CAN - Controller Area Network

Introduction and applications

Anna Bonaldo

anna.bonaldo@bluewind.it

bluewind

Introduction to Controller Area Network (CAN)

Definition 1

A controller area network (CAN bus) is a vehicle bus standard designed to allow microcontrollers and devices.

From Wikipedia - CAN bus

Definition 2

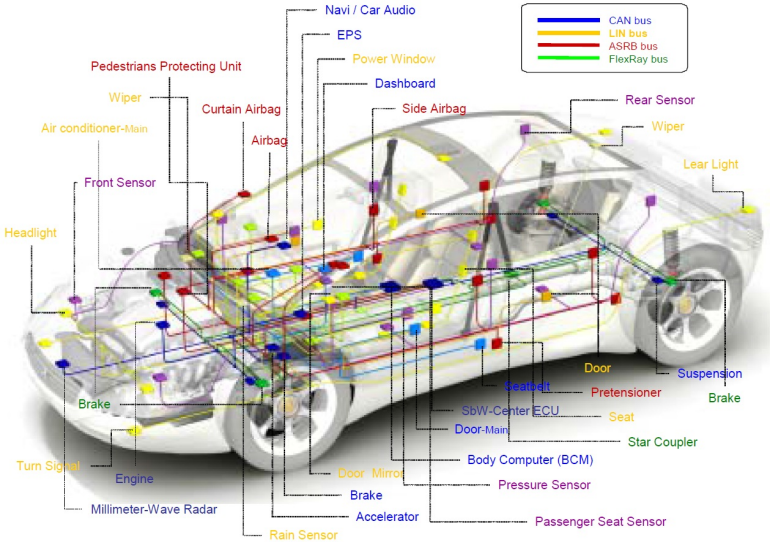
The CAN bus is a serial communication bus, designed for robust performance within harsh environments, primarily in industrial and automotive applications.

Applications

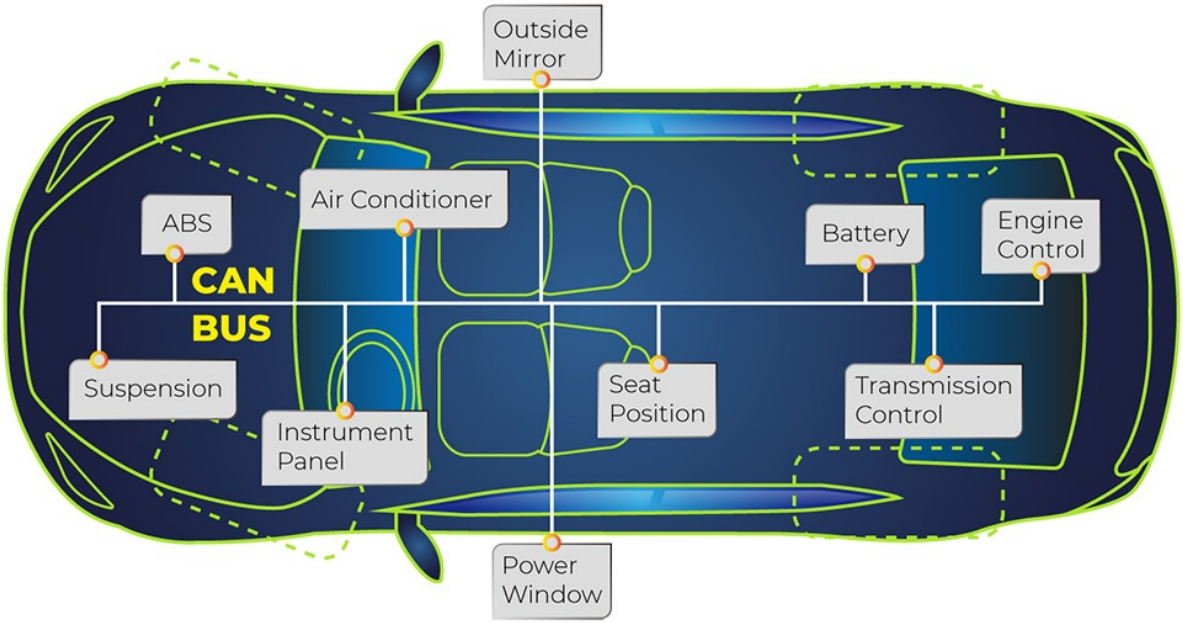
- Passenger vehicles, trucks, buses (combustion vehicles and electric vehicles)
- Electronic equipment for aviation and navigation
- Industrial automation and mechanical control
- ...
- Model railways/railroads
- Ships and other maritime applications
- ...
- Robotics/Automation

From Wikipedia - CAN bus

Communication busses in a moder vehicle



CAN bus application (automotive)



Why CAN bus?

- It has been introduced by Bosch to support communication in industrial automation environments
- When CAN bus was created, its goal was to make wiring simpler and electronics less complicated.

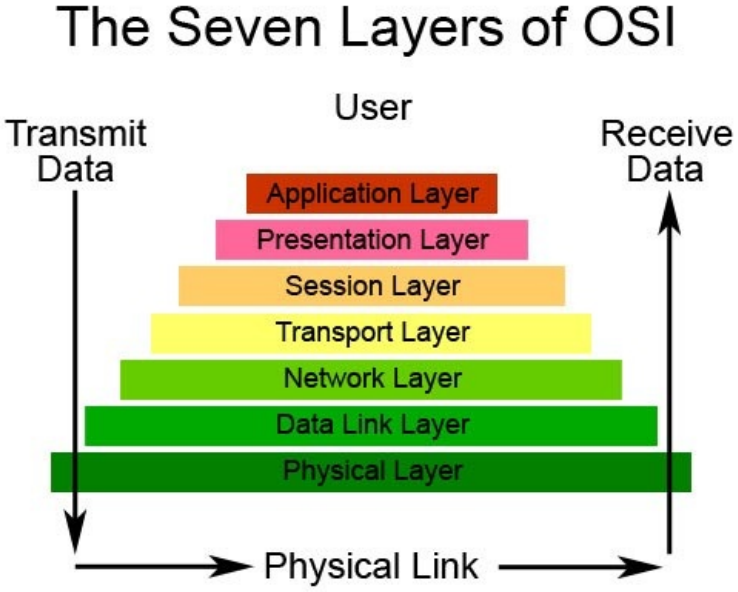
Top 5 CAN Bus Protocol Advantages

Many vehicles now use the CAN bus system because of its clear benefits, such as:

- Durability
- Cost-Effective
- Speed Varieties
- Adaptable Design
- Efficient Data Handling

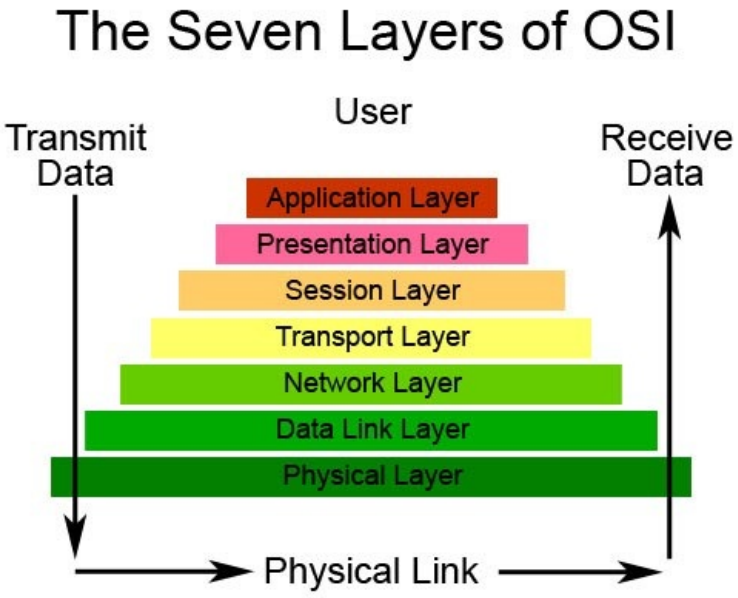
What is CAN bus?

The Controller Area Network (CAN bus) is a communication network designed to interconnect MCU-based boards using the “computer network paradigm”



CAN Bus is defined in ISO 11898 but this only covers the bottom two layers.

Other standards such as CAN Open and SAE J1939 are extensions to the CAN standard that define high level layers.



ISO 11898-1 (Data Link Layer):

- Addresses the communication protocol and its mechanisms.
- Defines message framing, arbitration, error handling, and more.
- Focuses on how data is transmitted and received within the network.

ISO 11898-2 (Physical Layer):

- Describes cable types, node requirements, and electrical signal levels.
- Specifies cable lengths: 40 meters for 1 Mbit/s and 500 meters for 125 kbit/s.
- Outlines factors like cable impedance, baud rate, and necessary terminations.
- Termination: Every CAN bus must be properly concluded with a 120 Ohms resistor at each bus end.
- CAN Nodes Connection: Linked via a two-wire bus system with baud rates of up to 1 Mbit/s for standard CAN and 5 Mbit/s for CAN FD.

What are the Key Components within a CAN Bus System?

1. Electronic Control Units (ECUs)

In an automotive CAN bus system, ECUs can e.g. be the engine control unit, airbags, audio system etc.

ECUs are the 'brains' of the CAN Bus, controlling specific functions like engine management or airbag deployment in vehicles. They process and act on the information received via the CAN Bus.

A modern car may have up to 70 ECUs - and each of them may have information that needs to be shared with other parts of the network.

2. CAN Controller

This component acts as a bridge between the ECUs and the CAN network. It manages the sending and receiving of data packets, ensuring messages are correctly formatted and transmitted.

3. CAN Transceiver

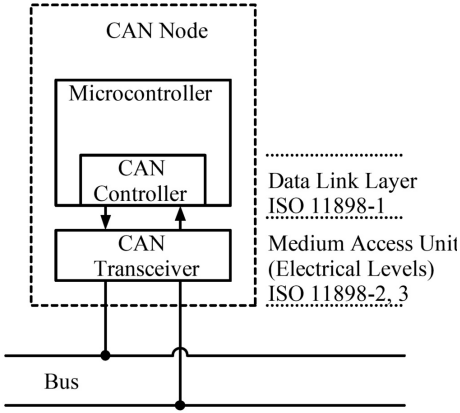
Working alongside the CAN Controller, the CAN Transceiver converts digital data into signals for transmission over the network and vice versa. This is essential for the communication across the network's nodes.

4. Bus Lines

CAN_H and CAN_L Wires: The physical wires of the network, CAN_H and CAN_L, are the backbone of the CAN Bus system. They transmit signals between nodes, even in environments with significant electrical noise.

Together, these components ensure the efficient and reliable operation of the CAN Bus system in various settings, from automotive to industrial applications.

CAN layers

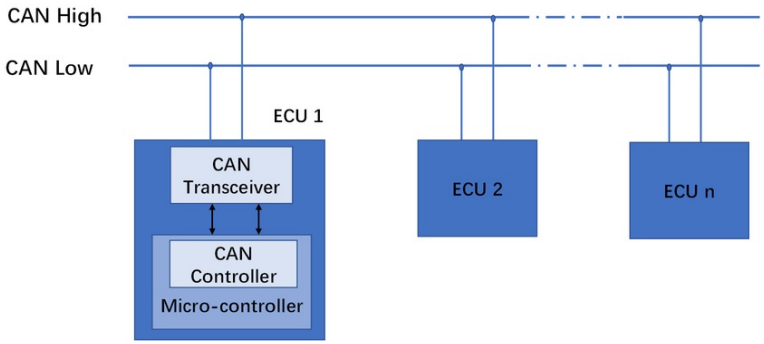

CAN Node
Microcontroller
CAN Controller
CAN Transceiver
Bus
Data Link Layer
ISO 11898-1
Medium Access Unit
(Electrical Levels)
ISO 11898-2, 3

CAN: Physical layer and Arbitration

- From the physical point of view, CAN uses a twisted-pair bus terminated, on both sides, by 120 Ω resistors
- Devices are attached to the bus by means of two signals called CANH (CAN high) and CANL (CAN low)

CAN: Physical layer and Arbitration

- All devices are “peers” and roles (e.g. master or slaves) do not exist
- Any device may decide to start to transmit in any moment, so an arbitration policy must be employed
- Data is transmitted serially, one bit at time
- The maximum speed is defined by the standard as 1 Mbit/s (Classical CAN) or 5 Mbit/s (CAN FD)



CAN Data layer

What is a CAN frame?

- Communication over the CAN bus is done via CAN frames.
- A standard CAN frame with 11 bits identifier (CAN 2.0A), which is the type used in most cars.
- The extended 29-bit identifier frame (CAN 2.0B) is identical except the longer ID.

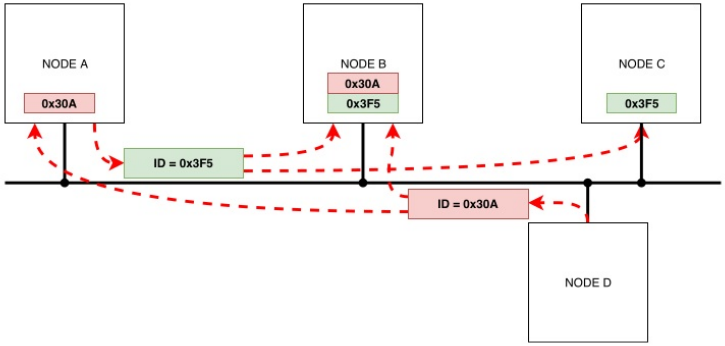
Data Frame

All data frames CAN Frame include three main parts:

- CAN-ID, 11 bits (29 bits for extended), used to identify the data and its priority
- Control bits, various bits for signaling
- Payload, 8 bytes (64 bits), application-dependent

The CAN-ID

- It is a tag that identifies the data that is being transmitted
- Addressing and transmission exploits the CAN-ID according to a publisher-subscriber paradigm
  - A node (publisher) sends its data using a certain CAN-ID
  - Each node interested in that CAN-ID (subscriber) “captures” the data packet and forwards it to the upper layers

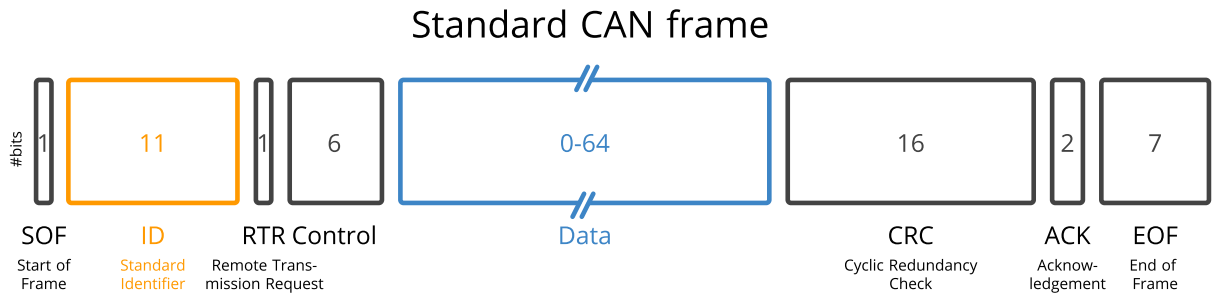


Frame types

- Standard The standard data CAN frame uses a 11-bit identifier. Is the kind predominantly found in vehicles.
- Extended

The extended CAN frame uses a 29-bit identifier with a couple of additional bits. The extended 29-bit identifier (CAN 2.0B) is identical, but has a longer ID and is usually used in the j1939 protocol - heavy-duty vehicles.

CAN Standard Frame (detail)

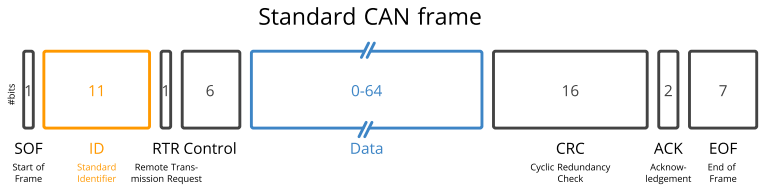


- SOF (Start of Frame):

```
> Length: 1 bit
```

Indicates the beginning of a CAN frame. It is always dominant bit (0 in CAN).

CAN Standard Frame (detail)

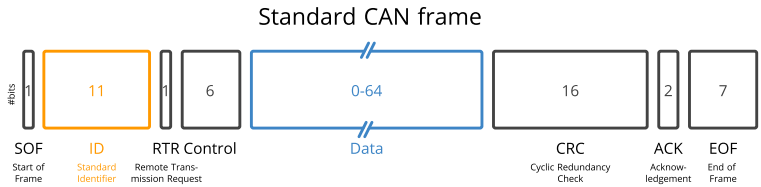


- ID (Identifier):

  > Length: 11 bits

Represents the priority and the address of the transmitting node. In CAN, the lower the identifier value, the higher the priority.

CAN Standard Frame (detail)

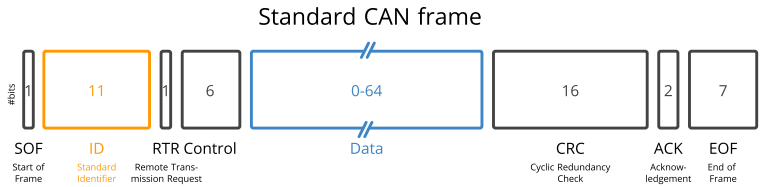


- RTR (Remote Transmission Request):

  `> Length: 1 bit`

Used to differentiate a data frame from a remote request frame (RTR = 0 for data frames and RTR = 1 for remote request frames).

CAN Standard Frame (detail)

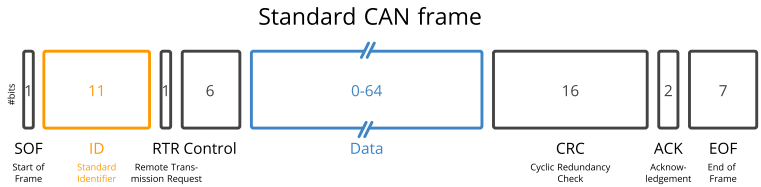


- Control

  > Length: 6 bits

Contains control information like the data length code (DLC) which indicates the number of bytes in the data field.

CAN Standard Frame (detail)

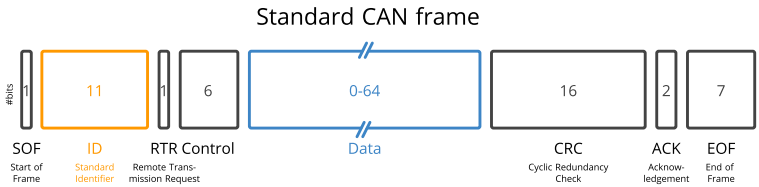


- Data:

```
>Length: 0 to 64 bits (0 to 8 bytes)
```

Contains the actual data being transmitted. Its length is determined by the DLC in the control field.

CAN Standard Frame (detail)

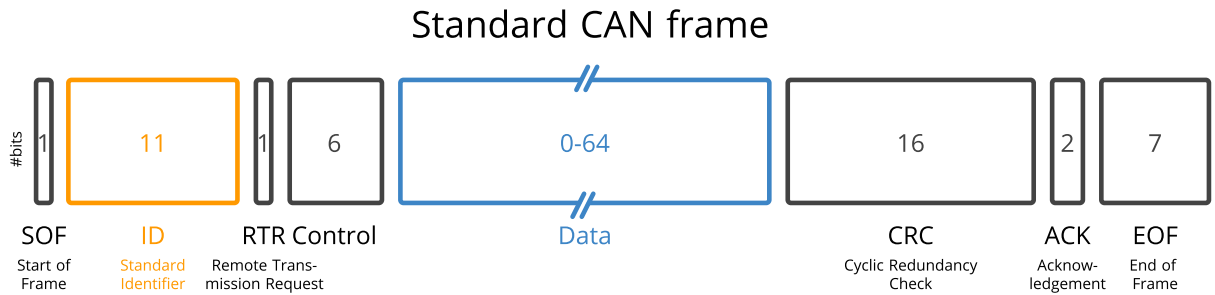


- CRC (Cyclic Redundancy Check):

  > Length: 16 bits

A polynomial code used to detect errors during data transmission. The transmitting node computers a CRC value based on the frame content and sends it along with the frame. The receiving node then calculates its own CRC from the received frame and compares it to the received CRC. If they match, it's assumed that the frame was received correctly.

CAN Standard Frame (detail)

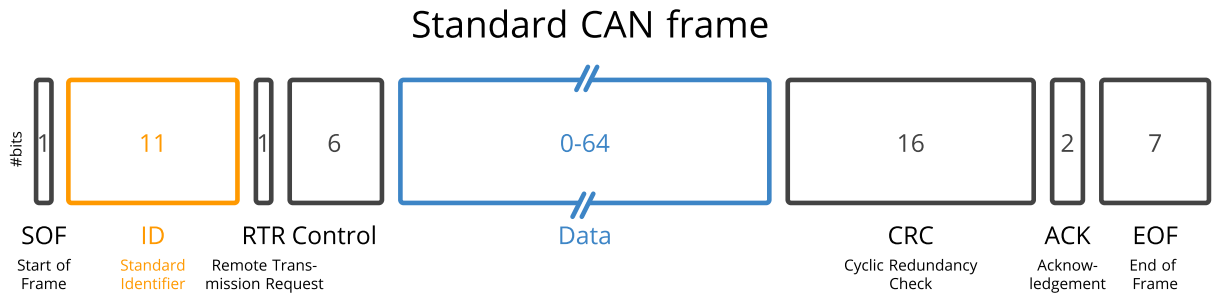


- ACK (Acknowledgement):

```
Length: 2 bits (one for the slot and one for the delimiter)
```

The ACK slot is overwritten with a dominant bit by nodes that correctly receive the frame. If the transmitting node sees a dominant bit in the ACK slot, it knows that at least one other node on the network received its frame correctly. Following, there is an ACK delimiter bit, which is always recessive (1 in CAN).

CAN Standard Frame (detail)



- EOF (End of Frame):

  `Length: 7 bits`

Purpose: Marks the end of a CAN frame. It consist of 7 consecutive recessive bits, ensuring that there's enough separation between consecutive frames.

In addition, CAN communication also involves some error handling mechanisms:

- Error Frames: If a node detects an error in a frame, it will transmit an error frame to notify other nodes of the error.

CAN on Infineon AURIX

> *https://github.com/Infineon/AURIX_code_examples - MCMCAN_1_KIT_TC375_LK*


bluewind

Thanks for listening!

Anna Bonaldo

anna.bonaldo@bluewind.it

https://www.bluewind.it