CAN bus

CAN bus (for **controller area network**) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer.

CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, maritime, industrial automation and medical equipment.

Development of Controller Area Network bus started originally in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE) congress in Detroit, Michigan. The first CAN controller chips, produced by Intel and Philips, came on the market in 1987. Bosch published the CAN 2.0 specification in 1991. In 2012 Bosch has specified the improved CAN data link layer protocol, called CAN FD, which will extend the ISO 11898-1.

CAN bus is one of five protocols used in the OBD-II vehicle diagnostics standard. The OBD-II standard has been mandatory for all cars and light trucks sold in the United States since 1996, and the EOBD standard has been mandatory for all petrol vehicles sold in the European Union since 2001 and all diesel vehicles since 2004.[1]

Applications

Automotive

A modern automobile may have as many as 70 electronic control units (ECU) for various subsystems.[2] Typically the biggest processor is the engine control unit (also engine control module/ECM or Powertrain Control Module/PCM in automobiles); others are used for transmission, airbags, antilock braking/ABS, cruise control, electric power steering/EPS, audio systems, 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.

The CAN bus may be used in vehicles to connect the engine control unit and transmission, or (on a different bus) to connect the door locks, climate control, seat control, etc. Today the CAN bus is also used as a fieldbus in general automation environments, primarily due to the low cost of some CAN controllers and processors.

Bosch holds patents on the technology, and manufacturers of CAN-compatible microprocessors pay license fees to Bosch, which are normally passed on to the customer in the price of the chip. Manufacturers of products with custom ASICs or FPGAs containing CAN-compatible modules may need to pay a fee for the CAN Protocol License.[*citation needed*]

Technology

CAN is a multi-master broadcast serial bus standard for connecting electronic control units (ECUs).

Each node is able to send and receive messages, but not simultaneously. A message consists primarily of an ID (identifier), which represents the priority of the message, and up to eight data bytes. The improved CAN (CAN FD) extends the length of the data section to up to 64 bytes per frame. It is transmitted serially onto the bus. This signal pattern is encoded in non-return-to-zero (NRZ) and is sensed by all nodes.

The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are not connected directly to the bus, but through a host processor and a CAN controller.

If the bus is idle which is represented by recessive level (TTL=5V), any node may begin to transmit. If two or more nodes begin sending messages at the same time, the message with the more dominant ID (which has more dominant bits, i.e., zeroes) will overwrite other nodes' less dominant IDs, so that eventually (after this arbitration on the ID.)

only the dominant message remains and is received by all nodes. This mechanism is referred to as priority based bus arbitration. Messages with numerically smaller values of IDs have higher priority and are transmitted first.

Each node requires a

- **Host processor**
  - The host processor decides what received messages mean and which messages it wants to transmit itself.
  - Sensors, actuators and control devices can be connected to the host processor.
- **CAN controller** (hardware with a synchronous clock)..
  - *Receiving*: the CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor (usually after the CAN controller has triggered an interrupt).
  - *Sending*: the host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.
- **Transceiver**
  - *Receiving*: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
  - *Transmitting*: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances (e.g., 500 m at 125 kbit/s). The improved CAN (CAN FD) extends the speed of the data section by a factor of up to 8 of the arbitration bit rate.

The CAN data link layer protocol is standardized in ISO 11898-1 (2003).[3] This standard describes mainly the data link layer (composed of the logical link control (LLC) sublayer and the media access control (MAC) sublayer) and some aspects of the physical layer of the OSI reference model. All the other protocol layers are the network designer's choice.

Data transmission

CAN features an automatic arbitration-free transmission. A CAN message that is transmitted with highest priority will succeed, and the node transmitting the lower priority message will sense this and back off and wait.

This is achieved by CAN transmitting data through a binary model of "dominant" bits and "recessive" bits where dominant is a logical 0 and recessive is a logical 1. This means open collector, or *wired or* physical implementation of the bus (but since dominant is 0, this is sometimes referred to as *wired and*). If one node transmits a dominant bit and another node transmits a recessive bit then the dominant bit "wins" (a logical AND between the two).

Truth tables for dominant/recessive, logical or, and logical and (for comparison)

	Dominant	**Recessive**
Dominant	Dominant	Dominant
Recessive	Dominant	Recessive

	0	**1**
0	0	1
1	1	1

	0	**1**
0	0	0
1	0	1

So, if a recessive bit is being transmitted while a dominant bit is sent, the dominant bit is displayed, evidence of a collision. (All other collisions are invisible.) A dominant bit is asserted by creating a voltage across the wires while a recessive bit is simply not asserted on the bus. If any node sets a voltage difference, all nodes will see it. Thus there is no delay to the higher priority messages, and the node transmitting the lower priority message automatically attempts to re-transmit six bit clocks after the end of the dominant message.

When used with a differential bus, a carrier sense multiple access/bitwise arbitration (CSMA/BA) scheme is often implemented: if two or more devices start transmitting at the same time, there is a priority based arbitration scheme to decide which one will be granted permission to continue transmitting. The CAN solution to this is prioritized arbitration (and for the dominant message delay free), making CAN very suitable for real time prioritised communications systems.

During arbitration, each transmitting node monitors the bus state and compares the received bit with the transmitted bit. If a dominant bit is received when a recessive bit is transmitted then the node stops transmitting (i.e., it lost arbitration). Arbitration is performed during the transmission of the identifier field. Each node starting to transmit at the same time sends an ID with dominant as binary 0, starting from the high bit. As soon as their ID is a larger number (lower priority) they will be sending 1 (recessive) and see 0 (dominant), so they back off. At the end of ID transmission, all nodes but one have backed off, and the highest priority message gets through unimpeded.

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 transmit the first six zeros of their ID with no arbitration decision being made. When the 7th bit is transmitted, the node with the ID of 16 transmits a 1 (recessive) for its ID, and the node with the ID of 15 transmits a 0 (dominant) for its ID. When this happens, the node with the ID of 16 will realize that it lost its arbitration, and allow the node with ID of 15 to continue its transmission. This ensures that the node with the lower bit value will always win the arbitration. The ID with the smaller number will win the right to use.

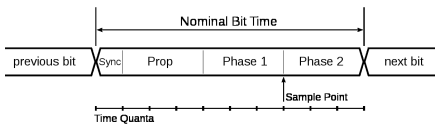
ID allocation

Message IDs must be unique on a single CAN bus, otherwise two nodes would continue transmission beyond the end of the arbitration field (ID) causing an error.

In the early 1990s, the choice of IDs for messages was done simply on the basis of identifying the type of data and the sending node; however, as the ID is also used as the message priority, this led to poor real-time performance. In those scenarios, a low CAN bus utilization of circa 30% was commonly required to ensure that all messages would meet their deadlines. However, if IDs are instead determined based on the deadline of the message, the lower the numerical ID and hence the higher the message priority, then bus utilizations of 70 to 80% can typically be achieved before any message deadlines are missed. It will use the CSMA/CD method in an efficient way.

Bit Timing

Each node in a CAN network has its own clock, and no clock is sent during data transmission. Synchronization is done by dividing each bit of the frame into a number of segments: synchronization, propagation, phase 1 and phase 2. The length of each *phase* segment can be adjusted based on network and node conditions. The sample point falls between phase buffer segment 1 and phase buffer segment 2, which helps facilitate continuous synchronization. Continuous synchronization in turn enables the receiver to be able to properly read the messages.



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

Layers

The CAN protocol, like many networking protocols, can be decomposed into the following abstraction layers:

Application layer

Object layer

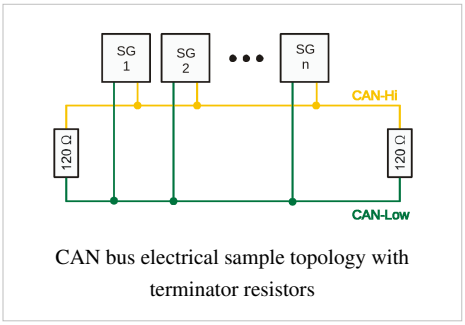
- Message filtering
- Message and status handling

Transfer layer

Most of the CAN standard applies to the transfer layer. The transfer layer receives messages from the physical layer and transmits those messages to the object layer. The transfer layer is responsible for bit timing and synchronization, message framing, arbitration, acknowledgement, error detection and signalling, and fault confinement. It performs:

- Fault Confinement
- Error Detection
- Message Validation
- Acknowledgement
- Arbitration
- Message Framing
- Transfer Rate and Timing
- Information Routing

Physical layer



CAN bus electrical sample topology with terminator resistors

CAN bus (ISO 11898-1:2003) originally specified the link layer protocol with only abstract requirements for the physical layer, e.g., asserting the use of a medium with multiple-access at the bit level through the use of dominant and recessive states. The electrical aspects of the physical layer (voltage, current, number of conductors) were specified in ISO 11898-2:2003, which is now widely accepted. However, the mechanical aspects of the physical layer (connector type and number, colors, labels, pin-outs) have yet to be formally specified. As a result, an automotive electronic control unit (ECU) will typically have a particular—often custom—connector with various sorts of cables, of which two are the CAN bus lines. Nonetheless, several de facto standards for mechanical implementation have emerged, the most common being the 9-pin D-sub type male connector with the following pin-out:

- pin 2: CAN-Low (CAN-)
- pin 3: GND (Ground)
- pin 7: CAN-High (CAN+)
- pin 9: CAN V+ (Power)

This de facto mechanical standard for CAN could be implemented with node having both male and female 9-pin D-sub connectors electrically wired to each other in parallel within the node. Bus power is fed to a node's male connector and the bus draws power from the node's female connector. This follows the electrical engineering convention that power sources are terminated at female connectors. Adoption of this standard avoids the need to fabricate custom splitters to connect two sets of bus wires to a single D connector at each node. Such nonstandard (custom) wire harnesses (splitters) that join conductors outside the node reduce bus reliability, eliminate cable interchangeability, reduce compatibility of wiring harnesses, and increase cost.

The absence of a complete physical layer specification (mechanical in addition to electrical) freed the CAN bus specification from the constraints and complexity of physical implementation. However it left CAN bus

implementations open to inter-interoperability issues due to mechanical incompatibility.

Noise immunity on ISO 11898-2:2003 is achieved by maintaining the differential impedance of the bus at a low level with low-value resistors (120 ohms) at each end of the bus. However, when dormant, a low-impedance bus such as CAN draws more current (and power) than other voltage-based signaling busses. On CAN bus systems, balanced line operation, where current in one signal line is exactly balanced by current in the opposite direction in the other signal provides an independent, stable 0 V reference for the receivers. Best practice determines that CAN bus balanced pair signals be carried in twisted pair wires in a shielded cable to minimize RF emission and reduce interference susceptibility in the already noisy RF environment of an automobile.

ISO 11898-2 provides some immunity to common mode voltage between transmitter and receiver by having a 0 V rail running along the bus to maintain a high degree of voltage association between the nodes. Also, in the de facto mechanical configuration mentioned above, a supply rail is included to distribute power to each of the transceiver nodes. The design provides a common supply for all the transceivers. The actual voltage to be applied by the bus and which nodes apply to it are application-specific and not formally specified. Common practice node design provides each node with transceivers which are optically isolated from their node host and derive a 5 V linearly regulated supply voltage for the transceivers from the universal supply rail provided by the bus. This usually allows operating margin on the supply rail sufficient to allow interoperability across many node types. Typical values of supply voltage on such networks are 7 to 30 V. However, the lack of a formal standard means that system designers are responsible for supply rail compatibility.

ISO 11898-2 describes the electrical implementation formed from a multi-dropped single-ended balanced line configuration with resistor termination at each end of the bus. In this configuration a dominant state is asserted by one or more transmitters switching the CAN- to supply 0 V and (simultaneously) switching CAN+ to the +5 V bus voltage thereby forming a current path through the resistors that terminate the bus. As such the terminating resistors form an essential component of the signalling system and are included not just to limit wave reflection at high frequency. During a recessive state the signal lines and resistor(s) remain in a high impedances state with respect to both rails. Voltages on both CAN+ and CAN- tend (weakly) towards ½ rail voltage. During a dominant state the signal lines and resistor(s) move to a low impedance state with respect to the rails so that current flows through the resistor. CAN+ voltage tends to +5 V and CAN- tends to 0 V. A recessive state is only present on the bus when none of the transmitters on the bus is asserting a dominant state. Irrespective of signal state the signals lines are always in low impedance state with respect to one another by virtue of the terminating resistors at the end of the bus.

This signalling strategy differs significantly from other balanced line transmission technologies such as RS-422/3, RS-485, etc. which employ differential line drivers/ receivers and use a signalling system based on the differential mode voltage of the balanced line crossing a notional 0 V. Multiple access on such systems normally relies on the media supporting three states (active high, active low and inactive tri-state) and is dealt with in the time domain. Multiple access on CAN bus is achieved by the electrical logic of the system supporting just two states that are conceptually analogous to a ‘wired OR’ network.

Frames

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 and/or remote frame

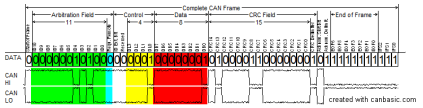
Data frame

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:

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Dominant (0) (see Remote Frame below)
Identifier extension bit (IDE)	1	Declaring if 11 bit message ID or 29 bit message ID is used. Dominate (0) indicate 11 bit message ID while Recessive (1) indicate 29 bit message.
Reserved bit (r0)	1	Reserved bit (it must be set to dominant (0), but accepted as either dominant or recessive)
Data length code (DLC)* (yellow)	4	Number of bytes of data (0–8 bytes)
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)

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	11	First part of the (unique) identifier for the data which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1). Optional
Identifier extension bit (IDE)	1	Must be recessive (1). Optional
Identifier B	18	Second part of the (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0)
Reserved bits (r0, r1)	2	Reserved bits (it must be set dominant (0), but accepted as either dominant or recessive)
Data length code (DLC)*	4	Number of bytes of data (0–8 bytes)
Data field	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)

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

* 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 and/or reception of a DLC greater than eight, but the actual data length is always limited to eight bytes.

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.

i.e.,

RTR = 0 ; DOMINANT in data frame

RTR = 1 ; RECESSIVE in remote frame

In the very unlikely 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. In this way, the node that transmitted the Remote Frame receives the desired data immediately.

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".

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.

Interframe spacing

Data frames and remote frames are separated from preceding frames by a bit field called interframe space. Overload frames and error frames are not preceded by an interframe space and multiple overload frames are not separated by an interframe space. Interframe space contains the bit fields intermission and bus idle and, for error passive stations, which have been transmitter of the previous message, suspend transmission. Interframe space consists of at least three consecutive recessive (1) bits.

Bit stuffing

In CAN frames, a bit of opposite polarity is inserted after five consecutive bits of the same polarity. This practice is called bit stuffing, and is due to the non-return to zero (NRZ) coding adopted. The stuffed data frames are destuffed by the receiver. Since bit stuffing is used, six consecutive bits of the same type (111111 or 000000) are considered an error.

Bit stuffing implies that sent data frames could be larger than one would expect by simply enumerating the bits shown in the tables above.

Standards

There are several CAN physical layer and other standards:

- **ISO 11898-1**: CAN Data Link Layer and Physical Signalling
- **ISO 11898-2**: CAN High-Speed Medium Access Unit

  ISO 11898-2 uses a two-wire balanced signalling scheme. It is the most used physical layer in car powertrain applications and industrial control networks.
- **ISO 11898-3**: CAN Low-Speed, Fault-Tolerant, Medium-Dependent Interface
- **ISO 11898-4**: CAN Time-Triggered Communication

  ISO 11898-4 standard defines the time-triggered communication on CAN (TTCAN). It is based on the CAN data link layer protocol providing a system clock for the scheduling of messages.
- **ISO 11898-5**: CAN High-Speed Medium Access Unit with Low-Power Mode
- **ISO 11898-6**: CAN High-speed medium access unit with selective wake-up functionality
- **ISO 11992-1**: CAN fault-tolerant for truck/trailer communication
- **ISO 11783-2**: 250 kbit/s, Agricultural Standard

  ISO 11783-2 uses four unshielded twisted wires; two for CAN and two for terminating bias circuit (TBC) power and ground. This bus is used on agricultural tractors. This bus is intended to provide interconnectivity with any implementation adhering to the standard.
- **ISO 15765-2** also called ISO-TP, is a standard for flow control and handling of messages larger than eight bytes.
- **SAE J1939-11**: 250 kbit/s, Shielded Twisted Pair (STP)
- **SAE J1939-15**: 250 kbit/s, Unshielded Twisted Pair (UTP) (reduced layer)

  The**SAE J1939** standard uses a two-wire twisted pair, −11 has a shield around the pair while −15 does not. SAE 1939 defines also application data and is widely used in heavy-duty (truck) and autobus industry as well as in agricultural & construction equipment.
- **SAE J2411**: Single-wire CAN (SWC)

Higher layer implementations

As the CAN standard does not include tasks of application layer protocols, such as flow control, device addressing, and transportation of data blocks larger than one message, and above all, application data, many implementations of higher layer protocols were created. Several are standardised for a business area, although all can be extended by each manufacturer. For passenger cars, each manufacturer has its own standard. Among these implementations are:

- ARINC 825 (for the aviation industry)
- CANaerospace (for the aviation industry)
- CAN Kingdom
- CANopen (used for industrial automation)
- CCP / XCP
- DeviceNet (used for industrial automation)
- EnergyBus (used for electrical vehicles)
- GMLAN (for General Motors)
- ISO 15765-4
- ISO 11783 or ISOBUS (agriculture)
- ISO 14229
- SAE J1939 (heavy road vehicles)
- ISO 11992 for heavy trailers
- MilCAN

- NMEA 2000 (marine industry)
- RV-C (used for recreational vehicles)
- SafetyBUS p (used for industrial automation)
- SmartCraft
- Smart Distributed System (SDS)
- VSCP (used for building automation)

Security

CAN is a low-level protocol, and does not support any security features intrinsically. Applications are expected to deploy their own security mechanisms; e.g., to authenticate each other. Failure to do so may result in various sorts of attacks, if the opponent manages to insert messages on the bus. Password mechanisms exist for data transfer that can modify the control unit software, like software download or ignition key codes, but usually not for standard communication.

Development tools

When developing and/or troubleshooting the CAN bus, examination of hardware signals can be very important. Logic analyzers and bus analyzers are tools which collect, analyse, decode and store signals so people can view the high-speed waveforms at their leisure. There are also specialist tools as well as CAN bus monitors.

References

[1] *Building Adapter for Vehicle On-board Diagnostic* (http://www.obddiag.net/adapter.html), obddiag.net, accessed 2009-09-09

[2] Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems A. Albert, Robert Bosch GmbH Embedded World, 2004, Nürnberg

[3] ISO 11898-1:2003 abtract (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33422)

External links

- Bosch specification (http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf) (old document — slightly ambiguous/unclear in some points, superseded by the standard
- ISO 11898 (http://www.iso.org/iso/search.htm?qt=Controller+Area+Network&searchSubmit=Search&sort=rel&type=simple&published=true))
- Bosch CAN FD Specification Version 1.0 (http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can_fd_spec.pdf)
- Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised (http://www.springerlink.com/content/8n32720737877071/)
- Pinouts for common CAN bus connectors (http://www.interfacebus.com/Can_Bus_Connector_Pinout.html)
- Independent discussion platform CANLIST (http://www.canlist.org/)
- Free Tutorials and Low cost development tools (http://www.dominantbit.com/)
- A webpage about CAN in automotive (http://marco.guardigli.it/2010/10/hacking-your-car.html)
- Controller Area Network (CAN) Schedulability Analysis with FIFO Queues (http://www.cs.york.ac.uk/ftpdir/reports/2011/YCS/462/YCS-2011-462.pdf)
- Controller Area Network (CAN) Implementation Guide (http://www.analog.com/static/imported-files/application_notes/AN-1123.pdf)
- Free Tutorial: Controller Area Network (CAN) Introduction and Fundamentals (http://microcontroller.com/CAN_(Controller_Area_Network):_Introduction_and_Fundamentals.htm)

Article Sources and Contributors

CAN bus *Source*: http://en.wikipedia.org/w/index.php?oldid=578994437 *Contributors*: 2001:8B0:1F4:CCDE:6542:6571:7CC9:6229, 2001:db8, 2A01:E35:8A15:9E60:E140:423E:8D06:166B, AGlossop, Aaron Nitro Danielson, AbuJazar, Accounting4Taste, Aechols, Agl, Ahoerstemeier, Ajn1, Akadruid, Albramallo, Alex.muller, Alexandreag, Allen Moore, Amorymeltzer, Anapaulasag, Andre maier, Andy Dingley, Arjayay, Armistej, Asher196, Attilios, Automotive joe, Axlq, BIL, Benburleson, Betsytimmer, Biker Biker, Bobblewik, Bsilverthorn, CAkira, Calltech, CanisRufus, Catverine, Cburnett, Chrisbolt, Colin Marquardt, Corwin8, D rock naut, DAALIYA, Deeptrivia, Denisarona, Depictionimage, DerHexer, Dicklyon, Dmorr, Dpotop, Drunken Pirate, Ee79, Egil, Electron9, Eliyahu S, EncMstr, Enric Naval, Erniotti, Fahidka, Finlay McWalter, Fleminra, Ft1, Fxhomie, Gabriel Kielland, Gejigeji, GoingBatty, Grafen, Gregmelson, Gtwfan52, Guy Harris, GyroMagician, Harriv, Heron, Hidayat ullah, Hmpeople, Hobsonlane, Hondrej, Hopp, IamNotU, IanOsgood, Idonra, Invernizzi.l, Inwind, Iviney, JacobBramley, Jidan, Jni, Joel7687, John of Reading, JohnCD, Johnny.cespedes, JonHarder, Jredders, Jschnur, JuergenKlueser, Juggers2k, Katharineamy, Kdub432, Khazar, Kinema, Kingdon, KnowledgeOfSelf, Kramer-Wolf, Kris iyer, Lakshmank85, Langerwolfgang, Leeraphael, Liptakokgabora, MC MasterChef, MStock, Marcelosr, Marco Guardigli, Marius siuram, MarkO555, MaxSem, Mc cappy, Mdem, Meestaplu, Meise, Miceduan, Michael Hardy, Millermk, MisterTS, Mortense, Mpeg4codec, MrOllie, Nasa-verve, Natevw, Niteowlneils, Nixdorf, Notyetinspace, Nyarcel, Ondrejandrej, Patpou77, Pentawing, Philippe, Phosphoricx, Pikamander2, Pingveno, Plindemann, Plupp01, Ponchobonjo, Quibik, Qxz, Radartooth, Ramtam, Rcpinto, Reply123, Robert K S, Rocketmagnet, Rocketrod1960, RolfBly, Ronz, Rotmat, Russella, Rvoorhees, SD5, SGBailey, Sagie, Sam8, Sbmeirow, Scientus, Selket, Sertrel, SethML, Shaddack, Simon04, Skomorokh, SlackerMom, Slgrandson, Stannered, StealthFox, Stobor827, Suruena, TFolsom, TastyPoutine, Tc.guho, Thaiio, TheAllSeeingEye, Thebigandroid, Thomas888b, Thunderbird2, TinyMark, Tony1, Tpikonen, Trailprice, Treekids, Triplestop, Tyler, Vanished user 34958, Vegaswikian, Versageek, Waheed446, Wdfarmer, Widr, Wikfr, Wikipedia tang, Wleizero, Wolfch, Woohookitty, Xoneca, Xylome, Yoenit, Zer0431, Zigger, ²¹², 508 anonymous edits

Image Sources, Licenses and Contributors

Image:CAN Bit Timing2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:CAN_Bit_Timing2.svg *License*: Public domain *Contributors*: en:User:Rocketmagnet, traced by User:Stannered

File:CAN-Bus Elektrische Zweidrahtleitung.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:CAN-Bus_Elektrische_Zweidrahtleitung.svg *License*: unknown *Contributors*: Alexander.stohr, Stefan-Xp

File:CAN-Bus-frame in base format without stuffbits.png *Source*: http://en.wikipedia.org/w/index.php?title=File:CAN-Bus-frame_in_base_format_without_stuffbits.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Erniotti

License