CAN Bus

for students

Funded by the
Erasmus+ Programme
of the European Union

STEP AHEAD II

**The support of Professional development of VET teachers and trainers in following of New trends in Automotive Industry
Automotive Innovation & Teacher training Academy
2018-1-SK01-KA202-046334**

CAN Bus

The aim of the lesson:
Understanding CAN bus working principles in automotive industry and learn basic diagnostic of CAN.

ANNEX 1

CAN NETWORK

What is it
A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles to save on copper, but is also used in many other contexts.

Applications

- Passenger vehicles, trucks, buses (gasoline vehicles and electric vehicles)
- Electronic equipment for aviation and navigation
- Industrial automation and mechanical control
- Elevators, escalators
- Building automation

- Medical instruments and equipment

The modern automobile may have as many as 70 electronic control units (ECU) for various subsystems.[7] Typically the biggest processor is the engine control unit. Others are used for transmission, airbags, antilock braking/ABS, cruise control, electric power steering, audio systems, power windows, doors, mirror adjustment, battery and recharging systems for hybrid/electric cars, etc. Some of these form independent subsystems, but communications among others are essential. A subsystem may need to control actuators or receive feedback from sensors. The CAN standard was devised to fill this need. One key advantage is that interconnection between different vehicle systems can allow a wide range of safety, economy and convenience features to be implemented using software alone - functionality which would add cost and complexity if such features were "hard wired" using traditional automotive electrics.

Examples include:

- **Auto start/stop:** Various sensor inputs from around the vehicle (speed sensors, steering angle, air conditioning on/off, engine temperature) are collated via the CAN bus to determine whether the engine can be shut down when stationary for improved fuel economy and emissions.
- **Electric park brakes**: The "hill hold" functionality takes input from the vehicle's tilt sensor (also used by the burglar alarm) and the road speed sensors (also used by the ABS, engine control and traction control) via the CAN bus to determine if the vehicle is stopped on an incline. Similarly, inputs from seat belt sensors (part of the airbag controls) are fed from the CAN bus to determine if the seat belts are fastened, so that the parking brake will automatically release upon moving off.
- **Parking assist** systems: when the driver engages reverse gear, the transmission control unit can send a signal via the CAN bus to activate both the parking sensor system and the door control module for the passenger side door mirror to tilt downward to show the position of the curb. The CAN bus also takes inputs from the rain sensor to trigger the rear windscreen wiper when reversing.
- Auto **lane assist**/**collision avoidance** systems: The inputs from the parking sensors are also used by the CAN bus to feed outside proximity data to driver assist systems such as Lane Departure warning, and more recently, these signals travel through the CAN bus to actuate brake by wire in active collision avoidance systems.
- **Auto brake wiping**: Input is taken from the rain sensor (used primarily for the automatic windscreen wipers) via the CAN bus to the ABS module to initiate an imperceptible application of the brakes whilst driving to clear moisture from the brake rotors. Some high performance Audi and BMW models incorporate this feature.
- **Sensors** can be placed at the most suitable place, and its data used by several ECU. For example, outdoor temperature sensors (traditionally placed in the front) can be placed in the outside mirrors, avoiding heating by the engine, and data used by both the engine, the climate control and the driver display.

CAN is a multi-master serial bus standard for connecting **Electronic Control Units [ECUs]** also known as **nodes**. Two or more nodes are required on the CAN network to communicate. The complexity of the node can range from a simple I/O device up to an embedded computer with a CAN interface and sophisticated software. The node may also be a gateway allowing a general purpose computer (such as a laptop) to communicate over a USB or Ethernet port to the devices on a CAN network.

All nodes are connected to each other through a two wire bus. The wires are a twisted pair with a 120 Ω (nominal) characteristic impedance.

ISO 11898-2, also called high speed CAN (512 Kbps), uses a linear bus terminated at each end with 120 Ω resistors. High speed CAN signaling drives the CAN high wire towards 5 V and the CAN low wire towards 0 V when transmitting a dominant (0), and does not drive either wire when transmitting a recessive (1). Designating "0" as dominant gives the nodes with the lower ID numbers priority on the bus. The dominant differential voltage is a nominal 2 V. The termination resistor passively returns the two wires to a nominal differential voltage of 0 V. The dominant common mode voltage must be within 1.5 to 3.5 V of common and the recessive common mode voltage must be within +/-12 of common.

ISO 11898-3, also called low speed or fault tolerant CAN (128 Kbps), uses a linear bus, star bus or multiple star buses connected by a linear bus and is terminated at each node by a fraction of the overall termination resistance. The overall termination resistance should be about 100 Ω, but not less than 100 Ω. Low speed/Fault tolerant CAN signaling drives the CAN high wire towards 5 V and the CAN low wire towards 0 V when transmitting a dominant (0), and does not drive either wire when transmitting a recessive (1). The dominant differential voltage must be greater than 2.3 V (with a 5 V Vcc) and the recessive differential voltage must be less than 0.6 V The termination resistors passively return the CAN low wire to RTH where RTH is a minimum of 4.7 V (Vcc - 0.3 V where Vcc is 5 V nominal) and the CAN high wire to RTL where RTL is a maximum of 0.3 V. Both wires must be able to handle -27 to 40 V without damage.

With both high speed and low speed CAN, the speed of the transition is faster when a recessive to dominant transition occurs since the CAN wires are being actively driven. The speed of the dominant to recessive transition depends primarily on the length of the CAN network and the capacitance of the wire used.

High speed CAN is usually used in automotive and industrial applications where the bus runs from one end of the environment to the other. Fault tolerant CAN is often used where groups of nodes need to be connected together.

The specifications require the bus be kept within a minimum and maximum common mode bus voltage, but do not define how to keep the bus within this range.

The CAN bus must be terminated. The termination resistors are needed to suppress reflections as well as return the bus to its recessive or idle state.

High speed CAN uses a 120 Ω resistor at each end of a linear bus. Low speed CAN uses resistors at each node. Other types of terminations may be used such as the Terminating Bias Circuit defined in ISO11783 [9]

A terminating bias circuit provides power and ground in addition to the CAN signaling on a four-wire cable. This provides automatic electrical bias and termination at each end of each bus segment. An ISO11783 network is designed for hot plug-in and removal of bus segments and ECUs.

CAN data transmission uses a lossless bitwise arbitration method of contention resolution. This arbitration method requires all nodes on the CAN network to be synchronized to sample every bit on the CAN network at the same time. This is why some call CAN synchronous. Unfortunately the term synchronous is imprecise since the data is transmitted without a clock signal in an asynchronous format.

The CAN specifications use the terms **"dominant"** bits and **"recessive"** bits where dominant is a logical 0 (actively driven to a voltage by the transmitter) and recessive is a logical 1 (passively returned to a voltage by a resistor). The idle state is represented by the recessive level (Logical 1). If one node transmits a dominant bit and another node transmits a recessive bit then there is a collision and the dominant bit "wins". This means there is no delay to the higher-priority message, and the node transmitting the lower priority message automatically attempts to re-transmit six bit clocks after the end of the dominant message. This makes CAN very suitable as a real time prioritized communications system.

The exact voltages for a logical 0 or 1 depend on the physical layer used, but the basic principle of CAN requires that each node listens to the data on the CAN network including the transmitting node(s) itself (themselves). If a logical 1 is transmitted by all transmitting nodes at the same time, then a logical 1 is seen by all of the nodes, including both the transmitting node(s) and receiving node(s). If a logical 0 is transmitted by all transmitting node(s) at the same time, then a logical 0 is seen by all nodes. If a logical 0 is being transmitted by one or more nodes, and a logical 1 is being transmitted by one or more nodes, then a logical 0 is seen by all nodes including the node(s) transmitting the logical 1. When a node transmits a logical 1 but sees a logical 0, it realizes that there is a contention and it quits transmitting. By using this process, any node that transmits a logical 1 when another node transmits a logical 0 "drops out" or loses the arbitration. A node that loses arbitration re-queues its message for later transmission and the CAN frame bit-stream continues without error until only one node is left transmitting. This means that the node that transmits the first 1 loses arbitration. Since the 11 (or 29 for CAN 2.0B) bit identifier is transmitted by all nodes at the start of the CAN frame, the node with the lowest identifier transmits more zeros at the start of the frame, and that is the node that wins the arbitration or has the highest priority.

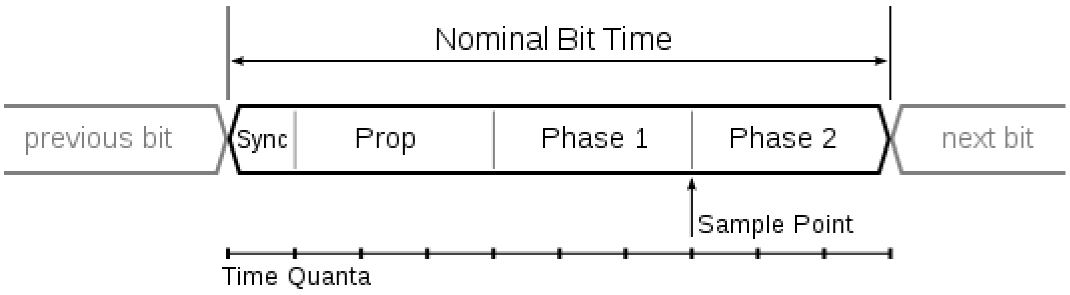
For example, consider an 11-bit ID CAN network, with two nodes with IDs of 15 (binary representation, 00000001111) and 16 (binary representation, 00000010000). If these two nodes transmit at the same time, each will first transmit the start bit then transmit the first six zeros of their ID with no arbitration decision being made.

All nodes on the CAN network must operate at the same nominal bit rate, but noise, phase shifts, oscillator tolerance and oscillator drift mean that the actual bit rate may not be the same as the nominal bit rate. Since a separate clock signal is not used, a means of synchronizing the nodes is necessary. Synchronization is important during arbitration since the nodes in arbitration must be able to see both their transmitted data and the other nodes' transmitted data at the same time.

Synchronization is also important to ensure that variations in oscillator timing between nodes do not cause errors.

Synchronization starts with a hard synchronization on the first recessive to dominant transition after a period of bus idle (the start bit). Resynchronization occurs on every recessive to dominant transition during the frame. The CAN controller expects the transition to occur at a multiple of the nominal bit time. If the transition does not occur at the exact time the controller expects it, the controller adjusts the nominal bit time accordingly.

The adjustment is accomplished by dividing each bit into a number of time slices called quanta, and assigning some number of quanta to each of the four segments within the bit: synchronization, propagation, phase segment 1 and phase segment 2.



An example CAN bit timing with 10 time quanta per bit.

The number of quanta the bit is divided into can vary by controller, and the number of quanta assigned to each segment can be varied depending on bit rate and network conditions.

A transition that occurs before or after it is expected causes the controller to calculate the time difference and lengthen phase segment 1 or shorten phase segment 2 by this time. This effectively adjusts the timing of the receiver to the transmitter to synchronize them. This resynchronization process is done continuously at every recessive to dominant transition to ensure the transmitter and receiver stay in sync. Continuously resynchronizing reduces errors induced by noise, and allows a receiving node that was synchronized to a node which lost arbitration to resynchronize to the node which won arbitration.

A CAN network can be configured to work with two different message (or "frame") formats: the standard or base frame format (described in CAN 2.0 A and CAN 2.0 B), and the extended frame format (only described by CAN 2.0 B). The only difference between the two formats is that the "CAN base frame" supports a length of 11 bits for the identifier, and the "CAN extended frame" supports a length of 29 bits for the identifier, made up of the 11-bit identifier ("base identifier") and an 18-bit extension ("identifier extension"). The distinction between CAN base frame format and CAN extended frame format is made by using the IDE bit, which is transmitted as dominant in case of an 11-bit frame, and transmitted as recessive in case of a 29-bit frame. CAN controllers that support extended frame format messages are also able to send and receive messages in CAN base frame format. All frames begin with a start-of-frame (SOF) bit that denotes the start of the frame transmission.

CAN has four frame types:

- **Data frame:** a frame containing node data for transmission

- **Remote frame:** a frame requesting the transmission of a specific identifier
- **Error frame:** a frame transmitted by any node detecting an error
- **Overload frame:** a frame to inject a delay between data or remote frame

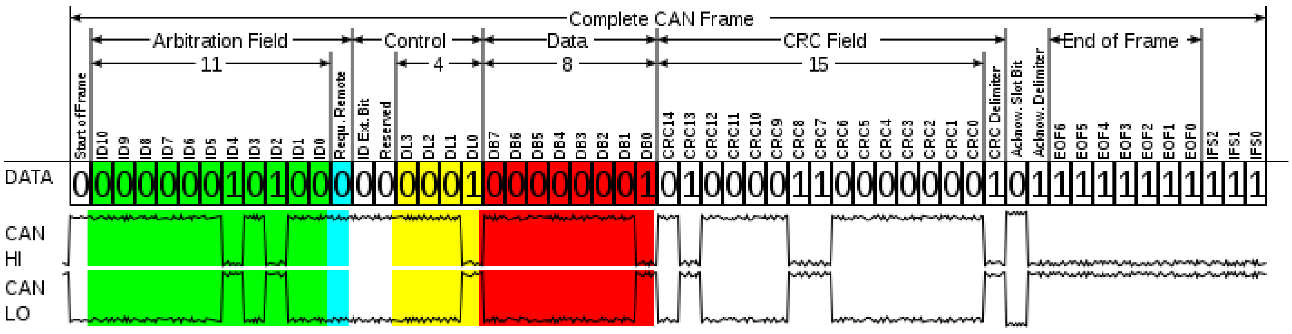
Data frame

The data frame is the only frame for actual data transmission. There are two message formats:

- Base frame format: with 11 identifier bits
- Extended frame format: with 29 identifier bits

The CAN standard requires the implementation must accept the base frame format and may accept the extended frame format, but must tolerate the extended frame format.

Base frame format



CAN-Frame in base format with electrical levels without stuffbits

The frame format is as follows: The bit values are described for CAN-LO signal.

Field name	**Length (bits)**	**Purpose**
Start-of-frame	1	Denotes the start of frame transmission
Denotes the start of frame transmission	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check

CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

1. It is physically possible for a value between 9–15 to be transmitted in the 4-bit DLC, although the data is still limited to eight bytes. Certain controllers allow the transmission or reception of a DLC greater than eight, but the actual data length is always limited to eight bytes.

Extended frame format

The frame format is as follows:

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier A (green)	11	First part of the (unique) identifier which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1)
Identifier extension bit (IDE)	1	Must be recessive (1) for extended frame format with 29-bit identifiers
Identifier B (green)	18	Second part of the (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Reserved bits (r1, r0)	2	Reserved bits which must be set dominant (0), but accepted as either dominant or recessive
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

1. It is physically possible for a value between 9–15 to be transmitted in the 4-bit DLC, although the data is still limited to eight bytes. Certain controllers allow the transmission or reception of a DLC greater than eight, but the actual data length is always limited to eight bytes.

The two identifier fields (A & B) combine to form a 29-bit identifier.

Remote frame

- Generally data transmission is performed on an autonomous basis with the data source node (e.g., a sensor) sending out a Data Frame. It is also possible, however, for a destination node to request the data from the source by sending a Remote Frame.
- There are two differences between a Data Frame and a Remote Frame. Firstly the RTR-bit is transmitted as a dominant bit in the Data Frame and secondly in the Remote Frame there is no Data Field. The DLC field indicates the data length of the requested message (not the transmitted one)

i.e.,

RTR = 0 ; **DOMINANT** in data frame
RTR = 1 ; **RECESSIVE** in remote frame

In the event of a Data Frame and a Remote Frame with the same identifier being transmitted at the same time, the Data Frame wins arbitration due to the dominant RTR bit following the identifier.

Error frame

The error frame consists of two different fields:

- The first field is given by the superposition of ERROR FLAGS (6–12 dominant/recessive bits) contributed from different stations.
- The following second field is the ERROR DELIMITER (8 recessive bits).

There are two types of error flags:

Active Error Flag
six dominant bits – Transmitted by a node detecting an error on the network that is in error state "error active".
Passive Error Flag
six recessive bits – Transmitted by a node detecting an active error frame on the network that is in error state "error passive".

There are two error counters in CAN:

1. **Transmit error counter** (TEC)
2. **Receive error counter (**REC)

- When TEC or REC is greater than 127 and lesser than 255, a Passive Error frame will be transmitted on the bus.
- When TEC and REC is lesser than 128, an Active Error frame will be transmitted on the bus.
- When TEC is greater than 255, then the node enters into Bus Off state, where no frames will be transmitted.

Overload frame

The overload frame contains the two - bit fields Overload Flag and Overload Delimiter. There are two kinds of overload conditions that can lead to the transmission of an overload flag:

1. The internal conditions of a receiver, which requires a delay of the next data frame or remote frame.
2. Detection of a dominant bit during intermission.

The start of an overload frame due to case 1 is only allowed to be started at the first bit time of an expected intermission, whereas overload frames due to case 2 start one bit after detecting the dominant bit. Overload Flag consists of six dominant bits. The overall form corresponds to that of the active error flag. The overload flag's form destroys the fixed form of the intermission field. As a consequence, all other stations also detect an overload condition and on their part start transmission of an overload flag. Overload Delimiter consists of eight recessive bits. The overload delimiter is of the same form as the error delimiter.

Links to videos:

- https://www.youtube.com/watch?v=FqLDpHsxvf8
- https://www.youtube.com/watch?v=Gi7mxVmzLkM
- https://www.youtube.com/watch?v=YrJn2AyWVBc
- https://www.youtube.com/watch?v=dwU5aEbsgLM
- https://www.snapon.com/Diagnostics/US/KB/CAN-Bus-Diagnostics.htm

Links to material:

- http://download.ni.com/pub/devzone/tut/can_tutorial.pdf
- http://www.ni.com/en-us/innovations/white-papers/06/controller-area-network--can--overview.html
- https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en
- https://www.aa1car.com/library/can_systems.htm

- http://www.esd-electronics-usa.com/CAN-Bus-Troubleshooting-Guide.html
- https://pmmonline.co.uk/technical/can-bus-fault-finding-tips-and-hints-part-1/
- http://pmmonline.co.uk/technical/can-bus-fault-finding-tips-and-hints-part-2/
- https://www.consulab.com/files/canBusHandout.pdf

NOTES: