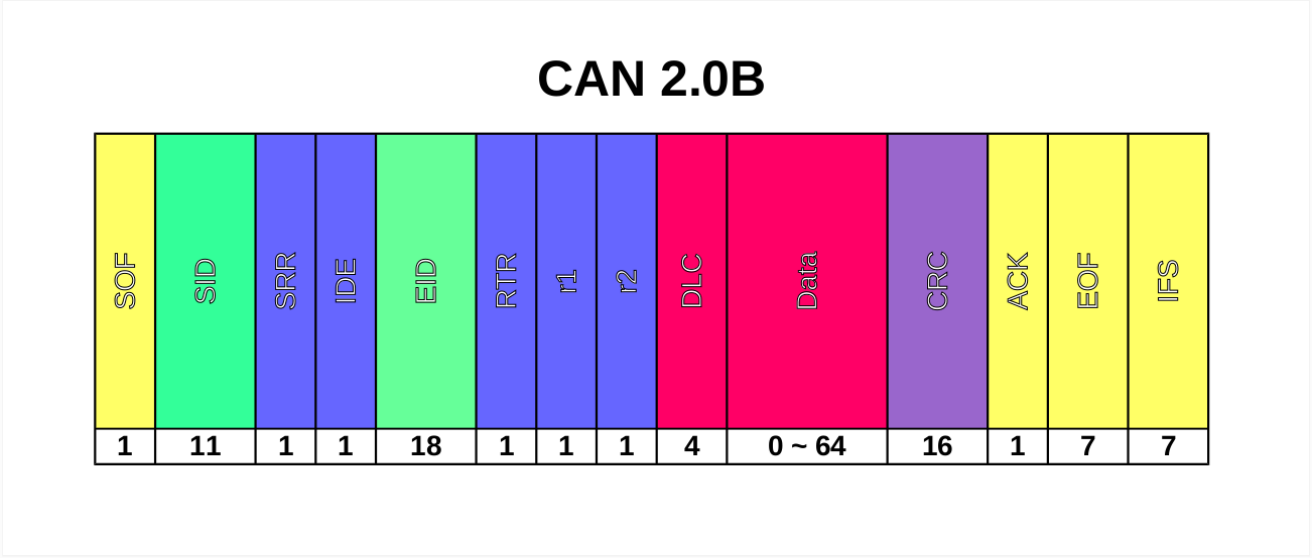
CAN Bus

Published: 22/06/2016 | Post categories: Communication Interfaces, Learn | Views: 8056



Why doesn't my car have a USB port? What is this big connector that looks like something connected to a 90's Television? Does that jack represent something better than most supported and favored USB or at least something simpler like UART? If you don't know answers to this questions - this post if for you to learn what is the protocol that stands behind the jack in you car. If you know that **CAN** bus stands behind it - this might be the place to learn more about it.

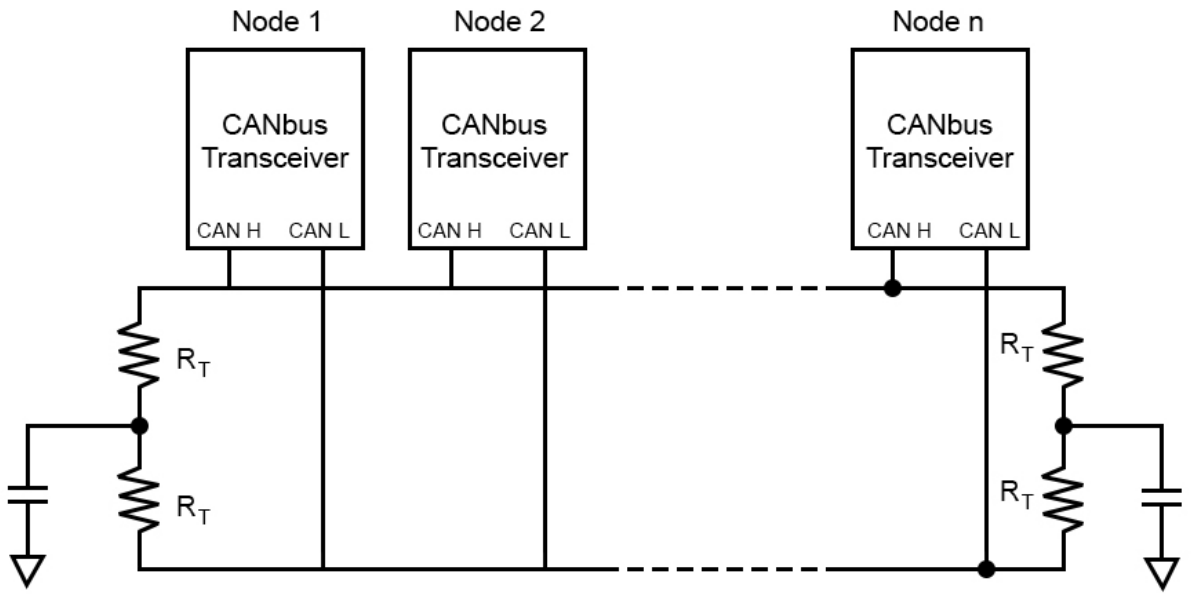
Well, not always and not only CAN is used for communication between the micro-controllers built-in your car but it is one of the most common. Behind that **OBD2** jack can stand one of the five standardized protocols but the CAN is recognized as the most effective. All of the USA cars produced after the 2008 have it and most of the European cars produced after the 2003.

Imagine communication in your car which approximately have 100 MCUs, some newer cars might have more than 1000, so the conclusion can be that CAN must be some very complex type of communication and protocol. That is the wrong conclusion - CAN is simple.

Only two wires are needed. This means no complex wiring system. Everything operates on those two wires and there is no master device that controls the communication between devices on the bus. It is all in arbitration between those devices attached.

Introducing the CAN

Controlled Area Network is developed by BOSCH in the 1980's for multiplexing electrical wiring within automobiles. It is also used in many other contexts such as industrial solutions. Communication is serial based and goes through the two lines (*CAN **H*** / *CAN **L***) and allows devices inside the network to communicate without a master, which means that a network is a multi master based system. All devices (nodes) on the bus can read the message sent by any device on the network.



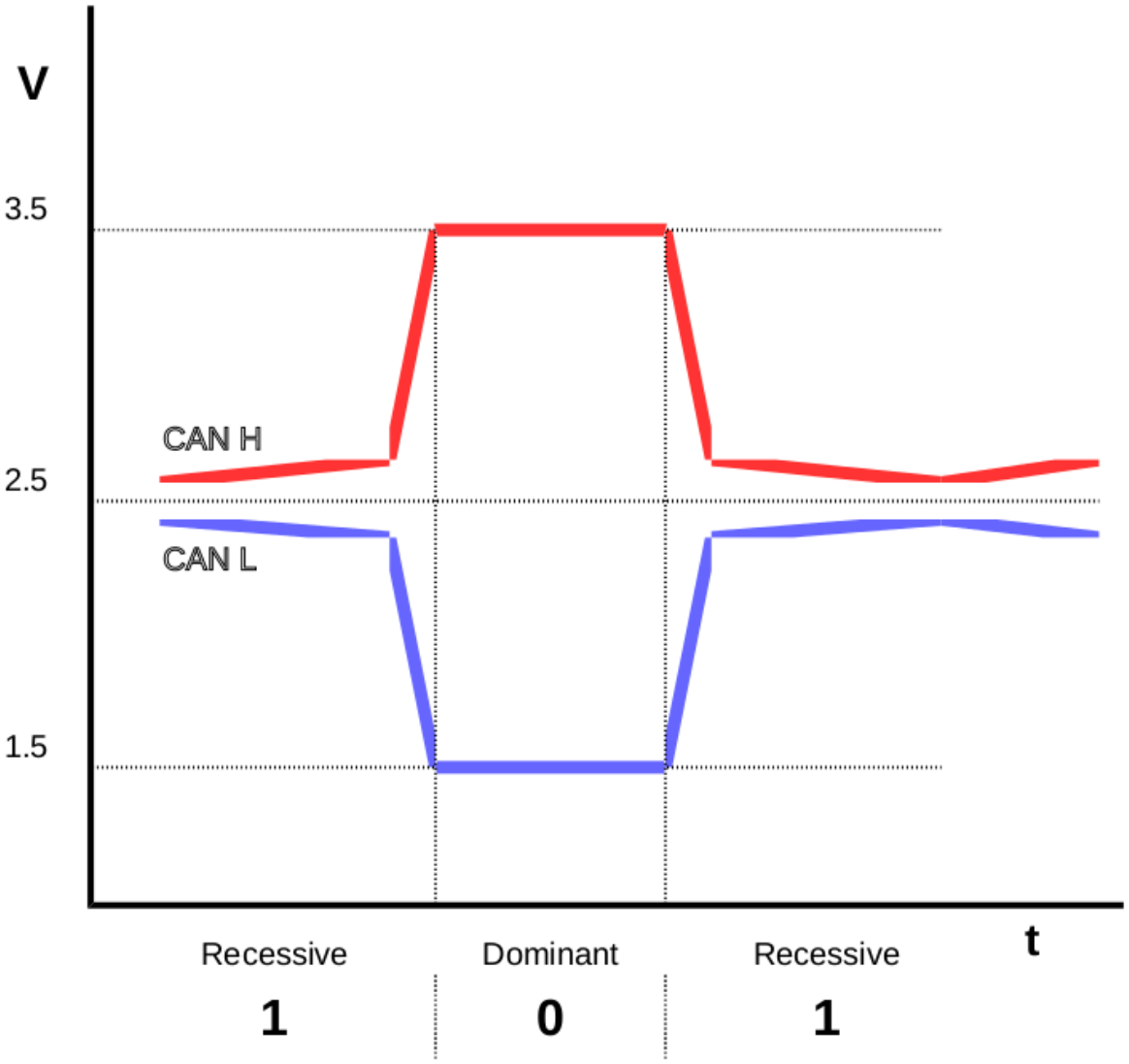
CAN communication is very effective in situations where there is no common ground. Lack of a common ground introduces a high noise level into the system. CAN handles this type of problem easily. CAN, however, is not power efficient if the distance is long. You can not expect a power effective device as a transmitter when looking at large distances through copper cabling.

Properties of CAN like speed of up to 1,000,000 bits per second, more than 1000 meters in distance between nodes, 130 bits of frame length, and internal fault mechanism gives us an answer why this protocol is found outside of its primary aim. Today the bus is used for the medical tools, industrial machines, building automation etc.

CAN Communication

One of the main characteristics is an opposite logic state between the bus sender input and receiver output. Transceivers have the driver input and receiver output pins passively pulled high internally, so that in the absence of any input the device automatically defaults to a recessive bus state.

Dominant bit always overwrites a recessive bit on a bus. That allows the transmitter to recognize that dominant bit on the bus. The receiver also can send the dominant bit on the bus while in receive state.



This technique is used for arbitration and the acknowledge bit by the receiver . Arbitration is the process of the negotiation between the devices and decision which of them can takeover the bus for itself. This process will be explained with more details later.

CAN Protocol

CAN protocol defines short messages which can be sent by any node inside the network and received by any node on the network. The protocol has its own rules and most important of them are :

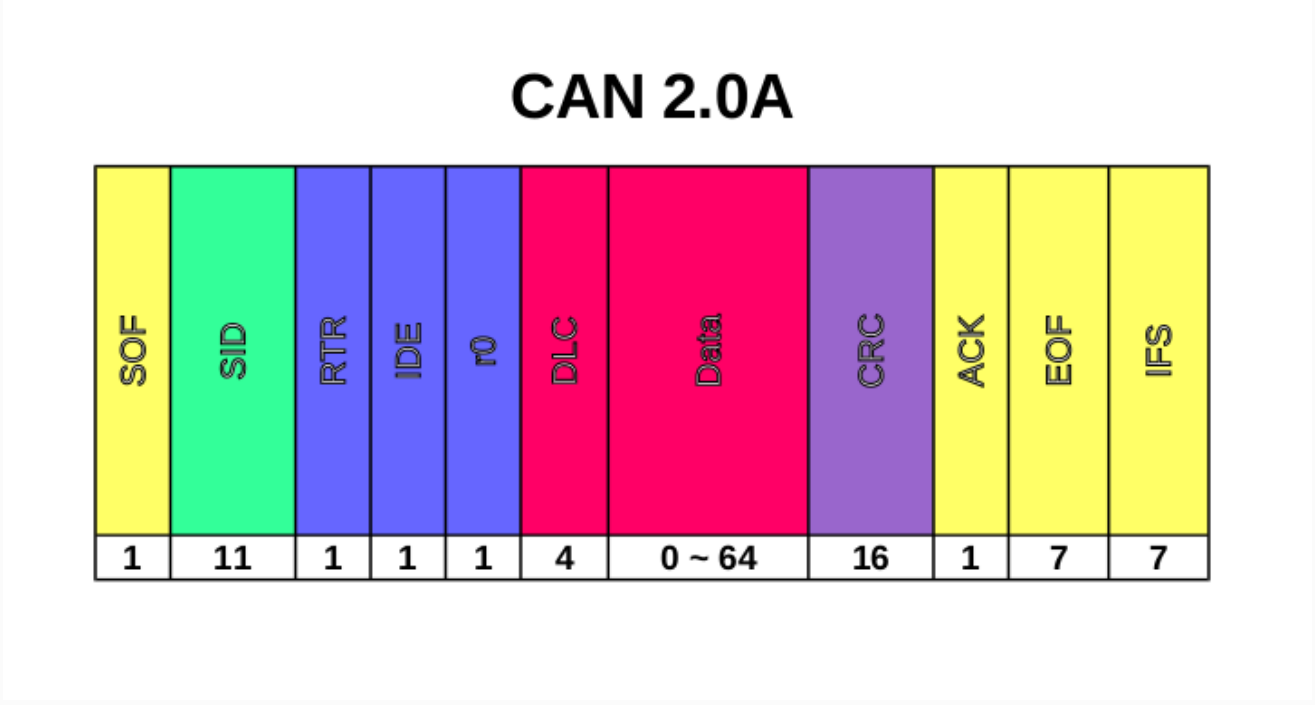
- Each node must wait for a prescribed period of inactivity before attempting to send a message.
- Collision between the nodes are resolved through the bit-wise arbitration based on pre-programmed priority of each message in the identifier field of the message.
- The device that won the arbitration continues sending the frame.
- The higher priority identifier always wins the arbitration.

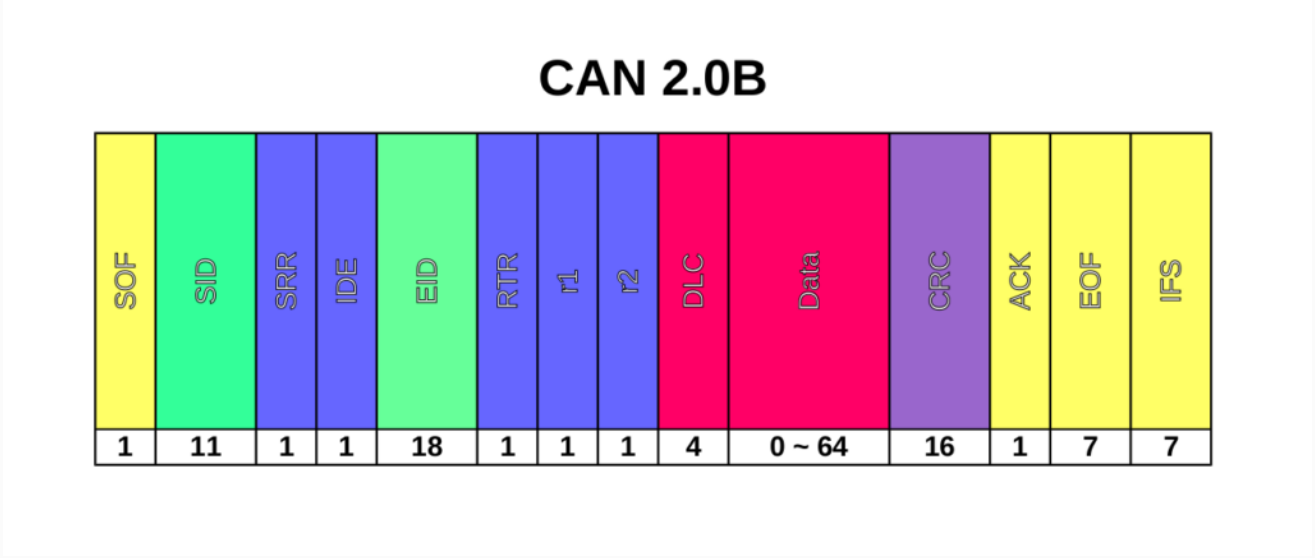
FRAME

Bosch published several versions of the CAN specification and the latest is the 2.0 version published in 1991. This specification has two parts :

- part A is for the standard format with an **11-bit** identifier
- part B is for the extended format with a **29-bit** identifier

A device that uses 11-bit identifiers is commonly called CAN **2.0A** and a device that uses 29-bit identifiers is commonly called CAN **2.0B**. The difference between the types is frame length.



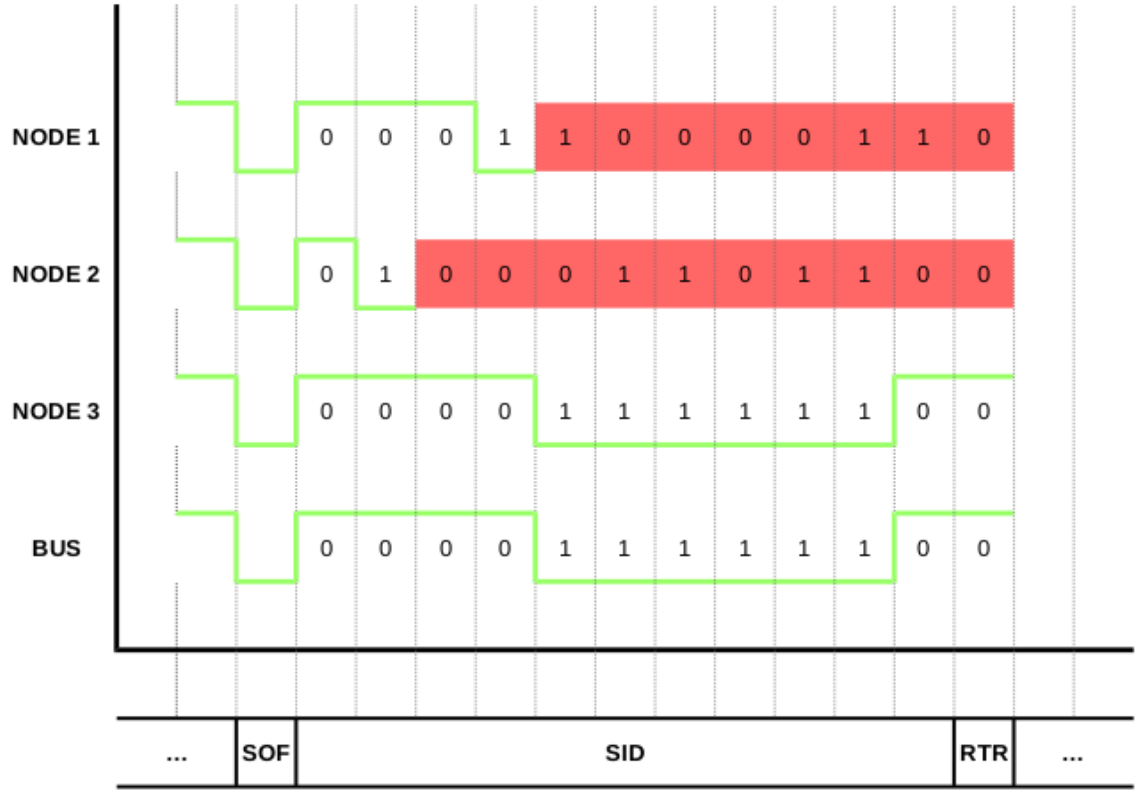


- **SOF** - *Start Of Frame* - single bit that marks the frame start used also for syncronization between nodes
- **SID** - *Standard Identifier* - 11 bits that represents ID and the priority of the message
- **SRR** - *Substitute Remote Request* - placeholder of the RTR in case of extended identifier
- **RTR** - *Remote Transfer Request* - bit is dominant when information is required from another node.
- **IDE** - *Identifier Enabled* - bit meaning is does extended identifier will be send after it
- **EID** - *Extended Identifier* - 18 bits that represents extended ID of the message
- **r0**, **r1**, **r2** - *Reserved bits* - bits reserved for future use
- **DLC** - *Data length* - 4 bits that represents the data length in bytes
- **CRC** - *Cyclic redundancy check*

- **ACK** - *Acknowledge bit* - Device that receiving the message overwrites this bit to indicate that error-free message is received - if no device overwrite the bit te sender will repeat the message after the rearbitration
- **EOF** - *End of frame* - 7 bits that marks the end of the message
- **IFS** - *Interframe space* - 7 bits that represents the space in time enough for the receiver to correctly handle the received frame and move it to the proper buffer area

ARBITRATION

The devices that transmitting are constantly comparing its output to the state of the bus. If the difference between the states exists, this means that another device is sending the data. That can be recognized only by device that is transmitting a recessive bit. In this case if the device recognizes a dominant bit, it stops transmission. This is the reason why the device that transmits lower SID wins the arbitration.



On the diagram you can see exactly what is happening when multiple nodes try to transfer data through the bus. The dominant bit on the bus always overrides the recessive one. Nodes that lose the arbitration stops transmission, which means that after the arbitration process only one device with the lowest SID will continue transmission (Node 3).

Summary

In conclusion this is an excellent case for small and / or closed networks. An example would be in the case of a network of sensors and micro-controllers inside the car. For complex machines that have also contains multiple sensors and even more MCUs than your car this also could be a good solution.

Devices like ours CAN SPI click or Microchips MCP25625 allows you to develop your own application easy by controlling the transceivers through the SPI and make this protocol even more understandable to you. Also our

development boards have ports and you can use it with micro-controllers that have built-in module for CAN communication.