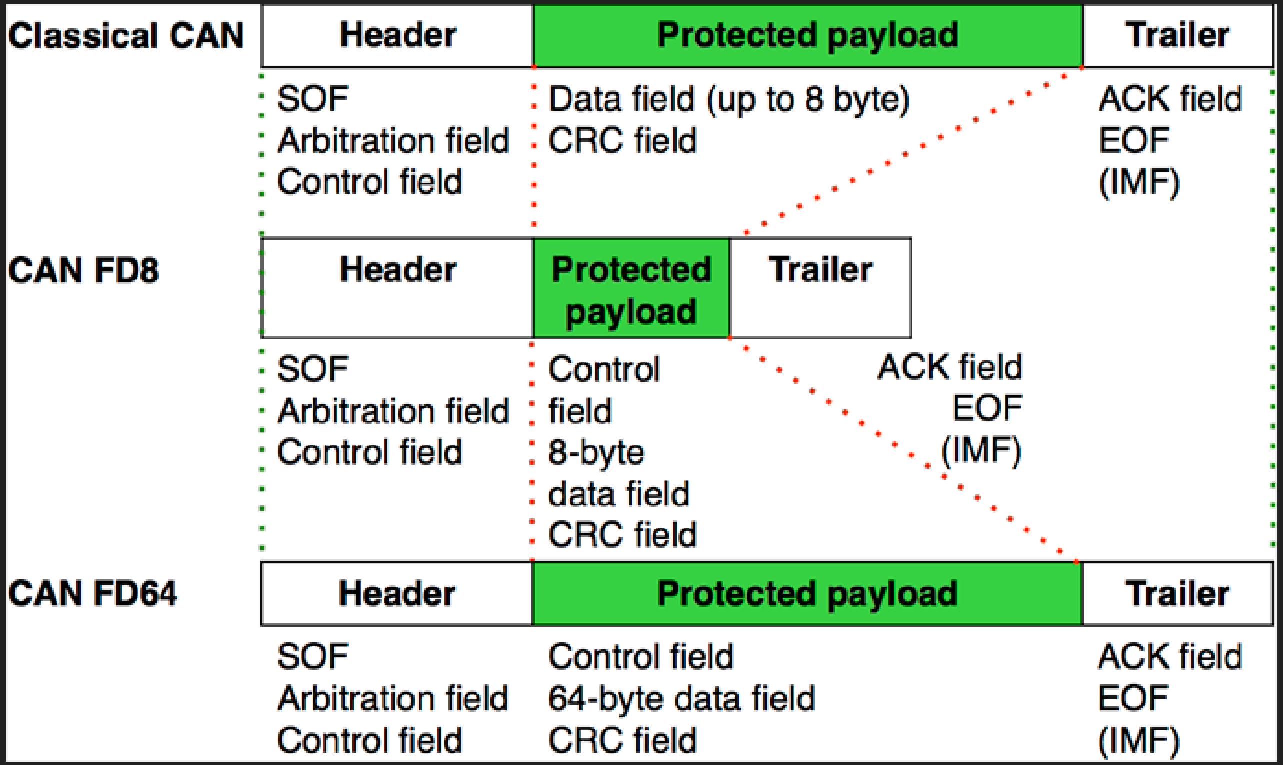
- Each bit is divided into four segments
  1. **Synchronization segment**, to synchronize the various nodes on the bus. (To correct the difference in clock oscillators of different nodes)
  2. **Propagation segment**, to compensate for the delay in the bus lines. (i.e. the controller transmits certain data and listen to the bus to know the data is received)
  3. **Phase Segment 1, Phase Segment 2** ,to be used lengthened or shortened by resynchronization.



CAN vs CAN-FD

CAN FD (flexible data-rate) vs CAN:

- The time for sending one data byte has been decreased in CAN FD new protocol.
- Using a ratio of 1:8 for the bit-rates in the arbitration and data phase leads to an approximately six-times higher throughput considering that the CAN FD frames use more bits in the header (control field) and in the CRC field.



LIN BUS

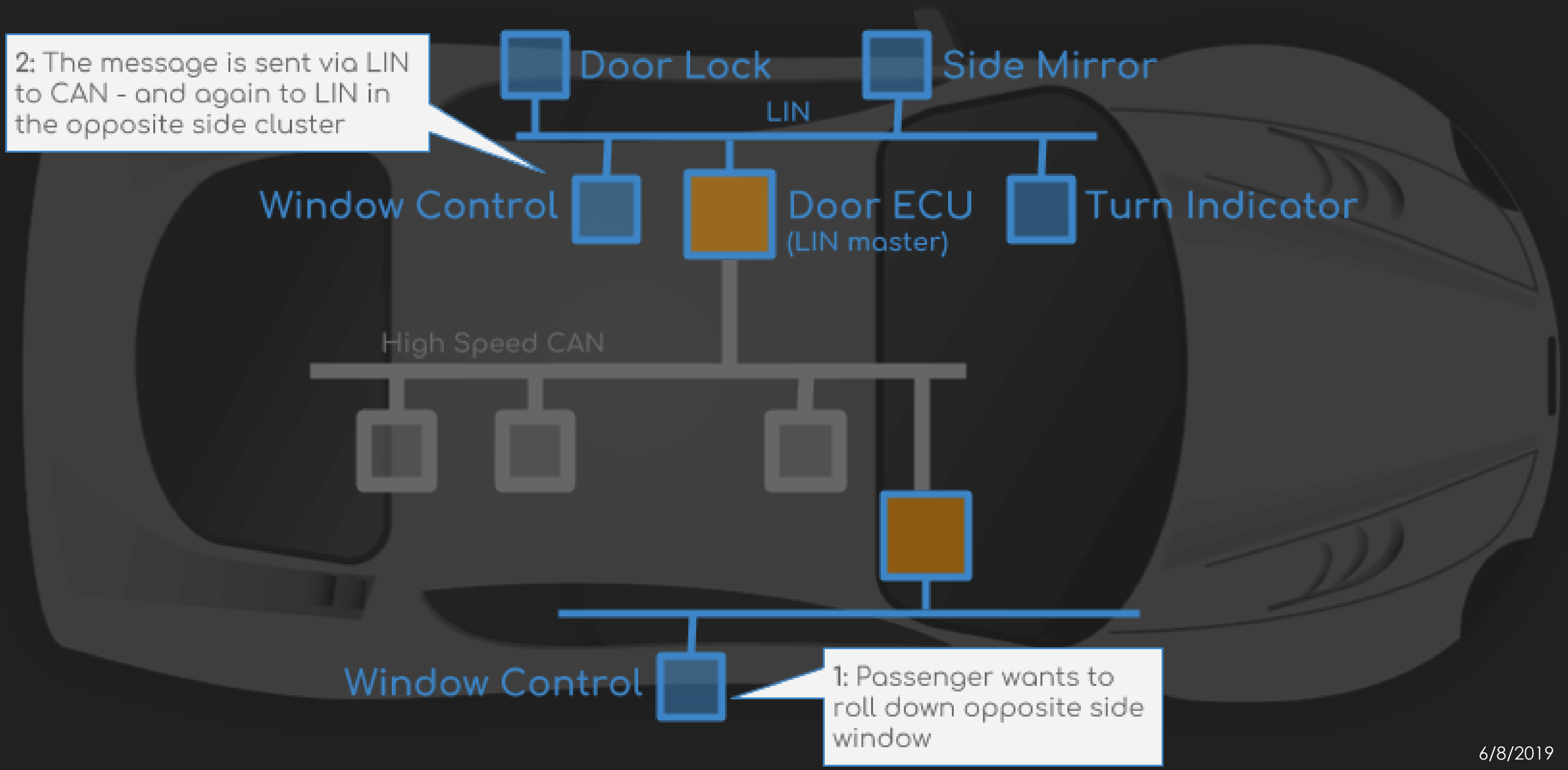
- LIN (Local Interconnect Network) is a low cost alternative to CAN bus in automotives.

- ***<u>QUICK OVERVIEW OF LIN BUS</u>***
  - Low cost option (if speed/fault tolerance are not critical)
  - Often used in vehicles for windows, wipers, air condition etc..
  - LIN clusters consist of 1 master and up to 16 slave nodes
  - Single wire (+ground) with 1-20 kbit/s at max 40 meter bus length

LIN BUS VS CAN BUS

- LIN is lower cost (less harness, no license fee, cheap nodes)
- CAN uses twisted shielded dual wires (5V), while LIN uses single wire (12V)
- A LIN master typically serves as gateway to the CAN bus
- LIN is deterministic, not event driven (i.e. no bus arbitration as there is a master and slave architecture)
- CAN can have multiple - LIN clusters have a single master
- CAN uses 11 or 29 bit identifiers vs 6 bit identifiers in LIN
- CAN offers up to 1 Mbit/s vs. LIN at max 20 kbit/s

Uses of LIN

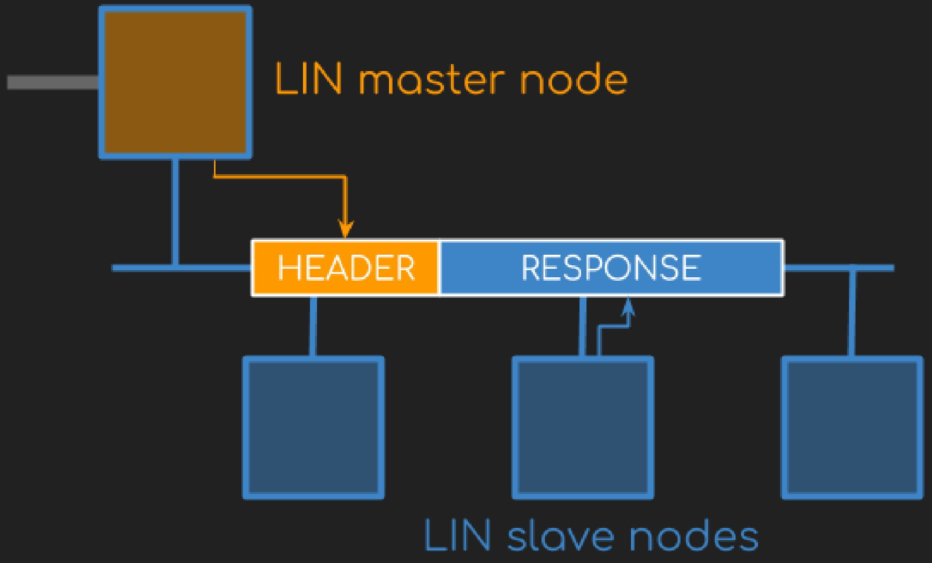


2: The message is sent via LIN to CAN - and again to LIN in the opposite side cluster

1: Passenger wants to roll down opposite side window

HOW DOES LIN BUS WORK?

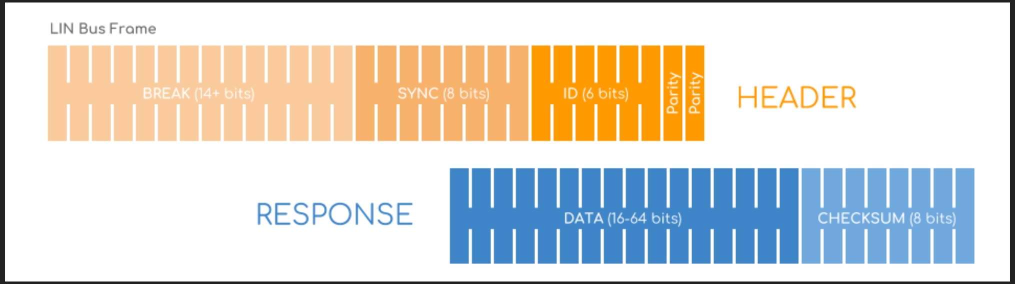
LIN bus at its core is relatively simple:
A master node loops through each of the slave node, sending a request for information - and each slave responds with data when polled.
However, with each specification update, new features have been added to the LIN specification - making it more complex.
Below we cover the basics: The LIN frame & six frame types.



THE LIN BUS MESSAGE FRAME

In simple terms, the LIN bus message frame consists of a **header** and a **response**.
Typically, the LIN master transmits a header to the LIN bus. This triggers a slave, which sends up to 8 data bytes in response.

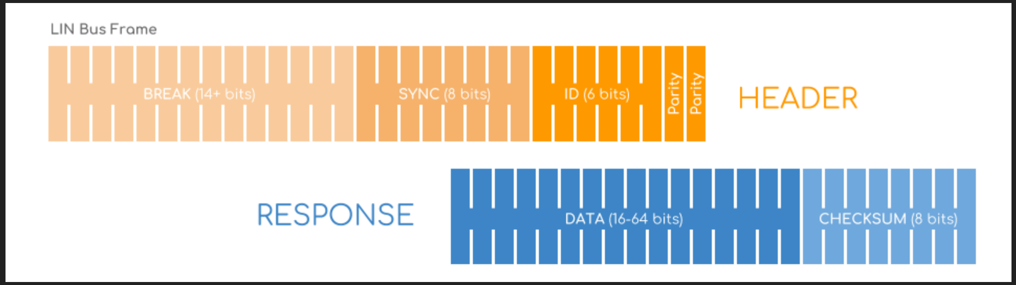
Break: Every LIN frame begins with the break, which comprises 13 dominant bits (nominal) followed by a break delimiter of one bit (nominal) recessive. This serves as a start-of-frame notice to all nodes on the bus.
Sync: The 8 bit Sync field has a predefined value of 0x55 (in binary, 01010101). This structure allows the LIN nodes to determine the time between rising/falling edges and thus the baud rate used by the master node. This lets each of them stay in sync.



THE LIN BUS MESSAGE FRAME

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.

Data Bytes
The data bytes field is transmitted by the slave task in the response. This field contains from one to eight bytes of payload data bytes.
Checksum
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, and enhanced checksum is calculated by summing the data bytes and the protected ID.


LIN Bus Frame
BREAK (14+ bits)
SYNC (8 bits)
ID (6 bits)
Parity
Parity
HEADER
RESPONSE
DATA (16-64 bits)
CHECKSUM (8 bits)