CAN Fundamentals

Simone SODERI

IMT School for Advanced Studies Lucca

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

- Overview the definitions of CAN bus
- Presentation of CAN bus technical specifications
- Present use cases
- Description of the database format used in the CAN bus

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

Introduction

- Controller Area Network (CAN) is a broadcast digital serial bus designed to operate at speeds from 10kbit/s to 1Mbit/s
- It was introduced by Bosch in the early 1980s for automotive applications

History

- **1983** Start of development of CAN at Robert Bosch GmbH
- **1986** v1.0 specification of CAN
  - at the Society of Automotive Engineers (SAE) conference in Detroit, Michigan.
- **1991** Specifications of the extended CAN2.0 protocol
  - Part 2.0A – 11-bit identifier
  - Part 2.0B – 29-bit identifier (extended frame format)
- **1993** First car equipped with CAN: Mercedes S-class
- **1994** First standard released: ISO 11898
- **1999** Explosion of CAN-linked equipment in vehicle and industrial applications
- **2012** Bosch released CAN Flexible Data-Rate (FD)

Why are we interested in the CAN bus?

- **Low cost**
  - Controller Area Network (CAN) was developed for use as an on-board network in vehicles. The adoption of the CAN bus has allowed to reduce the wiring on board of the vehicle and consequently the weight and the overall cost.
- **Built-in Error Detection**
  - The bitwise arbitration avoids multiple communication over the bus at the same time.
- **Robustness**
  - CAN bus lines are highly resistant to electrical disturbances;
  - Error handling scheme that results in retransmitted messages when they are not properly received.
- **Standard Physical layer**
  - It supports low-speed CAN (max 125 kbps) and high-speed CAN (max 1 Mbps)
- **Flexibility**
  - Message-based protocol: nodes on the bus have no identifying information associated with them. As a result, nodes can easily be added or removed.

CAN bus vs Ethernet (1/2)

WHAT	CAN BUS	ETHERNET
Hardware level	CAN bus works by a differential voltage between a pair of twisted lines	Ethernet bus is current-driven and is coupled through transformers at both ends, providing galvanic insulation from the network
Network architecture	multi-point with the management of the multi-master case	point-to-point (multi-point with additional devices)
Medium access	CSMA/CR (collisions-resolution method)	CSMA/CD (collisions occur and retransmission after random time)
Speed	1 Mbps	1 Gbps
Protocol efficiency	Best for short message	Best for longer message
Network extension	40 m @1 Mbps	100 m @1 Gbps
Cost	Low cost	3 to 5 times the cost of a CAN interface

CAN bus vs Ethernet (2/2)

WHEN	CAN BUS	ETHERNET
Data rate	≦ 1 Mbps	high bit-rate applications
Network architecture	easy network setup	wider network extension
Cost	low cost	higher than CAN bus
Signal interferences	robust communication for hostile environments (CAN bus works by a voltage differential between a pair of lines)	providing galvanic insulation (Ethernet bus is current-driven and is coupled through transformers at both ends)

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

CAN bus Standards (1/2)

- **ISO 11898** Road vehicles – Controller Area Network (CAN)
  - **Extensions to Data Link Layer**
    ISO 11898-4 Time-triggered CAN
    ISO 11898-5 relates to high-speed CAN and low-power applications
  - **Data Link Layer**
    ISO 11898-1 Data link layer and physical signalling
  - **Physical Layer**
    ISO 11898-2 High-speed medium access unit
    ISO 11898-3 Low-speed fault-tolerant medium-dependent interface

Bosch Specification CAN 2.0A and CAN 2.0B

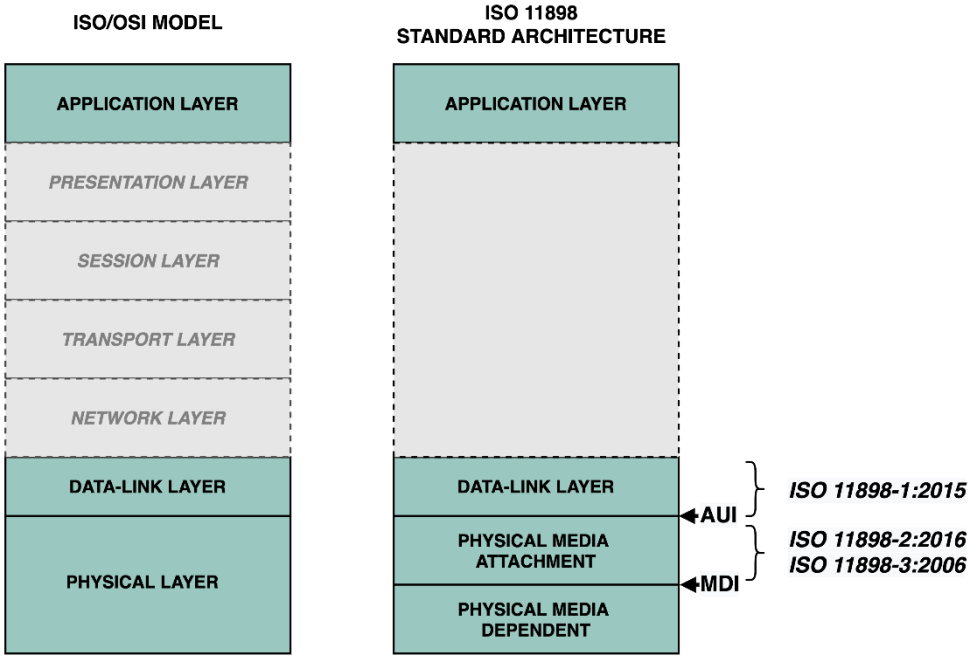
CAN bus Standards (2/2)

- Application standards that use ISO 11898 at lower layers
  - **SAE J2284:**
    High Speed CAN (HSC) For Passenger Vehicle Applications
  - **SAE J1939:**
    Recommended Practice for Control and Communications Network on Truck and Bus Applications
  - **CiA - CANopen:**
    Application layer and communication profile

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

CAN in ISO/OSI Reference Model

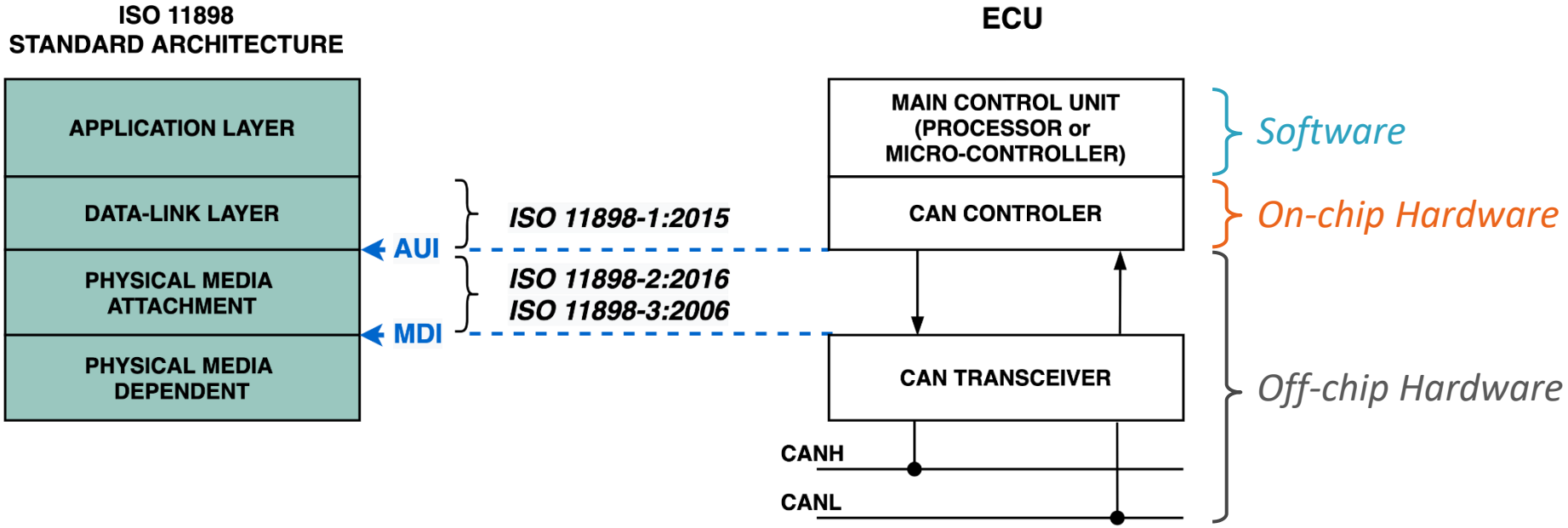


- In the ISO/OSI reference model, the original **CAN specification**, developed by Bosch, covers only the **Physical and Data link layers.**
- CiA and SAE provided their own specifications for application layer of the CAN protocol.
- In the OSI terminology, two parts are not defined by the original standard:
  - **AUI** = Attachment Unit Interface
  - **MDI** = Media Dependent Interface

Electronic Control Unit (ECU)

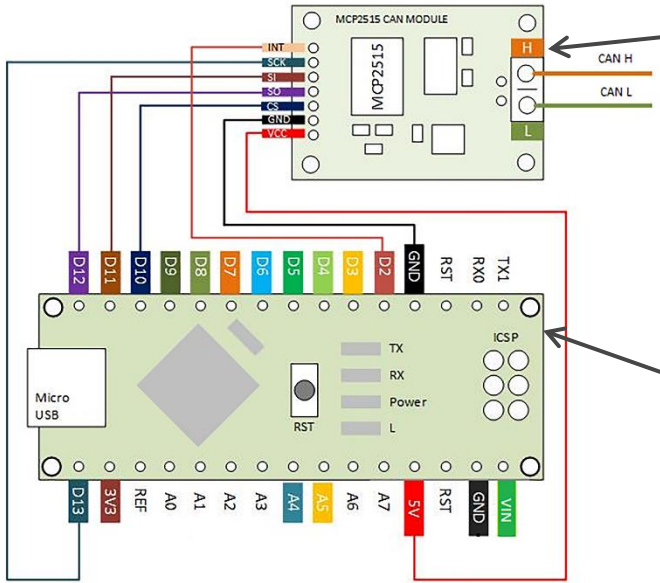
- In the CAN bus network's **nodes** are called Electronic Control Units (ECU).
- Each ECU refers to a **single function**
  - E.g., in a vehicle: the engine control unit, airbags, or audio system.
- ECUs communicate in **broadcast** through the CAN bus.
  - ECU prepare and broadcast information.
- The broadcasted data is read (received) by **all** other ECUs on the CAN network
  - Each ECU can then check the data and decide whether to process or ignore it (this depends on the ID field (which we explain later) within the received data).

Electronic Control Unit (ECU)



- ECU = CAN network's nodes
- Each ECU represents a single function in the vehicle

ECU example of implementation by Arduino and MCP2515



CAN Controller
(MCP 2515)
+
CAN Transceiver
(MCP 2551)

µP
(Arduino Nano)

CAN bus: protocol overview (1/2)

- CAN protocol uses a **broadcast transmission**, every node (i.e., ECU) receives every message.
- CAN is a **message-based protocol**. A message is organized in a structure called **frame**.
  - The data carried in each byte of the frame is defined in the CAN protocol.
- CAN specification covers physical and data-link layers.
- CAN **physical layer** defines
  - the encoding of bits into electrical signals
  - bit timing and synchronization
  - cable and connectors.
- CAN **data-link layer** defines
  - services for data transfer and remote data request.
  - conditions upon which received messages should be accepted and message filtering
  - framing, arbitration of the communication medium, error detection and management

CAN bus: protocol overview (2/2)

- When a CAN node **is ready to transmit data**, it checks to see if the bus is busy and then simply writes a CAN frame onto the network.
- The CAN frames that are transmitted **do not contain addresses** of either the transmitting node or any of the intended receiving node(s). Instead, an arbitration ID - that is unique across the network - labels the frame.
- All nodes on the CAN network receive the CAN frame, and, depending on the arbitration ID of that transmitted frame, each CAN node decides whether to accept the frame, or not.
- If **multiple nodes try to transmit a message** onto the CAN bus at the same time, the node with the highest priority (lowest arbitration ID) automatically gets bus access.

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

CAN Physical Layer: overview

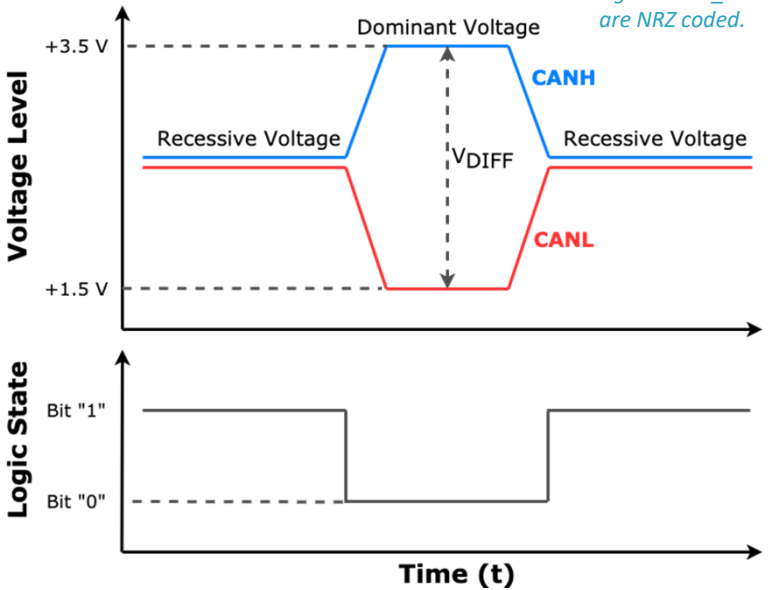
- **Physical Line Signalling:** responsible for *bit-by-bit* translation of data into/from HW
  - Bit Timing
  - Bus Encoding
  - Synchronization
  - Error detection
- **Physical Medium Attachment (PMA)**
  - Transmitter/Receiver Specification
- **Medium-Dependent Interface (MDI)**
  - Bus Cable, Bus Termination Networks, Connectors

CAN Physical Layer: bit timing (1/2)

- CAN physical layer **transmission model**, relies on
  - **dominant signals**, or CAN High (CAN_H) (encoded as logical 0 s)
  - **recessive signals**, CAN Low (CAN_L) (encoded as logical 1 s).

- **CAN_H** and **CAN_L** signals are digital with **Non-Return to Zero (NRZ)** bit encoding:
  - the polarity of the signal on the bus changes only when the incoming signal changes from a 1 to a 0 or from a 0 to a 1;
  - With NRZ the signal is constant for the whole bit time;
  - NRZ encoding ensures a minimum number of transitions.

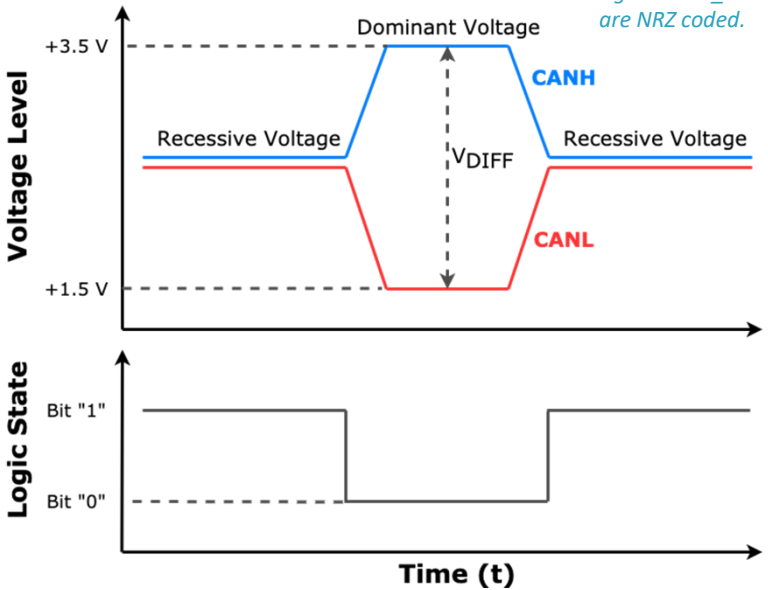
In the CAN bus, the single signals CAN_H and CAN_L are NRZ coded.



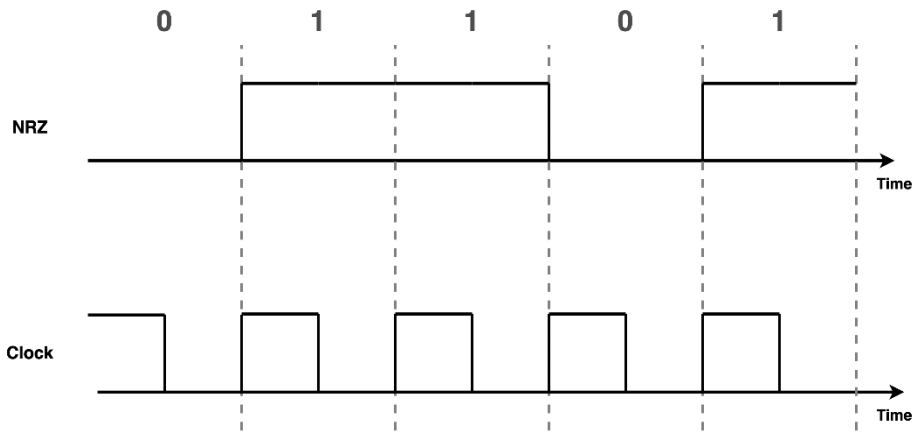
CAN Physical Layer: bit timing (2/2)

- If multiple ECUs try to drive the bus state at the same time, the **“dominant”** configuration always prevails upon the **“recessive”**
- CAN is based on transmission of **differential voltages** (differential signal transmission).
  - This effectively **eliminates the negative effects of interference voltages** induced by motors, ignition systems and switch contacts

In the CAN bus, the single signals CAN_H and CAN_L are NRZ coded.



Non-Return-to-Zero (NRZ)

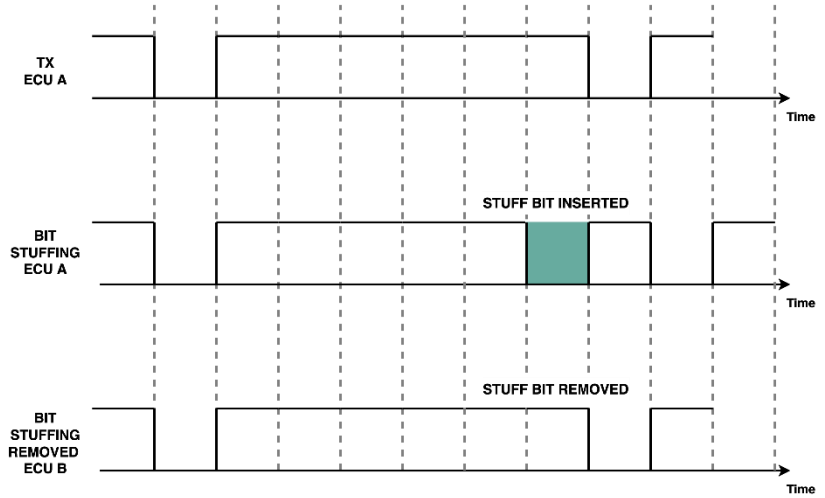


In the CAN bus, the single signals CAN_H and CAN_L are NRZ coded.

- **binary signals** to be transmitted are mapped directly:
  - a logic “1” to a high level
  - a logic “0” to a low level.
- Characteristic of NRZ coding is that consecutive bits of the same polarity exhibit no level changes.
- The use of NRZ encoding ensures a minimum number of transitions and high resilience to external disturbance.
- NRZ coding has no synchronization properties.
  - If no level change occurs over a longer period, the receiver loses synchronization.
  - **CAN bus** uses bit stuffing as the synchronization mechanism.

CAN Physical Layer: bus encoding

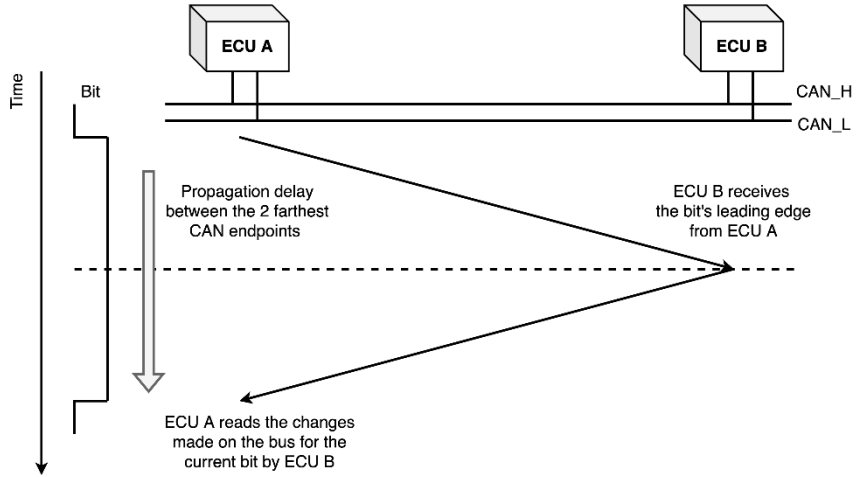
- Bus Encoding: NRZ with bit stuffing
  - Standard NRZ may have **not enough signal edges for synchronisation**
    - Without fall edges the device might lose the bit timing synchronization
    - We do need to insert fall edges artificially
  - **Bit stuffing**: after 5 consecutive bits with a same polarity (dominant or recessive):
    - The transmitter inserts a bit with complementary polarity;
    - Receiver removes the bit inserted



CAN Physical Layer: synchronization

- Nodes are requested to be **synchronized on the bit edges** so that every node agrees on the value of the bit currently transmitted on the bus.
- Each node implements a protocol that keeps the receiver **bit rate aligned** with the actual rate of the transmitted bits → bit length known!
- Long sequences without transitions avoided by applying **bit stuffing**
- **Hard synchronization** with dominant Start Of Frame (SOF) bit after bus idle
- No synchronization with dedicated clock

CAN Physical Layer: bit rate ⇆ bus length



- To calculate the maximum bus length, we must **consider the farthest ECU** (ECU B) from the transmitter (ECU A) that can change polarity on the bus.
- Empirically we can calculate the **speed of the CAN signal** with the following formula

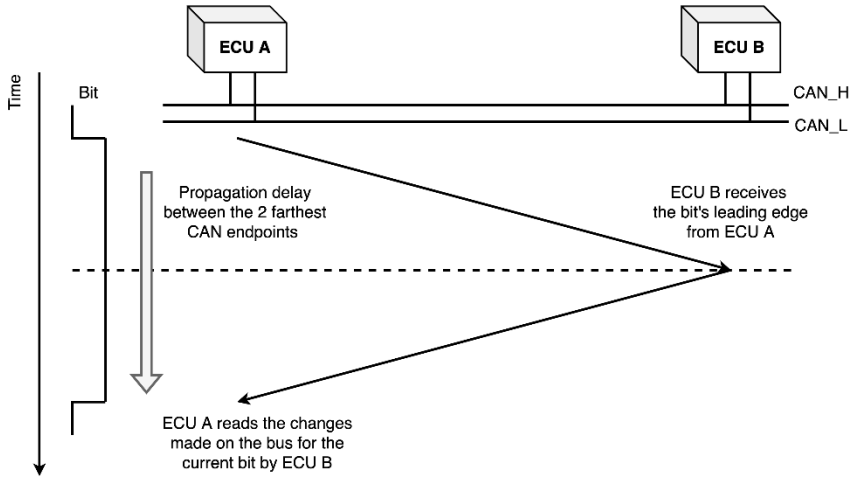
$$signal\ speed = \frac{1}{cable\ delay} \cong 0.3 * speed\ of\ light$$

- The type of arbitration implies that the **bit time is at least twice the propagation latency** on the bus

$$Max\ bus\ length < \frac{(signal\ speed * Bit\ time)}{2}$$

CAN Physical Layer: bit rate ⇆ bus length

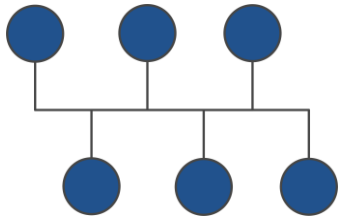
Bit rate	Bit time (ms)	Max Bus length (m)
1 Mbps	1	45
800 kbps	1.25	55
500 kbps	2	90
250 kbps	4	180
125 kbps	8	360
62.5 kbps	16	720
20 kbps	50	2250
10 kbps	100	4500



The type of arbitration implies that the bit time is at least twice the propagation latency on the bus

CAN Physical Layer: topology

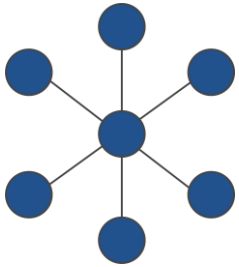
BUS



All the nodes are connected in line with each other

industrial application (assembly line plant)

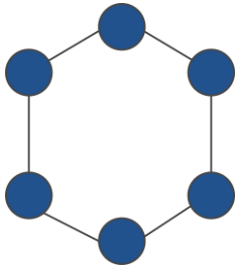
STAR



All the nodes are connected to a central hub

automotive or robotics applications

RING



Each node connects to exactly two other nodes

used for high reliability and fault tolerance applications

CAN Physical Layer: topology

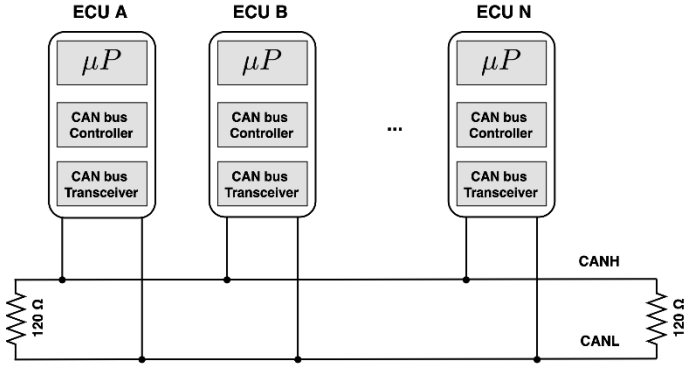
- The wiring-topology of a CAN-network **should be as close as possible to a single line structure** in order to avoid cable-reflected waves.
  - Avoiding complicated network structures.

- The topology **affects the bit-rate**. Simpler topologies allow for higher bit-rates.

CAN Physical Layer: bus

- Automotive CAN according to ISO 11898-2/3 uses **twisted** pair with differential voltages on a bus topology (tolerant to single wire disturbance)

- The bus must be terminated with resistor of **120 Ω** to:
  - remove signal reflections at the end of the bus
  - ensure the bus gets correct voltage levels



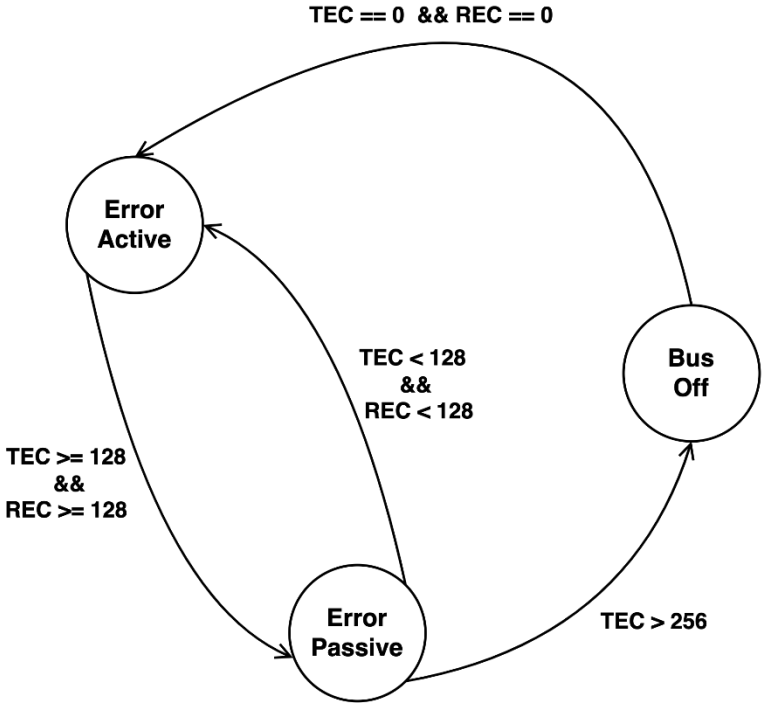
CAN Physical Layer: error detection

- Sender
  - **Bit errors:** Senders monitor the bus during transmission. Differences between the bit sent and the bit monitored are considered as transmission errors.
- Receiver
  - **Stuff error**: Receiving nodes can detect errors by checking if the bit sequence is compliant with the bit stuffing rule (e.g., an ECU receives 5 (or more) consecutive bits with a same polarity).
- An ECU detecting an error transmits an ERROR FLAG.

CAN Physical Layer: error management

- Each node can be in **3 states**:
  - **Error active:** the unit can normally take part in bus communication
  - **Error passive**: units in this state are suspected of faulty behaviour have limited error signaling capacity
  - **Bus off**: units in this state are very likely corrupted and cannot have any influence on the bus.

- For **fault confinement**, each node has two counters:
  - Transmit Error Count (TEC)
  - Receive Error Count (REC)



CAN Physical Layer: fault confinement

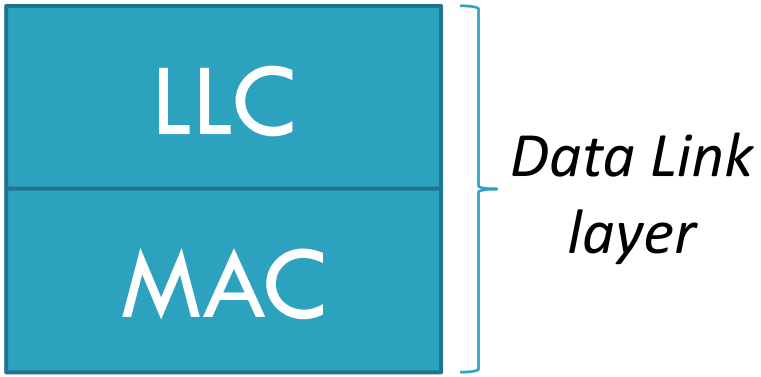
- Fault confinement is also a major benefit of CAN.
- Faulty nodes are **automatically dropped from the bus.** This prevents any single node from bringing a network down and assures that bandwidth is always available for critical message transmission.
- **Hot-plugging:** the fault containment also allows nodes to be added to a bus while the system is in operation, aka *hot-plugging*

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

CAN Data Link Layer: overview

- Logical Link Control (LLC)
  - Data transfer and remote request
  - Message filtering
  - Flow control
  - Error recovery
- Medium Access Control (MAC)
  - Message Framing
  - Arbitration of the communication
  - Acknowledgement management
  - Error detection
  - Monitoring the error status and limiting ECU's operations



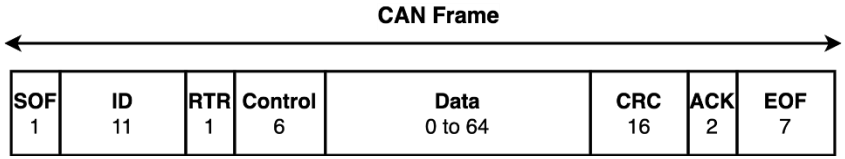
CAN Data Link Layer: data-sheet

- Message framing
  - CAN 2.0A (standard format): 11 bits identifier
  - CAN 2.0B (extended format): 29 bits identifier
- Broadcast
  - Every ECU on the bus receives all messages and filters what it needs
- Real Time
  - Being contention-free medium, CAN assures real-time capabilities
- Prioritization
  - The message identifiers assures non-destructive arbitration
- Error Management
  - Error detection and error handling assures ECU's operations confinement

CAN Data Link Layer: frames types

- **Data frame:**
  - Data frames are used to transmit information between a source node and one or more receivers. Data frames **do not use explicit addressing** to identify the message receivers. Instead, each receiver node states the messages that will be received based on their information content, which is encoded in the Identifier field of the frame.
- **Remote frame:**
  - A remote frame is used to request the transmission of a message with a given identifier from a remote node.
- **Error frame:**
  - The Error frame is not a real frame, but rather the result of error signaling and recovery action(s).
- **Overload frame:**
  - The purpose of the Overload frame is to inject a bus state at the end of a frame transmission that prevents the start of a new contention and then transmission.
  - It is transmitted in 2 cases: when receiver needs delay internally or when a dominant bit is detected during Intermission
- **Interframes:**
  - Data frames and remote frames are separated by an Interframe space (7 recessive bits) on the bus.

CAN Data Link Layer: CAN 2.0A Data Frame



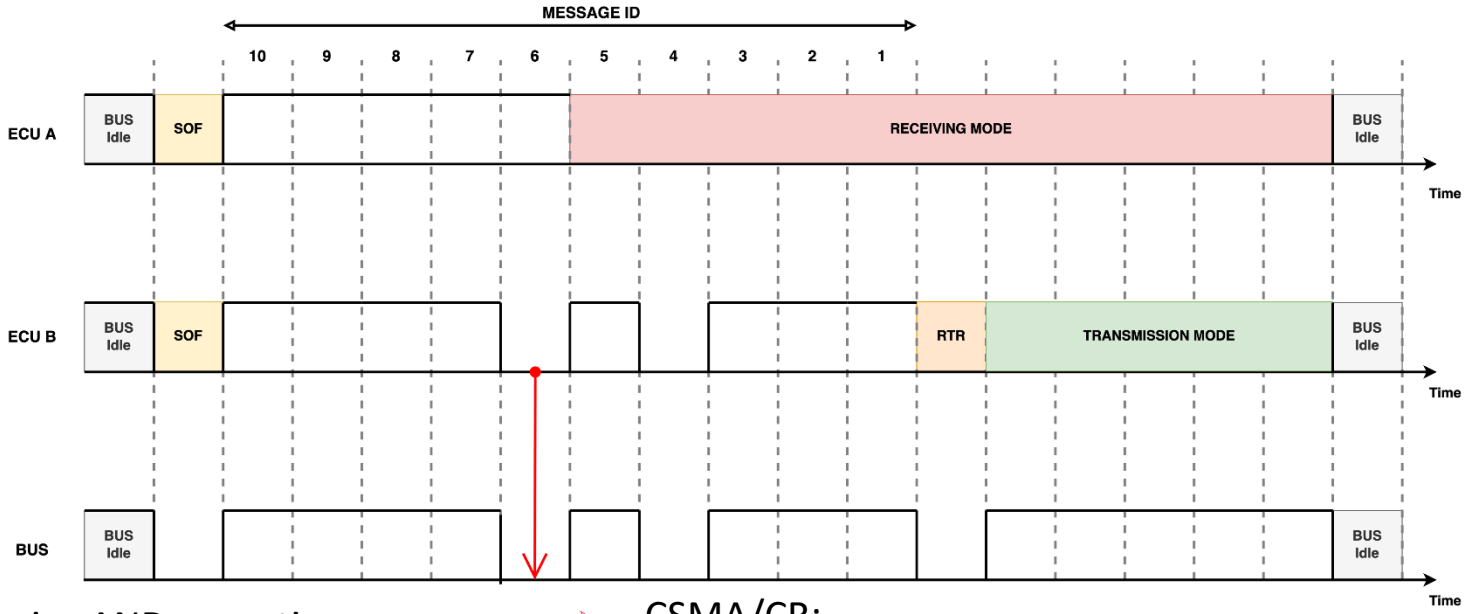
Most Significant Bit (MSB) - First

- **SOF (Start of Frame):**
  - Single dominant bit
- **ID: Arbitration field:**
  - 11 bits identifier
- **Control:**
  - 6 bits partially reserved
- **RTR (Remote Transmission Request):**
  - RTR = Dominant for data frames
  - RTR = Recessive for remote frames
- **Data:**
  - Payload consists of 0…64 bits
- **CRC (Cyclic Redundancy Check):**
  - 15 bits CRC + 1 recessive bit
- **ACK:**
  - 1 bit for acknowledgement + 1 recessive
- **EOF (End of Frame):**
  - 7 recessive bits

CAN Data Link Layer: arbitration (1/2)

- Let's assume we have two nodes that want to communicate on the bus: ECU A and ECU B
- All ECUs are synchronized on the SOF bit
- Each begins to communicate by ignoring the existence of the other
- The bus access is handled via the protocol Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) with **non-destructive arbitration**.
  - The bus behaves as a **wired-AND**
  - MAC protocol implements the arbitration using the ID field for the wired-AND
  - Collision of messages is avoided by bitwise arbitration without loss of time
- Arbitration solves the problem on which ECU can continue to transmit

CAN Data Link Layer: arbitration (2/2)



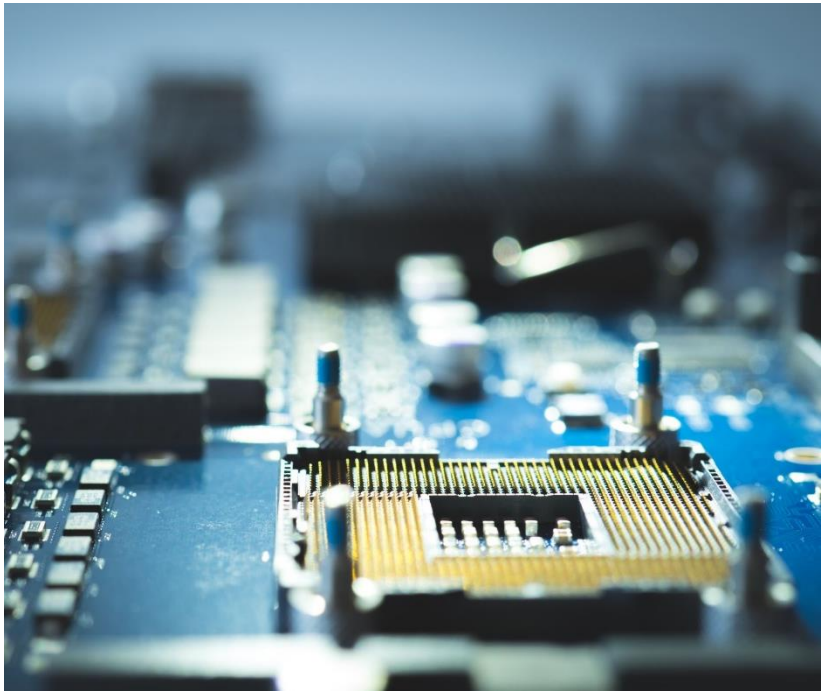
- Bitwise AND operation:
  - Bits contained in the ID
- CSMA/CR:
  - Non-destructive CSMA/CR: sender with lowest identifier (=highest priority) wins the arbitration

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

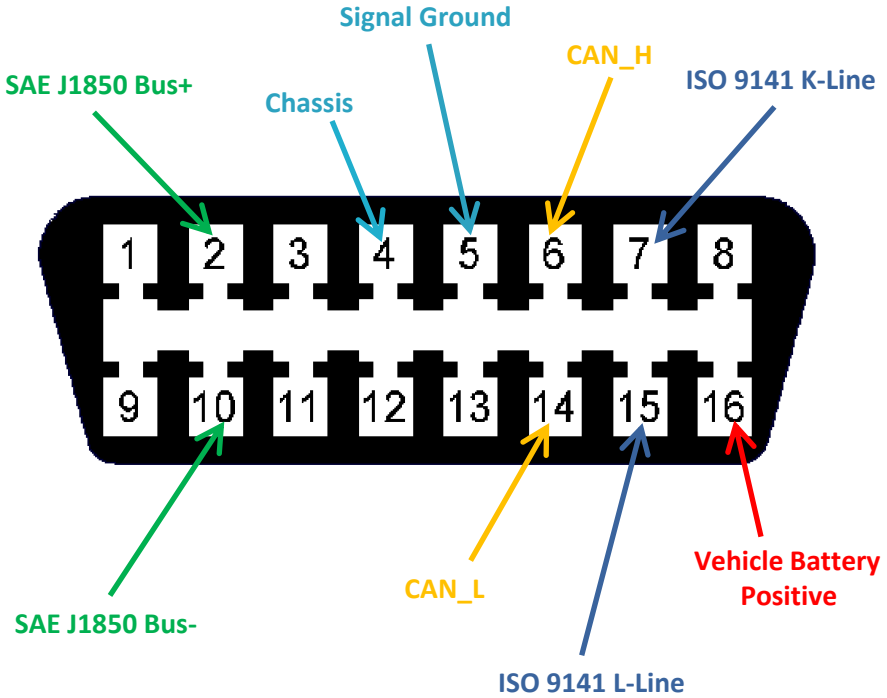
Application fields

- Automotive, transportation, aviation, space, maritime industry
- Industrial and home automation
- Medical equipment
- Household appliances
- Consumer electronics

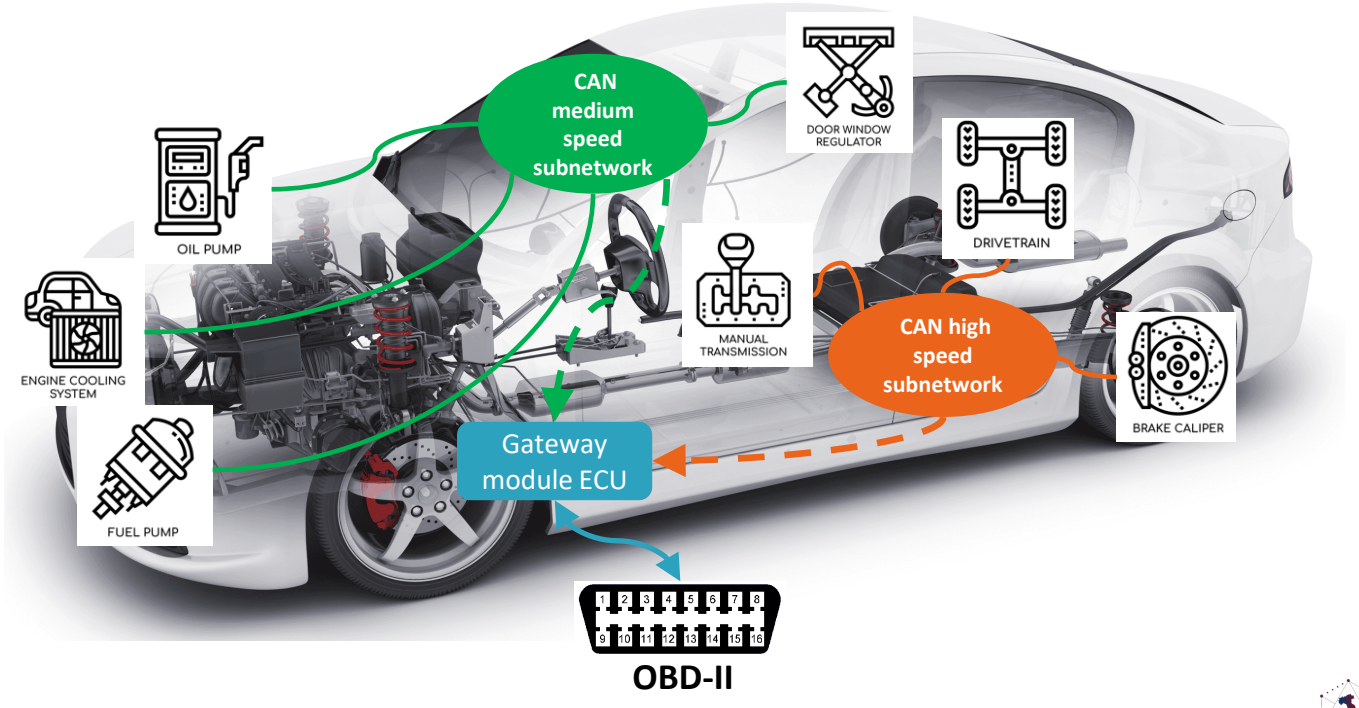


Vehicles: On-board diagnostics (OBD)-II

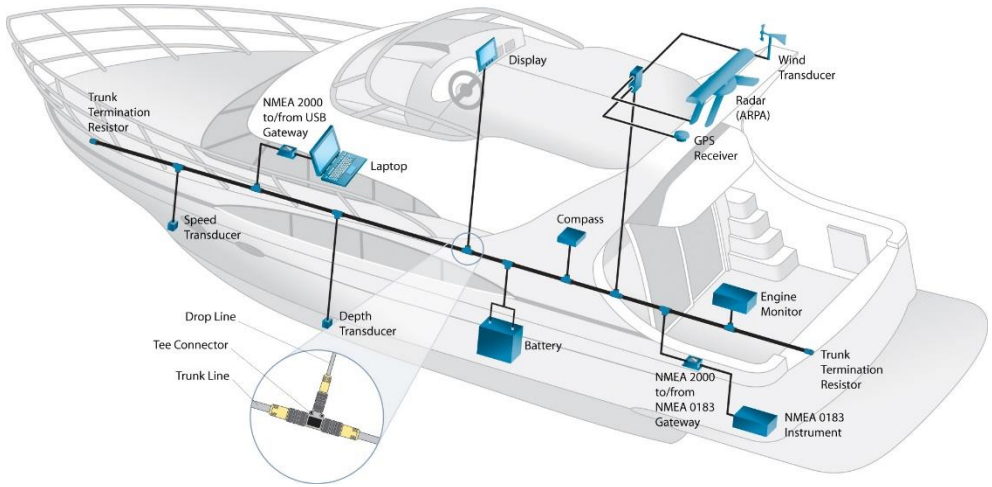
- OBD-II provides access to data from the engine control unit (ECU) and offers a **valuable source of information** when troubleshooting problems inside a vehicle.
- Real-time data in addition to a standardized series of diagnostic trouble codes (DTCs) which allow a person to **rapidly identify and remedy malfunctions** within the vehicle.



CAN bus: car scenario



Maritime Communications over the CAN



Source:
https://upload.wikimedia.org/wikipedia/commons/b/bf/NMEA2000_Modified_motor_yacht.jpg

- National Marine Electronics Association (NMEA) standards are the primary specifications employed on boats and ships of all sizes to interconnect instrumentation
- NMEA 2000 is the standard that operates over the CAN bus

Outline

- Introduction
- Standards
- Model
- CAN Physical Layer
- CAN Data Link Layer
- Applications
- CAN database

CAN database: why?

- **Message-based communication protocol:**
  - In CAN bus the nodes on the bus have no identifying information associated with them.
  - Hot-plugging: nodes can easily be added or removed without performing any software or hardware updates on the system.

*CAN bus **database** describes all messages on the network and makes them available in a readable format*

CAN database: DBC file

- The CAN DBC file format was developed by Vector Informatik GmbH in the 1990s.
- DBC is an **ASCII-based** translation, and the type of data that communicates over a **CAN bus can be read and understood using DBC files.**
- The DBC standard is proprietary, but it is also widely used, making it a de-facto standard!

```
CAN ID    Data bytes
0DA00500  FF FF FF 28 32 FF FF FF
```

RAW

```
Message       Signal       Value  Unit
EngineMsg     EngineRPM    1605    rpm
```

DBC

CAN database: DBC file

- **DBC contains:**
  - Messages
  - Information to decode message/signals from the CAN bus data stream.
- **Message:**
  - It is a group of signals
  - A message is a single transmission on the CAN bus
  - All signals within a message are sent/received at the same time
- **Signal:**
  - It contains the magnitude and the value of interest within the scope of the message

DBC: Message

Syntax: A message starts with **BO_** followed by

- **Message ID:**
  - 11-bit identifier (29-bit for CAN extended version)
  - DBC adds 3 extra bits
- **Message Name:**
  - The name must be unique, 1-32 characters and may contain [A-z], digits and underscores. It is used to help identify the message
- **Message Length:**
  - Number of bytes of the message
- **Sender:**
  - Used identify which device transmitted a message on the bus
- **Bus name:**
  - determines the bus on which this message is decoded when using a multi-bus accessory

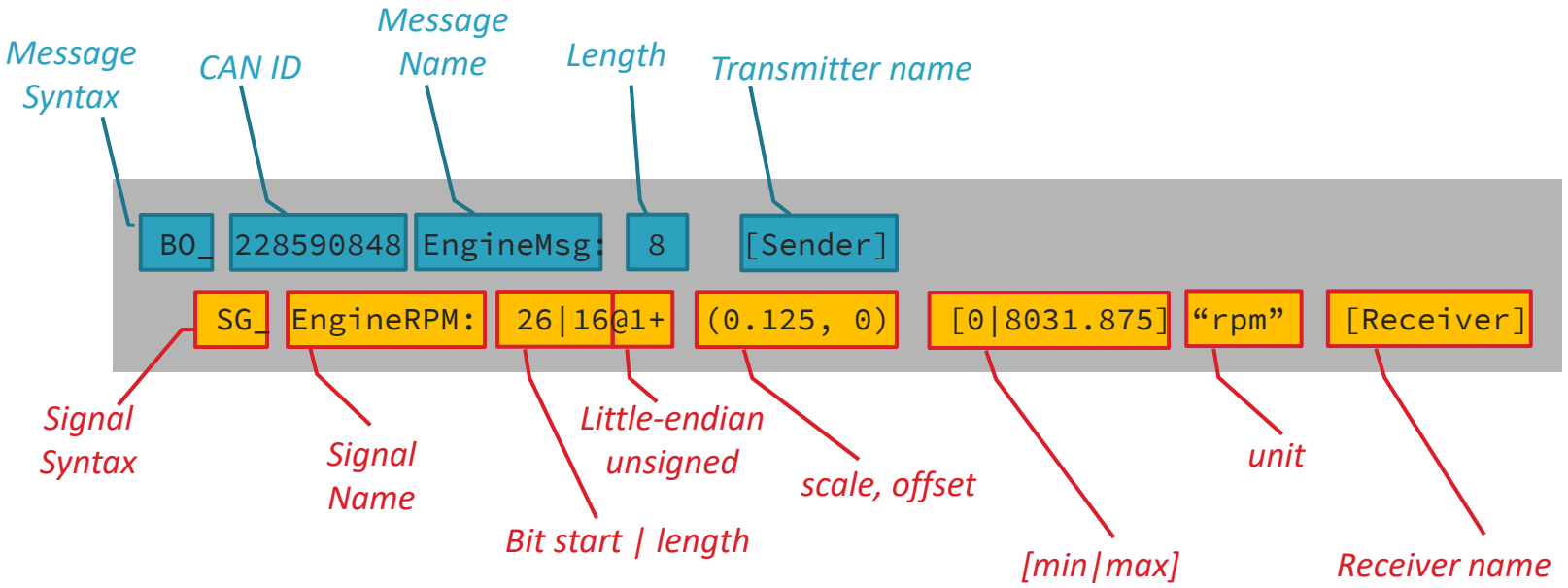
DBC: Signal

Syntax: A signal starts with **SG_** followed by

- **Signal Name:**
  - The name must be unique, 1-32 characters and may contain [A-z], digits and underscores. It is used to help identify the signal
- **Bit start:**
  - counts from 0 and marks the start of the signal in the data payload
- **Signal Length:**
  - Number of bytes of the message
- **Byte order @x:**
  - **@1** specifies that the is little-endian/Intel; **@0** for big-endian/Motorola
- **Signed/unsigned:**
  - + informs that the value type is unsigned. Instead, - is used for signed signals
- **Scale/offset:**
  - The (scale,offset) values are used in the physical value linear equation
  - physical_value = offset + scale * raw_value_decimal
- **[Min|Max]:**
  - [min|max] and unit are optional meta information
- **Receiver:**
  - The receiver is the name of the receiving node

DBC: Rules to decode

DBC file includes **rules** that describe how to **decode** CAN messages and signals



CAN Fundamentals

Simone SODERI

IMT School for Advanced Studies Lucca

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

CAN Fundamentals
In depth-material

Simone SODERI
IMT School for Advanced Studies Lucca

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

Access to the Medium

- When node has packet to send
  - transmit at full channel data rate R.
  - no a priori coordination among nodes

- two or more transmitting nodes → "**collision**"

- **MAC protocol** specifies the **access to the medium** and specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)

MAC Protocols

- MAC protocols control the message transmission

- Examples are
  - Random access protocols such as
    - **CSMA/CD** (carrier sense multiple access/collision detection) → Ethernet
    - **CSMA/CA** (carrier sense multiple access/collision avoidance) → WLAN
    - **CSMA/CR** (carrier sense multiple access/collision resolution) → **CAN bus**
  - Fixed-assignment protocols such as
    - TDMA (time division multiple access)
    - FTDMA (flexible TDMA)

Carrier Sense Multiple Access (CSMA)

- Carrier Sensing:
  - check if the medium is **idle** or **busy** before starting transmitting
- If the medium is busy, CSMA protocols wait for some time (the **backoff time**) before a transmission is tried again.

CSMA/CD

- CSMA/CD: carrier sensing as in CSMA
- Collision Detection
  - collisions detected within short time
  - colliding transmissions aborted → reducing channel wastage
  - **Retransmission**: wait for some time (generated by the backoff algorithm) before retransmitting the message in order to reduce the risk of the same messages colliding again.
- However, due to the possibility of successive collisions, the temporal behaviour of CSMA/CD networks can be somewhat hard to predict!

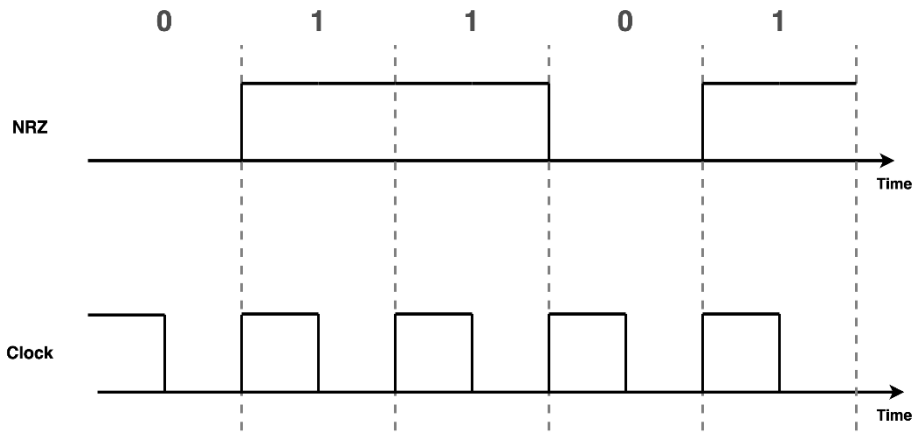
CSMA/CA

- CSMA/CD: carrier sensing as in CSMA
- In some cases, it is not possible to detect collisions.
  Examples:
  - using a wireless medium often makes it impossible to simultaneously read and write (send and receive) to the medium.
- Collision Avoidance
  - usage of some handshake protocol in order to guarantee a free medium before the initiation of a message transmission.

CSMA/CR

- CSMA/CD: carrier sensing as in CSMA
- Collision Resolution
  - Once there is collision CSMA/CR does not go into a backoff mode
  - **resolves collisions** by determining one of the message transmitters involved in the collision that is allowed to go on with an uninterrupted transmission of its message.
  - The other messages involved in the collision are retransmitted at another time
- Due to the collision resolution feature of CSMA/CR, it has the possibility to become more predictable in its temporal behavior compared to CSMA/CD

Non-Return-to-Zero (NRZ)



In the CAN bus, the single signals CAN_H and CAN_L are NRZ coded.

- binary signals to be transmitted are mapped directly:
  - a logic “1” to a high level
  - a logic “0” to a low level.
- Characteristic of NRZ coding is that consecutive bits of the same polarity exhibit no level changes.
- The use of NRZ encoding ensures a minimum number of transitions and high resilience to external disturbance.
- NRZ coding does not have any synchronization properties.
  - If no level change occurs over a longer period, the receiver loses synchronization.
  - **CAN bus** uses bit stuffing as the synchronization mechanism.

CAN Fundamentals
In depth-material

Simone SODERI
IMT School for Advanced Studies Lucca

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it