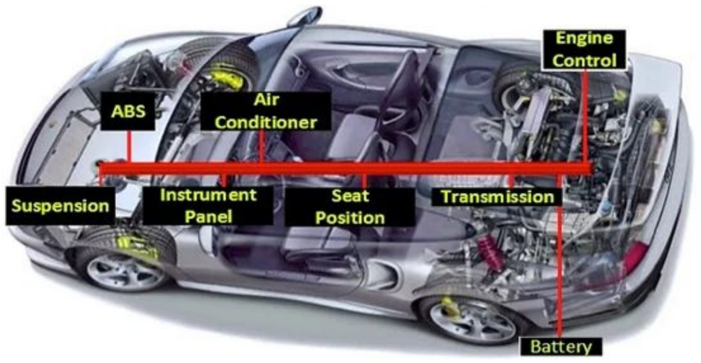
Controller Area Network (CAN Bus)

Communication Protocols

Controller Area Network (CAN Bus)

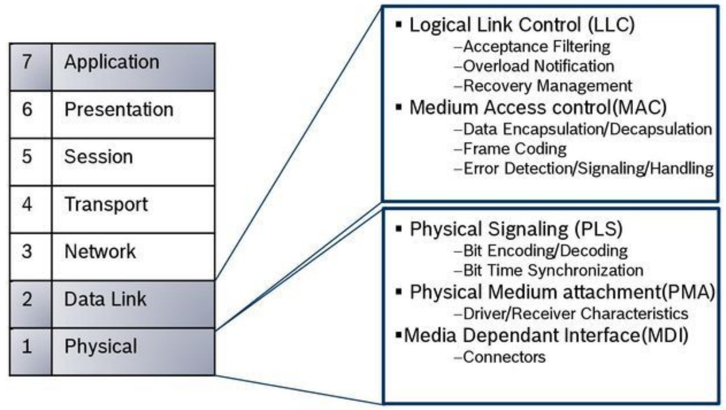
- An asynchronous serial bus system protocol introduced by Bosch in 1986
  - Bosch published several versions of the CAN specification
  - Standard CAN (ISO 11898)
    - Low Speed CAN (CAN 2.0A) in 1993
      - Up to 125Kbps and 11-bit message Identifier
    - **High Speed CAN** (CAN 2.0B / Extended CAN) in 1995
      - Up to 1Mbps and 11-bit/29-bit message Identifier
    - CAN FD (Flexible Data-Rate) in 2015
      - Up to 15Mbps and 11-bit/29-bit message Identifier

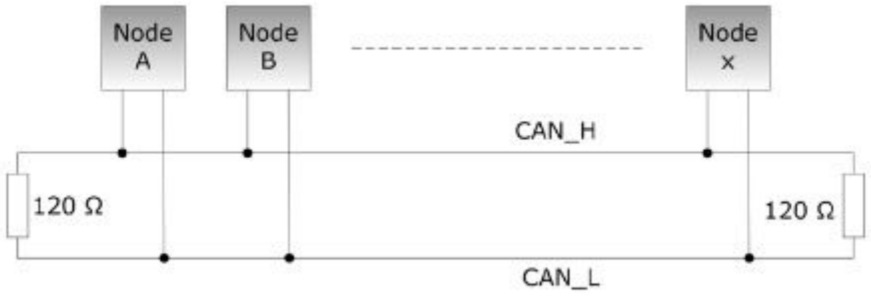


- Primarily designed for building networks in automotive applications
  - Nowadays it is also used in industrial automation and medical equipment

Controller Area Network (CAN 2.0B - Features)

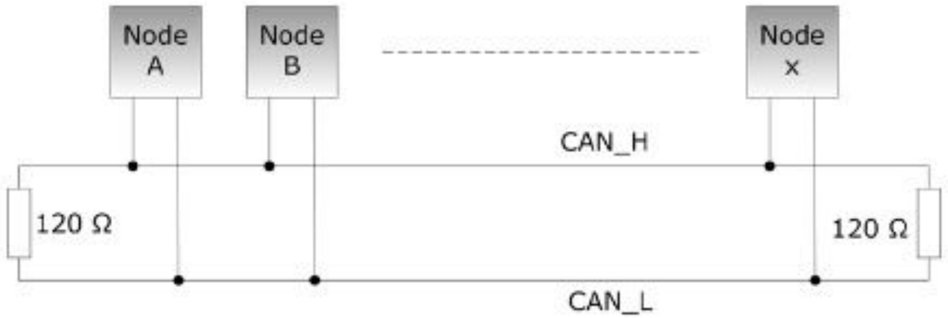
- ❖ CAN as a closed network in the OSI model
  - ➢ No need for security, sessions, user interface, addressing and etc.
  - ➢ Physical and Data Link layers are standardized
- ❖ Supports half-duplex communication
- ❖ Allows multiple nodes communicate via a pair of wires
- ❖ Is a message oriented transmission protocol
  - ➢ Nodes have no addresses
  - ➢ Messages have unique identifiers are broadcasted
  - ➢ Nodes can receive all the messages and filter them
- ❖ High performance and real-time communication
  - ➢ Up to 1Mbps. Bus access conflicts resolved by arbitration

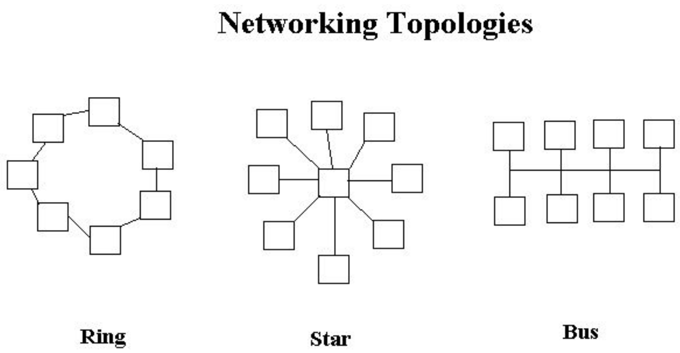




Controller Area Network (CAN 2.0B - Features)

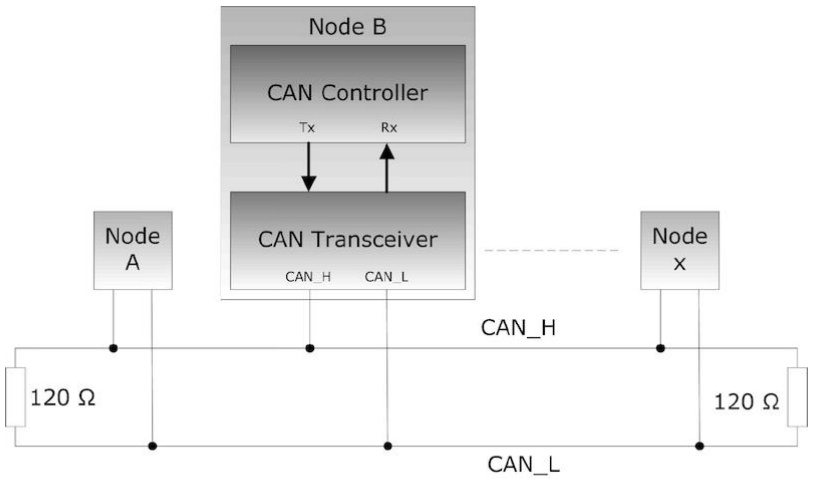
- ❖ Provides high noise immunity by using differential signaling
  - ➢ The signal lines should be terminated in both the ends with 120Ω resistors
- ❖ Extensive error checking
  - ➢ Different checks like CRC, ACK and etc.
  - ➢ Every connected node participate
  - ➢ A message is accepted by all nodes or none
- ❖ Is an asynchronous communication
- ❖ Is a low cost bus system
- ❖ CAN supports different
  - ➢ Topologies like bus, ring and star
  - ➢ Bus management methods like master-slave

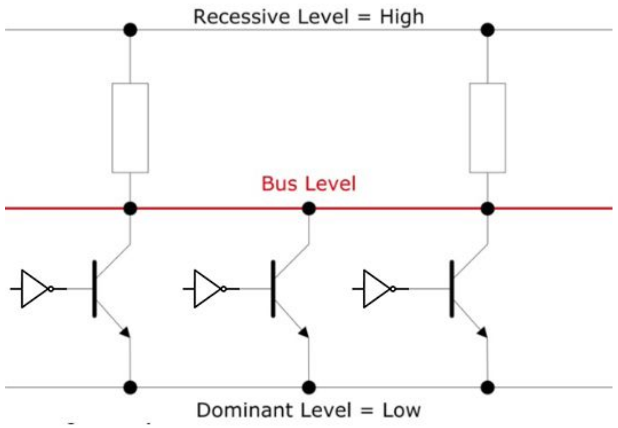


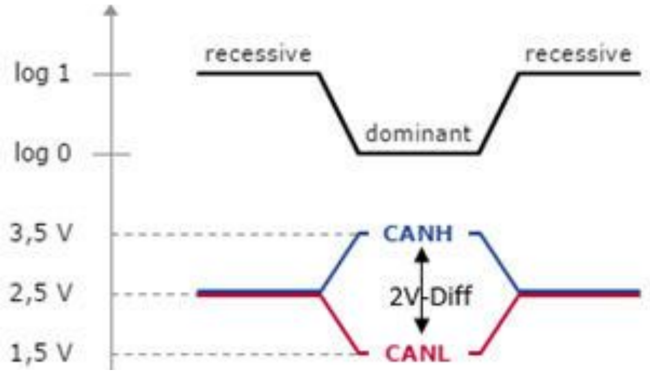


Controller Area Network (CAN 2.0B - Physical Layer)

- Physical medium: a twisted pair wires
- 120Ω resistors are used to terminate the lines
- LOW level is dominant and HIGH is recessive
- LOW on the bus by a node wins over a HIGH on the bus
- CAN uses differential signaling (CAN_H and CAN_L)
- Every node has a CAN controller and a CAN transceiver
- Logical values are inverted
  - Logical 0 is dominant on the bus
  - Logical 1 is recessive on the bus
- CAN bus uses
  - NRZ signals and encoding
  - Bit-stuffing for resync the nodes

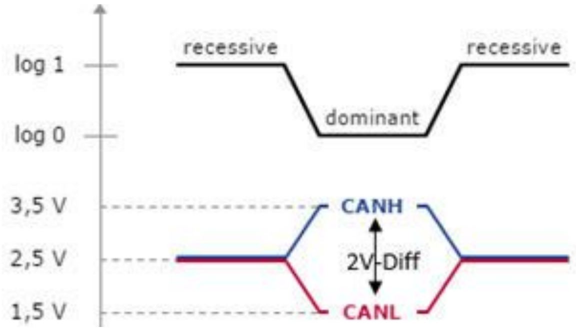


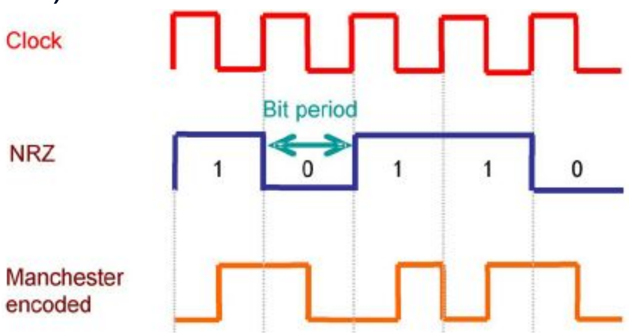




Controller Area Network (CAN 2.0B - Physical Layer)

- Non Return to Zero (NRZ)
  - As signal level
    - Logical "1" is represented by a voltage level (usually a positive voltage)
    - Logical "0" is represented by another level (usually a negative voltage)
    - E.g. In RS-232; "one" is −12V and "zero" is +12V
    - For example in CAN_H, “one” is +2.5V and “zero” is +3.5V
  - As signal encoding
    - Logical “1” is sent as a HIGH and “0” send as a LOW
    - Logical values are not sent as transition of the levels (0 -> 1 and 1 -> 0)
    - Signal is not a self-clocking signal
  - NRZ signaling and encoding
    - Have a better Electromagnetic Compatibility(EMC)
    - Need resynchronization. Because the signal is not self-clocking.

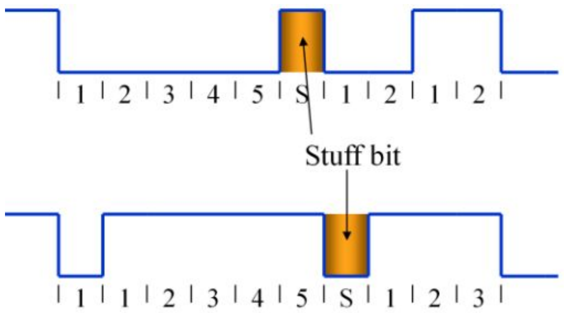




Controller Area Network (CAN 2.0B - Physical Layer)

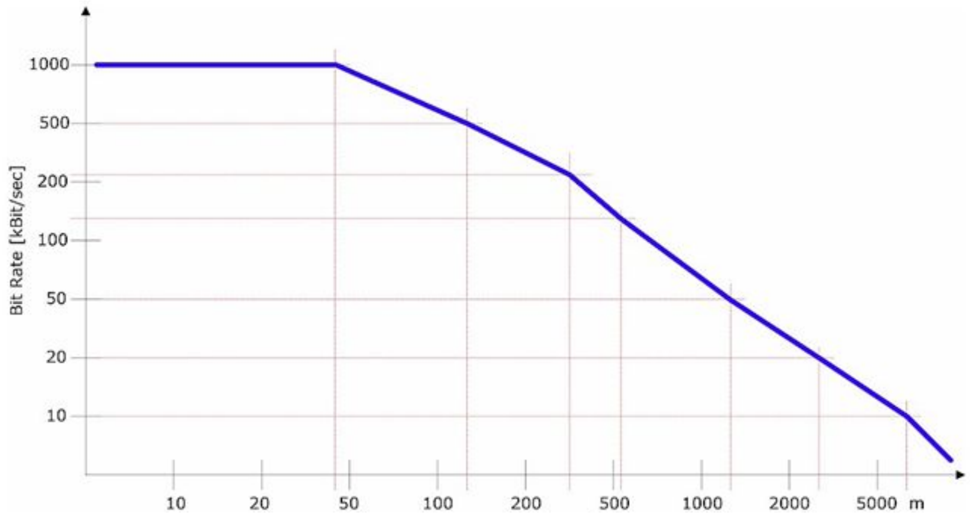
- Synchronization
  - No clock in the bit stream
  - Receivers synchronize the communication on the recessive to dominant transitions
  - **Hard Synchronization** occurs at the **SOF** and resets the bit clock
  - **Resynchronization** occurs at recessive-to-dominant edges and it adjusts the bit clock

- Bit stuffing
  - If there is no edge for a long time, receivers lose track of bits
  - Periodic edges allow receivers to resynchronize to sender clock
  - Bit stuffing is used to ensure synchronization of all bus nodes
  - A stuff bit occurs after 5 consecutive bits with same polarity



Controller Area Network (CAN 2.0B -Transmission)

- Speed: Up to 1 Mbit/sec.
  - Common baud rates: 1 Mbps, 500 Kbps, 250 Kbps and 125 Kbps
- CAN is an asynchronous communication
  - All nodes should have the same baud rate
- Max length: 40 to 5000m
  - Depends on the speed
    - At 1 Mbps, 40m
    - At 125 Kbps, 500m

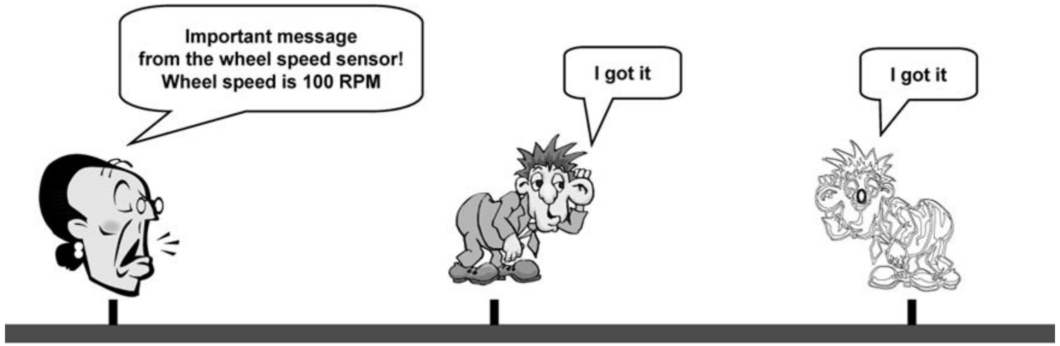


- The standard allows a maximum cable length of 40 m with up to 30 nodes and a maximum stub length (from the bus to the node) of 0.3 m. Longer stub and line lengths can be implemented, with a trade-off in signaling rates.

Controller Area Network (CAN 2.0B - Message Broadcasting)

- Each node is receiver & transmitter
- A transmitter broadcasts its message on the bus
- All nodes read message, then decide if it is relevant to them
- All nodes verify reception was error-free
- All nodes acknowledge reception

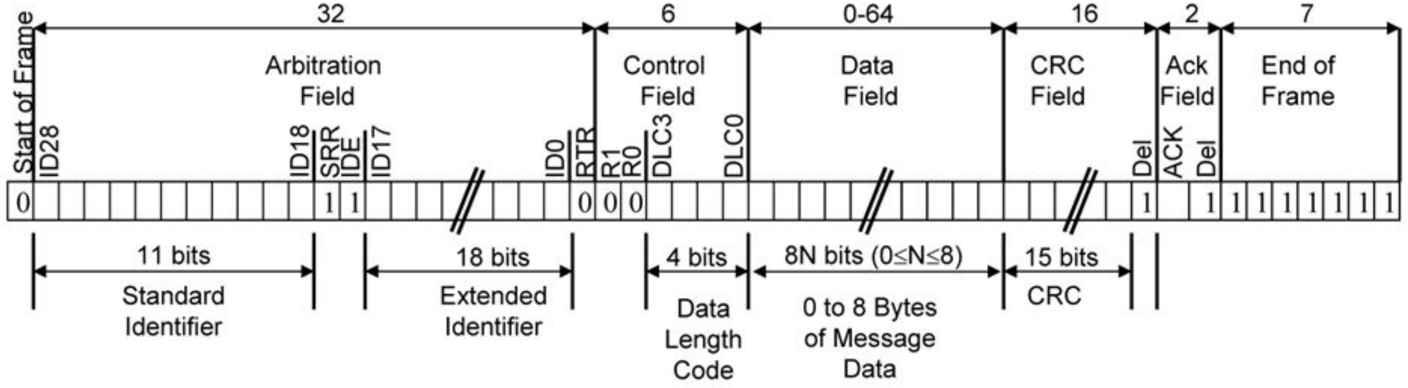
- Message Types in CAN
  - Data Frame
  - Remote Frame
  - Error Frame
  - Overload Frame

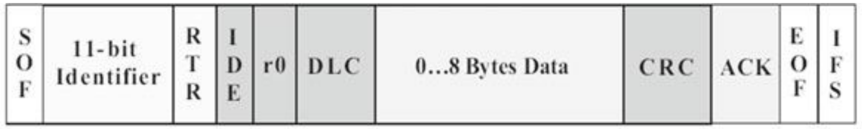


CAN Bus

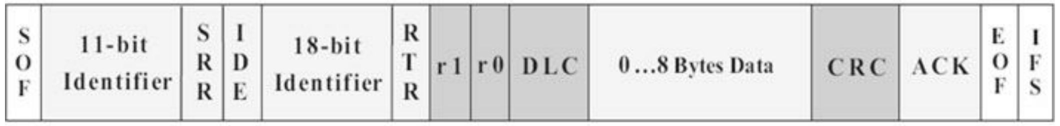
Controller Area Network (CAN 2.0B - Data Frame Format)

- Each data frame (message) has
  - Start of Frame
  - Arbitration Field
  - Control Field
  - Data Field
  - CRC Field
  - Acknowledgement Field
  - End of Frame
  - Intermission Frame Space (IFS)





(a)



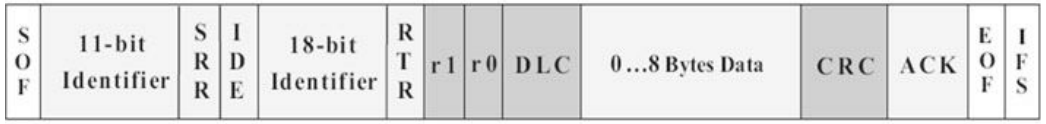
(b)

- Start of Frame: A dominant bit (0)
- End of Frame: 7 recessive bits (111 1111)
- Intermission Frame Space: 3 or more recessive bits

Controller Area Network (CAN 2.0B - Data Frame Format)

❖ Arbitration Field

(a)



(b)

- ➢ Message identifier: 11 bits or 29 bits
  - In the Standard CAN this field is 11 bits
  - In the Extended CAN this field is 29 bits
- ➢ RTR (Remote Transmit Request) bit
  - If this bit is recessive (1) the frame is a remote request
- ➢ This field sets the priority of the message
  - Lower value for this field means that the message is more important
- ➢ In the case of collision, bus arbitration is done using this field
  - A message with lower ID has more priority and it will be send before messages with higher ID
- ➢ CAN controllers use this field to filter the received messages
- ➢ SRR (Substitute Remote Request) and IDE (IDentifier Extension) in Extend CAN are recessive (1)

Controller Area Network (CAN 2.0B - Data Frame Format)

- Control Field
  - IDE (IDentifier Extension) and R0 in the Standard CAN are dominant (0)
  - R0 and R1 in the Extended CAN are dominant (0)
  - DLC (Data Length Code) can have the value 0..8
- Data Field
  - This field can be from 0 up to 8 bytes
  - It is always full 8-bit bytes
  - The bytes can have any value
  - Some CAN controllers can extend ID filtration into the data field

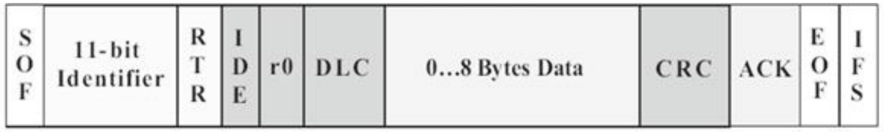
(a)

(b)

- CRC Field
  - A 15-bit CRC checksum of the bits in the message and a recessive bit as a delimiter

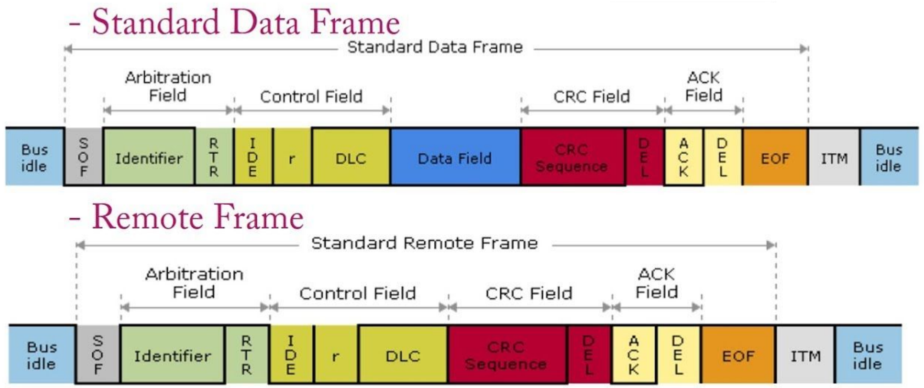
Controller Area Network (CAN 2.0B - Data Frame Format)

- Acknowledgement Field
  - The ACK bit and a recessive bit as a delimiter
  - The transmitter sets the ACK bit to 1
  - Any receiver set the ACK bit to 0 when the message is found OK
- Remote Frame (when RTR bit is 1)
  - DLC must be identical to DLC in corresponding DATA Message!
  - There is no Data Field in a remote request frame



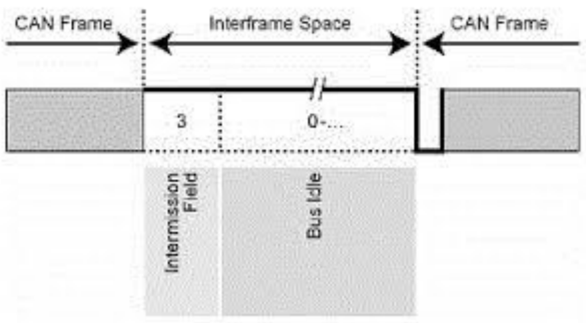
(a)

(b)

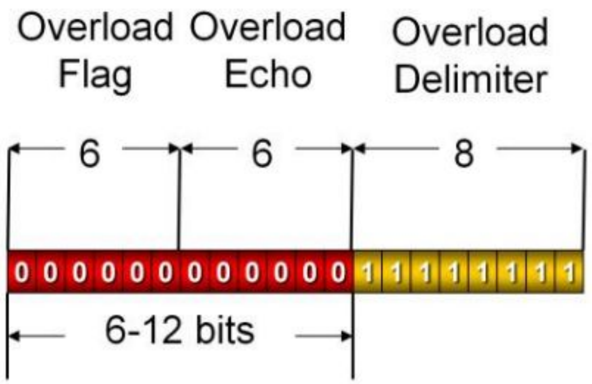


Controller Area Network (CAN 2.0B - Frames)

- Intermission Frame Space: 3 or more recessive bits
  - Contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area

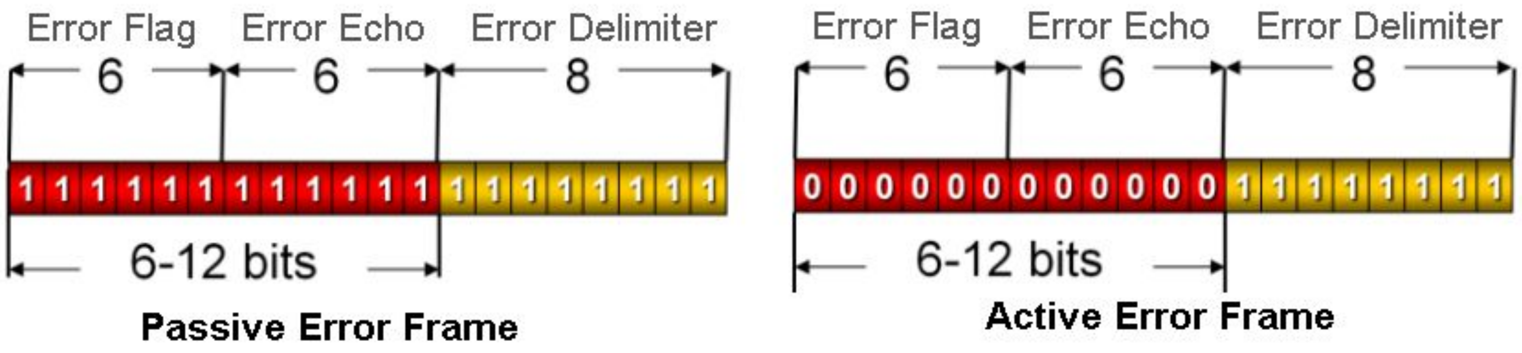


- Overload Frame
  - It is primarily used to provide an extra delay between messages for a busy receiver
  - Its format is similar to an active error frame and is transmitted during the interframe space, before the next data or remote farme
  - Up to two consecutive overload frames can be sent
  - Other nodes echo the Overload Flag



Controller Area Network (CAN 2.0B - Frames)

- Error Frames
  - Are transmitted by any node who detects an error with the data or remote Frame
  - Node in the error active state will transmit out an active error frame.
  - Node in the error passive state will transmit out a passive error frame.
  - Other nodes echo the error flag (if there is more than two nodes on the bus)
  - An error frame will destroy the current data or remote frame on the bus
  - The transmitter will retransmit the data/remote frame at the next available time

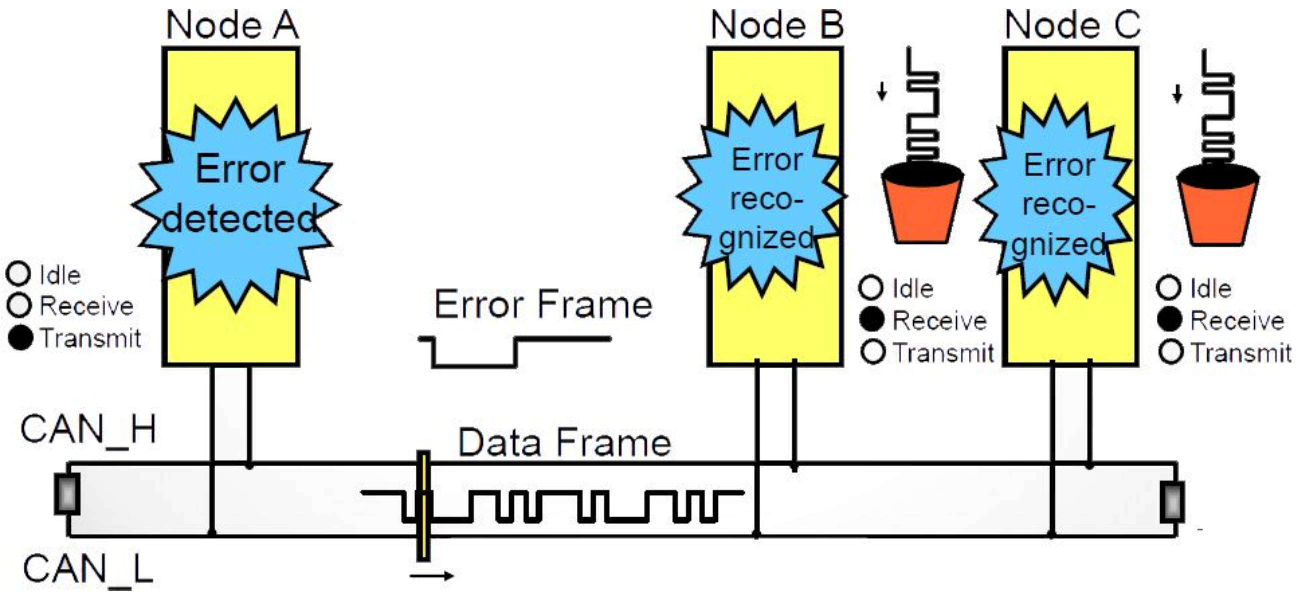


Controller Area Network (CAN 2.0B - Error Checking)

- There are 5 error detection mechanisms to ensure the integrity of messages
  - CRC Error Checking
    - Receivers calculate the CRC and verify it against the CRC received in the message
  - Acknowledge Error Checking
    - This error occurs when the transmitter detects a recessive bit in the ACK field of the message
  - Form Error Checking
    - This error occurs when a recessive bit in a message become dominant. E.g. ACK delimiter.
  - Stuff Error Checking
    - This error occurs when any node detects 6 consecutive bits of the same polarity between the SOF and end of CRC field. (***Note***: *error frames intentionally violate the bit stuffing rule*)
  - Bit Error Checking
    - This error occurs when the transmitter monitors a signal on the bus different from what it sent.
    - Exceptions: During arbitration and Acknowledgment

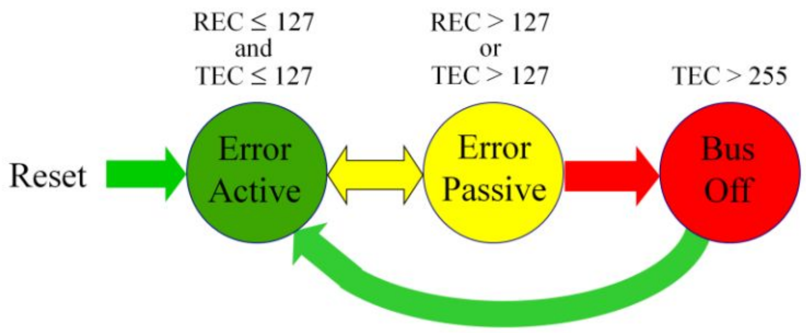
Controller Area Network (CAN 2.0B - Error Handling)

- All the nodes can detect error conditions
- Detected errors are made public to all other nodes via error frames
- Error frames will destroy the current data or remote frame on the bus
- The transmitter will automatically retransmit the message at the next available time



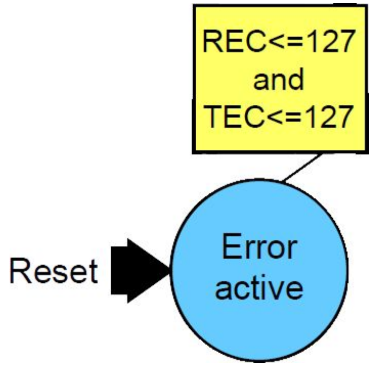
Controller Area Network (CAN 2.0B - Error Handling)

- The error conditions increment internal receive or transmit counters
- The error counters are updated according specific rules
  - For each failed transaction the error counters are incremented
  - For each successful transaction the error counters are decremented
- By using the RX and TX error counters a node is able to change its error state
- Every node has three error states according to its error counters
  - Error Active
  - Error Passive
  - Bus Off

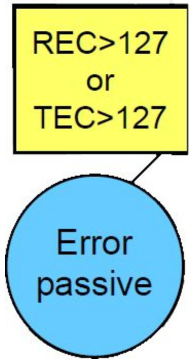


Controller Area Network (CAN 2.0B - Error Handling)

- ❖ Error Active Node State
  - ➢ This is the normal mode. It is the state after reset.
  - ➢ Allows to receive and transmit messages
  - ➢ Allows to transmit active Error Frames
  - ➢ Both the error counters are smaller than or equal to 127
  - ➢ A "warning" interrupt will be triggered when either error counters exceed 95

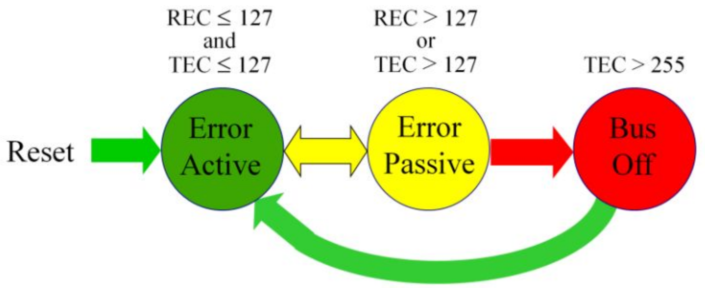


- ❖ Error Passive Node State
  - ➢ Either the REC or the TEC counter reach to128
  - ➢ Allows to receive and transmit messages
  - ➢ Allows to transmit passive Error Frames
  - ➢ After transmission, the node gets suspended in order to limit its disturbance of the bus
    - ■ It must wait 8 bit times longer than error-active nodes before it may transmit another message



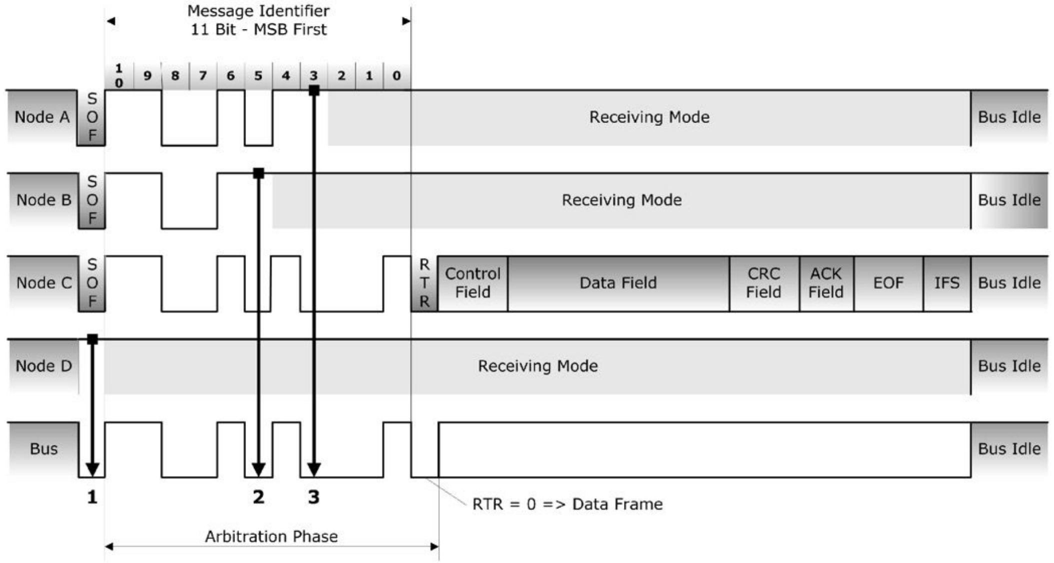
Controller Area Network (CAN 2.0B - Error Handling)

- Bus Off Node State
  - Transmit error counter exceeds 255
  - All bus activities are stopped and the node is dropped off the bus
    - The node can not transmit anything on the bus
    - The node can not receive data or remote frames
  - It prevents a single node from overloading the bus with error frames thus preventing any valid data frames onto the bus
- Recovering of a node From Bus Off
  - Reinitialize the node
  - Detects 128 occurrence of 11 consecutive recessive bits
    - Long bus idle or 128 valid messages, or a combination of both



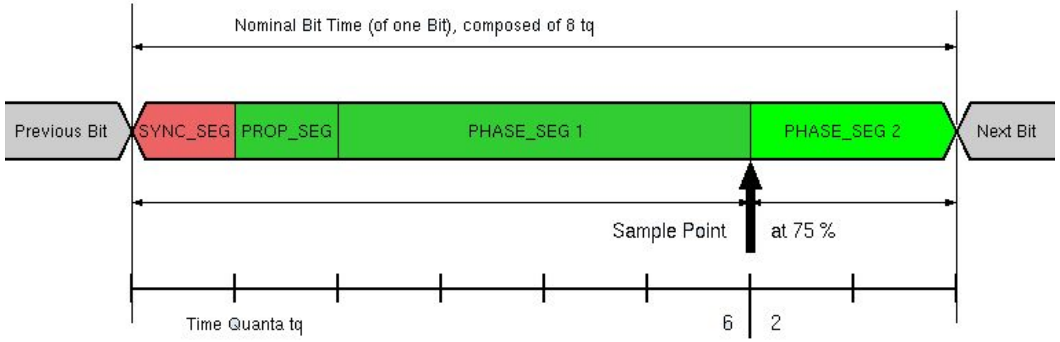
Controller Area Network (CAN 2.0B - Bus Arbitration)

- ❖ Arbitration is needed when multiple nodes try to transmit at the same time
- ❖ Only one transmitter can transmit at a time.
- ❖ Nodes wait for bus to become idle.
- ❖ Message importance is encoded in message ID
- ❖ Nodes with more important messages continue transmitting
- ❖ As a node transmits each bit, it verifies that the same bit value is on the bus
- ❖ A “0” on the bus wins over a “1” on the bus
- ❖ The losing node stops transmitting and winner continues



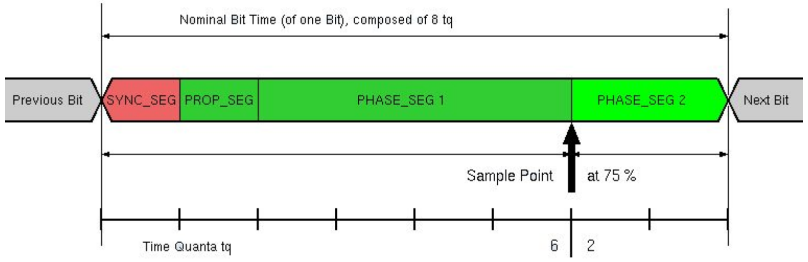
Controller Area Network (CAN 2.0B - Timing)

- CAN is an asynchronous serial bus with NRZ bit coding
- The NRZ bit coding does not encode a clock into the signal
- The receivers must synchronize to the transmitted data stream
  - To ensure messages are properly decoded
- The CAN protocol allows the user to program
  - The bit rate
  - The sample point of the bit
  - The number of times the bit is sampled



- The CAN bit time is made up of 4 segments
  - SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2
  - Each of these segments are made up of **integer** units called Time Quanta (**Tq**)

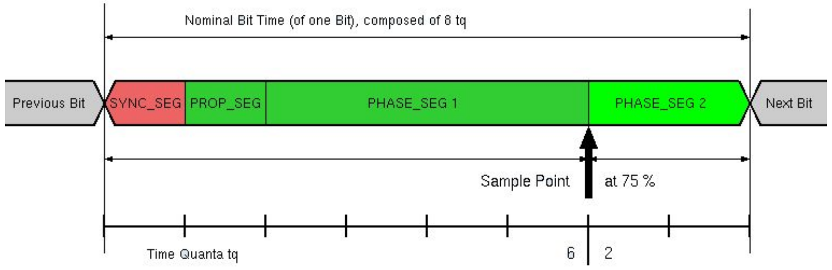
Controller Area Network (CAN 2.0B - Timing)



- Generally the bit timing parameters depend on
  - The physical bus propagation delays
  - The oscillator(clock) tolerances all over the system
- The Nominal Bit Rate is defined as the number of bits per second transmitted by
  - An ideal transmitter with no resynchronization (ideal oscillators).
  - NBR = 1 / t_{bit} and t_{bit} = SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2
- The Synchronization Segment (SYNC_SEG)
  - This segment is used to synchronize the nodes on the bus.
  - Falling edge of the bit is expected to occur within this segment.
  - This segment is fixed at **1 Tq**.

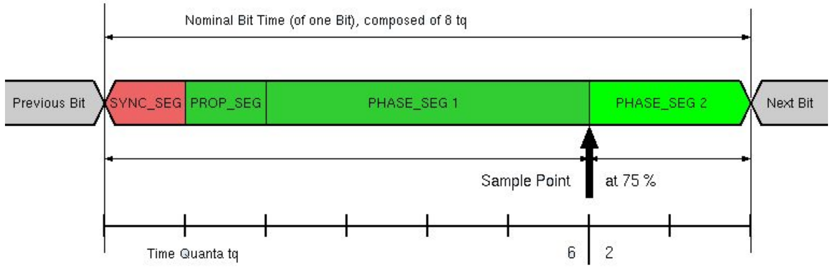
Controller Area Network (CAN 2.0B - Timing)

- The Propagation Segment (PROP_SEG)
  - This segment is used to compensate the physical delays between nodes
  - The propagation delay is defined as **twice the sum of**
    - the propagation time of the signal on the bus line and the delays associated with the bus drivers
  - The PROP_SEG is programmable from 1 - 8 Tq.
- The Two Phase Segments (PHASE_SEG1 and PHASE_SEG2)
  - PHASE_SEG1 can be **stretched** or PHASE_SEG2 can be **shrinked** by **resynchronization**
  - They are used to compensate for edge phase errors on the bus
  - PHASE_SEG1 is programmable from 1 - 8 Tq
  - PHASE_SEG2 is programmable from 2 - 8 Tq



Controller Area Network (CAN 2.0B - Timing)

- **The Sample Point(s)**
  - It is the point in the bit time in which the logic level is read and interpreted
  - If the sample mode is one sample per bit. It is located at the end of PHASE_SEG1
  - If the sample mode is three samples per bit
    - First two samples are taken prior to the end of PHASE_SEG1
    - And the third sample is taken at the end of PHASE_SEG1
    - Value of a bit is determined by a majority decision



- **The Resynchronization Jump Width (RJW)**
  - The bit clock is adjusted as necessary by **1 - 4 Tq** to maintain resynchronization within the Synchronization Segment in a message by stretching or shrinking one of the phase segments.
  - RJW is the max. number of Tq that a bit time can be changed by one resynchronization

Controller Area Network (CAN 2.0B - Timing)

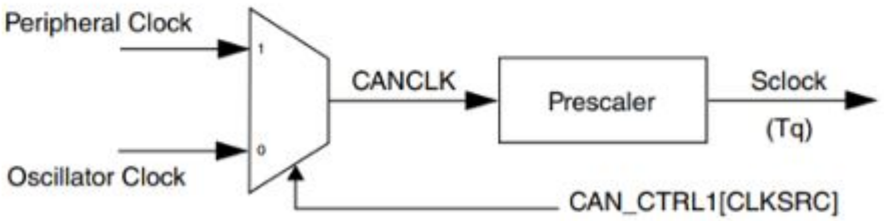
- Time Quanta (**Tq**)
  - It is determined by clock of the CAN controller
  - It is one period of the CAN controller clock
- The are some requirements for programming the CAN bit timing segments
  - Max number of the Tq in a bit is 25
  - PROP_SEG + PHASE_SEG1 ≥ PHASE_SEG2
  - 1 Tq ≤ PHASE_SEG1 ≤ 8 Tq
  - 2 Tq ≤ PHASE_SEG2 ≤ 8 Tq
  - RJW < PHASE_SEG2
  - RJW ≤ MIN (4, PHASE_SEG1)
  - SYNC_SEG = 1 Tq

$$t_{PROP\_SEG} = 2(t_{Bus} + t_{Tx} + t_{Rx})$$

$$PROP\_SEG = ROUND\_UP\left(\frac{t_{PROP\_SEG}}{t_Q}\right)$$

Controller Area Network (CAN 2.0B - Timing)

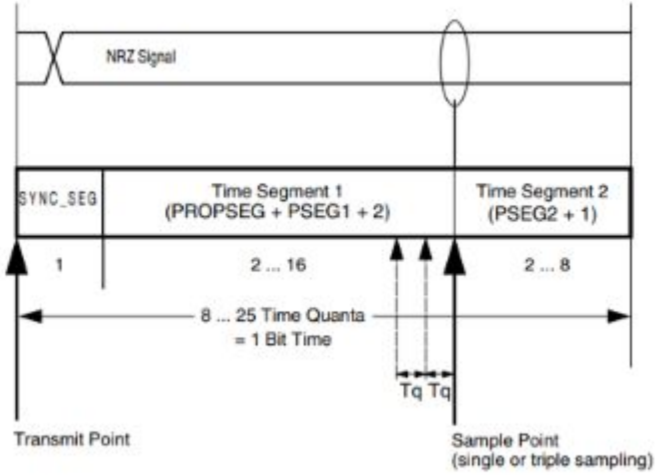
❖ Example; FlexCAN in MK64FX512VLL12 (CAN Controller in Teensy 3.5)



$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{\text{CANCLK}}}$$

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$



49.3.3 Control 1 register (CANx_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: 4002_4000h base + 4h offset = 4002_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	0		SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Controller Area Network (CAN 2.0B - Timing)

- Example: CAN Controller in Teensy 3.5 (FlexCAN in MK64FX512VLL12)
- Calculate the timing parameters for bit rate of 1 Mbps
  - If we use the peripheral clock (32 MHz)
  - The end to end length of the bus is 20m (propagation delay of twisted pair = 5 ns/m)
  - Propagation delay of the bus drivers is 150 ns

CLKSRC = 1, PRESDIV = 3 => Tq = (3 + 1) ÷ 32 MHz = 125 ns

Total propagation delay = 2 ✕ (150 + 20 ✕ 5) = 500 ns

Bit time = 1 ÷ Bit rate = 1 ÷ 1 Mbps = 1us => Bit time = 1us ÷ Tq = 1000 ns ÷ 125 ns = **8 Tq**

SYNC_SEG = **1 Tq**

PROPSEG = ROUNDUP (total propagation delay ÷ Tq) = ROUNDUP (500 ÷ 125) = **4 Tq**

PSEG1 = **1 Tq** and PSEG2 = **2 Tq**

RJW = MIN(4, PSEG1) = MIN(4, 1) = **1 Tq**

Controller Area Network (CAN 2.0B - Message Filtration)

- Message filtration is done by the CAN controller in order to
  - Offload the processor
  - Save RX buffer space
- Uses FILTER and MASK
  - The MASK is used to determine which bits of message IDs are compared with the FILTER
  - Mask 0: Don’t care. Mask 1: Match with filter
  - The “1”s in the ID must match the “1”s in the filter to be allowed to pass (bitwise AND)
- Only messages that match the filter “pass”
- Operates on the CAN message ID
- Example: we want to accept only frames with IDs of 0x00001560 thru to 0x00001567
  - Set the filter to 0x00001560
  - Set the mask to 0x1FFFFFF8

Controller Area Network (CAN 2.0B - Application)

- List all the signals on the bus
  - Signal name and description
  - Sender and receiver node(s)
  - Cycle time of the signal (reading and writing)
  - The priority/importance of the signal
  - Data type and range/value of the signal
- Pack and unpack signals in messages
  - Group the signals in messages by
    - Sender, importance, cycle time and etc.
  - Specify ID for the messages according the priorities
  - Automatically generate C libraries using a scripting language or a code generator

Controller Area Network (CAN Bus)

- Some useful links
  - SparkFun - How CAN BUS Works
  - CAN Bus Explained - A Simple Intro
  - Fun and Easy - How the CAN bus Protocol Works
  - CAN Bus System Explained
  - Introduction to CAN bus
  - Kvaser - CAN Protocol Tutorial (Part 1, Part 2, Part 3, Part 4, Part 5, Part 6, Part 7)
  - Introduction to the Controller Area Network (CAN)
  - Technical Comparison: CAN bus, CAN FD, and Ethernet
  - How CAN FD Improves CAN Real-time Performance