



CAN Protocol



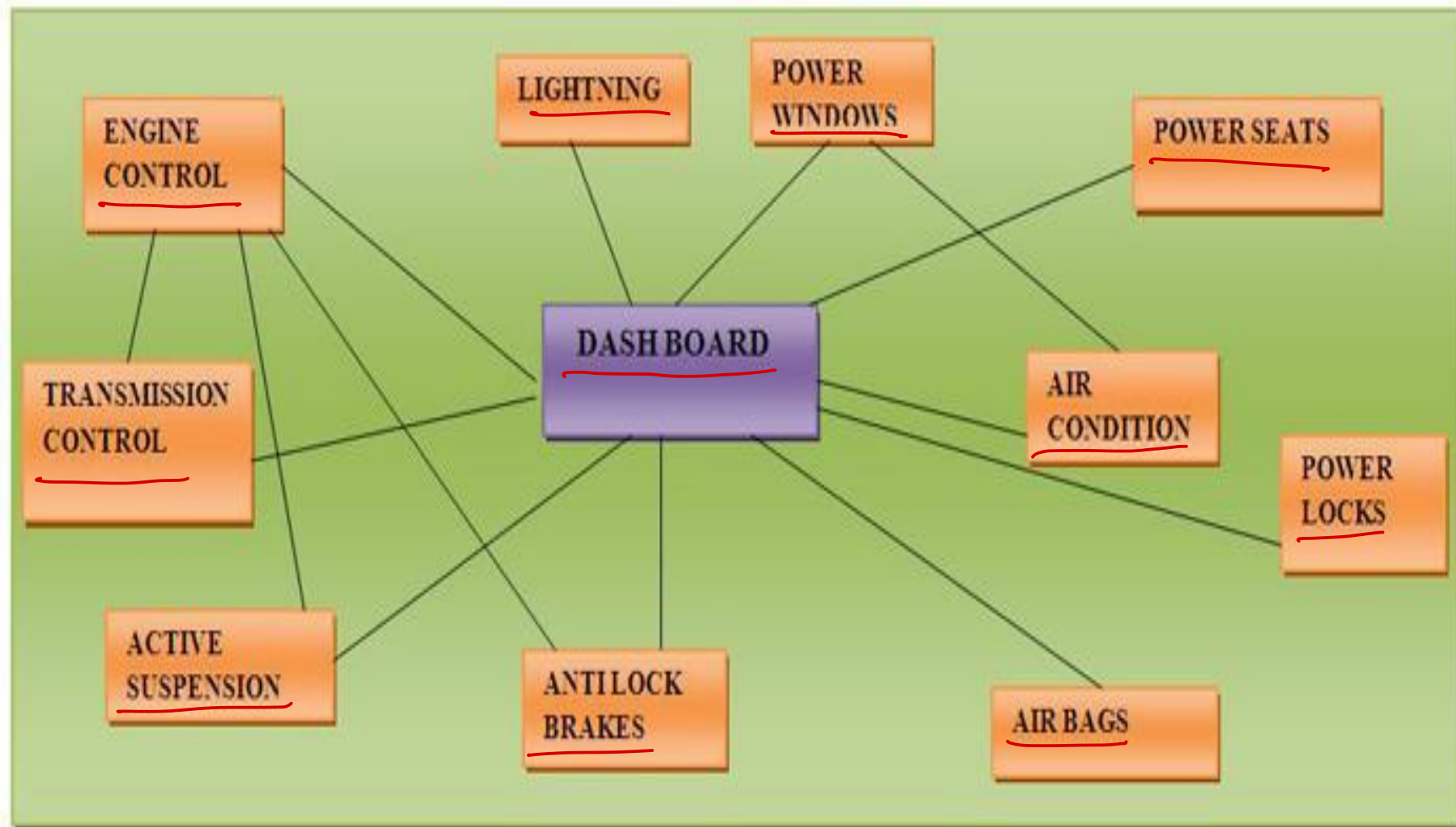
Controller Area Network

Introduction

- ▶ CAN protocol is method of communication between various electronic devices. It defines set of rules for communication in a network of devices. → bus protocol
- ▶ Original idea initiated Robert Bosch in 1983. However first release of CAN protocol is done in 1986.
- ▶ Protocol is implemented in hardware and software to communicate between different controllers present in the automobiles.
- ▶ Nowadays this protocol is used in various industries including Healthcare (ICUs & Operation Rooms), Entertainment (light control, door control in studios, gambling machines), Science (high energy experiments, astronomical telescopes).



Automobiles – Prior invention of CAN

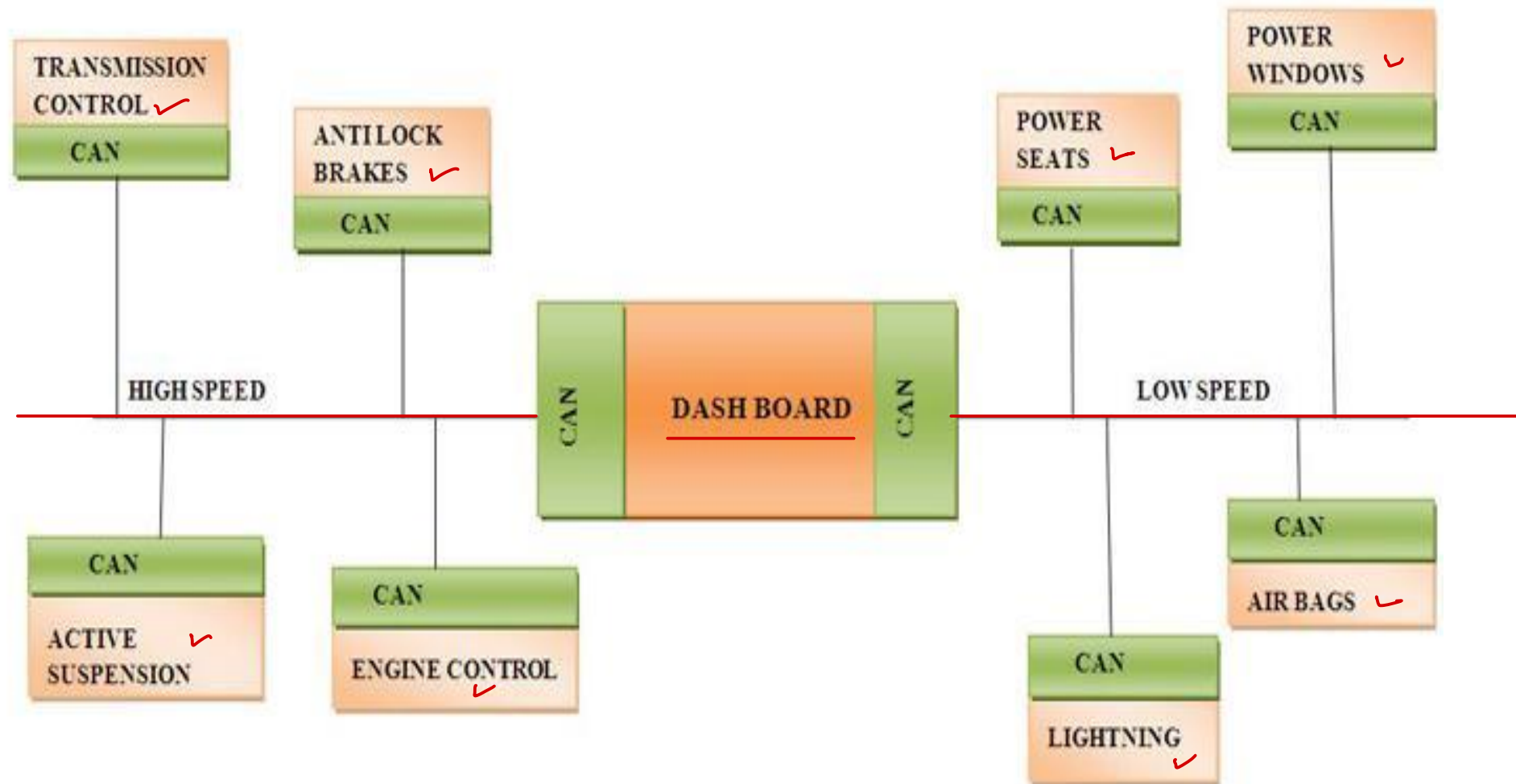


Drawbacks & Limitations of Wired System

- ▶ Number of wires in various subsystem makes the system complicated and difficult to maintain.
- ▶ Passing real time information among subsystems was tedious implementation (serial protocols used).
- ▶ Asynchronous transmitter/receiver do not support multi-domain communication e.g. communication between air-conditioning system and door/window system.
- ▶ Multiple domains in automobiles includes power generation (engine), chassis (driving mechanism), body (climate control/wipers), telemetric (entertainment units) and passive safety (air bags, etc).



Automobile System – with CAN protocol

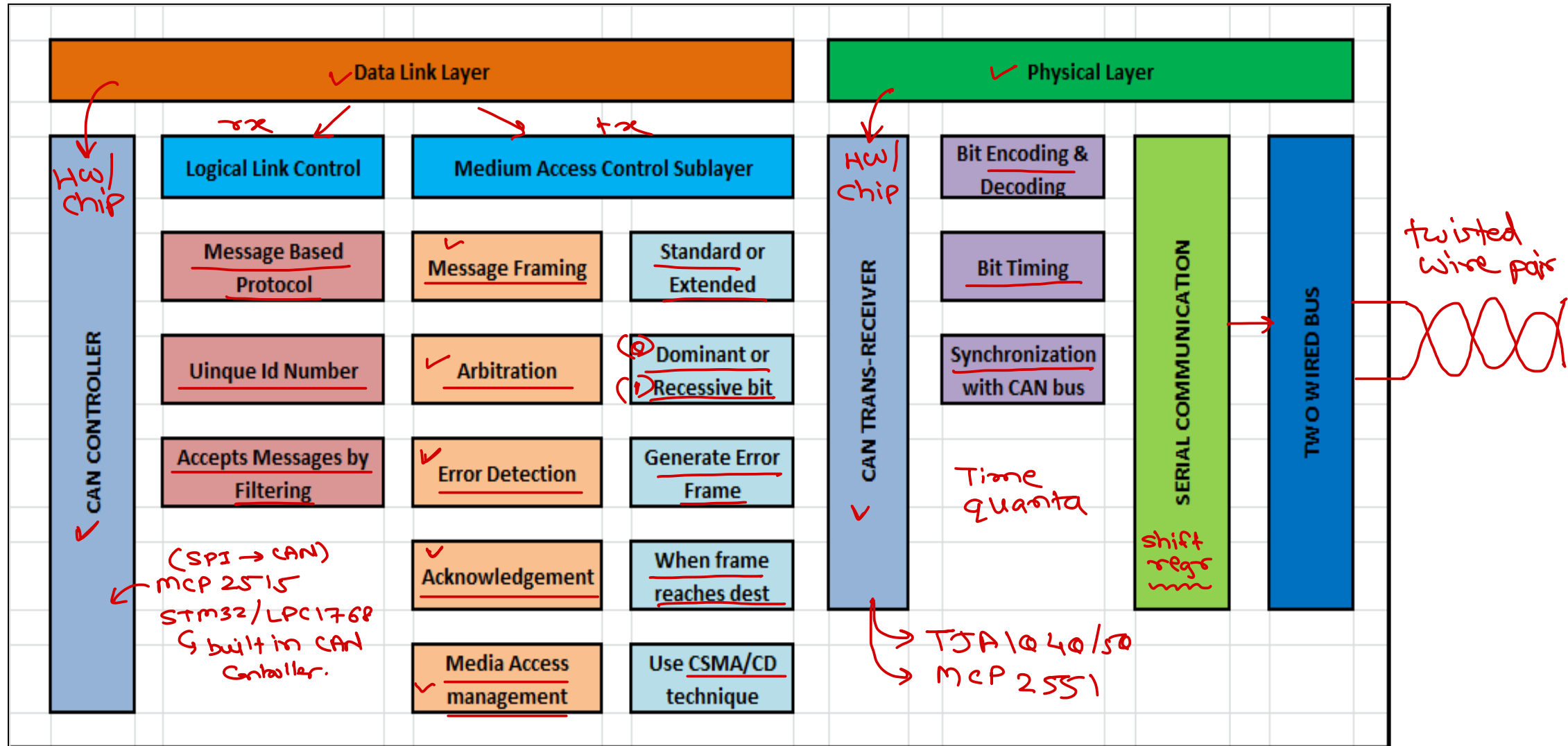


CAN Architecture

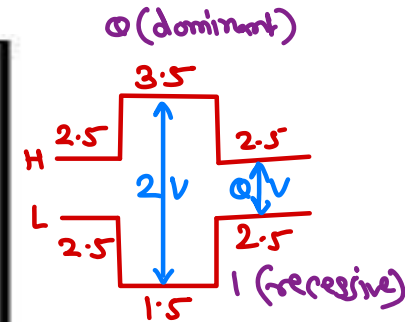
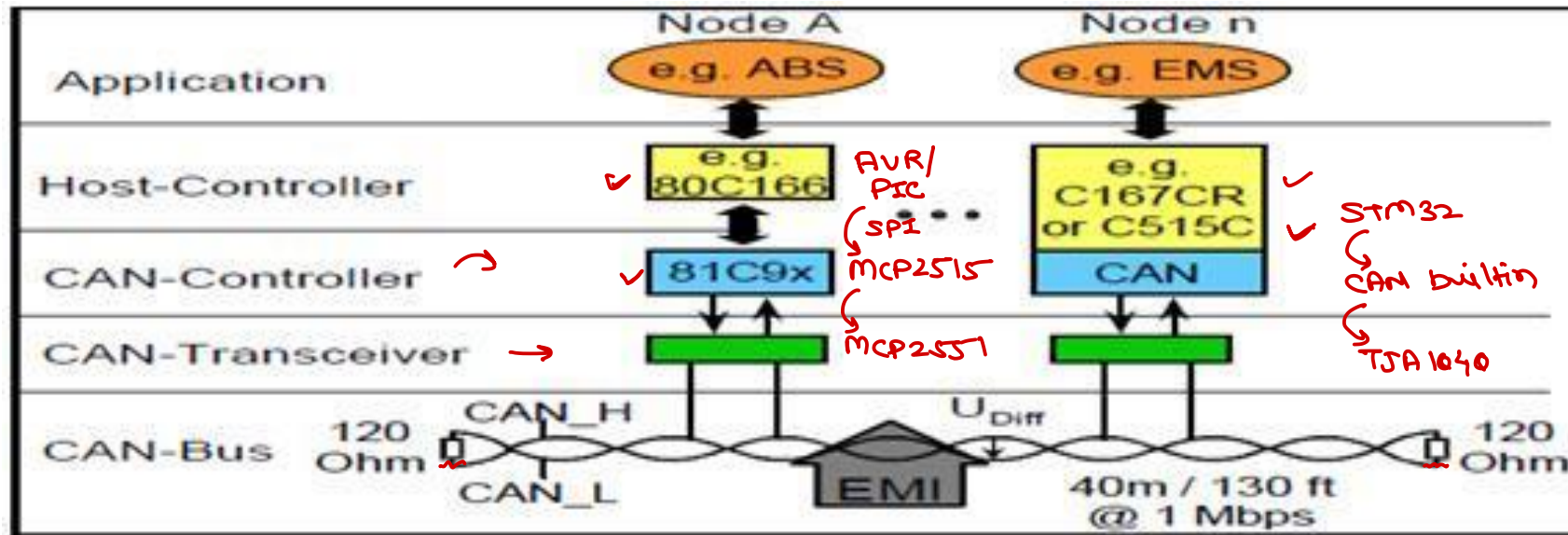
- ▶ CAN protocol is implemented with OSI reference model.
- ▶ It implements two layers of OSI model and rest are left for implementation specific to the requirement.
- ▶ Data Link Layer
 - ▶ Logical link control ← 82
 - ▶ Allows filtering of messages based on UID.
 - ▶ Medium access control → 82
 - ▶ Prepare message frame and handle arbitration.
- ▶ Physical Layer
 - ▶ Send bits to the CAN two wire bus as per timing requirements.

- ✓ application layer
- ✓ presentation layer
- ✓ session layer
- ✓ transport layer
- ✓ network layer
- ✓ data link layer
- ✓ physical layer

CAN Architecture

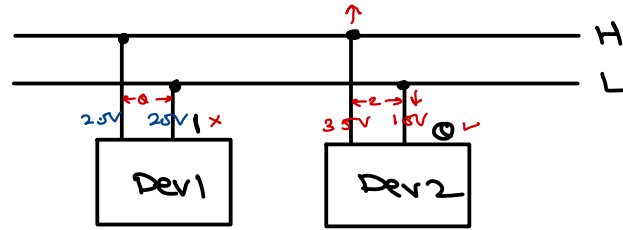


CAN Node

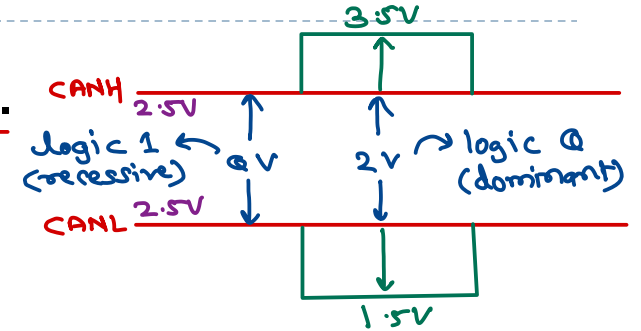


- ▶ Each electronic device is called as Node.
- ▶ Host-controller is MCU responsible for functioning of node.
- ▶ CAN controller converts messages of node as per CAN protocol. Can be a separate chip or embedded in MCU.
- ▶ Trans-receiver is to transmit bits on CAN bus.

CAN Bus



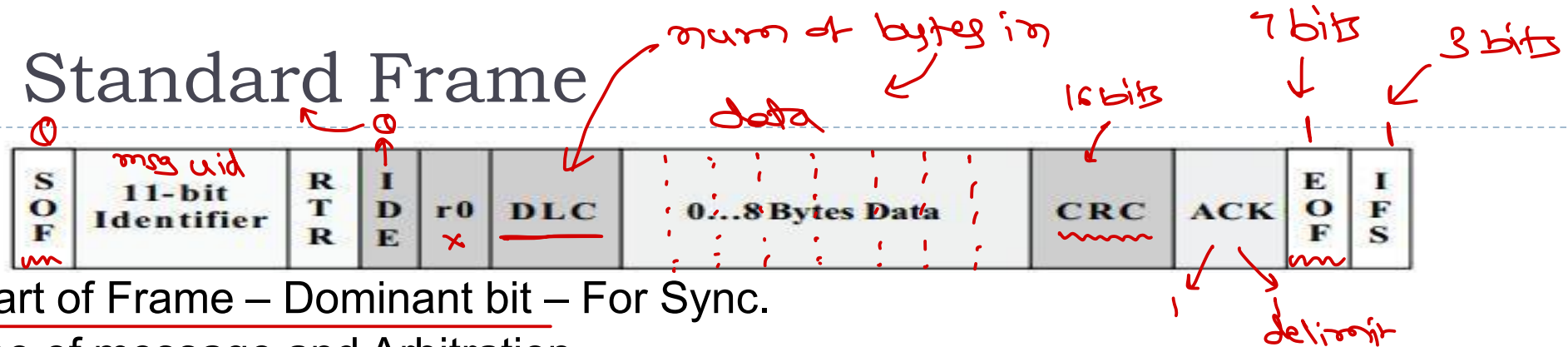
- ▶ CAN bus is a two twisted wire bus i.e. CANH & CANL.
 - ▶ The passive voltage of each line is 2.5 V.
 - ▶ The active voltages are 3.5 V and 1.5 V.
 - ▶ When both lines are 2.5 V, difference is 0 V. It represent logic 1 & called as “recessive bit”.
 - ▶ When both lines are pulled to 3.5 V and 1.5V respectively, then difference is 2 V. It represent logic 0 & called as “dominant bit”.
 - ▶ Note that dominant bit can always overwrite recessive bit.
 - ▶ CAN bus is a linear bus terminated with 120 Ω . Also input impedance of each node is 120 Ω .
- ▶ CAN bus is not a master slave bus i.e. Any node can write the data on the bus in certain format (frame) provided bus is available.



CAN Frame

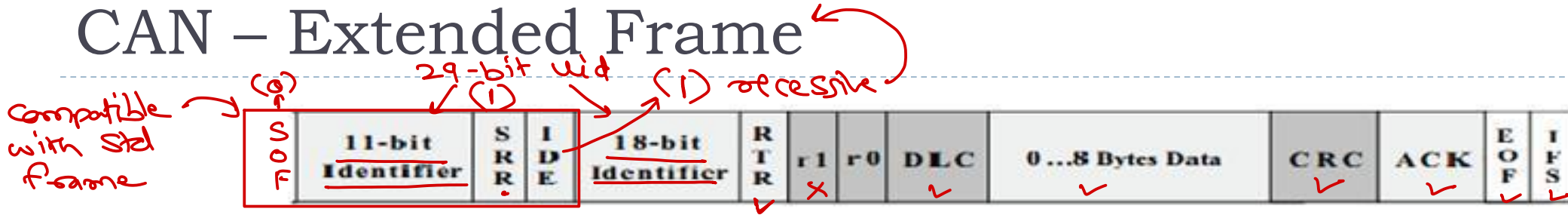
- ▶ CAN is a message based protocol (not address based).
- ▶ Message contains a pre-defined unique id (rather than addresses).
- ▶ Messages are accepted or rejected by any node based on this UID. If multiple nodes send messages at same time, node with highest priority gets bus access. *↪ arbitration* *one who write first 0 (dominant) bit.*
- ▶ CAN message is made up of 10 bytes.
- ▶ Each message is coded into meaningful sequence of bits/bytes called as **frame**.
- ▶ Framing is done by Medium Access Layer. *↪ Can controller*
- ▶ There are two types of frames: *Formats:*
 - ▶ Standard CAN Frame *↪ CAN 1.0 → 11 bit uid*
 - ▶ Extended CAN Frame *↪ CAN 2.0 → 29 bit uid*

CAN – Standard Frame



- ▶ SOF: Start of Frame – Dominant bit – For Sync.
- ▶ UID: Type of message and Arbitration.
- ▶ RTR: Type of frame i.e. Data Frame [Dominant] or Remote Transmission Request (RTR) Frame [Recessive].
 - ▶ RTR frame don't have data, instead request other node to send data.
- ▶ IDE: UID Extension. Standard (dominant) or Extended (recessive) frame.
- ▶ R0: Reserved for future use.
- ▶ DLC: 4-bit data length code [0 to 8 bytes – data length]
- ▶ DATA: 0 to 8 bytes
- ▶ CRC: 15 bits CRC + 1 bit delimiter (recessive)
- ▶ ACK: Transmitter sent recessive bit, Rcvr overwrite with dominant. + 1 bit delimiter (recessive)
- ▶ EOF: 7-bits (recessive) to indicate End of Frame.
- ▶ IFS: 3 recessive bits - Intermission bits – Separation between two frames.

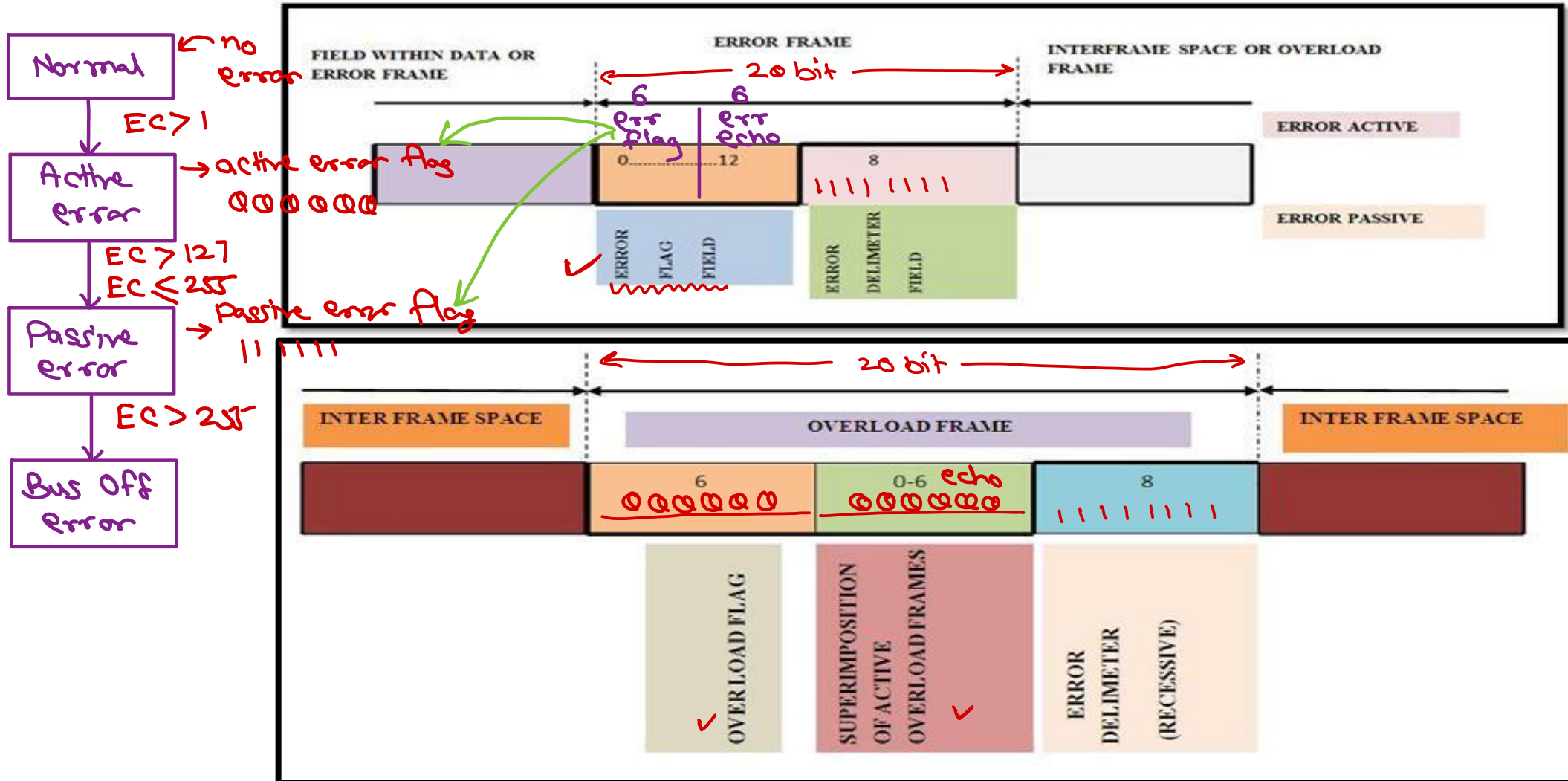
CAN – Extended Frame



- ▶ SRR: Substitute Remote Request : Recessive bit ~~like~~ RTR
- ▶ 11-bit Id + 18 bit Id = 29 bit Id – Extended Message Id.
- ▶ R1: Additional reserved bit.
- ▶ Types of Frames:
 - ✓ **Data Frame**
 - ✓ Remote Frame To request data from other device.
RTR=1. No data sent in RTR frame.
 - ✓ Error Frame: When error is detected, transmission aborts and send this frame.
 - ✓ Overload Frame: Like error frame, sent by node when busy in internal processing.

TEC } Tx/Rx
REC } Error Count

CAN – Error and Overload Frames



CAN Error Detection & Handling

- ▶ There are five methods of error detection.
 - ▶ Message Level Error Detection.
 - ▶ CRC check ✓
 - ▶ ACK slots ✓
 - ▶ Form error ✓
 - ▶ Bit Level Error Detection.
 - ▶ Stuff error ✓
 - ▶ Bit error ✓
- ▶ If node detects an error, following steps occurs:
 - ✓ Transmits error flag.
 - ✓ Destroys transmitted frame.
 - ✓ Transmitting node resends the frame.



CAN – Error Detection

▶ CRC check:

- ▶ Calculated and sent by transmitter node.
- ▶ Receiver node recalculate CRC and if differs, raise error.

▶ ACK slots:

- ▶ Transmitter send recessive bit & Receiver overwrite dominant
- ▶ If none of the node overwrite dominant bit, error is raised.

▶ FORM (FORMAT) error:

- ▶ EOF, IFS, ACK delim bits are always recessive.
- ▶ If dominant bit is found, error is raised.

▶ BIT error:

- ▶ Transmitter always monitor sent bit.
- ▶ If sent bit is not validated error is generated, except in case of arbitration and acknowledgment bit.

Handwritten note: 1 → 0 → arbitration

Handwritten note: 1 → 0
Tx Rx





CAN – Error Detection

▶ Bit Stuff Error:

- ▶ CAN bus is never IDLE as it follows NRZ method (non-returning to zero i.e. 0 & 1 is represented as non-zero values - differential).
- ▶ For sake of synchronization one bit of opposite polarity is added after consecutive 5 bits of same polarity, called as bit-stuffing.
- ▶ Stuffed data frames are de-stuffed by data link layer of receiver.
- ▶ If error is found in stuffing, error is raised.
- ▶ In CAN, 6 consecutive recessive/dominant bits represent error bits.
- ▶ All fields in the frame are stuffed with the exception of the CRC delimiter, ACK field and end of frame which are a fixed size.

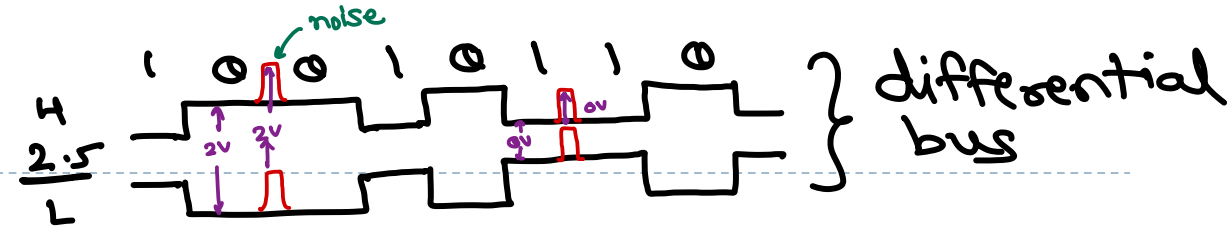
⑦

①

③



CAN protocol – Advantages



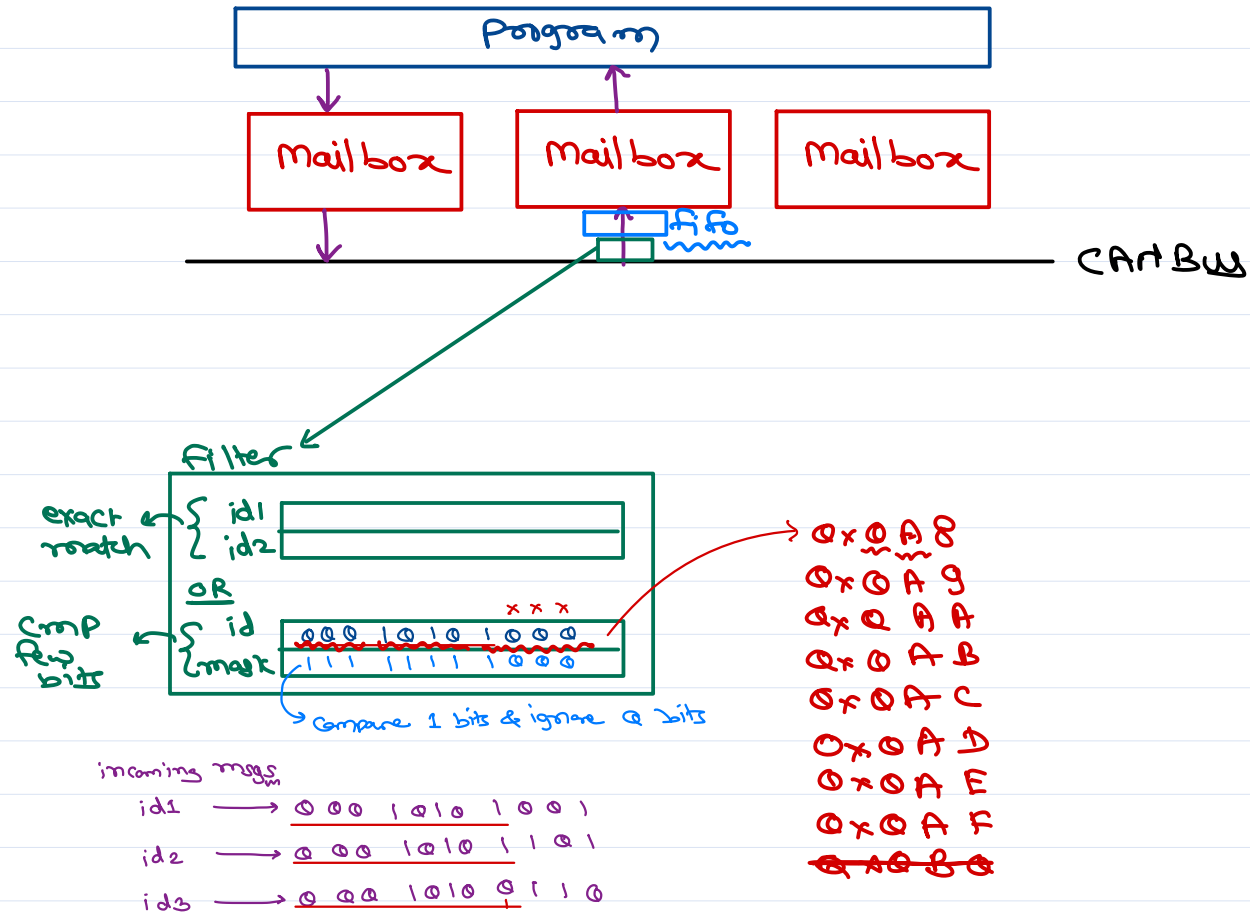
- ▶ Low cost: Only two wire serial bus.
- ▶ Reliable: Error detection & handling. Immune to noise.
- ▶ Flexibility: Nodes can be easily added or deleted.
- ▶ High speed: Support data rate of 1 Mbits/sec @ 40m bus.
- ▶ Multi-master bus: Any node can access bus.
- ▶ Fault confinement: Faulty nodes do not disturb commn.
- ▶ Broadcast capability: One to One/Many/All commn.
- ▶ Standardization: ISO standardized.
 - ▶ ISO-DIS 11898 : High speed communication
 - ▶ ISO-DIS 11519-2 : Low speed communication

Mailbox and Acceptance Filter

① Mailbox

② Acceptance Filter

③ Time quanta



CAN bit timing

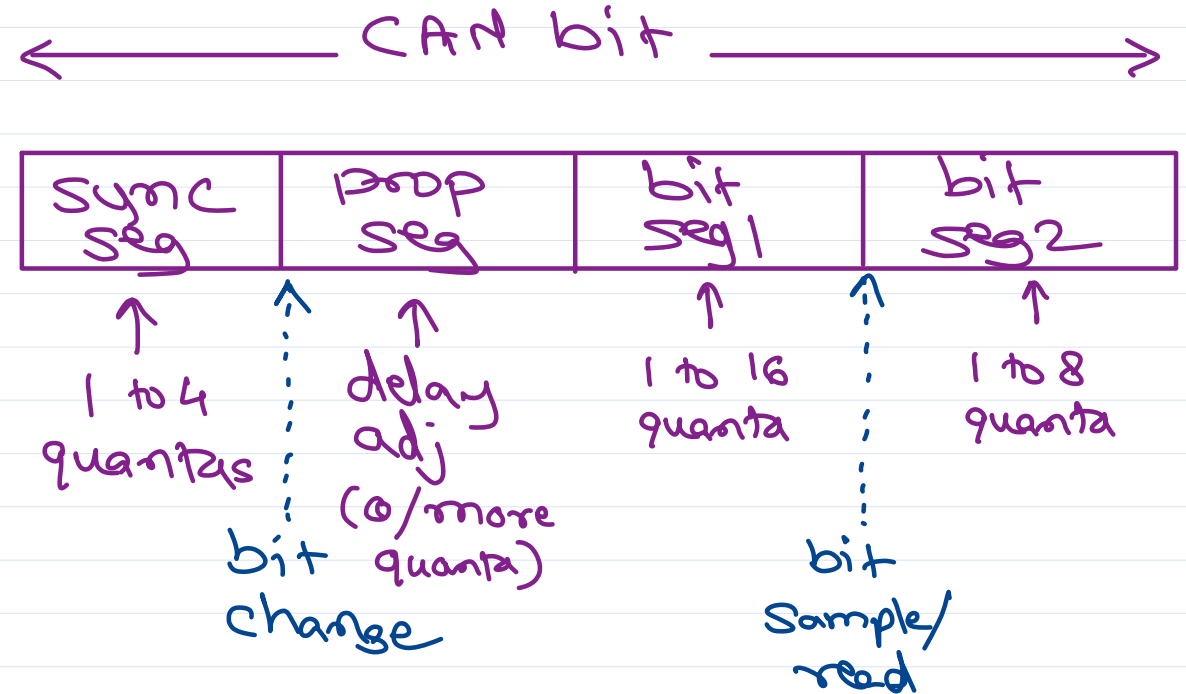
CPU clock = 72 MHz

PCLK = 36 MHz (APB1)

CAN Prescaler = 18

$$\text{CAN Freq} = \frac{\text{PCLK}}{\text{PR}} = \frac{36}{18} = 2 \text{ MHz}$$

$$\begin{aligned}\text{CAN Time quanta} &= \frac{\text{Prescaler}}{\text{PCLK}} \\ &= \frac{18}{36 \text{ MHz}} = 0.5 \mu\text{s} \\ &= 500 \text{ ns}\end{aligned}$$



Our Example :

$$\begin{aligned}&= 1 + 0 + 2 + 1 \\&= 4 \text{ quanta} \leftarrow \text{bit nominal time} \\&= 4 \times 500 \text{ ns} = 2000 \text{ ns} = 2 \mu\text{s}\end{aligned}$$

$$\text{Bit rate} = \frac{1}{2 \mu\text{s}} = 0.5 \text{ MBit/s} = 500 \text{ KBit/s}$$





Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

