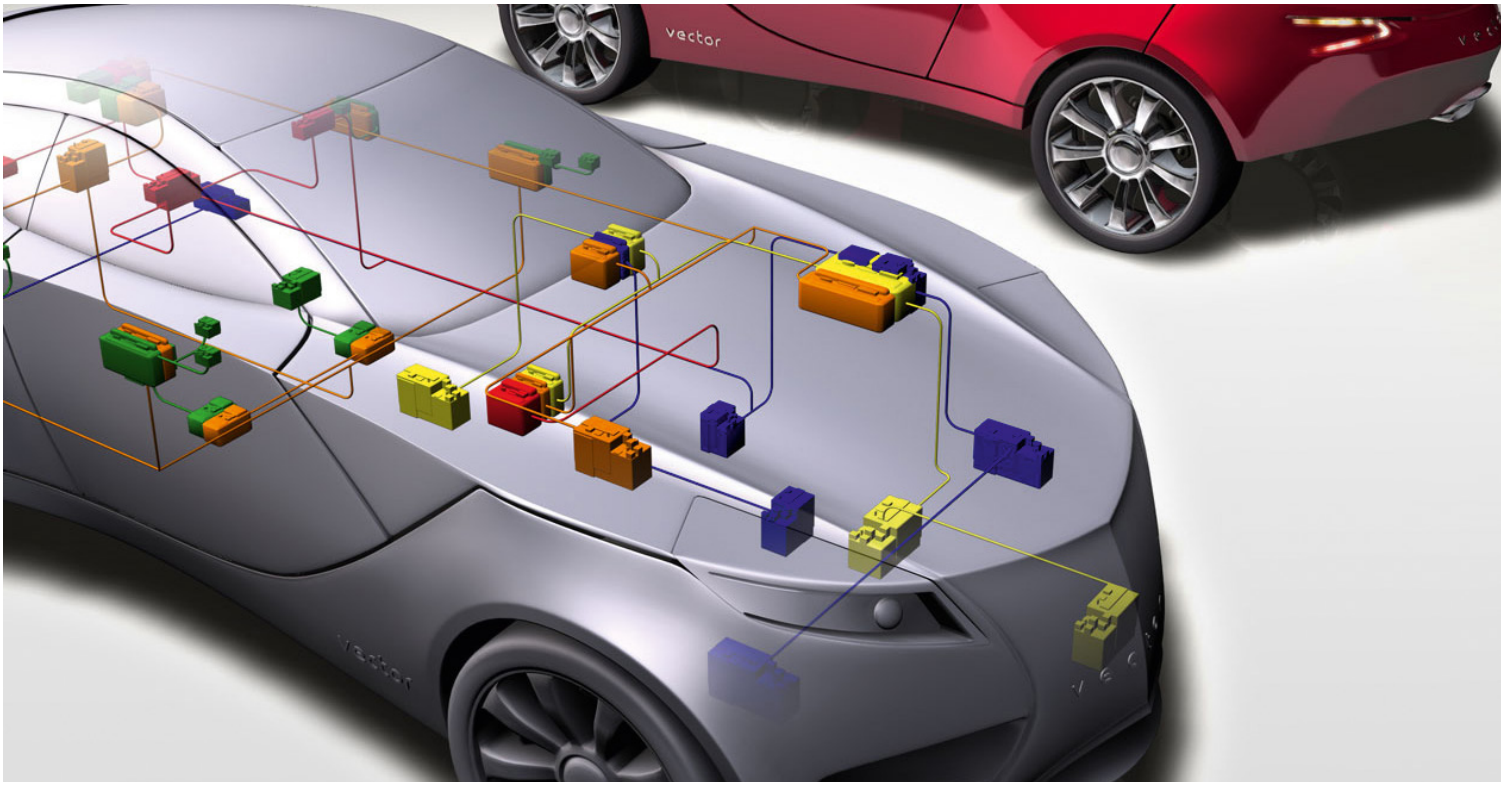
VECTOR



Data Communication in the Automobile – Part 2:

Reliable Data Exchange with CAN

As presented in part 1 of our article series, the increasingly complex electronic systems in automobiles are calling for a higher level of data exchange between the ECUs. In order to ensure this with sufficient reliability and speed, the CAN bus was developed. CAN stands for Controller Area Network. As the name implies, the CAN bus can link a larger area and reach a length up to multiple kilometers.

The CAN bus was developed by Bosch [1] and became a standard in 1993. It is currently available as ISO 11898 **(Figure 1)**. The standard is divided into multiple parts. The first part specifies the CAN protocol and covers all aspects of the data link layer (framing, addressing, bus access, data integrity) and the physical signal coding as part of the physical layer of the ISO 7498 reference model – the so-called OSI layer model (OSI: Open Systems Interconnection). The CAN controller was developed for handling the CAN protocol.

Parts 2 and 3 of ISO 11898 describe two versions of the physical layer, namely CAN high-speed and CAN low-speed. The latter is also often called fault-tolerant CAN because it continues functioning if one of its two wires breaks, although with diminished reliability. Parts 2 and 3 also cover the physical layer of the ISO reference model, including the physical bus connection, data rates, and the voltage levels on both CAN wires.

CAN uses differential signal transmission, which reduces noise sensitivity and requires two communication wires (CAN high and CAN low) that are terminated at both ends with characteristic impedance R_T of 120 Ω.

CAN high-speed is mainly used in drive and chassis applications. It is primarily implemented by the CAN high-speed transceiver, which supports a maximum data rate of 1 Mbit/s. The CAN low-speed physical layer has been used mainly in the convenience area. It is placed in doors, seats,

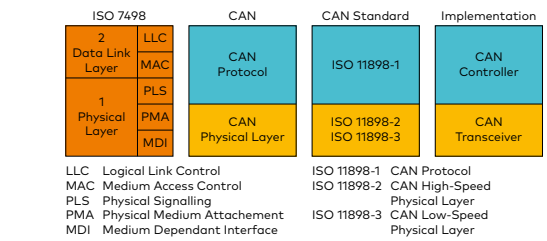


Figure 1: CAN Standard

and sunroofs where it is subject to bending that can cause wire breaks. Here the fault-tolerant CAN low-speed transceiver with a maximum data rate of 125 kbit/s is used. It can also be operated using a single wire. Although this type of CAN bus is rarely used.

The CAN interface consists of a CAN controller and a CAN transceiver **(Figure 2)**.

The CAN controller handles the CAN protocol. The CAN transceiver connects the CAN controller physically to both of the CAN wires and measures or generates the voltage levels on these two wires.

What Has to Go from Where to Where?

CAN uses the receiver-selective form of addressing. The identifier (ID) indicates the content of the transmitted data and not the destination. A message can thus be received and evaluated by all ECUs on the bus (message distribution). The application of a receiving ECU decides whether it evaluates a message. It can even set an acceptance filter in its CAN controller when starting, which hides unneeded CAN messages in the message stream based on their identifiers. CAN offers two sizes of identifiers: 11 bit and 29 bit. The smaller identifier (standard format) is used in passenger cars. It provides 2048 different messages, while the larger identifier offers 536 870 912 messages. The latter is mainly required in commercial vehicles for CAN-based software protocols, such as SAE J1939, but is now found in passenger cars as well.

Receiver-selective addressing offers the following advantages:

> Cost savings through shared use of sensors by all ECUs on the bus.
> Easy implementation of distributed functions
> It allows different configurations without adaptation of hardware or software

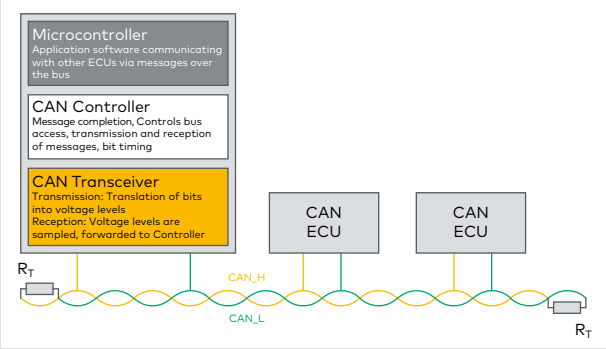


Figure 2: Structure of a CAN ECU

Events Trigger the Transmission of Messages

If a newsworthy event happens in everyday life, it is communicated in newscasts. In the world of serial buses, the term "event" is also used to describe an occurrence that requires information to be transmitted. For fast communication of information, the ideal situation is for the underlying event to directly trigger transmission of the respective data. This is referred to as event-driven transmission.

The alternative would be to transmit information according to a prescribed schedule or time pattern. But, if information is now produced that requires transmission and it's not the ECU's turn to send, the transmission must wait.

To avoid this wait, CAN was developed as an event-driven bus system. Every CAN node is authorized to access the CAN bus immediately after occurrence of an event and to send data that has been created. The only exception is if another ECU is already transmitting data. Courtesy dictates that another transmission is not to be interrupted.

The important thing here is that the other transmission does not last too long. CAN limits the message length to a maximum of 130 bits (for 11 bit identifiers). With the usual data transmission rate of 500 kbit/s in passenger cars, this leads to a transmission duration of approximately 0.25 milliseconds. The bus is available again after that. This is an important precondition for data transmission that must be sufficiently fast for applications like drive and chassis.

However, there is still a risk of collisions, namely when multiple ECUs want to start transmitting messages simultaneously after the bus becomes available again after a transmission. This danger rises with increasing bus load. If messages were to be destroyed as a result of a collision, this would cause the bus load to increase and initiate a vicious cycle that would place the availability of sufficient data transmission speed in doubt. To avoid this, the CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) bus access method is used in the CAN network.

Priorities Instead of Collisions

When an ECU wants to send, it must check whether the bus is free (Carrier Sense – CS). If the bus is busy, the ECU must wait. When the bus becomes available again, there is a possibility that other ECUs have been waiting for it too. In this case, all ECUs start sending messages (Multiple Access – MA). To avoid the impending damage from this collision (Collision Avoidance – CA or Collision Resolution – CR), a process now occurs that is referred to as bitwise arbitration.

All ECUs with a transmission request simultaneously send the identifier of their respective CAN message to be transmitted, bitwise from the most significant to least significant bit.

A bit with significance 0 is dominant on the CAN bus. This means if two ECUs simultaneously transmit different bit

values, the 0 value prevails over the 1 value on the bus. This is called "Wired AND bus logic" (0 = dominant, 1 = recessive). For each ID bit that is sent, each ECU compares the value on the bus with the value it sent. This is called bit monitoring. The rules of the arbitration logic determine whether an ECU may continue sending or must stop **(Figure 3)**.

One bit after the transmission of the identifier the arbitration ends. All ECUs that transferred a logical 1 when sending their identifier but found a 0 on the bus, had to stop sending. Only one ECU did not have to stop and can now transmit its entire message uncontested.

Every loser of an arbitration changes to receive mode and waits until the winner has finished transmitting its message. As soon as the bus is free again, the ECUs access it for another send attempt. In this way, the bus logic and arbitration logic not only prevent collisions (Collision Avoidance) but also ensure a priority-controlled bus access, since the smaller the numerical value of an identifier, the higher the priority of the related CAN message. Assuming the bus is free, the message with the smallest identifier (ID = 0) is therefore transmitted without delay. In the case of collisions, messages with numerically higher ID values have a risk of delay as a result of losing the arbitration.

With a lower bus load, the probability of collisions is low. This type of random, nondestructive, and priority-controlled bus access ensures fair and very fast bus access. When the bus load increases, collisions and delays of, in particular, low priority messages accumulate. In the worst case, information arrives too late at their receivers or not at all. For this reason, it is necessary to ensure when planning and developing a CAN bus that the bus load will not be too high.

The bitwise arbitration has yet another small obvious disadvantage: A bit must have a long enough duration so that delays in sending do not pose problems. An electrical signal has a propagation speed of approximately 200 000 km/s or 0.2 m/ns in copper. Additionally, there are delays through the CAN controller and transceiver. A bit level must last long enough on the bus that the ECUs most distant from the sender can still detect and evaluate it. As a result, there is a reciprocal relationship between the bus length and transmission rate.

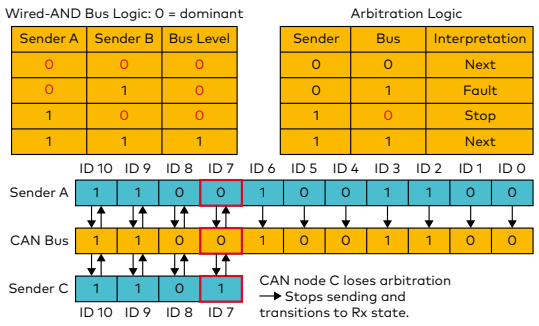


Figure 3: Dominance of Zero Bit and Arbitration

Ordering and Delivering Data

Road traffic includes public transit buses that run according to schedule and charter buses that run only when ordered. We find both models in CAN, too. Messages can be sent cyclically at regular intervals or only on request.

CAN buses in vehicles mainly use cyclic transmissions. CAN buses are shorter than 40 m in passenger cars and shorter than 200 m in commercial vehicles. As a result, high transmission rates are possible that allow cyclic sending without the bus load becoming too high.

On long CAN buses (over a kilometer), found in building complexes or industrial plants, only low transmission rates are possible. Cyclic sending of messages makes sense only if the cycles are correspondingly long. However, data can be sent on request.

There is the Data Frame for transmitting data **(Figure 4)** and the Remote Frame for requesting data. Because an identifier indicates the content of the data, it can be used by either, the Remote Frame and the Data Frame for transmitting the requested data. The Remote Transmission Request (RTR) bit, included at the end of the identifier, is also used for arbitration. This prevents problems if a remote frame and the associated data frame collide during arbitration. If RTR = 1, a Remote Frame is present. Otherwise, a Data Frame is present.

A Bit for the Clock Comparison

Synchronization between sender and receivers is a basic precondition for the transmission of Data Frames and Remote Frames. For reasons of cost and effort, CAN has no clock line. Synchronization is realized with signal edges and a defined mechanism. If nothing is being sent on CAN (bus idle), the bus level is recessive, as would be the case if only bits with value 1 were being sent. For this reason, a message begins with the transmission of a dominant synchronization bit (Start of Frame – SOF). All receivers adjust their clocks at the falling signal edge of this bit.

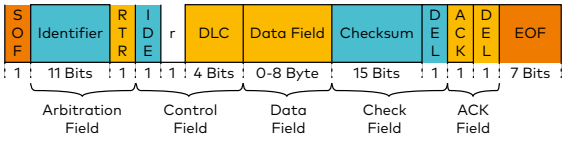


Figure 4: CAN Data Frame

Each receiver ensures synchronization throughout the transmission by evaluating each falling signal edge and adjusting its own time pattern (Bit Timing), if necessary. Long series of homogeneous bits do not exhibit any signal edges. To break these series, CAN generates additional signal edges by inserting a complimentary bit after five of equal value. This is called bit stuffing. In this way, a falling edge is forced after 10 bits at the latest. After synchronization, the receivers discard these stuff bits.

The SOF (Start of Frame) is followed by either the 11 bit ID or 29 bit ID. Its length is determined in the IDE bit (Identifier Extension). IDE = 0 means short IDs. IDE = 1 means long IDs. CAN allows a maximum of eight byte payload in the data field. The exact number of payload bytes is indicated using a DLC (Data Length Code). The data field is followed by the 15 bit check field, also called the checksum or cyclic redundancy check – CRC.

The sender calculates this checksum from all bits to be transmitted. In mathematical terms, it divides the bit string by the 15^{th} degree polynomial 0xC599 or $1x^{15} + 1x^{14} + 1x^{10} + 1x^{8} + 1x^{7} + 1x^{4} + 1x^{3} + 1x^{0}$ on the binary field, thus $x \in \{0,1\}$.

Each receiver does the same thing with the arriving bits. It then compares both sequences and rates the received message in the acknowledge slot (ACK slot) following the recessive CRC delimiter. 0 stands for good. 1 stands for error. A CAN message is concluded after the recessive ACK delimiter with the 7 bit recessive EOF (End of Frame). Because no more stuff bits are inserted after the CRC, the EOF is a one-to-one identifier for the end of the message. The EOF is followed by three recessive bits, but these no longer belong to the message. They are called an Intermission (ITM) or Interframe Space (IFS). Only after this is another message permitted to be sent.

Error Detection and the Bull in the China Shop

The probability that errors in CAN messages remain undetected is very low at $4.7 * 10^{-11}$ or less [2]. The CAN protocol has defined error detection mechanisms. On the receiver side, these include not only the CRC but also the check of the format (Form Check) and the check according to the bit stuffing rule (Stuff Check). The sender uses bit monitoring to check whether a bit level on the bus corresponds to the send request, and it evaluates the ACK slot.

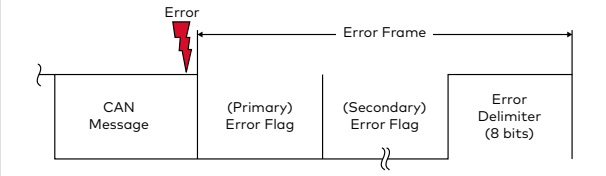


Figure 5: CAN Error Frame

For example, if statistics indicate that every thousandth message on a CAN bus exhibits an error and the CAN bus operates 1000 hours per year at a data rate of 500 kbit/s, an average bus load of 25% and an average message length of 80 bits, statistically a CAN message with an undetected error would only occur every 4000 years.

As soon as an ECU notices an error, it aborts the message transmission by sending six dominant bits. This so-called Error Flag is the proverbial bull in the china shop of the CAN protocol. Wherever it appears, it deliberately violates applicable rules, i.e. in most cases the bit stuffing rule. This impairs the affected CAN transmission to such an extent that all ECUs on the CAN bus can also detect only locally detectable errors. A second Error Flag follows because all other ECUs now see the error generated by the first Error Flag. Then eight recessive bits are awaited. This is called the Error Delimiter. The two Error Flags and the Error Delimiter together are called the Error Frame **(Figure 5)**. It is essential for network-wide data consistency.

Errors must be corrected! The sender of the corrupted message does this by retrying as soon as the CAN bus is free again, thus, after the Error Frame and ITM. However, there is no guarantee that the retry takes place immediately because every ECU is allowed to start sending after each ITM and which message is passed through is determined by the arbitration process. The error recovery time thus depends on the message priority and the bus load.

What would happen if an ECU signals an error where none existed? Active message transmissions would be aborted needlessly and the communication would be disrupted. This is prevented through self-monitoring of the CAN controller using an error counter. When an ECU detects an error first, i.e. sends the first of the two Error Flags, it must increase its counter by 8 points. As the sender of the second Error Flag, it only has to increment its counter by 1. If a message is received without error, the counter is reduced by 1. If the ECU detects the error when sending, it increases the Transmission Error Counter (TEC). Otherwise, it increases the Receive Error Counter (REC). If one of these counters exceeds the value 127, the CAN controller changes over to Error Passive mode. Consequently it can no longer send Error Flags consisting of dominant bits. Generating Error Flags with recessive level, it no longer interrupts or disturbs other ECUs, disturbing only itself while sending. The TEC continues to be incremented when errors occur until it forces the shutdown of the ECU when value 255 is reached **(Figure 6)**.

CAN Messages Are Graded

Every message transmission is graded by all receivers simultaneously. They all return an evaluation exactly within the ACK bit during the transmission of the sender. This is what is called an "In Transmission Response" or an "In

Frame Acknowledgment". A dominant level corresponds to a Good grade and a recessive level to a Poor grade. Because the sender sends the ACK slot recessively, already one Good grade is enough to confirm the correctness of the message transmission. Poor grades of other ECUs would thus be overwritten and remain unheard initially. However, these ECUs send an Error Flag after the ACK delimiter.
If there is no Good evaluation at all and the ACK slot remains recessive, the sender specifies an ACK error and cancels its message transmission with an Error Flag.

The Limits of CAN

CAN is still the most commonly used bus technology in automobiles. However, CAN has systemic limits. Due to the principle of arbitration, CAN itself is not deterministic even during cyclic sending of messages. Using CAN, extremely time-critical applications do not function with sufficient reliability. In addition, only a maximum of one million bits per second can be transmitted with CAN. Another is the high degree of reliability of CAN that makes it too expensive for simple applications that can manage without it.
Driver assistance or autonomous driving as well as comfort applications in the audio area and especially the video area require significantly higher data transmission rates and place higher demands on timely availability of data. Bus systems such as FlexRay and networks such as MOST and Ethernet have been established in these areas. But CAN also has been further developed, and an expansion of CAN, called CAN FD, is now available that enables significantly higher transmission rates. The revised CAN standard ISO 118981:2015 containing CAN FD has been in place since December 2015.
On the other end, LIN (Local Interconnect Network) fills the gap when it comes to cost-effective networking of sensors and actuators in the comfort area, such as for control of power windows, seats, external mirrors, sunroofs, and the air conditioning.

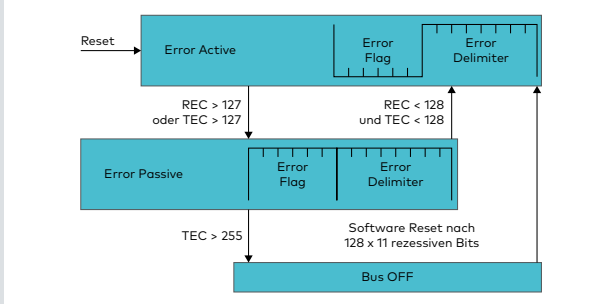


Figure 6: States of a CAN Controller

Reliable ECU Networking

Our specialists at Vector [3] support manufacturers and suppliers in the automotive industry and other industries not only in CAN networking, but also in communication systems such as LIN, FlexRay, CAN FD, and Ethernet.
We offer consistent tool chains for planning, design, development, and maintenance as well as software components and basic software for AUTOSAR ECUs. For getting started with ECU networking, we offer seminars on the basics of CAN, LIN, FlexRay, and Ethernet. To gain the necessary skills for handling the above named tools, engineers and technicians attend workshops where they learn all about the multifaceted development tasks concerning electronics in automobiles [4].
In the first part of this series [5], we addressed serial bus systems in general. The serial bus systems LIN, FlexRay, and MOST will be addressed in the remaining articles in the series. Interested readers will find additional in-depth information on the previously published topics on the Vector E-Learning platform [6].

Literature References:

[1] de.wikipedia.org/wiki/Controller_Area_Network
[2] Unruh, J.; Mathony, H. J.; Kaiser, K.H: Error Detection Analysis of Automotive Communication Protocols. SAE International Congress 1990.
[3] www.vector.com
[4] www.vector-academy.com
[5] Christmann, E.: Data Communication in the Automobile – Part 1: Architecture, Tasks, and Advantages of Serial Bus Systems
[6] elearning.vector.com

Ernst Christmann, Physicist, Mathematician
is a Senior Technical Trainer in the area of software tools for ECU testing and development and has worked for Vector Informatik GmbH in Stuttgart since 2004.