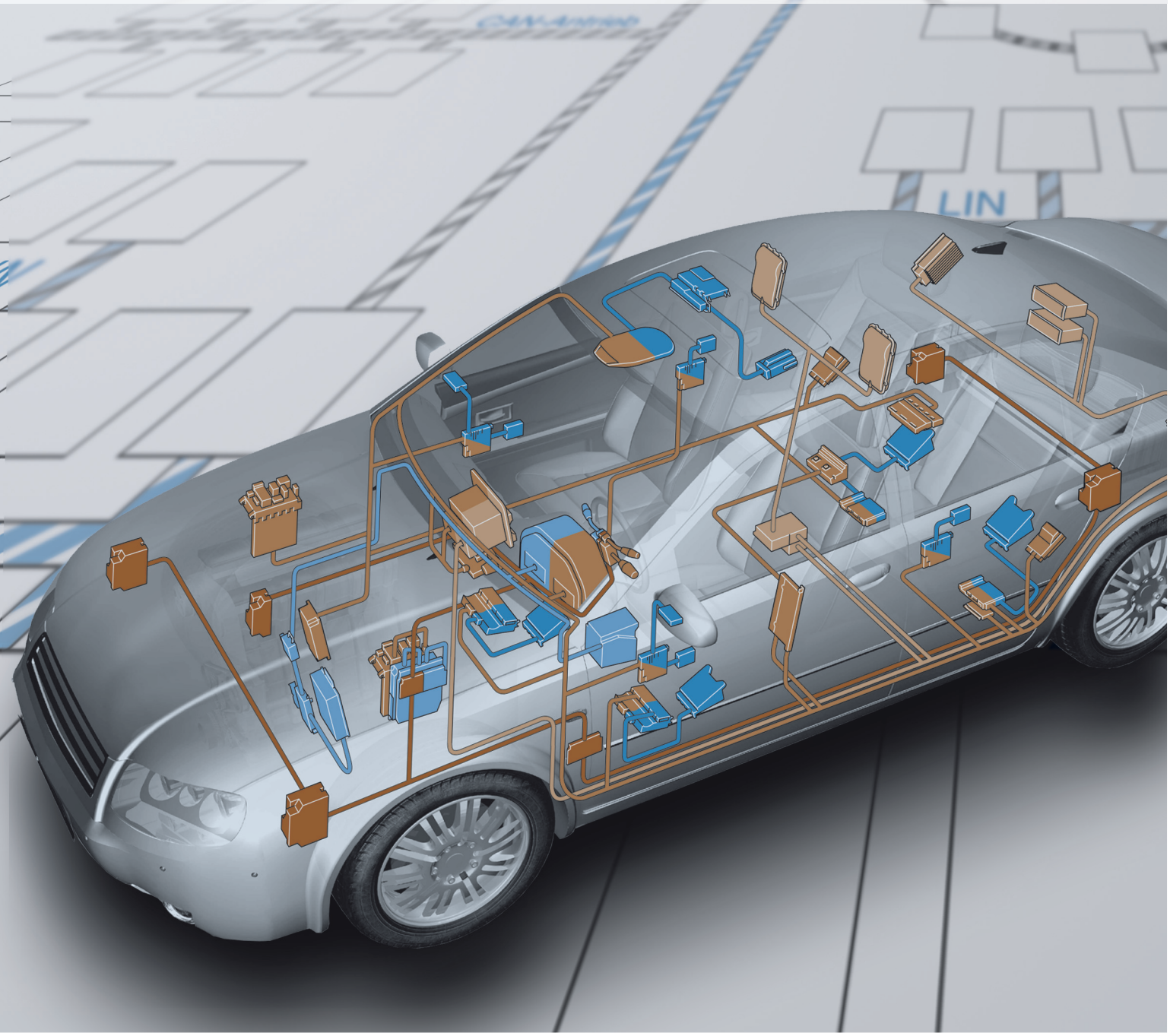
Automotive Networking

- Basic principles of networking
- Examples of networked vehicles
- Bus systems
- Architecture of electronic systems

Edition 2007 | Expert Know-How on Automotive Technology
Automotive Electrics/Automotive Electronics

BOSCH
Invented for life



Imprint

Published by:
© Robert Bosch GmbH, 2007
Postfach 1129
D-73201 Plochingen.
Automotive Aftermarket
(AA/MKK2)

Editorial team:
Dipl.-Ing. Karl-Heinz Dietsche.

Authors and contributors:
Dipl.-Ing. Stefan Mischo;
Dipl.-Ing. (FH) Stefan Powolny;
Dipl.-Ing. Hanna Zündel;
Dipl.-Ing. (FH) Norbert Löchel;
Dipl.-Inform. Jörn Stuphorn,
Bielefeld University;
Dr. Rainer Constapel,
DaimlerChrysler AG Sindelfingen;
Dipl.-Ing. Peter Häussermann,
DaimlerChrysler AG Sindelfingen;
Dr. rer. nat. Alexander Leonhardi,
DaimlerChrysler AG Sindelfingen;
Dipl.-Inform. Heiko Holtkamp,
Bielefeld University

and the editorial team in cooperation with the responsible technical departments of Robert Bosch GmbH.
Unless otherwise specified, the above are all employees of Robert Bosch GmbH.

Printed in Germany.
Imprimé en Allemagne.

1st Edition, July 2007.

(1.0)

Automotive Networking

Robert Bosch GmbH

Contents

As electronic systems and components continue to take hold in the automotive sector, so the number of electronic systems used in motor vehicles is constantly increasing. Such systems used to operate autonomously to a large extent, but today are combined to form an interconnected network. Powerful bus systems enable data to be exchanged between these systems.

Networking has taken hold right down to the small-car segment. The engine-management system exchanges data with other systems which have become standard features (e.g. antilock braking system, ABS). Many comfort and convenience functions (e.g. power-window units) are likewise controlled by bus systems. New functions – e.g. the further development of ACC (Adaptive Cruise Control) for stop & go traffic – require data from different vehicle domains. In other words, all the electronically controlled systems in a vehicle must communicate with each other.

Data transfer in the various vehicle domains is subject to different requirements. Various bus systems have therefore become established. This publication starts by explaining the basic principles of networking, then provides an overview of automotive networking, and uses several examples to demonstrate the need for networking in modern vehicles. A further chapter deals with the most important buses in detail.

The complexity of electrics/electronics continues unabated. It will be necessary in the future to pursue new approaches in order to be able to keep on top of this complexity. Standardized structures in the development of electronic systems ensure that in-vehicle electronics performs its functions reliably in spite of the increasing complexity.

Basic principles of networking

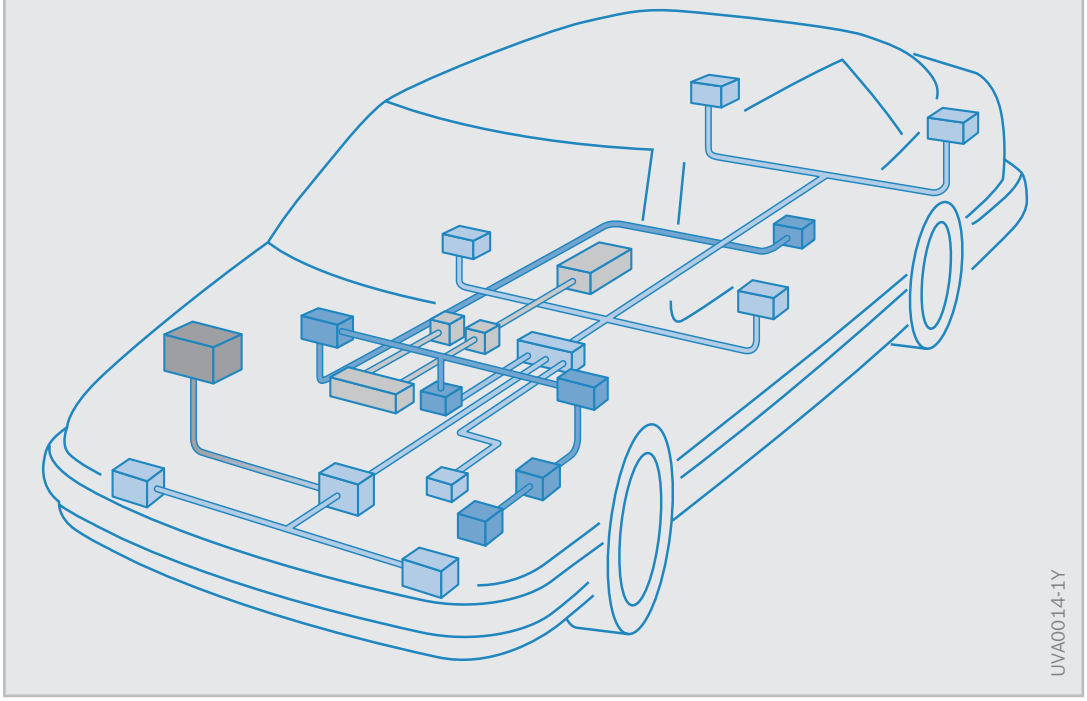
With the tremendous speed at which computer technology is advancing, the number of electronic systems in use is increasing more and more. This growth is also continuing in automotive engineering. However, this also means that the complexity of an overall system (the vehicle in this case) is on the increase. Individual systems such as engine management have been improved over the last few years. However, innovations are mainly achieved by means of interaction between several individual systems. The individual components need to be networked so that the multitude of information that is managed by the individual systems can also be used elsewhere throughout the system. Different communication systems are used depending on requirements (e.g. transmission reliability, fault tolerances, costs).

Network topology

A network is understood to be a system in which a group of elements can exchange information via a transportation medium. If the elements are visualized as nodes and the communication relationships as lines, a picture is created of a network where many nodes are related to several other nodes. The nodes in a communication network are also often referred to as network subscribers or stations.

In motor vehicles, complex control units such as those for the engine management system (Motronic or electronic diesel control, EDC), the electronic stability program (ESP), the transmission control system or the door modules can be network subscribers (Fig. 1). However, a sensor with a conditioning circuit that merely prepares and digitizes a measured value can also act as a network subscriber and make the measured signals available to other net-

1 Vehicle system networking



work subscribers. The transport medium via which the communication takes place is referred to as a bus or a data bus.

A network topology is understood to be the structure consisting of network nodes and connections. This merely shows which nodes are interconnected, but does not depict underlying details such as the length of the connection. Every network subscriber must have at least one connection to another network subscriber in order to participate in network communication. Different requirements are made of the network topology for a variety of communication network applications, while the topology determines some of the characteristics of the overall network. All network topologies are based on the following four basic topologies

- Bus topology
- Star topology
- Ring topology and
- Mesh topology

Other structures (hybrid topologies) can be created by combining these basic topologies.

Bus topology

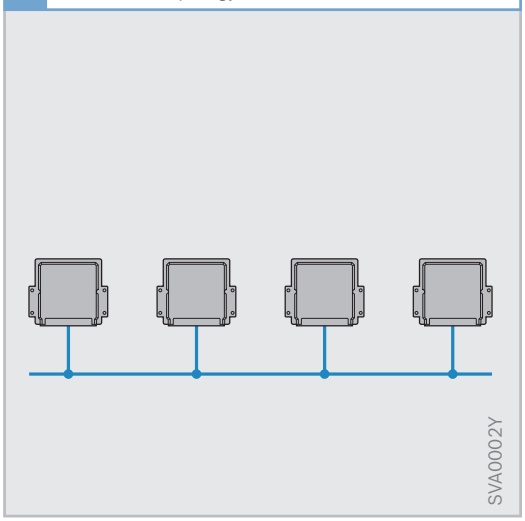
This network topology is also referred to as a linear bus. The core element of a bus topology is a single cable to which all nodes are connected via short connecting cables (Fig. 2). This topology makes it extremely easy to add other subscribers to the network. Information is transmitted by the individual bus subscribers in the form of so-called messages and distributed over the entire bus.

Nodes transmit and receive messages. If a node fails, the data that is expected from this node is no longer available to the other nodes on the network. However, the remaining nodes can continue to exchange information. However, a network with a bus topology fails completely if the main line is defective (due to a cable break, for example).

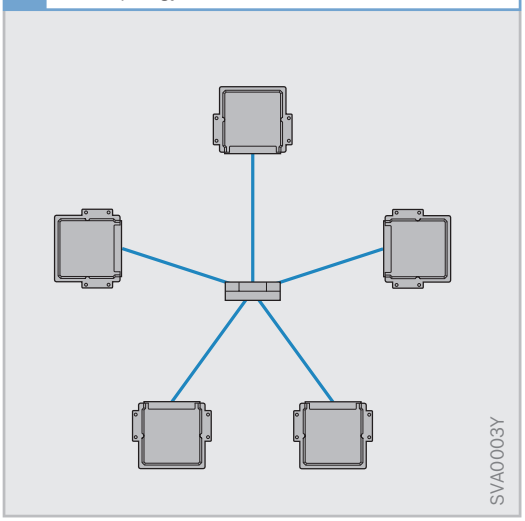
Star topology

The star topology consists of a main node (repeater, hub) to which all other nodes are coupled via a single connection (Fig. 3). A network with this topology is therefore easy to extend if free capacity is available (connections, cables).

2 Linear bus topology



3 Star topology



In star topologies, data is exchanged between the individual node connections and the main node, whereby a distinction is made between active and passive star topologies. In active star topologies, the main node contains a computer that processes and relays information. The performance capability of a network is essentially determined by the performance capability of this computer. However, the main node does not have to have special control intelligence. In passive star systems, it merely connects the bus lines of the network subscribers together.

The following applies to active and passive stars: if a network subscriber fails or a connecting line to the main node is defective, the rest of the network continues to operate. However, if the main node fails the entire network is disabled.

In the automotive area, star structures are under discussion for safety and security systems such as brakes and steering. In this case, the risk of a complete network failure is prevented by designing the main node to be physically redundant. This means that several main nodes are used to which the nodes whose information is needed for safe operation of the vehicle can be connected in parallel.

Ring topology

In the ring topology, each node is connected to its two neighbors. This creates a closed ring (Fig. 4). A distinction can be made between single rings and double rings.

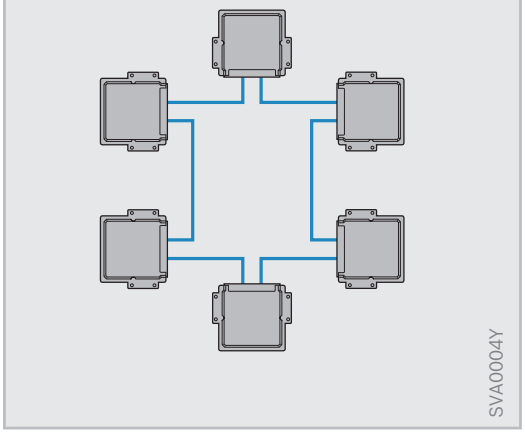
In a single ring, data transfers are unidirectional from one station to the next. The data is checked when it is received. If the data is not intended for this station it is repeated (repeater function), boosted and relayed to the next station. The data that is being transferred is therefore relayed from one station to the next in the ring until it has reached its destination or arrives back at its point of origin, where it will then be discarded.

As a soon as a station in a single ring fails, the data transfer is interrupted and the network breaks down completely.

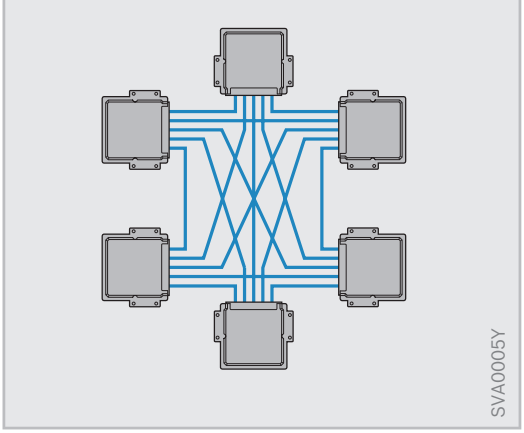
Rings can also be set up in the form of a double ring (e.g. FTTI), in which the transfer of data is bidirectional. In this topology, the failure of a station or a connection between two stations can be overcome, since all data is still transferred to all operational stations in the ring.

However, if several stations or connections fail, the possibility of a malfunction cannot be ruled out.

4 Ring topology



5 Mesh topology



Mesh topology

In a mesh topology, each node is connected to one or more other nodes (Fig. 5). In a fully meshed network, each node is connected to every other node.

If a mode or connection fails it is possible for the data to be rerouted. This type of network therefore has a high degree of system stability. However, the cost of networking and transporting the message is high.

Radio networks form a type of mesh topology, since the transmissions from each station are received by every other station that is within range.

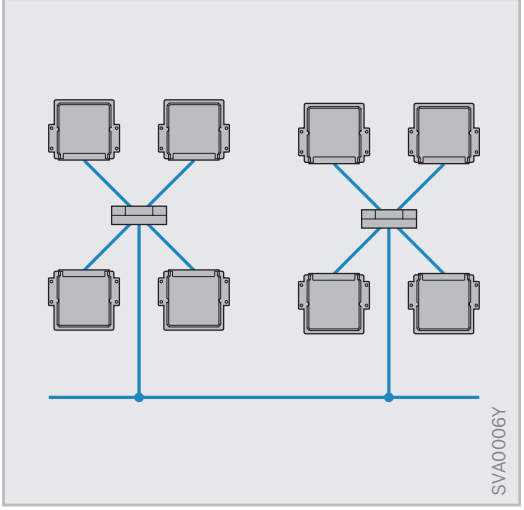
A mesh topology is bus-like as far as exchanging messages is concerned, and star-like regarding data transfers, since every station receives all transmissions from every other station, but connection failures can be overcome.

Hybrid topologies

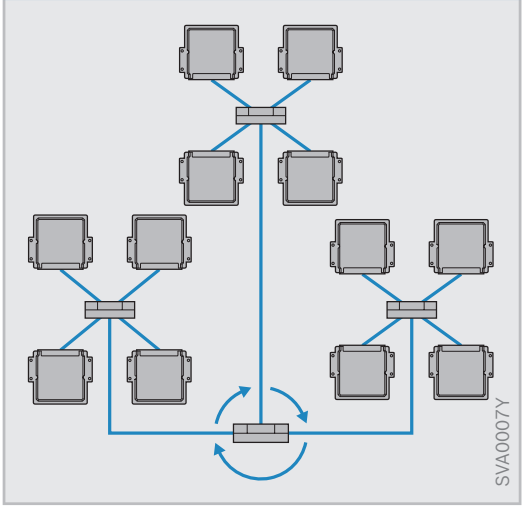
Hybrid topologies are a combination of different network topologies. Examples of such combination are:

- Star bus topology: the hubs of several star networks are interconnected as a linear bus (Fig. 6).
- Star ring topology: the hubs of several star networks are connected to the main hub (Fig. 7). The hubs of the star network are connected in the form of a ring in this main hub.

6 Star bus topology



7 Star ring topology



Network organization

Addressing

In order to make it possible to transmit messages via a network and evaluate the contents thereof, the useful data (payload) that is transmitted is also accompanied by data transfer information. This can be explicitly contained within the transmission or implicitly defined using preset values. Addressing represents important information for data transfer information. It is needed in order for a message to be sent to the correct recipient. There are different ways of doing this.

Subscriber-oriented method

The data is exchanged on the basis of node addresses. The message sent by the transmitter contains the data to be transmitted and also the destination node address (Fig. 8a). All receivers compare the transmitter receiver address to their own address, and only the receiver with the correct address evaluates the message.

The majority of conventional communication systems (such as Ethernet) operate using the subscriber addressing principle.

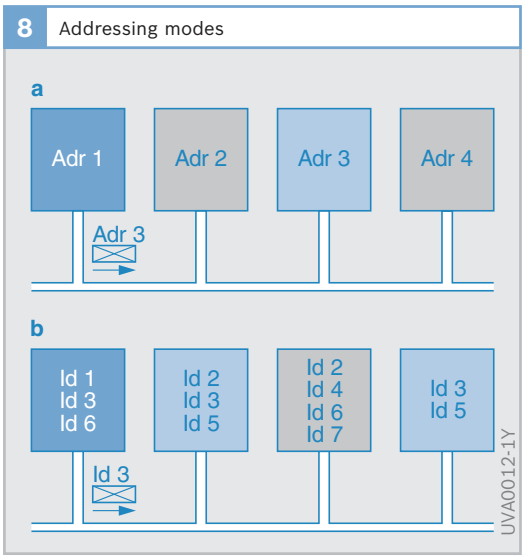


Fig. 8
a Subscriber-oriented method
b Message-oriented method

Message-oriented method

In this method it is not the receiver node that is addressed, but the message itself (Fig. 8b). Depending on the content of the message, it is identified by a message identifier that has been predefined for this message type. In this method, the transmitter does not need to know anything about the destination of the message, since each individual receiver node decides whether or not to process the message. Of course, the message can be received and evaluated by several nodes.

Transmission-oriented method

Transmission characteristics can also be used to identify a message. If a message is always transmitted within a defined time window, it can be identified on the basis of this position. By way of a safeguard, this addressing is often combined with message or subscriber-oriented addressing.

Bus access method

A node must access the bus in order to transmit a message. In the bus access method a distinction is made between

- Predictable methods in which the bus access is determined by certain time-dependent network characteristics, whereby only one node can transmit at a time and
- Random methods whereby any node can attempt to transmit data if the bus appears to be free

In the predictable method the bus access right is determined before bus access. It can thereby be ensured that only one subscriber is using the bus at a time. Access collisions because of simultaneous bus usage will be prevented if all subscribers use this method.

In the random method, the nodes can simultaneously attempt to use the bus as soon as it appears to be free. The timing of the bus access is therefore random. There is a risk of transmission collisions using this method, which will require attention. This can be dealt with by repeating transmissions after a collision has been detected (e.g. Ethernet), by giving the transmissions different codings (CDMA), controlling communication via a master or prioritizing message types or transmitters.

Time division multiple access (TDMA)

TDMA is a deterministic (predictive) access method. In this case each node is assigned a time window in which it is allowed to transmit (a priori). A fixed schedule is therefore required for the network. There is not usually a main communication subscriber controlling the communication procedure. However, concepts exist in which in which it is possible to switch between different schedules if necessary. The internal clocks of the different stations must run extremely synchronously with TDMA, since the transmit windows have to be adhered to with extreme precision.

Master-slave

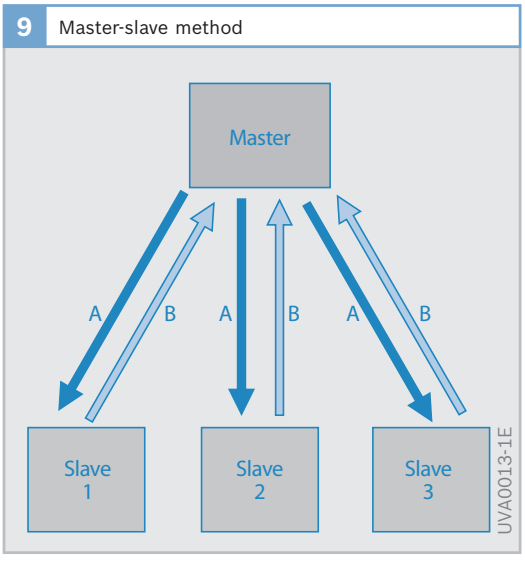
In the master-slave system, one node on the network operates as the master. This node determines the communication frequency by interrogating its subordinate nodes (slaves). A slave only replies if it is spoken to by the master (Fig. 9). However, some master-slave protocols allow a slave to contact a master in order to transmit a message (e.g. transmit information about the position of the power-window unit to the door module).

Multimaster

In a Multimaster network, several nodes can access the transport medium independently without the assistance of another node. Bus access is uncontrolled. Every node can access the bus and transmit a message if the bus appears to be free. This means that each node is its own master, and that any node can start a message transfer with equal status. However, this also means that collision detection and handling methods have to be in place. For example, this may be in the form of a decision-making phase with prioritization or delayed transmission repeats. The use of priority control prevents a bus conflict if several nodes wish to use the bus at the same time, since the network node that has high priority or wishes to transmit a message with high priority forces its way through in the event of a conflict and transmits its message first. Normal message transmission resumes when the line is free again.

The Multimaster architecture has a positive effect on the availability of the system, since no individual node is in control of communication whose failure would lead to total communication breakdown.

9 Master-slave method



OSI reference model

Network protocols are usually defined in layers, which combine properties and tasks. The properties of the deeper-lying layers are assumed in the next level up. This has the advantage that individual layers are exchangeable, provided that the interfaces that are provided between the layers remain unchanged.

The ISO OSI reference model (Open Systems Interconnection) provides a basis for describing and comparing many communication protocols. This was developed by the ISO (International Standardization Organization) and led to the adoption of international standards by ISO and IEEE (Institute of Electrical and Electronic Engineers).

In the OSI model, data communication systems are depicted in different layers (Fig. 10). The complex task of data communication is distributed among clearly arranged functional areas (layers). Not all of the layers in the OSI model are needed in a simple communication system. Layers can also be combined for many applications. Network protocols in the automotive area are often divided up into

- Physical layer
- Communication layer and
- Application layer (user layer)

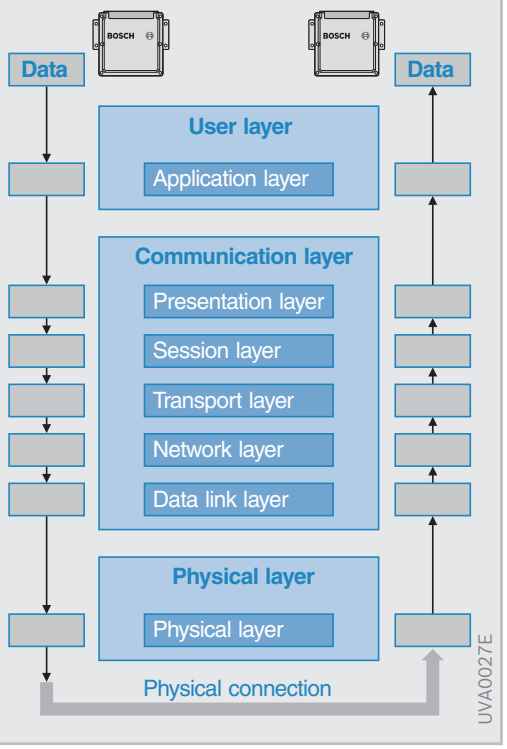
Physical layer

The electrical and procedural parameters of the physical connection between the network subscribers is defined in the physical layer.

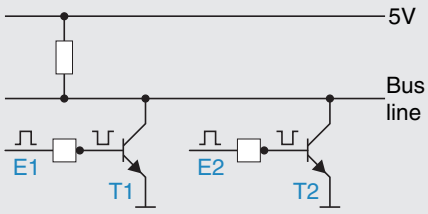
Signal level

In digital technology, data is represented by sequences of the two binary statuses, 0 and 1. In order to transmit the data on a bus, these statuses must be represented on the transmission agent. It is particularly important to avoid short-circuits on the

10 OSI reference model



11 Creating dominant and recessive levels



Control E	Status T	Bus level
0	conducts	0 V
1	blocks	5 V

E1	E2	Bus level	
0	0	0 V	dominant
0	1	0 V	dominant
1	0	0 V	dominant
1	1	5 V	recessive

SVA0008Y

bus when one node is transmitting a status of 1 and another is transmitting a status of 0.

The binary statuses can be depicted in many different ways. The serial interface of the PC, for example, uses +12 V and -12 V, and CAN-B uses voltages of 0 V and 5 V. The voltages of the serial interface are unsuitable for a bus, since short-circuits can occur if several subscribers wish to transmit conflicting binary statuses simultaneously.

If the coding allows one level to overwrite another, the overwriting level is referred to as dominant, and the subordinate level as recessive.

It is also possible to depict dominant and recessive levels using visual media. A status of 1 (recessive) then corresponds to e.g. dark, and a status of 0 (dominant) corresponds to light. In an optical fiber, an individual node can override all of the others by feeding light into the conductor.

Bit stream

The application information cannot usually be transmitted directly. In order to make transmission possible, the information is first incorporated as a payload in the frame of a message that contains information to be transmitted. Since all protocols have been developed in accordance with different requirements, the frame format differs from protocol to protocol.

The frame needs to converted into a bit stream to actually transmit the information, i.e. a sequence of bits that can be transmitted via the transport medium as physical states.

Example of the UART interface

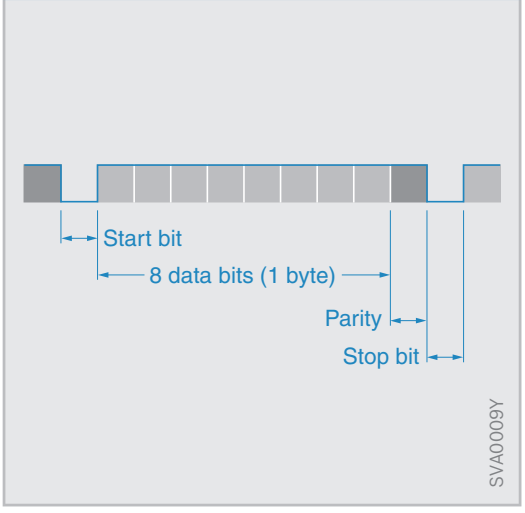
The microcontrollers that are used in control units have a simple interface (UART, Universal Asynchronous Receiver/Transmitter) on the chip, via which they can communicate with the outside world (e.g. with a PC). The essential features of a data transmission are read out via this interface.

If no data is being exchanged, the bus level is 5 V (microcontroller operating voltage, Fig. 11). When the start bit is transmitted (dominant level), the other station connected to the bus (receiver) is notified that a data transfer is starting (Fig. 12). The length of the start bit determines a bit time that represents the basis for the entire data transfer. Every subsequent data bit has the same length. The reciprocal of this time corresponds to the data transfer rate, i.e. the number of bits that can be transmitted in one second in a continuous data stream. All participating stations must be set to the same data transfer rate.

After the start bit has been received, the transmission of an 8-bit data word commences (1 byte) with the lowest significant bit (LSB, Low Significant Bit). The receiver that has synchronized itself to the start bit scans the data bus between each data bit and therefore assembles the transferred data byte.

The eighth data bit is followed by the parity bit. This bit indicates whether the number of transmitted ones is odd or even. It therefore allows the receiver to perform a simple check for possible transmission errors. The sequence is completed with the stop bit, which is placed onto the bus

12 UART interface transmission frame



with a dominant level. The next data transfer can then take place.

Communication layer

Control units can only interconnect and exchange data if they speak the same "language". This language determines the rules that are used to exchange data between the individual network subscribers.

The communication layer accepts data from the application layer, prepares it for transmission and forwards it to the physical layer.

The essential features of this protocol layer are:

- Message frame format
- Bus access control
- Message addressing
- Detection and handling of collisions
- Network node synchronization
- Checksum calculation

Application layer

The application layer consists of the application that processes and provides the information. The application layer is the only protocol layer to be affected by user or sensor input.

Control mechanisms

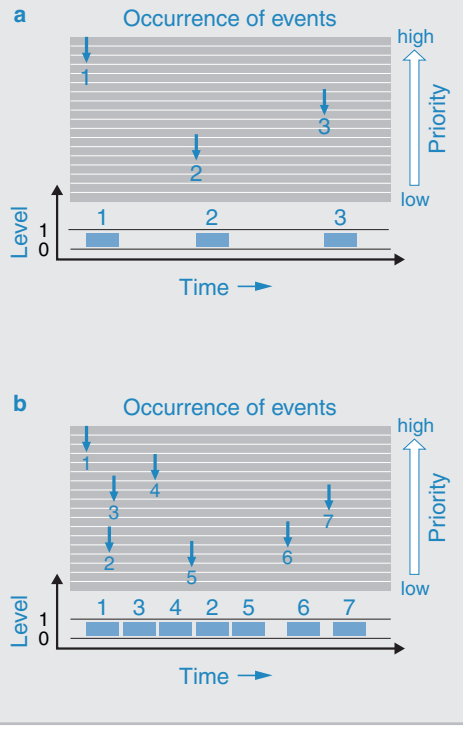
Event control

In an event-driven bus system, messages are transmitted as soon as an event that triggers the transmission of a message has occurred (Fig. 13a). Examples of such events are:

- Pressing a button on the air conditioning system control panel
- Operating the hazard warning flasher switch
- Incoming message that requires a reaction (e.g. information from rpm sensor to speedometer needle motor)
- Expiration of a fixed time period (time frame, e.g. 100 ms), after which messages are transmitted cyclically

Since the stations are not synchronized with each other, situations where several stations wish to access the bus simultaneously are unavoidable. In order to allow

13 Event control



a message to be transmitted without falsification, only one station at a time can transmit data on the bus. Collision avoidance mechanisms are available for preventing or solving bus conflicts.

If a node wishes to transmit a message whilst the bus is occupied, the transmission is delayed (Fig. 13b). A station that is ready to transmit must then wait until the transmission that is currently in progress has been completed.

Since bus access is subsequently renegotiated, the transmission may be delayed yet again. These delays become problematic if the bus becomes overloaded by a large number of network subscribers that wish to transmit messages. In this case messages may be lost if the transmitter abandons the transmission due to excessive delays.

Event-driven bus systems are suitable for reacting to asynchronous (unforeseen) events as quickly as possible. In an ideal case, they reduce the delay between the occurrence of the event and the message transmission (latency time) compared to time-driven systems. However, the latency time can vary considerably depending on the network loading.

Advantages

- High level of flexibility and capability of retrofitting new nodes in the network
- Good response time to asynchronous external events
- Bus usage depending on event frequency in line with requirements
- No network loading by unused events, since only events that have actually occurred trigger a transmission

Disadvantages

- Static bus occupancy, non-deterministic (i.e. not possible to prove that a message was transmitted at the right time)

Timer control

In the most recent developments in dynamic driving systems such as brakes and steering, an increasing number of mechanical and hydraulic components are being replaced with electronic systems (x-by-wire). Mechanical connections such as the steering column are becoming superfluous, and the functionality thereof is being taken over by sensors and actuators. The reliability, safety and failure tolerance requirements of these systems are extremely high. This means:

- Messages must be received on time
- The latency time of critical messages must be extremely small
- The system must have a redundant design
- The failure of a node must affect the rest of the system as little as possible and
- It must be possible to achieve a safe operating status from any fault situation

X-by-wire systems require close networking by the various components. The external increase in complexity places new demands on the safety, failure tolerance and availability of the communication system. The demands that are made of the electronic and network architecture therefore also increase. A reliable, fault-tolerant network architecture is required so that data is transmitted with guaranteed transmission characteristics, and electronic system malfunctions are handled in the most efficient way.

System architectures for real-time applications meet these requirements because their behavior is predictable and verifiable because of the way in which they are constructed. In these protocols, time windows within which a node is permitted to transmit are assigned to the control units in the communication network (nodes) during network planning (Fig. 14). In order to comply with the time window, the nodes must be synchronized as precisely as possible.

All transmissions are processed sequentially in accordance with the network planning (without collisions). Once each node has transmitted its message, the cycle restarts with the first transmitter. This makes it possible to determine how chronologically up-to-date the data is at any time. Since missing messages are detected immediately, time-triggered concepts are more reliable than event-driven systems.

If a fast data rate is required in a time-triggered system, the time delay between the occurrence of an event and the transmission of the data can be so small that the system complies with strict real-time requirements.

The bus can be protected from unauthorized access by a bus guardian. The bus guardian prevents a defective node from interfering with network communication by transmitting messages outside the relevant transmit window.

These characteristics make it possible to create redundant, fault tolerant systems in which transmission errors can be remedied and faults in the network can picked up by network nodes that can provide the functionality without errors.

Advantages

- Deterministic system
- Punctual data transmission
- Reliable detection and isolation of defective network nodes

Disadvantages

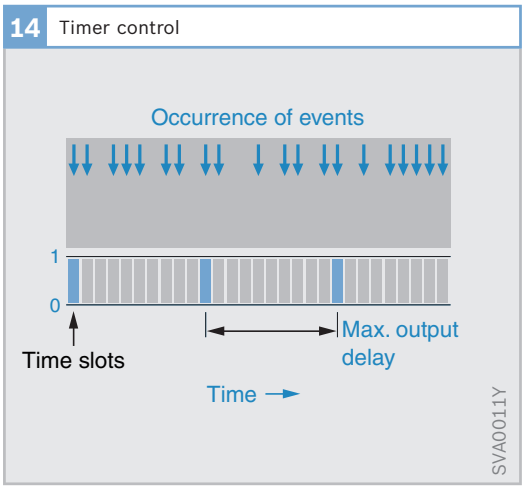
- Overall system must be planned for distributed developments
- Capacity for expanding the communication system must be planned in
- Good response time to asynchronous external events

Composability

If a communication system allows independently developed subsystems to be integrated in an overall system, it is said to support composability. An important criterion when doing this is that the properties that have been assured for the functionality of a subsystem are not adversely affected by adding other subsystems. If this has been ensured, the checking of system functionality is restricted to subsystem checking that can be carried out by the constructor of the subsystem.

If a communication system supports composability, changes can be made to a control unit without affecting the functionality of other control units. It is therefore not necessary to recheck the entire system after integrating a modified control unit – it is sufficient to check that the individual subsystems are operating reliably. Composability therefore reduces the time and cost of integrating new subsystems. This is the only way to increase the complexity of the electronics in the vehicle.

14 Timer control



Overview of bus systems used in vehicles

	CAN-C high-speed CAN	CAN-B low-speed CAN	LIN	TTP
Definition	Controller area network	Controller area network	Local interconnect network	Time-triggered protocol
Bus type	Conventional bus	Conventional bus	Conventional bus	Conventional and optical bus
Domains	Drivetrain	Comfort/ convenience	Comfort/ convenience	Safety-related networking
Applications	Engine management, transmission control and ABS/ESP networking	Body and comfort and convenience electronics networking	Low-cost expansion of CAN bus for simple applications in the comfort and convenience electronics area	Networking in safety-related environments such as brakes, steering, railway signal boxes or aircraft landing gear
Most frequently used topology	Linear bus	Linear bus	Linear bus	Star topology
Data transfer rate	10 kbit/s to 1Mbit/s	Max. 125 kbit/s	Max. 20 kbit/s	Unspecified, typ. 10 Mbit/s
Max. number of nodes	10	24	16	Unspecified
Control mechanism	Event-driven	Event-driven	Time-driven	Time-driven
Bus lines	Copper conductors (twisted pair)	Copper conductors (twisted pair)	Copper conductor (single wire)	Copper conductors (twisted pair)
Deployment	in all vehicles	in all vehicles	in all vehicles	Premium class vehicles, aircraft, rail control systems
Standard	ISO 1198	ISO 11519-2	LIN consortium	TTA group
SAE classification	Class C	Class B	Class A	Drive-by-wire

	MOST Bus	Bluetooth	Flexray
Definition	Media oriented systems transport	Proprietary name (Danish king)	Proprietary name
Bus type	Optical bus	Wireless	Conventional and optical bus
Domains	Multimedia and Infotainment	Multimedia and Infotainment	Deployment across all domains
Applications	Transmission of control, audio and video information	Data transfers over short distances, e.g. mobile phone integration in the infotainment system	A network system for use in safety-related and simple applications
Most frequently used topology	Ring topology	Network topology (radio)	Star topology
Data transfer rate	Max. 22.5 Mbit/s	Max. 3 Mbit/s (v2.0) Max. 723 kbit/s (v1.2)	Typ. 10 Mbit/s Max. 20 Mbit/s
Max. number of nodes	64	8 active (up to 256 passive)	Theoretically up to 2,048 Max. 22 per passive bus/star
Control mechanism	Time and event-driven	Event-driven	Time and event-driven
Bus lines	Plastic or glass optical waveguides	Electromagnetic radio waves	Copper conductors (twisted pair)
Deployment	Premium class vehicles made by European manufacturers	All vehicles, connection between multimedia equipment and infotainment system	Pilot application
Standard	MOST cooperation	Bluetooth SIG	Flexray consortium
SAE classification	Mobile Media	Wireless	Drive-by-wire

Table 1

Automotive networking

Electrical and electronic systems in motor vehicles are often not independent of each other but influence and complement each other. For this reason, signal lines were used in previous injection and ignition systems in order to simplify communication between these two systems. However, the increasing number of electronic systems rapidly increased the demand for and the scope of the information that was being exchanged. The number of signal lines and plug connections that is required increased accordingly, meaning that the technology that has so far been used was approaching the limit of its capability.

The solution was provided by the development of serial bus systems, with which large volumes of data from different sources can be transferred. A serial bus system was first used in a vehicle in 1991, when the CAN bus was used in the Mercedes-Benz 500E.

The demand for additional driving safety, convenience, economy and stricter legal requirements on the environmental compatibility of motor vehicles can only be achieved with the aid of additional electronics. The number of electronic systems in vehicles is therefore increasing all the time (Fig. 2).

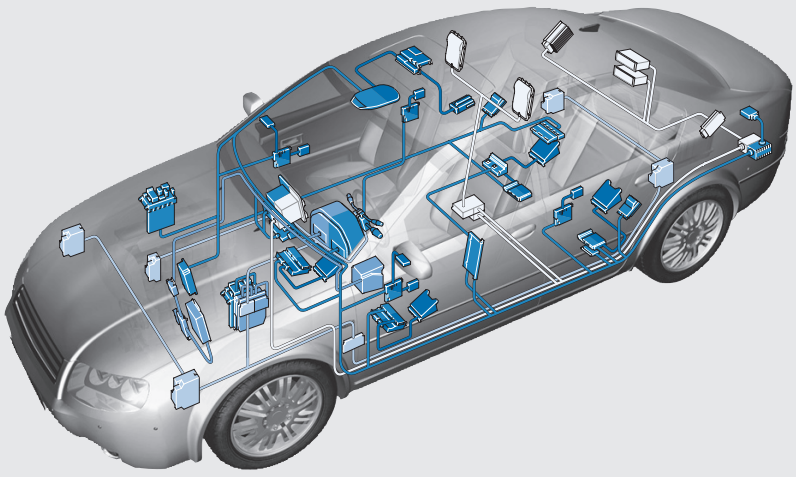
Cross-system functions

If you examine the signals that are processed in the individual systems, it becomes evident that many signals are needed in several control units. For example, the driving speed is evaluated in the electronic stability program (ESP) for the vehicle dynamics control, in the engine management for the automatic speed control (cruise control) and in the car sound system for the speed-dependent volume control.

The preparation of these variables from sensor signals requires computing power and therefore hardware and software resources. It is therefore advisable for these variables always to be calculated in a control unit and transmitted to other control units via a communication network.

However, in order to implement cross-system functions, electronic systems can also exchange information and therefore influence each other. Intelligent sensors are also considered to be electronic systems that prepare the sensor signal in an evaluation circuit and put the information on the data bus via a bus interface. Pre-crash sensors detect a pending collision, for example; the airbag control unit then sends the door modules and the overhead control panel a request to close the windows

1 Automotive networking



and the sliding roof. This protects the occupants from penetrating objects.

Another example of a system-encompassing function is the adaptive cruise control (ACC), in which the radar sensor, the engine management, the electronic stability program (ESP) and the transmission control communicate with each other. Distance control from the vehicle in front that is adapted to the flow of traffic is made possible in this system by means of engine torque adjustments, automatic brake system intervention and gear selection.

Coordination between the individual systems is therefore required for cross-system functions. Large volumes of data must be exchanged to do this. As well as powerful components, a powerful communication system is also required, with a low-cost network that is suitable for automotive vehicles. Special serial databus systems have been developed for this purpose.

The use of bus systems has the following advantages in comparison to a solution that uses conventional wiring:

- Reduced costs with less weight and installation space because of fewer cables in the wiring harness
- Better reliability and functional reliability due to fewer plug-in connections
- Simplification of vehicle assembly during production
- Multiple use of sensor signals
- Simple connection of system components to a bus
- Easier handling of equipment and special equipment variants in a vehicle

Requirements for bus systems

Both financial (e.g. cable costs, component costs) and technical constraints must be taken into consideration when a bus system is being selected. The most important technical selection criteria are explained in the following.

Data transfer rate

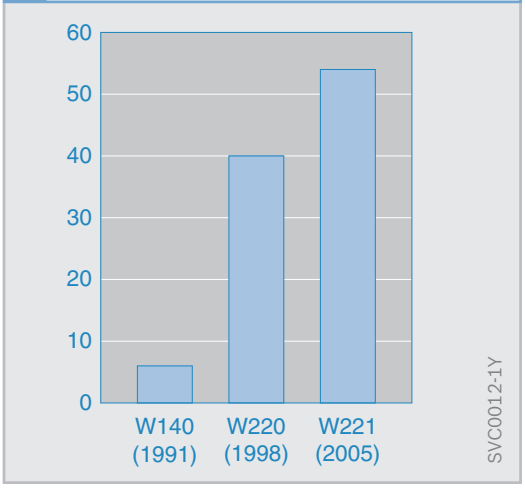
This variable specifies the volume of data that is transmitted during a time unit. The smallest unit of data is the bit, and the data transfer rate is usually specified in bits/seconds. Alternative names for this expression are transfer rate, data rate or bit rate.

The required data rate is dependent on the application. A slower transfer rate is required to switch the air-conditioning compressor on and off than to transfer audio signals, for example.

Interference immunity

Ideally, the data should be transferred without interference. However, this cannot be guaranteed in a motor vehicle because of electromagnetic effects. The interference immunity requirements that are made depend on the safety relevance of the electronic systems concerned. Lesser requirements are made of comfort and

2 Number of control units in the Mercedes S-class that are networked via CAN



convenience systems than the antilock brake system (ABS), for example.

In order to meet these requirements, mechanisms that detect transmission errors are incorporated in the network protocols. A simple check can be carried out using the parity bit, which is calculated in the transmitter and is transmitted together with the useful data. This specifies whether the number of 1's in the transferred byte is even or off. This information is checked by the receiver. Single errors can be detected using this method.

Another method is the checksum check. If several data bytes are being transmitted, the transmitter calculates a checksum from the individual data bytes using a predefined formula and transmits this value. The receiver also calculates the checksum of the data bytes that have been received and compares it with the checksum that has been received. If a data transmission error is detected, the received data is not used and a repeat transmission is requested.

Real-time capability

A real-time system guarantees that its results are calculated within a fixed time interval. The duration of the time interval depends on the application. The antilock brake system (ABS) must react to the incipient locking of a wheel within a few milliseconds (wheel speed reduction), whereas response times of 100 ms are adequate for actuating the power-window motor. Human beings cannot perceive delay periods of less than 100 ms.

Different demands are made of real-time behavior depending on the application:

- Soft real-time requirement: the system generally adheres to the specified response time, and if these times are occasionally exceeded, it does not produce any serious effects (e.g. image jerking during picture transmission).
- Hard real-time requirement: the time specification must be strictly adhered to. If the specified response time was exceeded, the calculated result would not be able to be used. This can lead to serious problems in safety-critical systems.

For example, if time allowances were exceeded in the ABS system, the incipient locking of the wheels would not be detected soon enough and the pressure in the master cylinder would not be reduced in time. This would result in locked wheels.

The time allowances must also be strictly adhered to for many engine-management system functions. Delays in transmitting injection and ignition signals could lead to engine judder and even misfiring. These reactions must be avoided, since they represent a potential danger. Hard real-time requirements must therefore be made of these systems.

However, this does not necessarily mean that the transmission of data via a bus system also has to be subject to these hard real-time requirements. Adherence to soft real-time requirements is usually sufficient. If signals from other control units are needed for functions (e.g. a torque reduction request during a shift operation), the bus system must transmit the data at a faster data transfer speed and with a smaller time delay so that the overall system complies with the specified real-time requirements.

Number of network nodes

The maximum number of nodes to be integrated varies for different areas of vehicle operation. The number of nodes for comfort and convenience systems may be high due to servomotor networking (e.g. seat adjustment) and intelligent sensors (e.g. rain sensors). Several identical busses can be used if necessary.

Classification of bus systems

Because of differing requirements, bus systems can be subdivided into the following classes.

Class A	
Transfer rates	Low data rates (up to 10 kBit/s.)
Applications	Actuator and sensor networking
Representative	LIN
Class B	
Transfer rates	Average data rates (up to 125 kBit/s.)
Applications	Complex mechanisms for error handling, control unit networking in the comfort functions
Representative	Low speed CAN
Class C	
Transfer rates	High data rates (up to 1 MBit/s.)
Applications	Real-time requirements, control unit networking in the drive and running gear functions
Representative	High speed CAN
Class C+	
Transfer rates	Extremely high data rates (up to 10 MBit/s.)
Applications	Real-time requirements, control unit networking in the drive and running gear functions
Representative	FlexRay
Class D	
Transfer rates	Extremely high data rates (> 10 MBit/s.)
Applications	Control unit networking in the telematics and multimedia functions
Representative	MOST

Applications in the vehicle

The overall vehicle system can be divided into four domains or functional areas from the point of view or electrics/electronics:

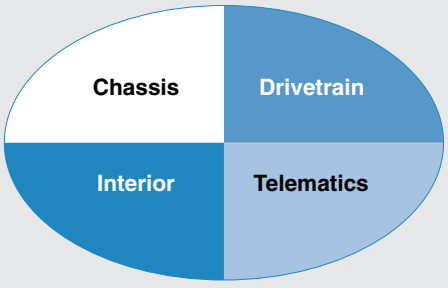
- Drivetrain
- Chassis
- Interior and
- Telematics

In the drivetrain and chassis domains, the emphasis is primarily on real-time applications. In the interior domain, the main focus is on multiplex aspects in networking. Mainly multimedia and infotainment applications are networked in the telematics domain.

Real-time applications

The networking of these systems makes considerable demands of the performance capability of the communication system. Crankshaft-synchronous processes or processes within a fixed time frame with cycle times of a few milliseconds are typical.
If the system response times are adequate for the task in hand, it is described as having real-time capability (e.g. rapid ignition-timing advance in the Motronic after a request from the traction-control system for reducing torque and therefore preventing the wheel from spinning).

3 Domains in the overall vehicle system



The drivetrain and chassis systems are assigned to class C. These require fast transfer rates in order to ensure the real-time behavior that is required for these applications. They also make considerable fault tolerance demands. These requirements are met by the event-driven CAN bus with a transfer rate of 500 kBaud (high-speed CAN).

Examples:
- Engine-management system (Motronic or electronic diesel control, EDC)
- Transmission control
- Antilock brake systems, ABS
- Vehicle dynamics control (e.g. electronic stability program, ESP)
- Chassis control systems (e.g. active body control, ABC)
- Support systems (e.g. adaptive cruise control, ACC)

Multiplex applications

The multiplex application is suitable for controlling and regulating components in the body and comfort and convenience electronics area (class B), such as
- Displays
- Lighting
- Access authorization with anti-theft warning device
- Air-conditioning
- Seat and mirror adjustment
- Door module (power-window unit, door-mirror adjustment)
- Windshield wipers
- Headlamp adjustment

The transfer rate requirements are not as high for class B systems as they are for class C systems. For this reason, low-speed CAN with a transfer rate of 125 kBit/s or single-wire CAN with 33 kBit/s. can be used.

If the transfer rate requirements drop to less than 20 kBit/s, the low-cost LIN is more frequently used. Applications are mainly in the mechatronics area; examples being the transfer of switch information or the activation of actuators.

Multimedia networking

Mobile communication applications combine components such as
- Car sound system
- CD changer
- Navigation system
- Driver-information systems
- Telephone
- Video system
- Voice input
- Internet, E-mail
- Back-up camera

The networking of these components makes it possible to have a centrally located display and control unit for several applications. Operating procedures can be standardized in this way, and status information can be summarized. Driver distraction is therefore minimized.

A distinction must be made between control data and audio/video data in multimedia networking. Transfer rates of up to 125 kBit/s are sufficient for control tasks (such as CD changer control), meaning that the low-speed CAN bus can be used, for example. The direct transmission of audio or video data requires extremely high transfer rates of more than 10 MBit/s. The MOST bus is used for this purpose, for example.

Coupling of networks

The network topologies and protocols that are most suitable for requirements are used for the different applications. However, the different network protocols are incompatible, meaning that data cannot be simply exchanged between networks.

In this case, help is provided by a gateway. A gateway can be compared to an interpreter that receives the "data" from a discussion partner, translates it and passes it to another discussion partner. Technically speaking a gateway is a computer that reads in the data that is transmitted by the networks and converts it into another format. The use of gateways therefore makes it possible to exchange information between different networks.

A central gateway (Fig. 4a) or several distributed gateways can be used (Fig. 4b) to interconnect the bus systems. All bus lines are routed to the central gateway. In the other case, one gateway connects two or more busses.

Examples of networked vehicles

Topology

The topologies of the communication networks can differ considerably depending on the vehicle equipment. Figure 5 shows examples of how the network can be structured for different vehicle classes. In some cases, different car manufacturers use different bus systems for communication.

Signal transmission

Signal types

A wide variety of information can be transmitted in a communication network in a vehicle. Some examples are:

- Engine operating conditions (e.g. engine temperature, engine speed, engine load)
- Physical measurements recorded by sensors (e.g. outside temperature)
- Control signals for activating servomotors (e.g. power-window units)
- Control-element switch positions (e.g. for the windshield wiper)
- Multimedia data (audio and video) for transmitting music and speech (e.g. from radio stations or when handsfree talking with a cellular phone) and moving pictures (e.g. when playing a DVD or the displays from a reversing camera)

4 Gateway structures

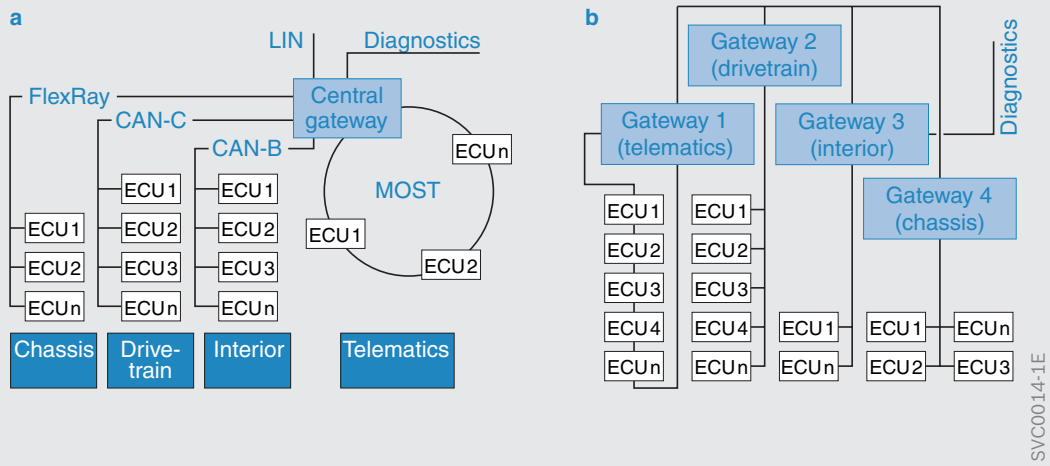


Fig. 4
a Network with central gateway
b Network with several distributed gateways

5 Network topologies

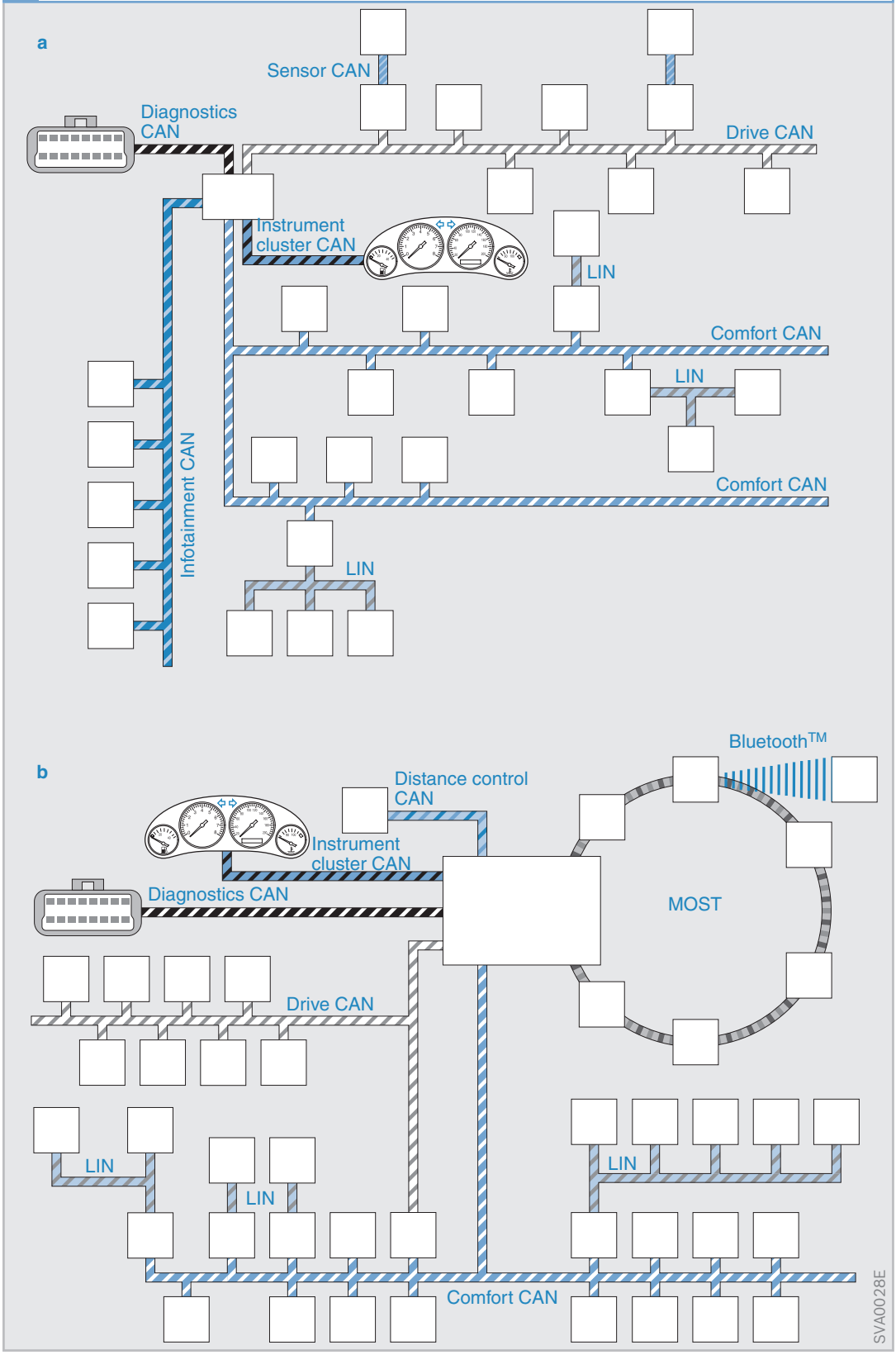


Fig. 5
Typical network topologies for new-generation vehicles
a Compact class
b Luxury class

Resolution

The signals must be available with a suitable resolution. Switch positions can simply be shown as a 1-bit value (0 for switch open, 1 for switch closed). Other signals such as digitized analog voltages from the engine-temperature sensor or the calculated engine speed are shown as 1-byte or 2-byte values, for example, depending on resolution requirements. One byte can be used to represent 256 values, and two bytes can be used to represent 65,536 values (= 256·256). The resolution for sensor signals with a voltage range of 0...5 V is approximately 20 mV with a 1 byte representation (= 5 V/256). A resolution of 5 mV requires a 10-bit data representation.

The conversion between the binary value and the physical value must be uniform so that the transferred signals represent the same physical value in all systems. A resolution of 30 rpm is sufficient for accessing the ignition map as far as the engine speed n is concerned. A value range of 0 to 255·30 rpm (= 7,650 rpm) and therefore the entire speed range can consequently be represented by one byte (8 bits). An incrementation of 30 rpm is too little for idle-speed control, on the other hand. More bits are required to represent the signal with a higher resolution, provided that the same measuring range is being recorded.

Output

In event-driven systems such as the ones that are primarily used in the automotive area, the signals can be transferred on the data bus when an event occurs. Examples of such events are the operation of switches for switching on the air-conditioning system or the windshield wiper. Signals that represent the operating state of the engine, for example, are not necessarily associated with an event. The engine temperature, which only changes slowly, is cyclically measured in a fixed time frame by the engine control unit (e.g. 1 second). The engine speed, on the other hand, can change extremely rapidly. The times at which the engine management carries out measurements and calculations depends on the crankshaft position and is performed once per combustion cycle. At fast engine speeds, this corresponds to a time interval of a few milliseconds, i.e. approx. 3.3 ms at an engine speed of 6,000 rpm for a 6-cylinder engine. However, not every system that needs the speed information for its control and regulation functions depends on this availability. The engine control unit therefore does not need to output the speed information on the data bus as soon as it is calculated. In this case too, data transmission takes place according to a cyclic time frame. A time frame of 10 ms is normal in the engine control area. This means that the speed information is transmitted on the bus 100 times per second.

Multimedia data

A multimedia-compatible digital bus system is frequently used in luxury-class vehicles to transfer audio data – as an alternative to using analog cables. In addition to improved audio quality, a bus system of this type offers the advantage that various audio streams and the associated check commands can be transferred in parallel. In the automotive industry, the optical MOST bus (Media Oriented Systems Transport) has proven successful as a the multimedia bus system.

The transfer of audio signals requires on the one hand synchronized transfer between the transmitter and one or more receivers and on the other hand a high data rate. In view of the fact that data transfer on the MOST bus takes place synchronously with a fixed clock-pulse rate, synchronization is already assured by its transfer mechanism.

The digital transfer of CD-quality audio signals, i.e. with a resolution of 16 bits and a clock-pulse rate of 44.1 kHz, requires a constant data rate of 1.35 Mbit/s for one stereo

channel. Up to 15 stereo channels can be transferred in parallel with the current version of the MOST bus (MOST25 with a total transfer rate of up to 24.8 Mbit/s).

Data transfer: examples

The following examples show which signals are measured and evaluated in which systems.

Driving speed

The ESP control unit calculates the driving speed from the wheel-speed sensors. This variable is transmitted on the CAN-C bus (drive CAN). The engine-management system needs this value for the cruise control, among other things, and the transmission control unit determines gear changes from the driving speed. The adaptive speed control (ACC, Adaptive Cruise Control) needs the current driving speed to calculate the necessary distance from the vehicle in front and use it as a setpoint value.

A gateway transmits the speed information via another CAN bus (instrument cluster CAN) to the instrument cluster, which displays the value via a needle instrument.

The CAN-B bus (comfort CAN) is also connected to the network via the gateway. Some luxury class vehicles are equipped with dynamic seats. The padding of the seats is inflated depending on the speed and the acceleration, counteracting the centrifugal force of the driver. This increases comfort considerably when cornering.

The speed information is sent to the Infotainment CAN via the gateway and relayed to the car sound system. This allows the volume to be adapted to the driving speed. The navigation system needs the speed to calculate the position if the GPS signal is missing (e.g. in a tunnel).

The diagnosis interface is directly connected to the engine and transmission control unit via the serial K-line. All other control units are connected to the diagnosis interface via a virtual K-line that is simulated on the CAN bus. This allows the driving speed to be read out in the workshop via the connected diagnostic tester (example: the correct assignment of the wheel-speed sensors must be checked for the ABS functional test).

Engine speed

The injection and (with a gasoline engine) the ignition timing are output with a resolution of less than 1° of the crankshaft angle. In order to ensure real-time behavior, the crankshaft position must be recorded in the engine control unit. The engine-speed sensor scans the crankshaft trigger wheel and relays the signal to the control unit, which calculates both the crankshaft position and the engine speed. This variable is used to calculate the injection time and the ignition angle, for example.

The engine speed is a variable that is needed in many other systems. The engine control unit therefore outputs it on the data bus. The shifting points are defined in the transmission control unit depending on the speed. The engine speed is needed for the ASR function (acceleration slip control) in the Electronic Stability Program (ESP) – ASR intervention (torque reduction) must not make the engine stall.

As in the previous example, the engine speed is transmitted to the diagnosis interface and the instrument cluster (display on the rev counter).

Turn signaling

The driver operates the turn-signal lever (Fig. 6, Item 1). A signal is relayed to the steering column control unit via a discrete line (2) depending on whether the driver is indicating a right or left turn. This may be a resistance-coded signal, for example. The control unit evaluates the signal and detects that the driver is indicating a left turn, for example.

The comfort CAN relays this information to the vehicle power supply control unit (3). The indication direction is defined on the basis of the received information (normal flash frequency, increased flash

frequency in the event of bulb failure). The front left and rear left turn-signal lights are then actuated via discrete lines (4, 5). The vehicle power supply control unit also transmits the "left turn signaling" information on the comfort CAN. The gateway (6) relays the information to the instrument cluster CAN. The indicator lamp then flashes on the instrument cluster (9).
If the vehicle has a trailer hitch, the information goes to the trailer-recognition control unit (7) via the comfort CAN. This actuates the turn-signal lamp on the trailer (8) via cables.

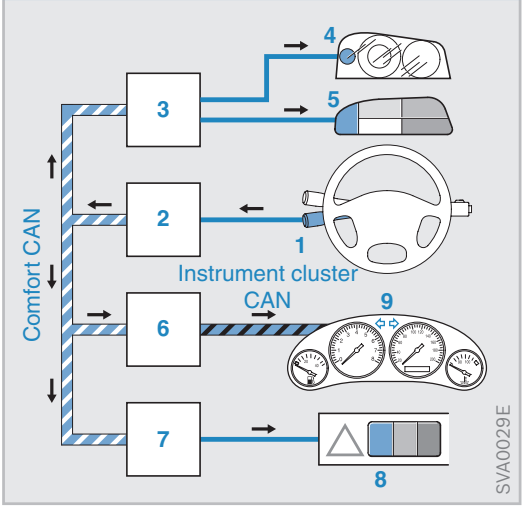
Wiper stage 1

The wiper switch (Fig. 7, Item 1) transmits a signal via a discrete line to the steering column control unit (2), which evaluates the information (e.g. wiper stage 1). The control unit transmits this information on the comfort CAN bus. The vehicle power supply control unit (3) picks up the information and relays it to the wiper motor (4) via the LIN bus. The vehicle power supply control unit acts as a gateway between the comfort CAN and the wiper LIN.

Load management

At low revs (idle speed) and when a considerable amount of power is being used up by the electrical consumers that are switched on, the battery or alternator voltage can drop to a low value. The vehicle power supply control unit calculates the current status of the vehicle power supply from the current battery voltage, the DF signal from the alternator (alternator utilization) and the information about heavy current consumers that are switched on with a short switch-on duration. The vehicle power supply control unit requests an idle speed increase via the CAN bus if the vehicle power supply is insufficient at idle speed. The engine control unit implements the request. If this action does not solve the problem, the vehicle power supply control unit switches off specific consumers such as the heated rear windshield, the seat heating or the heated outside mirror. These consumers are connected to the vehicle power supply control unit via discrete lines.

6 Data transfer during turn signaling



7 Data transfer during windshield-wiper operation

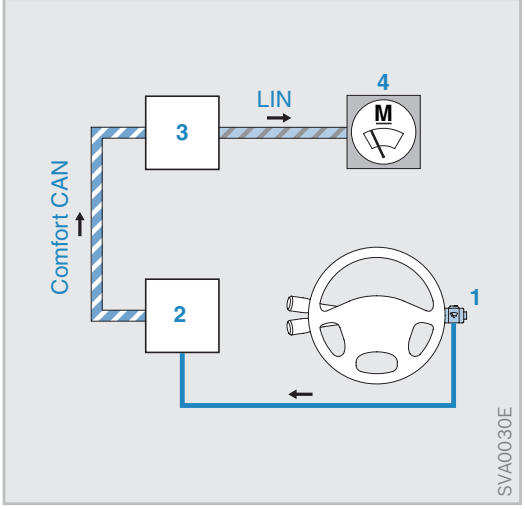


Fig. 6
1 Turn-signal lever
2 Steering column control unit
3 Electrical-system control unit
4, 5 Turn-signal lights
6 Gateway
7 Trailer-recognition control unit
8 Turn-signal light on trailer
9 Instrument cluster

Fig. 7
1 Windshield-wiper lever
2 Steering column control unit
3 Electrical-system control unit
4 Wiper motor

Multimedia application

The signal sequences during the output of a radio signal on the amplifier is described in the following as an example of the transfer of signals and multimedia data in a luxury-class vehicle infotainment system with MOST bus (Fig. 5b).

The functions of the vehicle infotainment system are controlled via the central operating unit, the head unit (Fig. 8, Item 1). In current systems, this unit usually has a screen in the instrument panel and a rotary/pushbutton controller in the center console, by means of which the user interacts with the system.

To output a radio station, the head unit, which in most systems is responsible for managing the audio channels of the infotainment system, establishes on the MOST bus a stereo channel of the required audio quality between the radio tuner (2) and the amplifier (3). It now tunes in the requested station at the radio tuner via corresponding check commands and if necessary makes further settings. Finally, the head unit connects the tuner output with the previously created stereo channel. For the information to be displayed in the user interface, the head unit receives from the radio tuner corresponding data relating to its current reception status (the current station name, for example) which are updated where necessary.

At the amplifier, the head unit connects the stereo channel with a selected input and uses corresponding check commands to set the properties for outputting the audio signal, e.g. the set volume. When all the settings have been made, it instructs the amplifier to fade in the audio signal of the radio signal, which is then output through the infotainment system's speakers.

Before this, the head unit has if necessary ensured that the output of a previously output audio signal has been faded out, the output of the associated device stopped, and the associated audio channel removed.

In parallel the CAN/MOST gateway (4) constantly transmits the driving-speed data, which it receives via the comfort CAN, via the MOST bus to the amplifier, which has requested a notification for this information when the system was started. The amplifier can use the speed together with further vehicle information to calculate additional settings for outputting the audio signal, e.g. adapting the volume depending on the current speed.

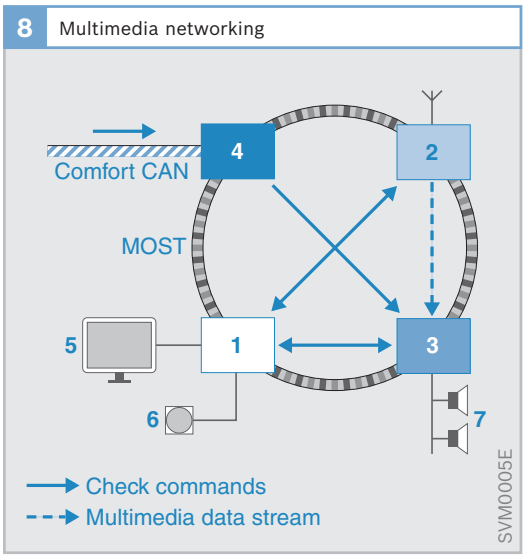


Fig. 8
1 Head unit
2 Radio tuner
3 Amplifier
4 CAN/MOST gateway
5 Screen
6 Control element
7 Speakers

Control of an automatic steel folding roof

Vehicles with automatic steel folding roofs are currently very much in vogue. These roofs are automatically opened and closed. The technical realization of this seemingly simple mechanical function poses a huge challenge for the networking of the electronics and control units.

The networking depicted in Figure 9 shows a typical present-day mid-size vehicle with the additional function of an automatic steel folding roof. At its maximum equipment specification, this vehicle has over 35 electronic control units (ECUs), which communicated with each other via four CAN buses and a further four LIN subbuses. In this example, a total of 13 control units are involved in the control of the automatic steel folding roof. The text below describes which functions are performed by the individual control units to move the roof.

Roof control unit

Power activation of the motors for the steel folding roof is performed by this control unit. This control unit also assumes the role of complete monitoring of the movement process. Proximity-type sensors which monitor, record and evaluate the

9 Control of an automatic steel folding roof

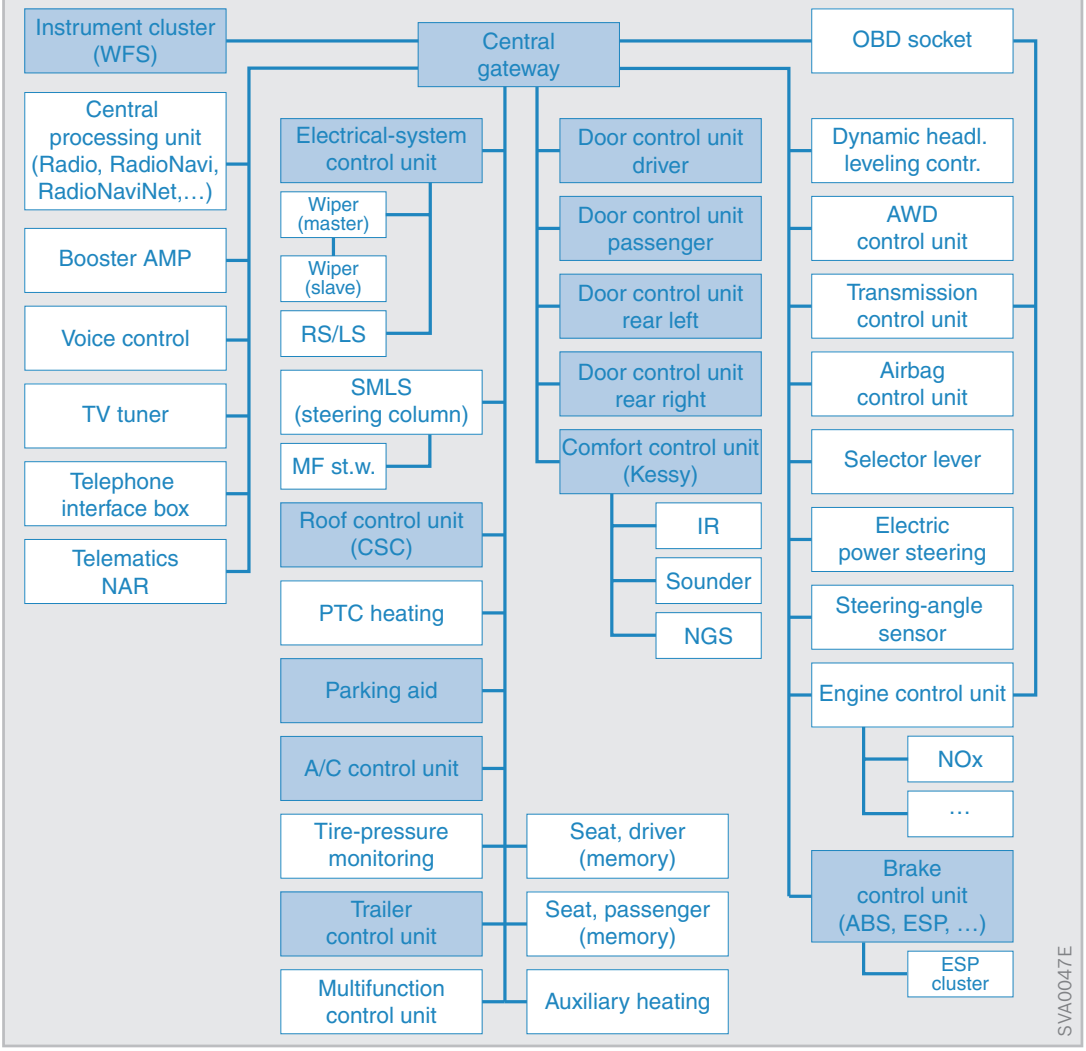


Fig. 9
Systems in blue boxes are involved in the control of the steel folding roof

movement of the roof are used for this purpose. In order to inform the driver of the current status while the roof is being opened and closed and to prevent any damage to the vehicle or the surroundings, the roof control unit receives numerous further parameters from other control units in the vehicle.

Instrument cluster
The instrument cluster receives via the gateway the current status from the roof control unit and from other participating control units, and informs the driver accordingly. Example: “Obstacle behind the vehicle! Roof movement stopped”.

Gateway
All the communication requests which are transmitted from one bus system to another are routed via the central gateway.

Vehicle power supply control unit
The vehicle power supply control unit checks whether the vehicle battery has sufficient charge to enable the roof to be moved. If necessary, the driver is informed and no roof movement is performed.

Parking-aid assistant
Before the roof is opened, the parking-aid assistant monitors the area behind the vehicle for obstacles to ascertain whether there is sufficient space available to unfold the roof. If an obstacle is detected, the driver is informed and movement is stopped. In this event, the driver can decide for him-/herself whether he/she wishes to continue moving the roof. In certain cases the parking-aid assistant can detect something which poses no danger to the vehicle.

A/C control unit
All windows are automatically closed when the air-recirculation switch is actuated. This also applies to the sliding sunroof, which is activated by the roof control unit.

Trailer control unit
The trailer control unit informs the roof control unit whether a trailer is hitched. If this is the case, opening and closing of the roof is disabled.

Door control units
The status of the windows and doors is interrogated by all four door control units and transmitted to the roof control unit. Opening or closing of the roof is only begun if all the doors are closed and the windows are in the correct positions. This setting is automatically corrected if the windows are not in the correct positions.

Comfort control unit
The comfort control unit informs the roof control unit of, among other things, the key position. Movement of the roof is enabled only if the correct key is inserted.

Brake control unit
The brake control unit uses sensors to record the wheel speeds and thereby identify whether the vehicle is moving. The roof control unit receives the speed information and enables movement of the roof only if the vehicle is stationary.

Future areas of FlexRay application

Up to now the high-speed CAN bus (CAN C) has been used to network control units in the drivetrain and in the chassis area. In future X-by-wire systems, the mechanical connections, e.g. between steering wheel and front axle (steer-by-wire) or brake pedal and wheel brakes (brake-by-wire), will be replaced by electrical communication systems in conjunction with driving-dynamics control systems.

Furthermore, vehicle architectures have up to new often used several CAN buses which are linked to each other via gateways to distribute the high data volume. Future architectures will use, for a fast, powerful connection between several master computers (which assume, for example, central functions in the safety and driver-assistance areas), a backbone bus with a high data rate to which in each case sub-buses (e.g. drivetrain CAN and sensor CAN in the chassis area) are connected.

The new FlexRay bus system satisfies the demands which will be placed in future on the vehicle architecture, such as, for example

- high data rates and guaranteed real-time capabilities in the drive and chassis areas,
- large date volume in the backbone, and
- high failure safety of safety-relevant applications (e.g. X-by-wire)

through properties such as

- high availability and redundancy by means of two physically independent channels,
- high data rates with up to 10 Mbit/s per channel,
- data transfer with guaranteed latency, and
- synchronicity of all communication users by means of a global time base.

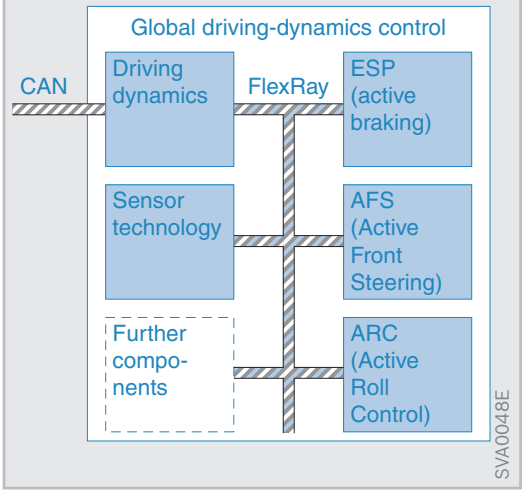
FlexRay is currently still in the development stage, although some initial systems are already in volume production. The following example is a current approach to a FlexRay topology.

Excerpt from chassis domain

Concepts for global driving-dynamics control are being developed to facilitate control which is as cooperative and thus as effective as possible. Components of these concepts are ESP (Electronic Stability Program), AFS (Active Front Steering), and ARC (Active Roll Control). Higher-level driving-dynamics control controls the individual systems without limiting their functionality and prevents negative interactions. Figure 10 shows the interconnection of these systems with bus system and sensors.

The yaw sensor transmits the vehicle's acceleration values and yaw rate to ESP. If a critical driving situation is detected by ESP using these data (e.g. vehicle oversteer), situation-conditioned brake pressures and engine-management interventions are calculated. Higher-level global driving-dynamics control also evaluates the driver-command steering angle from the steering-angle sensor and calculates a supplementary steering angle, which is converted by AFS. More effective and comfort-enhancing control can be achieved by this interaction of the individual systems. In this way, the vehicle can, for example, be stabilized by a corrective steering movement already at a very early stage such that braking interventions can be partially or even completely avoided.

10 FlexRay



Bus systems

CAN bus

In 1991 the CAN bus (Controller Area Network) was the first bus system to be introduced to a motor vehicle in mass production. It has since established itself as the standard system in the automotive sector, but the CAN bus is also commonly used as a field bus in automation engineering in general. In imitation of other network types, such as the local area network (LAN), wide area network (WAN) or personal area network (PAN), this bus system was given the name, CAN.

Applications

The CAN bus is used in various domains in the motor vehicle. These domains differ in the requirements they demand of the network. Due to the fast processes involved in the area of engine management, information is required much faster here than in the area of comfort/convenience where the controlled systems are located further apart and as such lines are more prone to damage. As a result of these different requirements, buses with different data rates are used that offer an optimum cost-benefit ratio for the field of application concerned. A distinction is made between high-speed and low-speed CAN buses.

1 Networking of electronic control units by CAN

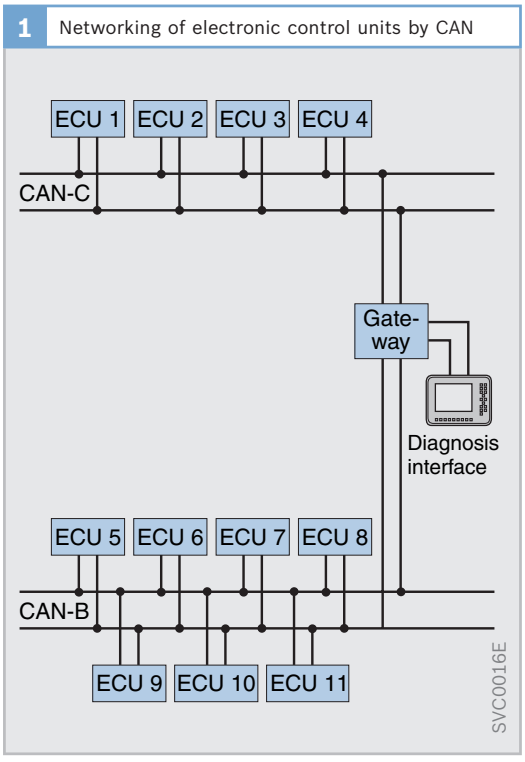


Fig. 1
ECU Control unit
ECU 1 to ECU 4
On the CAN-C high-speed bus
ECU 5 to ECU 11
On the CAN-B low-speed bus

High-speed CAN (CAN-C)

CAN-C is defined in ISO Standard 11898-2 and operates at bit rates of 125 kBit/s to 1 MBit/s. The data transfer is therefore able to meet the real-time requirements of the drivetrain.

CAN-C buses are used for networking the following systems:

- Engine-management system (Motronic for gasoline engines or EDC for diesel engines)
- Electronic transmission control
- Vehicle stabilization systems (e.g. ESP)
- Instrument cluster

Low-speed CAN (CAN-B)

CAN-B is defined in ISO Standard 11898-3 and operates at a bit rate of 5 to 125 kBit/s. For many applications in the comfort/convenience and body area, this speed is sufficient to meet the real-time requirements demanded in this area. Examples of such applications are:

- Control of the air-conditioning system
- Seat adjustment
- Power-window unit
- Sliding-sunroof control
- Mirror adjuster
- Lighting system
- Control of the navigation system

The CAN bus is finding ever more use in vehicle diagnostics. Here, the electronic control unit is connected directly to the CAN bus and thus receives the information it needs for diagnostics immediately. Previous diagnosis interfaces (e.g. KWP2000) are becoming less important.

Topology

Bus topology

In the development of the CAN, particular focus was placed on eliminating the need for a central control element for communication. This approach is most effectively supported by a bus topology in which all network nodes are connected to a bus and each node is able to receive all information sent on the bus. The bus topology is most commonly selected during the conceptual design of the communications system.

In addition to offering favorable electrical properties, the linear bus topology has the advantage that the failure of one station would not affect the functionality of the data transmission system. Furthermore, additional stations can be connected to the system with little extra effort.

Star topology

The use of a central coupler makes it possible to build star topologies. Active as well as passive couplers can be used. The use of a star topology achieves a high level of flexibility in adapting to the networking task.

The coupler is used to build a star topology and simply forwards the messages to the individual segments. Since the signal transit times remain unchanged, the circuit length of the star topology is the same as that of a bus topology.

Data transmission system

Network nodes

A network node (Fig. 2) comprises the microcontroller for the application software, the CAN controller and the CAN transceiver (bus driver). The CAN controller is responsible for the transmit and receive modes. It generates the bit stream for data communication from the binary data to be transmitted and forwards it to the transceiver on the TxD line. This amplifies the signals, generates the voltage level required for differential data transfer and transmits the processed bit stream serially on the bus line (CAN_H and CAN_L).

Incoming messages are processed by the transceiver and sent to the CAN controller on the RxD line.

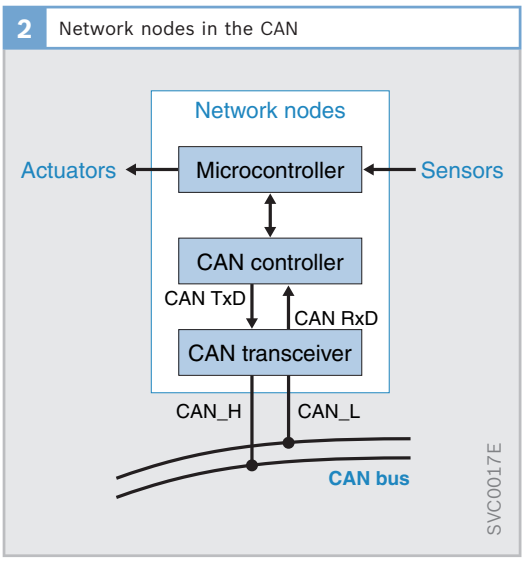
The microcontroller, which runs the application program (e.g. Motronic), controls the CAN controller, prepares the data to be sent and evaluates the data received.

Logic bus states and coding

CAN uses two states for communication, dominant and recessive, with which the information bits are transmitted. The dominant state represents a binary “0”, the recessive a binary “1”. NRZ (Non-Return to Zero) is used as the encoding method for the data transmission. With this method, there is no compulsory return to zero between two transmission states of the same value.

When it receives messages, the CAN transceiver converts the signal level back to logical states. In the process, a differential amplifier subtracts the CAN_L level from the CAN_H level (Fig. 3). If lines become twisted, disturbance pulses (e.g. from the ignition system) have the same effect on both lines. Differential data transfer therefore makes it possible to filter out interference on the line.

2 Network nodes in the CAN



Some transceivers also evaluate the voltage level on the CAN_H and CAN_L line separately. It would then be possible for operation to continue in single-line mode if one of the two bus lines were to fail as a consequence of a short-circuit or cable break. However, the bus subscribers would have to share a common ground that would assume the function of the failed line.

Transmission agent and bus coupling

Two-wire line

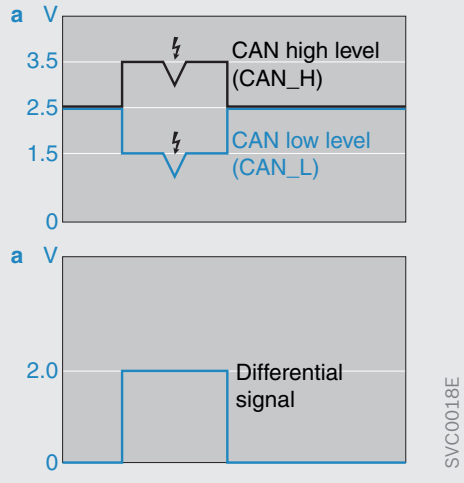
For the CAN, any transmission agent on which dominant and recessive states can be transmitted is worthy of consideration. An untwisted or twisted pair of wires, depending on the ambient conditions, is usually used, and the wires are either galvanically coupled or decoupled. The two bus lines are designated CAN_H and CAN_L. The two-wire line supports symmetrical data transfer whereby the bits are sent on both bus lines and represented by different voltages. This reduces sensitivity to in-phase interference because the interference affects both lines equally and can be filtered out. Additional shielding of the lines reduces their own radiation emissions, especially at high baud rates.

Single-wire line

The single-wire line is a means of reducing manufacturing costs by dispensing with the second line. For this to be possible, however, all bus subscribers must share a common ground that would assume the function of the second line. The single-wire version of the CAN bus is therefore only an option for a communications system of limited spatial dimensions.

The data transfer on the single-wire line is more prone to interference radiation because it is not possible to filter out disturbance pulses as it is with the two-wire line. For this reason, a higher level increase is required on the bus line to improve the signal-to-noise ratio. This in turn has a negative effect on interference radiation. The flank steepness of the bus signals must therefore be reduced in comparison to the two-wire line. This is accompanied by a lower data transfer rate. Consequently, the single-wire line is only used for the low-speed CAN in the body electronics and comfort/convenience electronics. Thanks to this feature, a low-speed CAN with two-wire line would still remain functional in the event of a line failure (CAN_H or CAN_L).

3 Filtering out interference on the CAN bus



Voltage level

The CAN transceiver converts the logical states 0 and 1 received by the CAN controller into voltage levels that are fed to the CAN_H and CAN_L bus lines.

The high-speed and low-speed CANs use different voltage levels for the transmission of dominant and recessive states. The voltage levels of the low-speed CAN are shown in Figure 4a, those of the high-speed CAN in Figure 4b.

In the recessive state, the high-speed CAN uses a voltage of 2.5 V on both lines. In the dominant state, a voltage of 3.5 V is present on CAN_H and a voltage of 1.5 V is present on CAN_L.

On the low-speed CAN, a voltage of 0 V is present on CAN_H in the recessive state, and 5 V on CAN_L. In the dominant state, voltages of 3.6 V and 1.4 V are present on the CAN_H and CAN_L respectively.

Reflection-free termination

Reflections of the electrical signals at open ends of lines would interfere with communication. To dampen these reflections, the bus lines are terminated at each end with a resistor of 120 Ω.

Alternatively, the terminating resistors may be integrated into the electronic control units themselves.

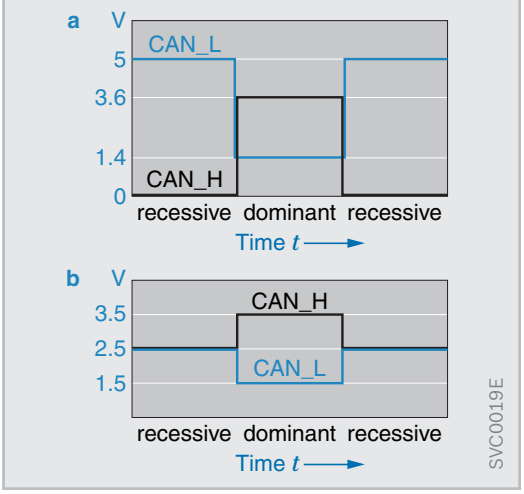
Emission limits

To permit correct evaluation of bit transmission, the signal must arrive at each node within the bit time without interference, and yet still at the relevant sampling point. Delays arise from the signal transit time on the data bus. The maximum permissible bit rate thus depends on the total length of the bus. ISO 11898 specifies the bit rate for a defined circuit length. The following recommendations exist for longer lines.

- 1 MBits/s for 40 m (specified)
- 500 kBits/s up to 100 m (recommendation)
- 250 kBits/s up to 250 m
- 125 kBits/s up to 500 m
- 40 kBits/s up to 1,000 m

It is possible to connect at least 30 network nodes to the bus without the need for additional measures.

4 Voltage level



5 Data transmission system

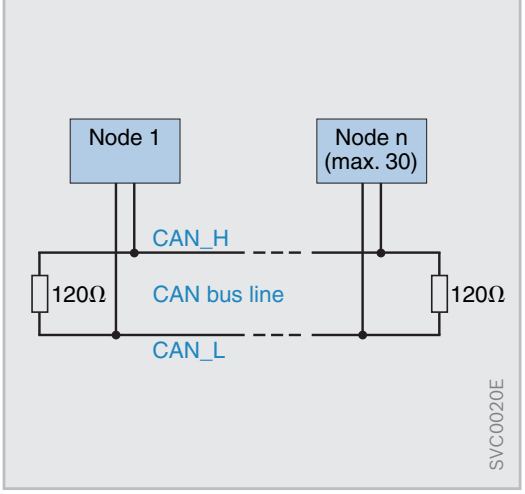


Fig. 4
a Voltage level of the low-speed CAN (CAN-B)
b Voltage level of the high-speed CAN (CAN-C)

CAN protocol

Protocol layers

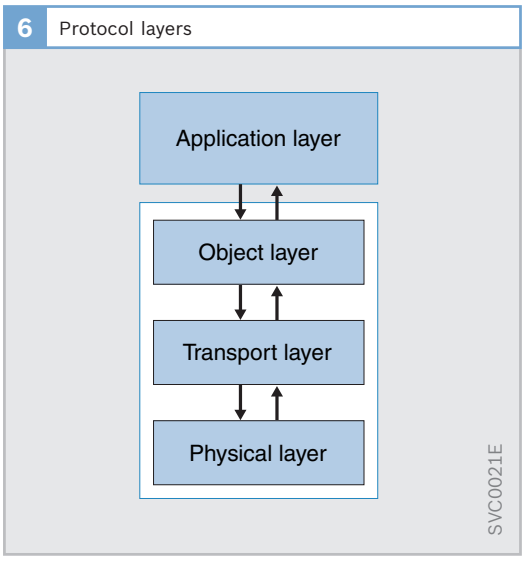
For communications protocols, it is standard practice to arrange thematically coherent tasks as a set of layers, which affords a high level of flexibility in the implementation of a bus system. With the CAN, both the CAN hardware and the CAN software are subdivided into several layers (Fig. 6).

Application layer
The application layer represents the information in the form of data structures used by the application. These sets of data which are to be transmitted are forwarded to the object layer for this purpose.

Object layer
The task of the object layer is to manage the messages. The functions of this layer are used to decide which message should be sent at which time. For incoming messages, this layer is responsible for message filtering.

Transport layer
The transport layer furnishes the object layer with received messages and processes the messages prepared by the object layer for sending in such a way that the physical layer is able to transmit this information. To fulfill this purpose, the transport layer is responsible for such functions as arbitration or fault detection and signaling.

6 Protocol layers



Physical layer
The physical layer is the lowest level in the transport stack. It consists of the physical components of the network, such as the wiring and the voltages used to send the information.

Multimaster principle

The CAN protocol supports communication between network nodes without the need for a central control unit. Each node may attempt to send messages at any time. Whether this attempt is successful or not essentially depends on two factors:

- Is the bus free before the start of transmission?
- Has the arbitration phase been passed successfully?

This design ensures that, even if any number of nodes were to fail, it can still be determined whether a node is authorized to send.

Content-based addressing

Unlike other networks, the CAN does not address the individual network nodes but the messages that have been sent. Each message has a unique marker, or identifier. The identifier classifies the content of the message (e.g. engine speed or power-window position). A station is therefore able to broadcast a message to all other stations (multicast or broadcast method). These stations read only those messages whose identifiers are stored in their acceptance list (message filtering, Fig. 7). In this way, each station decides for itself whether or not it needs a message sent on the bus. The identifier has 11 bits (standard format, CAN 2.0 A) or 29 bits (extended format, CAN 2.0 B). With 11 bits in the standard format, it is possible to distinguish between 2,048 different CAN messages;

in the extended format, this number rises to over 536 million.

The advantage of this addressing method is that the network nodes do not require any information about system configuration and are thus free to operate fully independently of each other. This results in a highly flexible complete system, which makes it easier to manage equipment variants. If one of the ECUs requires new information which is already on the bus, all it needs to do is call it up from the bus. It is possible to integrate additional stations into the system (provided they are receivers) without having to modify the existing stations.

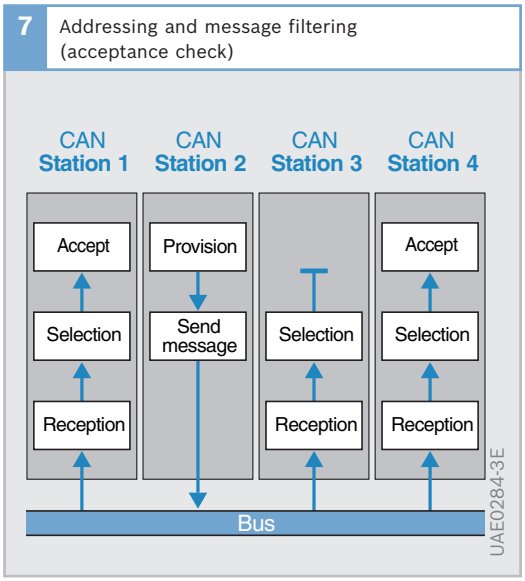
Controlling bus access

Arbitration phase

If the bus is unoccupied (recessive state) and messages are available for sending, each station is free to initiate the sending of its message. The message begins with a dominant bit (start-of-frame bit), followed by the identifier. When several stations start to transmit simultaneously, the system responds by employing "wired-and" arbitration (arbiter = logical AND operator) to resolve the resulting conflicts over bus access. The message with the highest priority (lowest binary value of the identifier) is assigned first access, without any data loss or delay (non-destructive protocol).

The arbitration principle permits the dominant bits transmitted by a given station to overwrite the recessive bits of the other stations (Fig. 8). Each station outputs the identifier of its message onto the bus bit by bit, with the most significant bit first. During this arbitration phase, each station wishing to send data compares the level present on the bus with the level it actually possesses. Each station that attempts to send a recessive bit but encounters a dominant bit loses the arbitration process. The station with the lowest identifier, i.e. the highest priority, makes its way onto the bus without having to repeat the message (non-destructive access control). The transmitters of lower-priority messages automatically become recipients of the message just sent by another station. They repeat their attempt to send as soon as the bus is free again.

Without this access control, bus collisions would result in faults. To guarantee unequivocal bus arbitration, therefore, it is not permissible for more than one node to send a message with the same identifier.



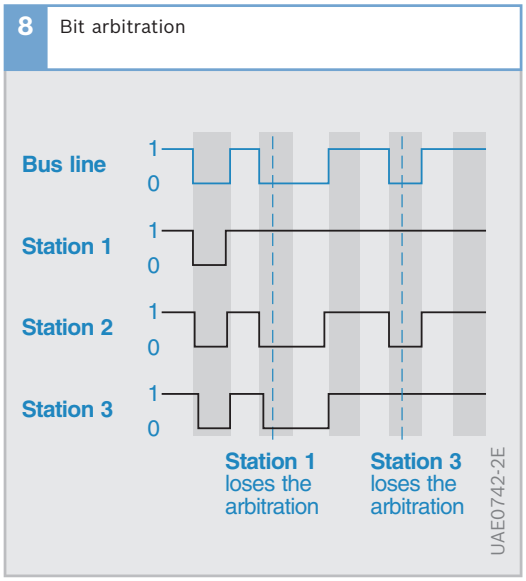


Fig. 7
Station 2 transmits
Stations 1 and 4 accept the data

Fig. 8
Station 2 gains first access (signal on the bus = signal from station 2)

0 Dominant level
1 Recessive level

With this access method, maximum-priority messages only have to wait for the transfer of the message currently being sent and, with bit times of 130 (CAN 2.0 A) or 150 (CAN 2.0 B), they have the lowest latency. At a data transfer rate of 500 kBits/s, this equates to 260 µs or 300 µs. The higher the load on the bus, the greater the data transfer's temporal offset becomes for messages of lower priority, and thus the greater the uncertainty over when a message to be sent will arrive at the recipient.

In order that all messages have a chance of accessing the bus, the data transfer rate must be matched to the number of bus subscribers.

Priority assignments

The direct consequence of the arbitration process is that the identifier also has the role of prioritizing the frame during transmission in addition to identifying the frame content. An identifier corresponding to a low binary number has high priority and vice versa. Message priorities are derived from the speed at which the message content changes or from the importance of the message to safety considerations, for example. It must not be possible for messages to have the same priority.

Message format

The message transfer on the CAN bus is based on four different frame formats:

- *Data frame*:
  The transmitted message contains data (e.g. current engine speed) that is provided by the transmitting station (data source).
- *Remote frame*:
  Stations can request the data they need from the data source (example: the windshield wiper requests how wet the windshield is from the rain sensor). The data source responds by sending the relevant data frame.
- *Error frame:*
  If a station detects a fault or error, it communicates this to the other stations using an error frame.
- *Overload frame*:
  This can be used to create a delay between a preceding and subsequent data frame or remote frame. The transmitting node reports that it cannot currently process another frame.

For data transfer on the data bus, a message frame is created. It contains the information to be transmitted arranged in a defined sequence. CAN supports two different formats of frame, which are specified in CAN 2.0 A and CAN 2.0 B. The most important difference between these two frame formats is the length of the identifier. A CAN 2.0 A frame has an 11-bit identifier, while a CAN 2.0 B frame has a 29-bit identifier divided into two parts (11 bit and 18 bit).

Both formats are compatible with each other and can be used together in a network. The frames compliant with CAN 2.0 A and B are shown in Figure 9. They have a maximum length of 130 bits (standard format) or 150 bits (extended format).

9 CAN message format

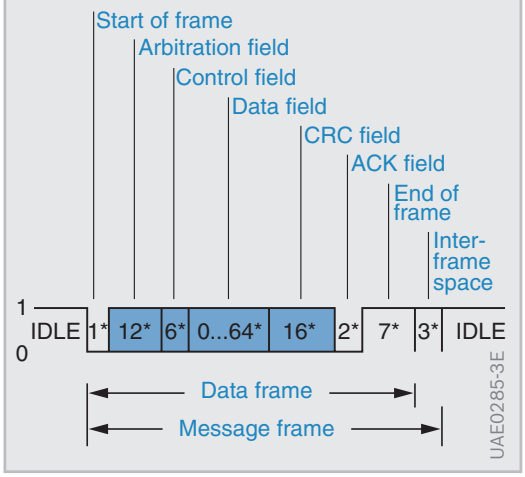


Fig. 9
0 Dominant level
1 Recessive level

In CAN 2.0 A and CAN 2.0 B data frames, the sequence of transferred information is arranged in the same way. The start-of-frame bit is followed by the arbitration, control, data and CRC fields. A frame is completed by the ACK field and the end-of-frame.

Start of frame
The bus is recessive in idle state. The start of frame, represented by a dominant bit, indicates the start of a transmission and, together with the process of bit stuffing, serves to synchronize all stations.

Arbitration field
With a CAN 2.0 A frame, the arbitration field comprises the 11-bit identifier and a control bit, the RTR bit (Remote Transmission Request).

With a CAN 2.0 B frame, the arbitration field comprises the 11-bit identifier, followed by the SRR bit (Substitute Remote Request) and the IDE (Identifier Extension Bit). Both bits are sent recessively and thereby ensure that a CAN 2.0 A frame always takes priority over a CAN 2.0 B frame if it has the same 11-bit identifier. The second 18-bit identifier follows. The RTR bit completes the arbitration field.

The RTR bit indicates whether the transmitted frame is a data or remote frame. The RTR bit is dominant in the data frame and recessive in the remote frame. If station A, for example, happens to send a message by data frame and station B requests this message by remote frame at the same time, the arbitration conflict cannot be resolved in this situation by means of the message identifier: the RTR bit is the decisive factor for access authorization. First of all, station A wins arbitration with the sending of the dominant RTR bit and continues to transmit the message. Station B, which has requested precisely this message, prepares to receive and is able to read the other data sent by station A.

Control field
In a CAN 2.0 A frame, the control field comprises the IDE bit (Identifier Extension Bit), which is always sent as dominant here, followed by a reserved bit for future extensions, which is sent recessively. The remaining four bits in this field define the number of data bytes in the next data field. This enables the receiver to determine whether all data has been received.

The structure of a CAN 2.0 B frame is practically identical. However, since the IDE bit already belongs to the arbitration field, a further reserved bit for future extensions takes its place and this reserved bit is sent recessively.

Data field
The data field contains the actual message information comprised of between 0 and 8 bytes. A data field in which the length of the data is expressed in 0 results in the shortest possible data frame with a length of 44 or 64 bits. A frame like this can be used to synchronize distributed processes. A number of signals can be transmitted in a single message (e.g. engine temperature and engine speed).

CRC field
The CRC field (Cyclic Redundancy Checksum) contains a 15-bit checksum (frame check word) across the preceding frame from the start bit to the final bit of the data field. The 16th bit (CRC delimiter) in this field is recessive and closes the checksum. The checksum is a means of detecting possible transmission interference.

ACK field
Unlike all preceding fields, the ACK field (acknowledgment) is not set by the sender of the frame but by a different node that is able to acknowledge receipt of the frame directly after the data field. This field comprises the ACK slot and the recessive ACK delimiter. The ACK slot is also transmitted recessively by the sender and over-

written as dominant by a receiver upon the message being correctly received. Here, it is irrelevant whether the message is of any significance for the particular receiver in the sense of an acceptance check (message filtering). Only correct receipt is confirmed. This signals to the sender that no malfunction occurred during data transfer.

End of frame
The end of frame marks the end of the message and comprises seven recessive bits. With seven bits of the same value in succession, the stuffing rule is deliberately broken (see Error detection).

Interframe space
The interframe space comprises a succession of three recessive bits which serve to separate successive messages. After a total of 10 of these recessive bits, the stations are permitted to begin transmitting within the network again. Until then, the bus remains in idle state.

Only data and remote frames need to include the interframe space. Error and overload frames can be sent immediately after the last frame. This enables immediate signaling of errors and problems.

Transmitter initiative

The transmitter will usually initiate a data transmission by sending a data frame. It is also possible for a receiving station to call in data from a sending station by depositing a remote frame. The network node that is able to deliver the information requested makes the information available in response to this query.

Error detection

The bus line may suffer interference, e.g. electromagnetic interference. To eliminate the risk of faulty behavior, the transmitted data must be checked for correctness. A number of control mechanisms for detecting errors are integrated in the CAN protocol.

Cyclic redundancy check
For every transmitted message, the transmitter calculates a check sequence from the start-of-frame, arbitration, control and data field. A 15-bit checksum is derived from the bit sequence by means of a generator polynomial. The checksum is then used to detect errors in the data transfer.

With CRC generation, a defined generator polynomial is used to carry out polynomial division across a given frame range. The remainder forms the checksum.

Once the CRC field has been received, the receiver is able to check that the frame has been transferred correctly by carrying out its own polynomial division of the received frame range using the generator polynomial and checking whether the received checksum matches the calculated remainder.

Frame check
The frame check involves all bus subscribers, senders as well as receivers, checking the sent/received data frame for compliance with the predefined frame structure. The CAN protocol contains a number of fixed-format fields (start and end-of-frame, delimiter) which are checked by all stations.

ACK check
With the ACK check, a receiver acknowledges that the frame has been received correctly by sending a dominant bit in the ACK slot. The sender of the frame can then verify whether a message was transferred correctly. An absence of this bit indicates that a transmission error has been detected.

Monitoring
The sender of a frame continuously monitors the bus level. It is able to detect a transmission error by comparing the sent bit and the sampled bit.

Bit stuffing

Compliance with bit stuffing is checked by means of the code check. The stuffing convention stipulates that in every data frame or remote frame a maximum of five successive bits of equal state may be sent between the start of frame and the end of the CRC field. As soon as five identical bit states have been transmitted in succession, the sender includes a bit with the opposite state. The receiving station clears all of these inserted bits after the message has been received (destuffing).

This measure permits detection of line faults, e.g. short-circuit or burst interference. Rare signal changes would affect the possibility for synchronization in the nodes.

Error handling

If a CAN controller detects a fault or format error, it interrupts the current transmission by sending an error frame comprising six successive dominant bits. This breaks the stuffing rule that prohibits this type of bit sequence. If the sender detects that its message has been interrupted by an error frame, it stops transmitting and makes another attempt at a later time. This effect prevents other stations from accepting the erroneous message and thereby ensures consistency of data across the entire system.

Fault confinement in the event of failures

Defective stations could weigh heavily on bus traffic if they were to frequently send erroneous messages or interrupt the transmission of correct messages by repeatedly sending an error frame. The CAN protocol localizes station failures by means of statististical error analysis. A station recognizes the probability of its own malfunction by how often it aborts messages before other stations send an error frame. The protocol's first measure in this case is to prevent a station such as this from continuing to abort transmissions. In an emergency, the station shuts down automatically.

Hardware

CAN controller of a subscriber

The CAN controller of a subscriber generates a frame from the data to be sent containing all the fields required for compliance with the CAN protocol. It then converts the frame into a bit stream.

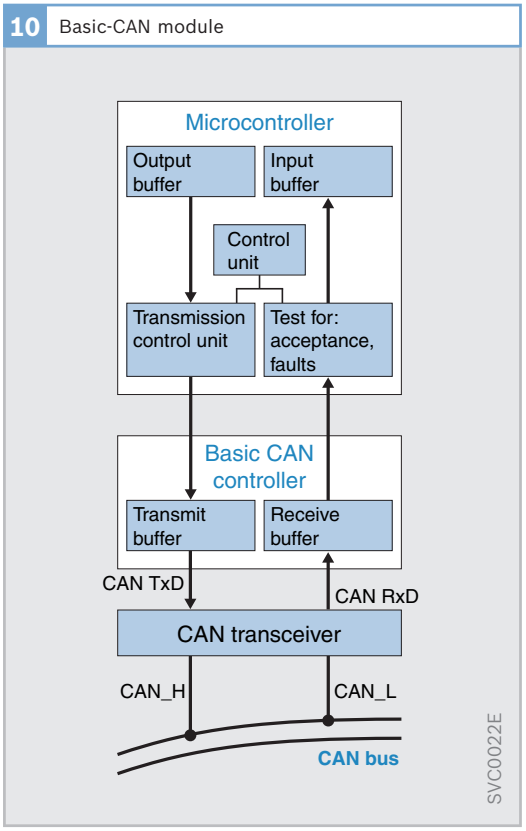
CAN only specifies the physical, transportation and object layers. The interfaces that are provided in the application layer can be arranged differently to suit the field of application. In order to provide proper CPU support (electronic control unit microcontroller for application software) for a wide range of different requirements, semiconductor manufacturers have brought onto the market products that provide a variety of capabilities. They differ neither in the frame format they produce, nor in their error-handling methods, but solely in the type of CPU support required for message administration.

If messages on the bus are managed in extended format (29-bit identifier), the CAN controllers must be compatible with the CAN 2.0 B specification. There are controllers that only support CAN 2.0 A and send and receive messages in the standard format, but that generate errors with messages in the extended format. Other controllers are able to tolerate the extended format without generating an error. These modules can be used in a CAN together with controllers that administer the sending and receiving of messages in the extended format.

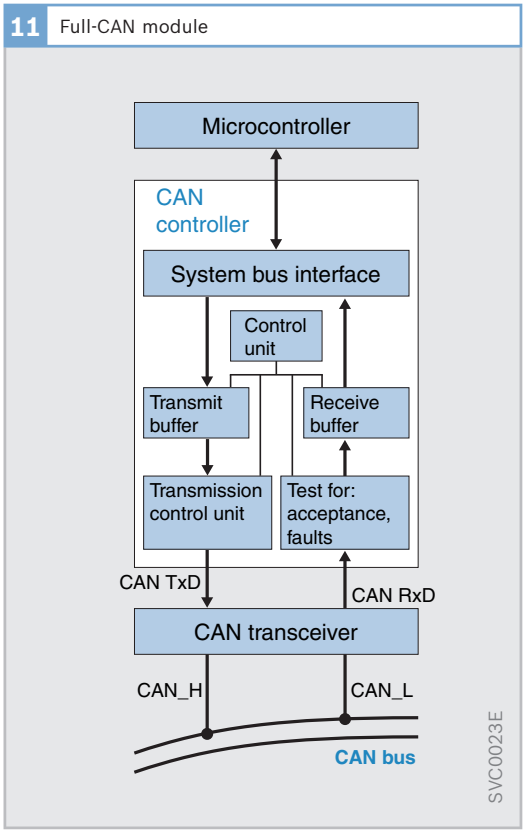
Basic CAN

In modules with basic-CAN implementation, only the basic function of the CAN protocol for the generation of the bit stream is implemented in the hardware. For the management of the messages to be sent and received, there is an intermediate

10 Basic-CAN module



11 Full-CAN module



buffer to which the local computer (application software microcontroller) has access (Fig. 10). Since the buffer capacity is limited, the computer must read the received data before new messages are received. Message filtering also takes place in this computer. A part of the computer's capacity is therefore used for CAN management. Since the computers do not usually have sufficient processing capacity, modules with basic CAN are primarily suitable for low bit rates, or for the transmission of fewer messages but at higher bit rates.

The advantage of these modules, in comparison to modules with full CAN, is the smaller chip surface and the lower manufacturing costs.

Full CAN

Full CAN implementation is the protocol of preference in cases where a station has to manage several messages at high bit rates and the local computer has no free capacity for communication tasks. They contain several “communication objects”, each of which contains the identifier and the data of a particular message. During the initialization of the CAN module by the local computer, it is decided which messages the CAN controller should send and which received messages it should process further. Received messages are only accepted (message filtering) if the identifier matches one of the communication objects.

CAN controllers with full-CAN implementation relieve the burden on the local computer by performing all of the communication including message filtering in the controller (Fig. 11).

The CAN controller can be coupled to the microcontroller in the electronic control unit as a stand-alone module by the address/data bus. Powerful microcontrollers have the CAN controller integrated on-chip. This type of bus coupling is the more cost-effective and thus the more common solution.

Modules without local computer

A further category of CAN module is one that is supplied without a local computer. These SLIOs (Serial Linked Input/Output) are able to input and output data via ports. They are therefore suitable for making sensors and actuators bus-compatible at low cost, but they do need a master that controls them.

Transceiver

The bit stream generated by the CAN controller is made up of binary signals. They do not yet correspond to the required voltage levels of the CAN bus. The CAN-bus interface module, or transceiver, generates the differential signals CAN_H and CAN_L and the reference voltage U_{ref} from the binary data stream.

Sleep mode

The CAN comfort bus must remain ready for operation even with the ignition switched off so that functions such as the radio, power windows or parking lamp may continue to operate. The bus subscribers must therefore be supplied by terminal 30 (permanent positive). After terminal 15 has been switched off (ignition off), a CAN node may enter sleep mode (standby) to relieve the vehicle electrical system of as much load as possible. The transmitter part of the transceiver module is switched off in this condition to minimize the power consumption in this mode of operation. However, the receiver part remains active and checks whether messages are being sent on the bus. In this way, the CAN controller, which also enters standby mode, is able to react to a wake-up message and fully activate the CAN node.

Data transfer sequence

Using the transfer of engine speed as an example, it shall be demonstrated how data is transferred on a CAN with full-CAN module.

Transmission

The engine management's application software calculates the engine speed from the signal from the engine-speed sensor. This value is calculated once for every combustion cycle. The measured value enters both the receive buffer and the transmit buffer of the microcontroller (Fig. 12a).

The engine-management microcontroller is coupled to the CAN controller by a parallel interface. The contents of the transmit buffer are cyclically passed (e.g. every 10 ms) to the transmit buffer of the CAN controller. A flag notifies the CAN controller that a message is ready to be sent. With this transmit instruction, the engine management has completed its part of the task.

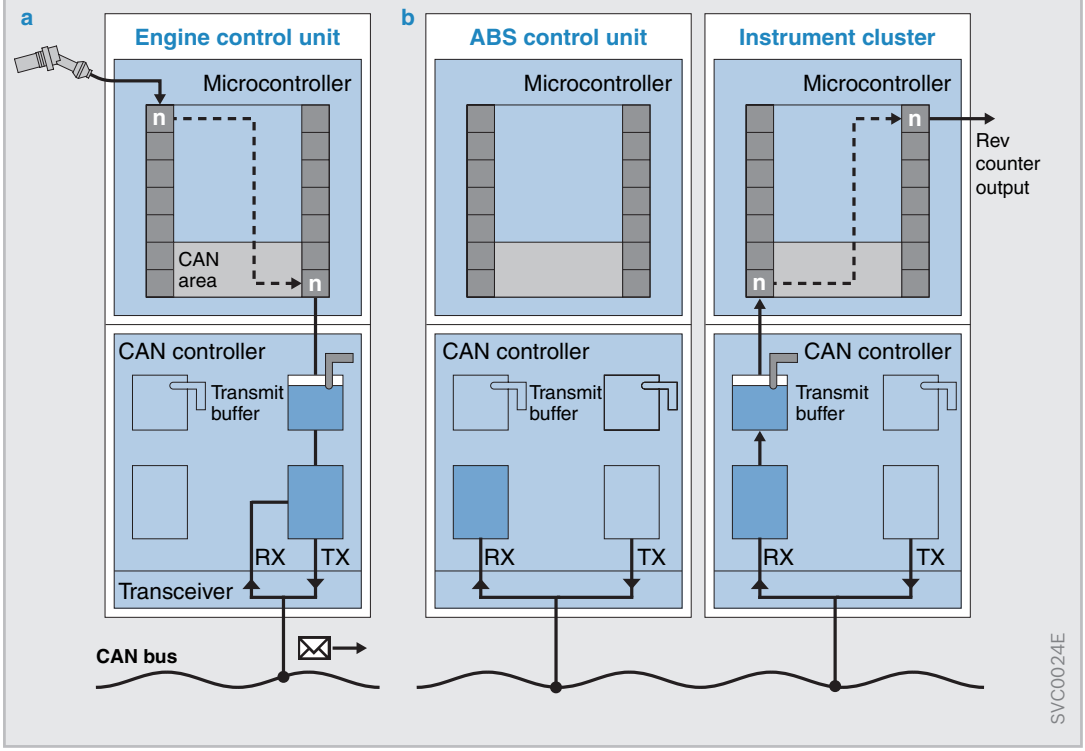
Once the engine-speed information has been stored in the transmit buffer, the CAN controller begins to compile the CAN frame. Over the RxD line, the CAN controller now monitors the bus. If the bus is free, it immediately starts to generate the bit stream and directs it to the transceiver on the TxD line; if the bus is occupied, it waits. From the bit stream, the transceiver creates a signal with the requisite voltage level.

Reception

The message originally sent by the engine management and forwarded on by the transceiver is received by all stations connected to the bus (Fig. 12b). The message reaches the CAN controller on the RxD lines in the form of a bit stream.

At the first stage (monitoring layer), the CAN controller checks the incoming message for errors using the CRC checksum. If the message is free of errors, each

12 Data transfer sequence



station responds with an acknowledge (ACK check).

At the second stage, or acceptance layer, the message undergoes message filtering. Each station checks whether the received identifier is addressed to the particular station and whether the message is required in the application software. If not, the message is rejected. Otherwise, it makes its way to the receive buffer. A flag notifies the application software that a new message is ready for processing.

The instrument cluster, for example, calls up the available message, processes the engine-speed information and calculates triggering signals for the actuator of the rev counter.

Standardization

The International Organization for Standardization (ISO) and SAE (Society of Automotive Engineers) have issued CAN standards for data exchange in automotive applications:

- For low-speed applications up to 125 kBit/s: ISO 11 519-2 and 11 898-3
- For high-speed applications faster than 125 kBit/s: ISO 11 898-2 and SAEJ 22 584 (passenger cars) or SAEJ 1 939 (commercial vehicles)
- An ISO Standard for diagnosis via CAN has also been published as ISO 15 765

Standardization makes it possible for components of different manufacturers to function together. No adaptations are required.

CAN is also widely used in industrial automation. These applications are supported by an alliance of companies in the “CAN in Automation” users group (CiA).

Bosch has concluded contracts with its licensees that guarantee that any CAN implementations will be able to communicate with each other. Users will be able to rely on the interaction of any CAN modules.

Characteristics

- Standardized in accordance with ISO 11898
- Prioritized communication
- Data transfer rates: up to 1 MBits/s
- Data capacity: up to 8 bytes per message
- Real-time response: the data protocol is sufficient for the real-time requirements in the motor vehicle
- Non-destructive bus-access method
- Low power consumption
- Flexibility of configuration
- Simple and economical design with twisted line pairs
- Very high reliability of data transfer
- Fault detection and signaling
- Localizing of failed stations
- Handling of intermittent and permanent faults
- Short-circuit resistance
- The number of nodes is theoretically unlimited. However, a limit arises in practice from the capacitive load of the bus and the increasing latencies of messages when a high number need to be sent.

LIN bus

Overview

The increasing use of mechatronic systems in the motor vehicle gave rise to the idea of designing a cost-effective bus system as an alternative to the low-speed CAN. In 1998, several automotive manufacturers founded a consortium with the aim of developing a specification for a serial bus for the networking of sensors and actuators in the body electronics area.

It was believed that a bus system with simple bus protocol and a simple sequence control would make it possible to use even low-capability microcontrollers without additional hardware for the communication interface.

The workgroup's resulting LIN bus specification was introduced into mass production with the Mercedes-Benz SL as early as 2001.

The name, LIN (Local Interconnect Network), is derived from the fact that all electronic control units are located within a demarcated installation space (e.g. in the door). The LIN, therefore, is a local subsystem for supporting the vehicle network by means of superordinate CAN networks.

The LIN bus is suitable for low data rates of up to 20 kBit/s and is typically limited to a maximum of 16 bus subscribers.

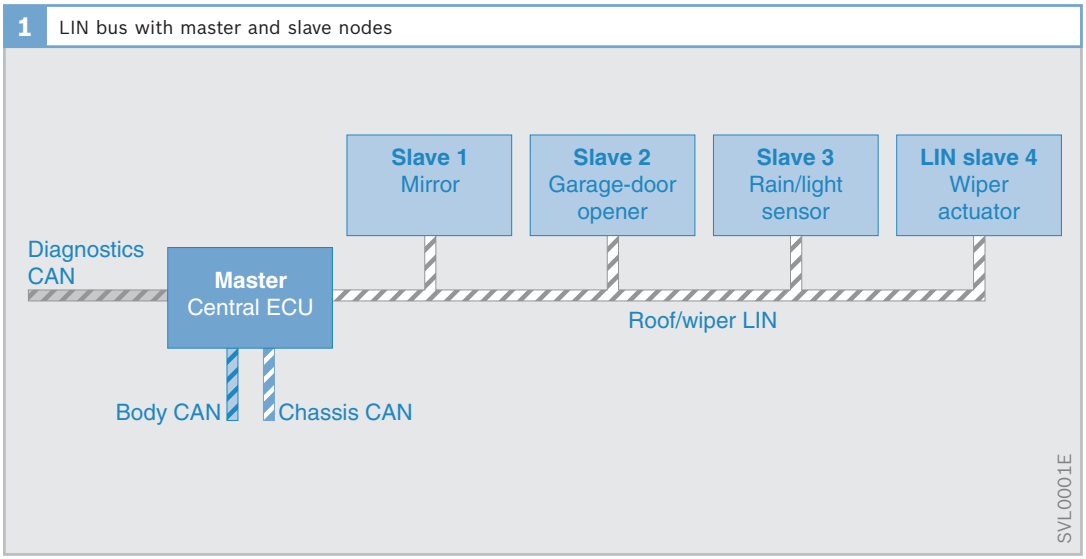
The electrical interface can be created easily and cost-effectively in the network nodes. As far as the nodes are concerned, a distinction is made between the master, which is generally an electronic control unit connected to a superordinate bus system, and the slaves. These are intelligent actuators, intelligent sensors or, quite simply, just switches with additional hardware for the LIN-bus interface.

The bus subscribers are usually arranged in a linear bus topology and connected to each other by a single-wire line. This topology, however, is not explicitly specified.

Communication on the LIN bus takes place in a time-synchronous manner, whereby the master defines the time grid. Consequently, there arises a strictly deterministic LIN bus response.

Figure 1 shows an example of a LIN network as a subbus in the roof/wiper area of the motor vehicle. Here, the bus comprises a central electronic control unit, as the master, and the four slaves: mirror, garage-door opener, rain/light sensor and wiper actuator. The master also functions as a gateway to the Chassis CAN, the Body CAN and the Diagnostics CAN.

1 LIN bus with master and slave nodes

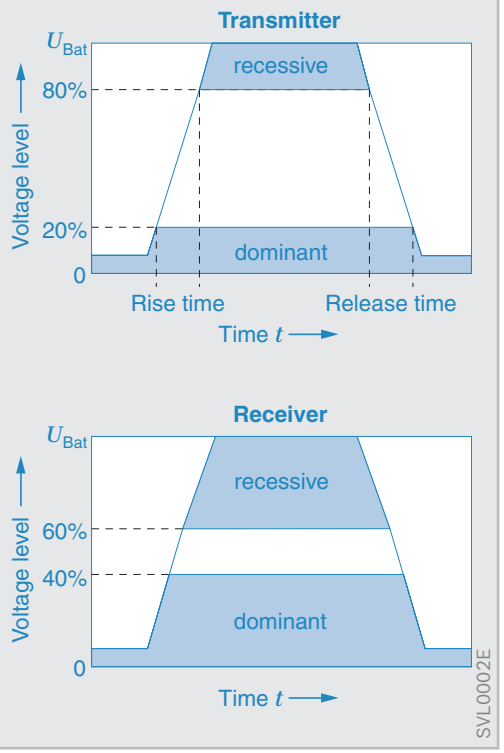


Applications

The LIN bus as a means of networking mechatronic systems can be used for many applications in the motor vehicle for which the bit rates and variability of the CAN bus are not essential. Examples of LIN applications:

- Door module with door lock, power-window drive and door-mirror adjustment
- Control of the power-sunroof drive unit
- Control of the wiper motor for the windshield wiper
- Sensor for rain and light detection
- Air-conditioning system (transmission of signals from the control element, activation of the fresh-air blower)
- Headlight electronics
- Control of motors for seat adjustment
- Theft deterrence
- Garage-door opener

2 Tolerance ranges for transmitters and receivers



Data transmission system

The LIN bus is designed as an unshielded single-wire line. The bus may adopt two logical states:

- The dominant level corresponds to the electrical voltage of approx. 0 V and represents logical 0
- The recessive level corresponds to battery voltage U_{Bat} and represents the logical 1 state

The recessive level is characterized by a resistance to battery voltage of 1kΩ in the master and 30kΩ in the slaves.

Due to circuitry variations, there may be differences in the voltage levels. A stable data transfer is ensured by the defining of tolerances for sending and receiving in the areas of the recessive and dominant levels. The tolerance zones on the reception side are wider (Fig. 2) to make it possible for valid signals to be received despite interference radiation.

The data rate of the LIN bus is limited to a maximum of 20 kBit/s. This is the outcome of a compromise between the demand for high edge steepness for easy synchronization of the slaves on the one hand, and the demand for lower edge steepness for improved EMC characteristics on the other. The standard bit rates of 2,400 Bit/s, 9,600 Bit/s and 19,200 Bit/s are recommended. The minimum permissible value for the bit rate is 1 kBit/s to prevent time-out conflicts. The edge steepness itself is defined in the LIN specification as 1 to 3 V/µs.

The maximum number of nodes is not specified in the LIN specification. It is theoretically limited by the number of available identifiers. In practice, the number of subscribers is restricted to 16 due to the maximum permissible total capacity of the bus system.

Bus access

Access to the LIN bus is determined by the master-slave principle. Each message is initiated by the master. The slave has the possibility to respond. The messages are exchanged between the master and one, several or all of the slaves (point-to-point, multicast, broadcast).

The following relationships are possible in the communication between master and slave:

- Message with slave response: the master sends a message to one or more slaves and asks for data (e.g. switch states or measured values)
- Message with master instruction: the master gives a slave a control instruction (e.g. switch on a servomotor)
- The master initiates communication between two slaves

No arbitration or conflict management is required because, with the master-slave access control, the master alone controls access to the data line.

LIN protocol

Frame

The information transferred on the LIN bus is embedded in a defined frame (Fig. 3). A message initiated by the master always begins with a header. The response (message field) contains different information depending on the type of message. If the master wishes to transmit control instructions for a slave, it will populate the response with data to be used by the slave. If the master is transmitting a data request, the slave that it is addressing will populate the response with the data requested by the master.

Header

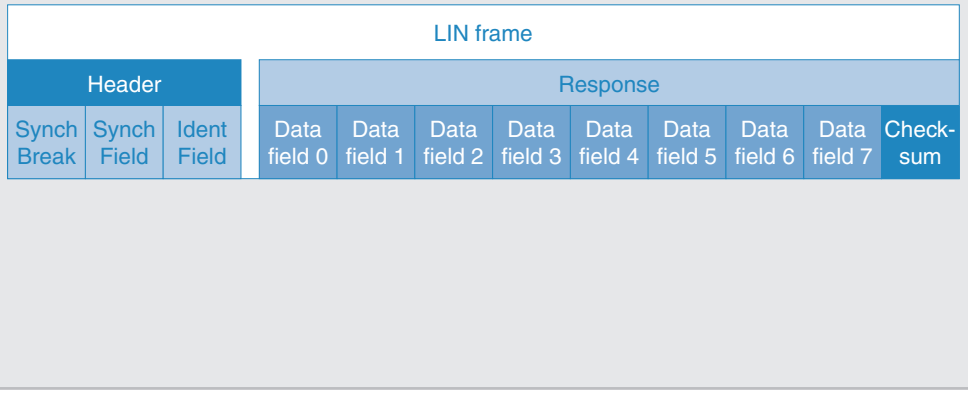
The head is composed of the following parts:

- The synchronization break
- The synchronization field
- The identifier field

Synchronization

To guarantee consistent data transfer between master and slaves, a synchronization takes place at the beginning of each frame. First of all, the beginning of a frame is unambiguously marked by the synchronization break (SynchBreak).

3 LIN message format



The SynchBreak comprises at least 13 consecutive dominant levels and one recessive level.

At the end of the synchronization break, the master sends the synchronization field (SynchField) consisting of the bit sequence 01010101. The slaves are then able to adjust themselves to the time basis of the master and thus synchronize.

The synchronization method described permits a loose specification of the timing of the bus subscribers. The clocking of the master should not deviate more than ±0.5 % from the nominal value. The clocking of the slaves is permitted to deviate by up to 15 % before synchronization as long as the synchronization reduces the deviation to a maximum of 2 % before the end of the message.

In this way, the slaves can be built without an expensive quartz oscillator, e.g. using a simple RC circuit.

Identifier

The third byte in the header is used as the LIN identifier. As with the CAN bus, a content-based addressing method is used – the identifier therefore provides information about the content of a message (e.g. engine speed). Based on this information, all nodes connected to the bus decide whether they would like to receive and process the message further or simply ignore it. This process is known as acceptance filtering.

Six of the eight bits of the identifier field determine the identifier itself. Their permutations give rise to a possible 64 different identifiers (ID). They have the following meanings:

- ID = 0 to 59: transmission of signals.
  ID = 60: master request for the commands and diagnosis
- ID = 61: slave response to ID 60
- ID = 62: reserved for manufacturer-specific communication
- ID = 63: reserved for future extensions to the protocol

Of the 64 possible messages, 32 may only contain two data bytes, 16 may only contain four data bytes, and the remaining 16 eight data bytes each.

The last bits in the identifier field contain two checksums, which are used to check the identifier for transmission errors and any resulting incorrect message assignments.

Data field

Once the header sent by the master node has been transmitted, it is time for the transfer of the actual data to begin. The slaves know from the transmitted identifier whether or not they are being addressed and, if they are, they reply with their response in the data field.

Several signals can be packed into one frame. In this case, each signal has precisely one generator, i.e. it is always written by the same node of the network.

During the data transfer of the bytes, it is always the least significant bit (LSB) that is output first. Each byte (8 bits) is preceded by a start bit and followed by a stop bit, which means that each byte involves the transmission of ten bits. The purpose of the start and stop bits is to resynchronize the nodes and thereby prevent transmission errors.

The data response of the slaves is verified by means of a checksum.

LIN description file

The configuration of the LIN bus, in other words the specification of network subscribers, signals and frames is managed in the LIN description file, or ldf. For this purpose, the LIN specification provides for an appropriate configuration language.

From the ldf, a set of C codes or header files is automatically generated using suitable tools. These are used as the basis for implementing the master and slave functions in the electronic control units connected to the bus.

The ldf is therefore a means of configuring the entire LIN network. It represents a common interface between the vehicle manufacturer and the supplier of the master or slave modules.

Message scheduling

The scheduling table in the ldf defines the sequence and time grid in which messages are sent. Frequently needed information is sent more often. Once the table has been worked through, the master begins with the first message again. The sequence in which the table is worked through may change depending on the operating status (e.g. diagnostics active/inactive, ignition on/off).

The transfer grid of each message is therefore known. This deterministic behavior is guaranteed by the fact that, with the master-slave access control principle, all messages are initiated by the master.

Network management

The nodes of an LIN network can be forced into sleep mode to minimize the no-load current of the entire electronics and electrical system in the vehicle. Sleep mode can be achieved in two ways:

- The master sends the go-to-sleep command with the reserved identifier 60
- The slaves enter sleep mode of their own accord if no data transfer has taken place on the bus for a relatively long period (4 seconds)

Both the master and the slaves are able to wake up the network. To do so, they must send the wake-up signal. This comprises data byte 128. After a break of 4 to 64 bit times (wake-up delimiter), all nodes must have been initialized and be ready to respond to the master.

Example: air-conditioning control

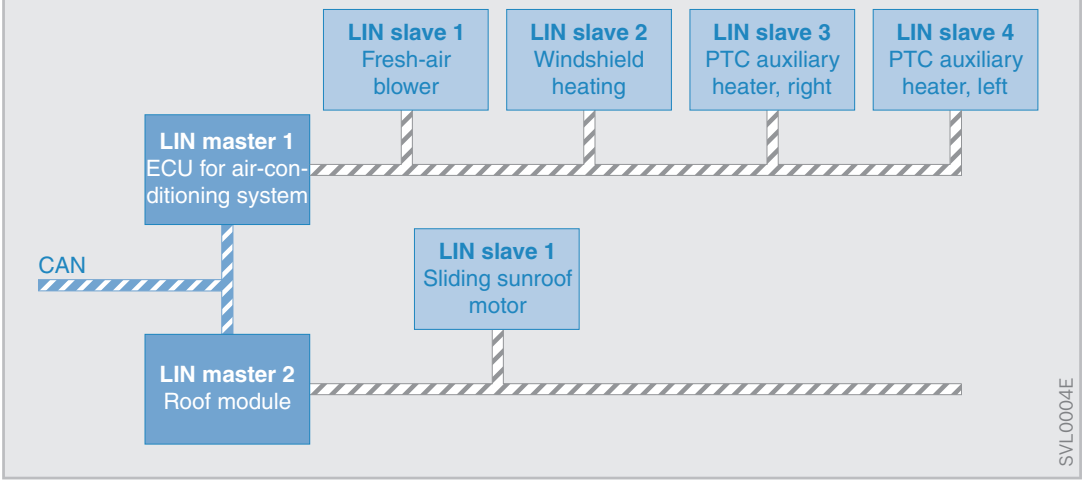
The LIN bus is typically used in the control of the air-conditioning system. The operating and display unit functions as the bus master. This is where the software for the open and control algorithms are stored.

One of its tasks is to adjust the speed of the fresh-air blower. The controlled variable for this is the actual temperature in the passenger cell and the desired temperature set by the driver. The electronic control unit receives the interior temperature from a temperature sensor located in an appropriate place in the interior. The desired temperature is set at the control unit, e.g. using a sensor ring.

From the input variables, the electronic control unit calculates that the blower speed needs to be increased to a value of 200 rpm, for example. The electronic control unit transmits a message containing the master instruction to the LIN bus at a defined time interval. The identifier in this example would be “set the blower speed”. In this subbus, this instruction corresponds to identifier 25, for example. Following the header containing this identifier, the master transmits a numerical value in the data field that equates to the physical value of 200 rpm. Each slave possesses a list of identifiers that are relevant for this node. The fresh-air blower is the one and only slave that responds to the “set blower speed” identifier and executes the master's request.

At speeds below idle, the fresh-air blower must be switched off. At this low speed, the heavy load could cause the engine to stall. The LIN bus is coupled to the CAN bus via the gateway and thus receives the current engine speed on a continuous basis. If the speed falls below the specified engine-speed threshold, the LIN master sends the message containing the “set blower speed” identifier where the data field contains the value 0. The fresh-air blower switches off in response.

4 LIN networking (example)



Intelligent actuators can send up-to-date operating-status information to the actuating unit. The fresh-air blower records the speed by means of a sensor and sends it back on the LIN bus as a numerical value. Due to the master-slave access method, the value is only contained in a message with slave response initiated by the master. The feedback signal with the current engine speed makes it possible to control, and therefore accurately maintain, the desired value.

Summary

The essential characteristics of the LIN bus are:

- Single-master/multiple-slave concept
- Master/slave access control
- Independent synchronization of the slave possible even without quartz
- Deterministic signal transmission
- Communication in the form of very short messages
- Character-based transmission (UART)
- Bit rate max. 20 kBit/s
- Data transfer over an unshielded single-wire line
- The reference potentials of the data line are battery voltage and ground
- Maximum bus length 40 m
- Maximum number of nodes 16, typically fewer than 12

Bluetooth

Overview

Mobile communication is gaining ever more importance in all domains. For convenient communication between various devices, a wireless link is indispensable. Infrared connections, as often used in the past, tended to be manufacturer-specific, they required a direct line of sight and imposed constraints on the area of movement. Only with a standardized, wireless link could mobile devices of different manufacturers communicate with each other without problems.

The development of the Bluetooth standard began in 1994 at the telecommunications company, Ericsson. As part of an initial study, Ericsson investigated the possibilities for completely replacing the cable connections between the cell phone and additional devices. In cooperation with other industry partners, the Bluetooth SIG (Special Interest Group) was formed in 1998 to create a uniform, internationally-accepted standard. The SIG set itself the aim of specifying a wireless technology with low manufacturing costs, low energy consumption and robust resistance to interference. The wireless interface had to be suitable for transmitting data for multimedia applications.

In the meantime, the Bluetooth SIG has expanded to involve around 2,000 companies from the telecommunications, data processing and automotive engineering fields.

Applications

Bluetooth is an industry standard for the networking of mobile multimedia devices, such as a car sound system, cell phone, headset, PDA (Personal Digital Assistant), PC and peripheral equipment (Fig. 1). Bluetooth stands for the simple exchange of data between portable terminals and the wireless transmission of audio and video signals for entertainment and information.

The short-range wireless connectivity of Bluetooth eliminates the need for connection cables between the cell phone and the hands-free system: no more tangling of cables when making a call. It is in this area that Bluetooth has been the most well received.

Meanwhile, a large number of consumer electronics devices, be they cell phones, PDAs, notebooks or car phones, now offer an address book function.

1 Bluetooth networking of mobile multimedia devices (example)

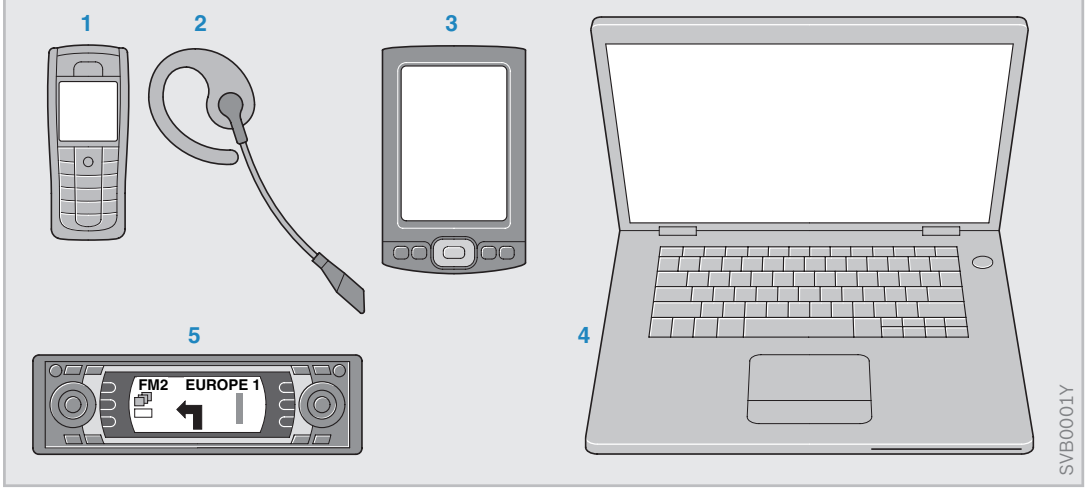


Fig. 1
1 Cellular phone
2 Headset
3 Personal digital assistant (PDA)
4 Laptop computer
5 Car sound system with navigation unit

With Bluetooth, it is possible to conveniently synchronize the entries stored in the various address books. The car driver is spared the laborious task of typing the phone number stored in the cell phone or PDA or inputting an address into the navigation system.

Bluetooth also makes it possible to listen to the radio through a headset in the rear of the vehicle and to control the car sound system using a PDA.

Bluetooth is predominantly used for the transfer of data in the multimedia domain. However, this technology can quite easily be used for diagnostic and service purposes. Workshops or breakdown services would be able to conveniently read vehicle information during fault diagnosis and record the condition of the vehicle.

Bluetooth versions

The first specification (Bluetooth 1.0) of the Bluetooth standard was adopted in July 1999. In December 1999 this specification was replaced by Bluetooth 1.0b, which contained improvements and clarifications for interoperability. February 2001 saw the release of the Bluetooth 1.1 specification, which fixed fundamental interoperability problems between the first devices and chips and is used in most Bluetooth devices today. Three years later, in November 2003, version 1.2 of the specification was published. This used the IEEE (Institute of Electrical and Electronics Engineers) language definition for technical documentation and introduced adaptive frequency hopping (AFH) as a technical advancement that improves resistance to interference in the frequency band. The most significant modification was introduced with version 2.0 of the specification in November 2004. With the enhanced data rate (EDR), it was possible to triple the bit rate from 1 MBit/s to 3 MBit/s and reduce power consumption. With each new version of the specification, great emphasis was placed on backward compatibility with the previous versions.

Transmission technology

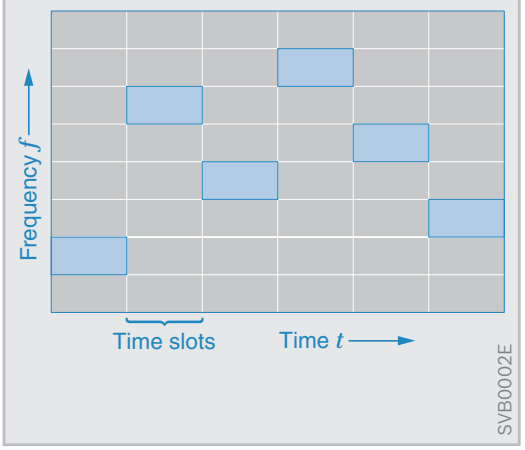
Frequency band

Bluetooth operates in the worldwide-license-free 2.4 GHz ISM band (Industrial Scientific Medicine). This is available practically worldwide without approval – apart from certain national limitations – and is reserved for applications in industry, science and medicine. For this reason, the possibility of interference from garage-door openers, microwave ovens or other appliances which operate using the same frequency band has to be accepted. To minimize interference, the best possible use should be made of the frequency spectrum. Devices that use the ISM band therefore implement spread-spectrum techniques.

Frequency-hopping method

Bluetooth divides the 2.4 GHz band into 79 channels at intervals of 1 MHz ($f = 2{,}402 + k$, $k = 0...78$). The band is terminated at both the upper and lower end by two guard bands with a width of 3.5 MHz (upper guard band) and 2 MHz (lower guard band). The transmission

2 Frequency-hopping method



uses a combined frequency-hopping/time division duplex (FH/TDD) method: the frequency-hopping spread spectrum (FHSS, Fig. 2). The channels are switched 1,600 times per second (i.e. in 625 µs slots). The slots are assigned in accordance with the TDD method, i.e. the transmitter and receiver are authorized to send alternately. The frequency-hopping method guarantees optimal and uniform use of the ISM band and makes Bluetooth resistant to interference from other transmitters in the same frequency band (e.g. WLAN). In addition, it also offers some security because a hacker would not know the destination of a frequency switch or indeed the time frame for which a particular frequency would be valid. This makes interception of a connection much more difficult.

The introduction of adaptive frequency hopping (AFH) in the Bluetooth 1.2 specification further increased Bluetooth's resistance to interference and reduced the effect of Bluetooth itself as a source of interference on other devices in the 2.4 GHz ISM band. The number of effective channels is dynamically adaptable to the conditions in the frequency band. Frequencies used by other transmitters are then removed from the list of 79 possible channels.

Modulation method

Bluetooth uses the Gaussian frequency-shift keying (GFSK) method of modulation. This is a special variant of frequency-shift keying (FSK) supplemented with a Gaussian low-pass filter. GFSK varies the frequency and delivers 1 bit per keying interval. Bluetooth has a symbol rate of 1 MBit/s. With the enhanced data rate (EDR) introduced with the Bluetooth 2.0 specification, it has been possible to increase this data rate to 3 MBit/s. This is made possible by the use of two variants of phase-shift keying (PSK): the π/4 differential quaternary phase-shift keying (π/4DQPSK) and the 8 differential phase-shift keying (8DPSK), which modulate the phase of the carrier.

π/4DQPSK represents one of four combinations of a bit pair (00, 01, 10 and 11) and so transfers a maximum of two bits; 8DPSK can represent any combination of three bits. This is used to increase the bit rate from 1 MBit/s (net 727 kBit/s) to 2 MBit/s (net 1,446 kBit/s) with GFSK, or 3 MBit/s (net 2,169 kBit/s) with EDR.

Power classes

Bluetooth devices are categorized into different power classes. Manufacturers are free to decide which power class they implement. The present specifications define three power classes:

- Class 1 with 100 mW (20 dBm) transmission output for a range of 100 to 150 m
- Class 2 with 2.5 mW (4 dBm) transmission output for a range of 10 to 25 m
- Class 3 with 1 mW (0 dBm) transmission output for a range of 10 m

Topology

Bluetooth supports the automatic configuration of ad-hoc networks. This means that two or more devices are able to spontaneously form a network with no prior knowledge of each other. Bluetooth is based on the master-slave principle. In principle, any Bluetooth device may be a master or a slave. The master is a special device that coordinates communication between the devices.

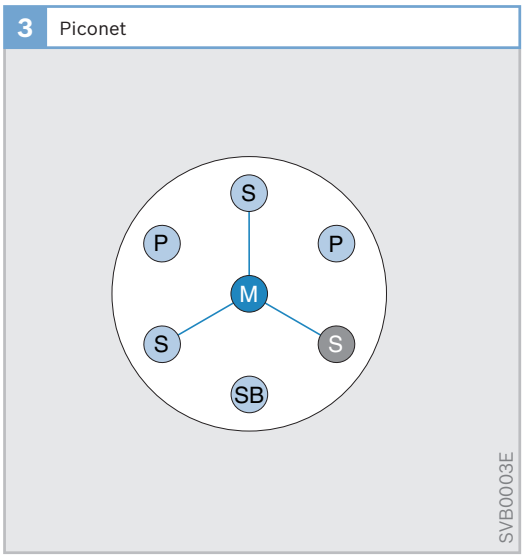
Piconet

A network of Bluetooth devices is known as a piconet (Fig. 3). For the devices within a piconet, there are two roles: master or slave. Which device assumes which role is not decided until communication is established. A device sets up a piconet and transmits its device ID and the value of its internal clock. This device becomes the master.

Any device that is not participating in a piconet but is not switched off remains in standby, which is marked by a small current draw.

The first stage in the creation of a piconet involves the potential master searching for other devices in its range (inquiry scan). Special frequency-hopping sequences (inquiry sequences) are available for this purpose. Devices that wish to enter into communication with other devices respond with an inquiry response message. By the end of the inquiry sequence, the information about all devices willing to communicate is available. The potential master is now able to address a specific device by means of a special paging sequence. Paging is a process whereby a fixed connection is established between a master and a slave. The slave contains all the information it needs to synchronize with the master.

3 Piconet



The piconet is defined by the frequency-hopping sequence, i.e. the sequence of switching between the 79 possible channels. The hopping sequence is specified by the master and is calculated from the master's device ID (a worldwide unique 48-bit identifier) and its internal clock. The slaves synchronize with the master's clock and its hopping sequence.

A master is able to manage up to seven active slaves. Each active slave receives a 3-bit active member address (AMA).

Active devices can send data or even simply remain connected. By deregistering, a device can switch back to standby. A Bluetooth device may also switch to any of three energy saving modes: hold mode, sniff mode and park mode. The devices continue to be synchronized with the hopping sequence of the piconet. The lowest energy saving mode is the park mode, where the device relinquishes its AMA and receives in its place an 8-bit parked member address (PMA). With the PMA, the device is still a subscriber in the piconet but makes way for another, active device. Parked devices remain synchronized with the hopping sequence of the piconet and can be addressed and reactivated by the master using the PMA.

Scatternet

In a piconet, only one slave is ever able to communicate with the master simultaneously. When more slaves enter the network, the data rate per device falls very quickly. This is where scatternets provide a solution. A scatternet consists of up to ten piconets in which the individual piconets partly overlap. In this way, it is possible to set up even larger networks in which all network subscribers may connect to one another. The scatternet in Figure 4 comprises two piconets. One of the Bluetooth devices is a member of both piconets. Through this device, the two networks are able to exchange data. However, this device, which is assigned

Fig. 3
M Master
S Slave
P Park mode
SB Standby

to more than one piconet, can only function as master in one of these networks.

A device that is assigned to two piconets in a scatternet must synchronize with the network with which it wishes to communicate. It cannot remain connected to both networks at the same time. Before the device leaves the old piconet, it must inform its master that it will be unavailable for a certain period of time. The remaining network subscribers can continue to communicate.

A master, too, may leave its own piconet and become a slave in another. All communication in its old network, however, is broken until it returns and assumes its master function again.

A master cannot become master of a second piconet. If this were to happen, both piconets would behave in an identical manner and form a single network.

Physical data channel

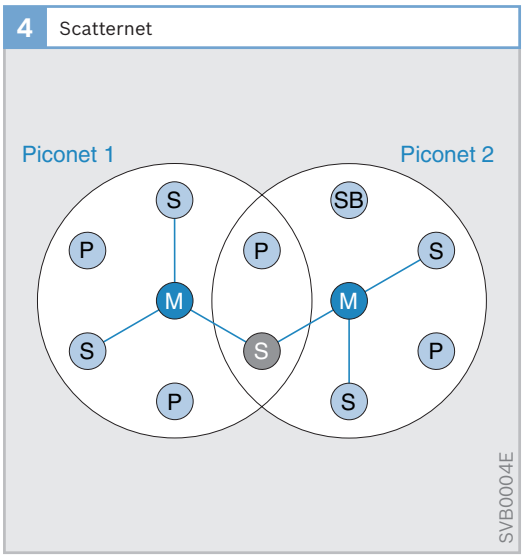
A data channel is represented by a hopping sequence between the 79 possible frequencies in the ISM band. Any device that is actively participating in a piconet must also hop at the same carrier frequency (frequency f_i). As soon as a master has sent data at frequency f_k, a slave can respond at frequency f_{k+1}. This procedure is illustrated in Figure 5.

It is conventional for the transmissions of the master to begin in the even time slots and those of the slaves to begin in the odd time slots. This characteristic conforms to the regular time-division multiplexing method (TDD here) whereby the master uses one half of the time slots and the slaves the other half.

In addition to packets that may occupy one time slot, Bluetooth provides for packets that may occupy three or five time slots (multi-slot packets). As soon as a master or slave sends a packet three or five time slots long, this transmitter will remain on the same frequency. There is no frequency change within a packet (Fig. 6).

Once the packet has been transferred, the frequency hops as determined by the hopping sequence (independently of the transmission process). In the illustration, for example, the frequency hops from f_k to f_{k+3} after the transmission of a 3-slot packet). The reason for this behavior is that not every station may have received the transmission and cannot therefore respond specifically to the transfer of data in several time slots. For this reason, all stations that are not involved in the transmission always continue to hop in the sequence specified by the master.

4 Scatternet



5 Timing of data transfer

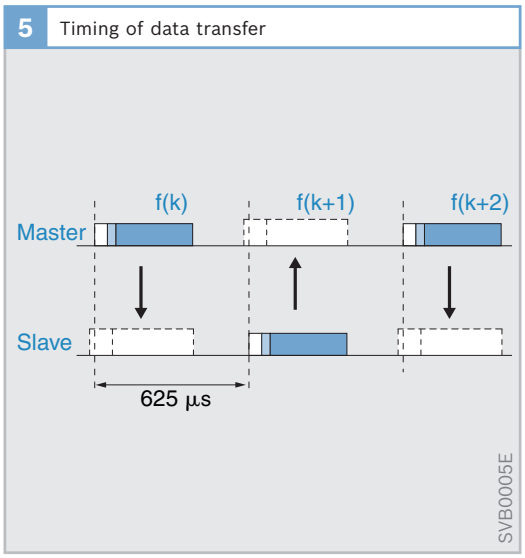


Fig. 4
M Master
S Slave
P Park mode
SB Standby

Physical connections

Bluetooth supports both circuit-switched and packet-switched data channels. For circuit-switched, synchronous communication, Bluetooth offers a synchronous connection-oriented link (SCO), and an ACL (Asynchronous Connectionless Link) for packet-switched, asynchronous communication. Between two Bluetooth devices, there may only ever be one ACL link but there can be up to three SCO links. The ACL links are the foundation for the data transfer. The SCO links are purely voice channels with a data rate of 64 kBit/s. The voice is transmitted by continuously variable slope delta modulation (CVSD) or pulse-code modulation (PCM). A Bluetooth master can maintain several different data channels between its slaves even at the same time.

Logical data channels

Logical channels refer to different types of channels streamed over a physical connection. The data transmitted over a physical channel has different logical meanings. Bluetooth distinguishes between two categories of channel: link channels, which are used for the exchange of control information between master and slave; and user channels, which transfer data from the user level – the applications.

Data packets

A Bluetooth data packet generally comprises three fields:

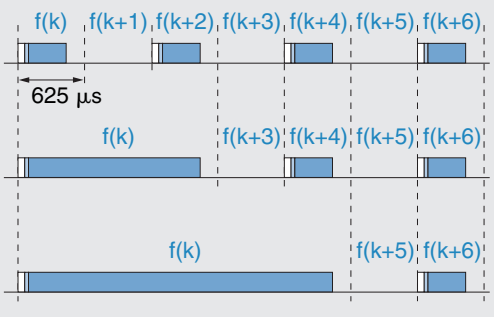
- Access code
- Header and
- Payload

A packet may simply comprise the access code, or the access code and a header, or the access code, header and payload (Fig. 7).

A packet begins with an access code. If a header follows, the access code has a length of 72 bits, otherwise it is 68 bits long. The access code is used for the synchronization and identification of a piconet, for device polling and for device calls. There are three different types of access code:

- Channel access code (CAC), the purpose of which is to synchronize and identify a piconet. The CAC is sent with each packet transmitted over the piconet.
- Device access code (DAC), which is used to transfer specific identifiers during the device call (e.g. when paging).
- Inquiry access code (IAC), which is used for device polling.

6 Frequency hopping with multi-slot packets



7 Structure of a packet

Access code 72 (68) bits	Header 54 bits	Payload 0 to 2,745 bits
4 preamble	3 AMA	
64 synchronization	4 type	
	1 flow	
(4) appendix	1 ARQN	
	1 SEQN	
	8 checksum	

SVB0007E

The header of a Bluetooth packet is 54 bits long and contains information about the link (link control information). It is made up of the address, packet type, flow control, error monitoring and checksum fields. The actual length of the header (i.e. the sum of the bits in the header) is 18 bits. The header is protected by forward error correction (FEC). The bits of the header are sent three times (1/3 FEC), giving a header length of 54 bits. A receiver can then simply take the majority decision; each bit triplet is resolved to the value with majority in the triplet.

The payload of a Bluetooth packet can be up to 341 bytes in size (1,023 bytes with EDR). The data field of the payload consists of up to three segments: the payload header, the payload itself and, in certain circumstances, a checksum.

Bluetooth uses 16 different packet types, the common feature of all being the access code and the header. The structure of the payload depends on the packet type concerned. Along with packets for ACL and SCO links, there are also packet types for polling the slaves, synchronizing the hopping sequence and acknowledging data transfers.

Bluetooth device addresses

A Bluetooth device is assigned a worldwide unique Bluetooth device address. The address is derived from the IEEE-802 standard and implemented in the device by the manufacturer. The 48-bit addresses are subdivided into three parts

- LAP (Lower Address Part, containing 24 bits)
- UAP (Upper Address Part, containing 8 bits)
- NAP (Non-significant Address Part, containing 16 bits)

The LAP and UAP fields form the significant part of a Bluetooth device address.

Bluetooth architecture

The Bluetooth architecture is complex. This is inherently associated with the idea behind general cable emulation. The architecture is designed to enable new protocols and applications to be adapted for the use of Bluetooth. Accordingly, the Bluetooth protocol stack is very extensive (Fig. 8 shows a simplified version of the protocol stack).

The Bluetooth technology can be subdivided into three logical sections:

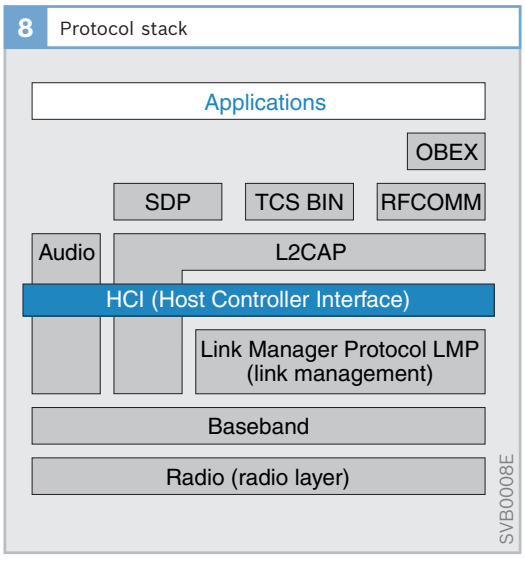
- Hardware
- Host stack
- Applications

The hardware layer and the host stack are defined in the Bluetooth Core Specification; the applications are part of the Bluetooth Profile Specification.

Hardware-side protocols

The hardware is represented by the Bluetooth chip. This is where the radio layer, baseband and link management are contained. The radio layer deals with the radio transmission, used frequencies, modulations and transmission output. The baseband represents the link-establishment mechanisms, the packet structure and the timing. The link management establishes and manages the link between

8 Protocol stack



two devices and also implements security and authentication functions.

The host controller interface (HCI) is the interface between the hardware of a Bluetooth module and the host-side protocols. The HCI is a hardware abstraction and realizes various interfaces for controlling the Bluetooth hardware and transferring data (e.g. via USB or UART).

Host stack

The host stack uses the HCI to control the Bluetooth hardware and transfer data. The format of this data is determined by the upper protocol layers. The logical link control and adaptation protocol (L2CAP) implements the abstraction of the hardware's properties, adapts the upper layers of the protocol stack to the capabilities of the baseband and hides transmission details such as the connectionless or connection-oriented transmission type. The L2CAP layer essentially has three main functions:

- It can receive packets with a length of up to 64 kB from the upper layers and decomposes them into smaller data packets (segments) for processing in the lower layers if necessary. At the other end, the segments are recomposed back into packets.
- It manages the multiplexing and demultiplexing of several packet sources. If a packet is being recomposed, the L2CAP layer determines to which protocol of the upper layers the packet is forwarded.
- L2CAP offers functions for negotiating quality of service and configuration parameters. This means, for example, that the maximum size of the payload can be negotiated so that a device with limited resources would not be overwhelmed by overly large packets. Using the configuration parameter for the quality of service, it is possible to define the properties of the data transfer: best effort (best attempt but with no guarantee of data transfer) or guaranteed.

In addition to L2CAP, there are also audio and control protocols that govern the handling of audio data and control data. Audio applications, for example, can make direct use of the baseband layer once the audio signals have been coded accordingly.

The service discovery protocol (SDP) is another important host-side protocol. It serves to identify and scan for services with specific properties and to describe services within range of a Bluetooth device. Bluetooth devices are meant to be able to interact on an ad hoc basis with other devices in different environments. It is therefore necessary to know which services are made available by which devices within range. All devices wishing to provide services must use an SDP server; for all other devices, an SDP client is sufficient. The SDP prepares the services available on a device in a service database. The service information in the possession of the SDP server is stored in a service record. The service record consists of a list with service attributes that describe the properties of the service more precisely and is identified by a 32-bit service record handle.

Two further protocols in the protocol stack form the fundamental basis for the interoperability of Bluetooth devices.

The RFCOMM (Radio Frequency Communication) cable emulation protocol above the L2CAP layer simulates up to 60 virtual serial interfaces derived in accordance with the ETSI 07.10 standard. As a result, almost any software that previously expected a serial interface can work with Bluetooth.

TCS BIN (Telephony Control Protocol Specification – Binary) is employed as the means of controlling telephone and telephony functions. This is a bit-oriented protocol for establishing voice and data connections between Bluetooth devices. The use of appropriate, indus-

try-wide standards has made it possible, here too, to ensure wide-ranging compatibility with legacy applications.

Applications

Many more protocols have been adapted for the Bluetooth standard (adapted protocols) and can be found in the protocol stack. Internet applications can, for example, continue to use TCP/IP through the point-to-point protocol (PPP) or the Bluetooth network encapsulation protocol (BNEP). For the exchange of vCalendars and vCards, it is possible to use the object exchange protocol (OBEX) covered by the IrDA standard.

Profiles

The uppermost layer contains the applications and profiles. Profiles represent "standard solutions" for a particular usage scenario. The Bluetooth specification currently brings together 13 different applications described as profiles. Each profile uses a certain choice of protocol; in principle, application profiles make a different protocol stack available for each application. Profiles describe the vertical slice through the Bluetooth protocol stack; the protocols represent the horizontal layers (Fig. 9). Within the profiles, the required and optional functions of the layers are defined. These standardized profiles make it possible to ensure interoperability between different devices.

The four fundamental profiles of the Bluetooth specification are:

- Generic access profile (GAP)
- Service discovery application profile (SDAP)
- Serial port profile (SPP) and
- Generic object exchange profile (GOEP)

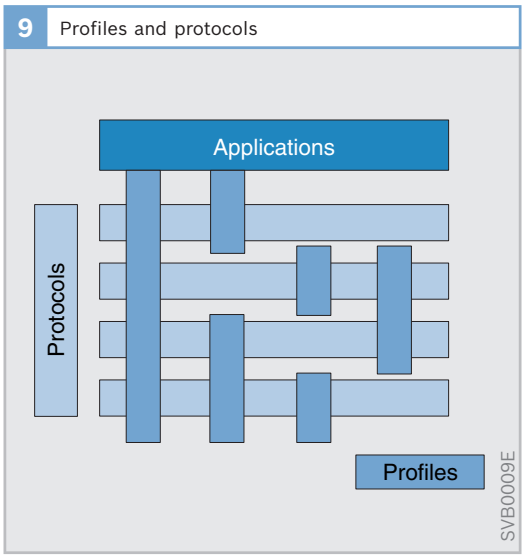
The Bluetooth profiles form the hierarchy (Fig. 10). The GAP forms the basis and describes all the essential functions that a Bluetooth device must fulfill at the lowermost level. These include, for example, the functions for establishing and managing the link, the supported operating modes and the security of a link.

The SDAP defines the access interface for the service discovery protocol (SDP) with which devices can discover or poll the services offered by other devices. SDP builds on the GAP. All Bluetooth devices must implement these two profiles.

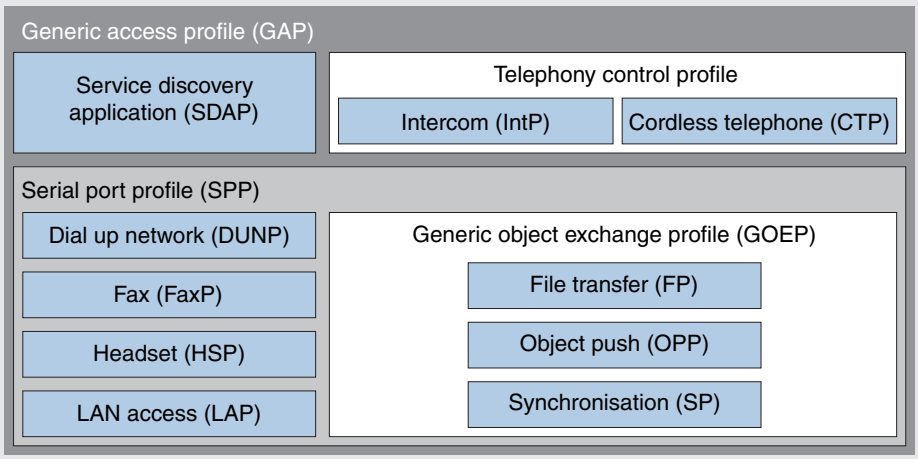
The SPP is used by most other Bluetooth profiles. One exception to this is the telephony control profile. This is always used when Bluetooth is being used for cable emulation or if a serial data connection is to be used. The SPP builds on the GAP and uses the RFCOMM protocol.

The GOEP defines the fundamental functions necessary for the exchange of complex objects. It defines a client-server relationship for the exchange of data. Like the SPP, the GOEP provides the foundation for further profiles.

9 Profiles and protocols



10 Profile hierarchy



Origin of the name, Bluetooth

The name, Bluetooth, originates from King Harald Gormsen, a Danish Viking. Harald had the nickname, Blatand. In the Middle Ages, Bla actually meant dark, while Blatand made reference to the dark figure of Harald Gormsen. Bluetooth, therefore, is not a direct translation of Blatand.

In the 10th Century, Harald Gormsen united large parts of Denmark and Norway. The Bluetooth wireless system developed 1,000 years later unified a wide range of different information, data processing and wireless mobile devices. This is why it was named after King Harald.

MOST bus

Introduction

The MOST bus (Media Oriented Systems Transport) was specifically developed for the networking of infotainment systems in motor vehicles. In addition to traditional entertainment functions, such as radio tuners or CD players, these infotainment systems (which are becoming an increasingly common feature of modern vehicles, and premium class vehicles in particular) also offer video functions (DVD and TV), route guidance capabilities and access to mobile communications and information.

Requirements

The functions of an infotainment system place high demands on a bus system. The transmission of multimedia data, both audio and video, requires a high data rate and a synchronization of the data transfer between source and sink as well as between sinks themselves.

For the transmission of informational data, e.g. detailed information about the music tracks on an mp3 player, and the sending of software updates, it is necessary for the data transfer to be flexible with varying and, at certain times, equally very fast data rates.

Likewise, the requirements for use in a motor vehicle, and the associated requirements of electromagnetic compatibility (EMC), must also be fulfilled.

MOST Cooperation

It was with these requirements in mind that the MOST bus was developed by the MOST Cooperation. This was founded in 1998 by BMW, DaimlerChrysler, Harman/Becker and Oasis SiliconSystems (now part of SMSC). By 2006, the MOST Cooperation had 16 automotive manufacturers and 67 suppliers and tool makers among its membership.

The MOST Cooperation creates and administers the specifications on which the MOST bus is based. Furthermore, it defines the requirements for implementation of MOST devices and provides appropriate compliance tests through accredited test houses.

Type of use

At present, the MOST bus is almost exclusively used for the networking of infotainment systems in motor vehicles. It can currently be found in over 35 vehicle models of various manufacturers, especially in the premium and mid-range classes (as at 2006).

Features of the MOST bus

The MOST bus supports the logical networking of up to 64 devices, at which point constraints associated with the chipsets may be encountered.

In its current version, the MOST bus offers a data rate of 24.8 MBit/s (MOST 25). Versions with higher data rates of 50 MBit/s (MOST 50) and 150 MBit/s (MOST 150) are already available as a development model.

Data transmission channels

For the transmission of data, the MOST bus supports the following channels and their different attributes:

- The control channel is used for the simple transmission of control commands, for the signaling of device statuses and for the sending of messages necessary to system management. With MOST 25, the control channel has a gross bandwidth of 705.6 kBit/s.
- For the transmission of multimedia data, the MOST bus has a flexible number of synchronous channels that are able to carry both audio and video data. In MOST 25, there are a maximum of 15 audio channels available in stereo quality.

- On the asynchronous channel, data is sent in packets. It is therefore suitable for transferring information that does not require a fixed data rate but does need a high data rate at certain times. Examples could be the transfer of track information of an mp3 player or a software update. With MOST 25, the asynchronous channel has a gross bandwidth of up to 12.7 MBit/s.

The available bandwidth can be flexibly distributed between the asynchronous channel and the synchronous channels by means of the "boundary descriptor", which can also be shifted to the transit period if relevant preconditions are met.

Topology

A MOST system is arranged in a ring structure whereby a device is connected to its predecessor or successor in the ring by an input or output respectively (Fig. 1). One of the devices acts as the "timing master" and generates the data frames for data transfer with which the other devices synchronize.

Each device has a bypass. When this is closed, the device forwards the signal directly and is therefore invisible to the MOST system. This service is useful, for example, when the system is starting and a device requires more time for initialization, or in the event of a temperature-dependent shutoff.

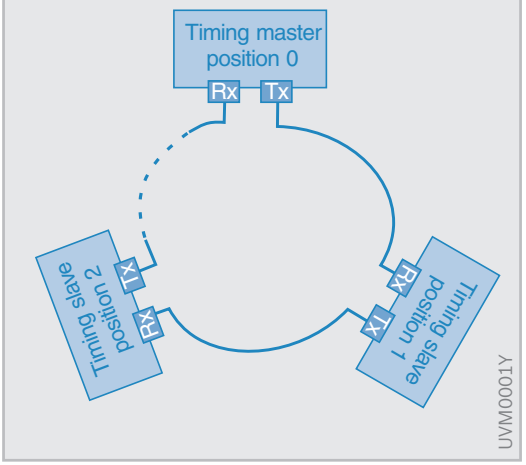
There are also other possible structures for a MOST system, e.g. a star topology. Internally, however, these must always be arranged to form a (logical) ring. With a star structure, this can be achieved through use of a central hub, for example.

Device model

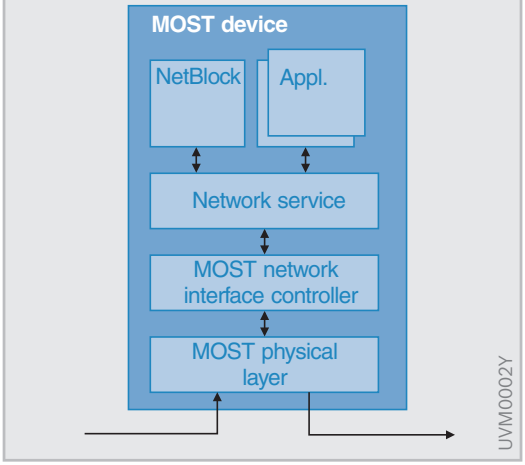
The MOST standard defines the model shown in Figure 2 for a MOST device for which the following elements are required:

At the lowermost level, the physical layer provides access to the transmission agent. Departing from the original definition that provided for optical transmission by POF cable (Plastic Optical Fiber), there are today various physical layers available with optical and electrical transmission as well as various speeds of transfer.

1 Ring structure of the MOST system



2 Model for a MOST device



The MOST network interface controller (NIC) is a hardware controller that is responsible for controlling the physical layer and implements the basic transfer services. While in the original version of the NIC many control tasks had to be carried out by the main processor of the device, the current version (Intelligent Network Interface Controller, INIC) is, for the most part, able to handle tasks autonomously and already implements substantial parts of the network service. Typical controllers for the MOST are the OS8104 (NIC) and the OS81050 (INIC), manufactured by SMSC.

The MOST network service is the driver layer through which applications and system services have access to the NIC. The lower layer of the network service (layer 1) offers basic communication and management functions, while the layer above (layer 2) offers functions for supporting the development of applications, such as a translator for the application protocol or support for the notification service.

The applications of the device are implemented on top of the network service. In a MOST system, the interface of an application is realized as a function block (FBlock). Each device must at least implement a special FBlock, the NetBlock, which is required for management functions within the MOST system. In addition, a device will usually implement one or more applications that can be used as FBlocks of applications on other devices. A device of a vehicle infotainment system is often integrated with several functions, e.g. those of a radio tuner and those of an amplifier, which can be represented in the MOST system as independent FBlocks.

Transmission agent

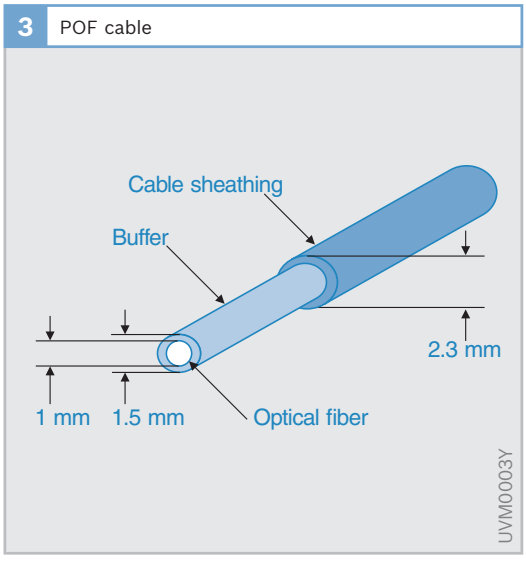
The MOST bus was originally defined as an optical transmission agent that used a plastic optical fiber (POF). The optical signal used here with a wavelength of 650 nm (in the red wavelength range) is generated by an LED on the transmit side, which is designated Tx-FOT (FOT = Fiber Optic Transceiver). On the receive side, Rx-FOT, the optical signal is converted back into an electrical signal by a PIN photodiode.

The POF cables used for automotive applications (Fig. 3) consist of a 980 µm thick optical core insulated by 20 µm thick optical cladding with a low refractive index. In total, the optical conductor therefore has a diameter of 1 mm. The optical fiber is insulated with a black buffer, which is in turn surrounded by a protective cable sheathing. This gives the cable a total diameter of 2.3 mm.

POF lines offer the following benefits when used in the motor vehicle:

- No electromagnetic interference radiation
- Insensitivity to interference irradiation
- Lower weight than equivalent shielded electrical lines
- More flexible routing in comparison to equivalent shielded electrical lines

3 POF cable



To date, modern vehicles are equipped exclusively with MOST 25, with POF data transmission via POF connections. In addition to this, however, further transmission techniques have been defined for the MOST:

- The optical transmission by glass fiber cable (PCS) with laser diodes (VCSEL), which offers a greater damping reserve, supports higher speeds and is less sensitive to high temperatures.
- An electrical transmission by copper cable, which is also less sensitive to temperature and is more economical by comparison, requires additional shielding measures at higher bandwidths. This has consequences for costs and cabling.

Data transfer

Data transfer on the MOST bus is organized into data frames, which are generated by the timing master with a fixed data rate and passed on by subsequent devices in the ring.

Data frames

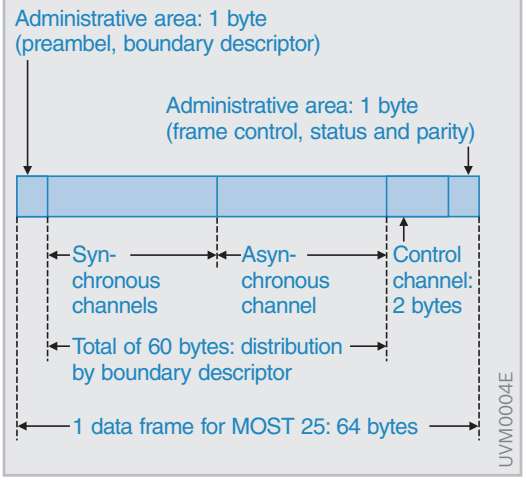
The timing master normally generates data frames with a cycle of 44.1 kHz, and in rarer cases 48 kHz. The cycle is defined by the system manufacturer (i.e. the vehicle manufacturer in conjunction with its device suppliers) to suit the predominant media formats in the system. The size of the data frames, by extension, determines the bus speed of a MOST system. With MOST 25, the size of a data frame is 512 bits.

A data frame is made up of the following ranges (Fig. 4):

- Areas for administrative information at the start and end of the data frame, the tasks of which include the synchronization of the data transfer, determination of the devices' ring positions and the transmission of the boundary descriptor
- An area for the control channel (with MOST 25, two bytes per data frame)
- An area for the synchronous channels, whereby one or more bytes are always assigned to a specific synchronous channel
- An area for the asynchronous channel

In the case of MOST 25, the synchronous and asynchronous area demand a 60-byte share of the data frame. The distribution between the synchronous channels and the asynchronous channel is determined by the value of the boundary descriptor with a resolution of 4 bytes. The synchronous area must have at least 24 bytes (6 stereo channels). This means that 24 to 60 bytes are permitted for the synchronous area and 0 to 36 bytes are permitted for the asynchronous area.

4 Data frame



System events

The fundamental system events described below indicate the current status of the data transfer in the MOST system. These are monitored by the NIC and forwarded by the network service to the applications, which must then respond accordingly.

- Lock, stable lock: a lock occurs as soon as communication is established in the ring, i.e. the correct device receives the correct data frames. A stable lock is set when the system has been in lock status for a specific period.
- Unlock, critical unlock: an unlock denotes the loss of the lock, i.e. that no correct data frames are being received. If the system is in the unlock state for a certain period, the status switches to critical unlock.
- Network change event: if the number of devices in a MOST system changes because one device has opened or closed its bypass, this is signaled by a network change event.

Addressing

Devices on the MOST bus are addressed by means of a 16-bit address. The following different addressing types are supported:

- Logical addresses: each device has a logical address that is unique in the system and is set and distributed when the system starts. The logical addresses can be set dynamically based on the ring position, for which, beginning with the timing master, the hexadecimal range 0x100 to 0x13F is reserved. Or they can be defined statically by the system manufacturer in a range reserved specifically for this purpose.
- Physical addresses: for system administration services, e.g. determination of logical addresses, each device also has a physical address that is derived from its current ring position. The physical addresses are assigned the range 0x400 to 0x43F.
- Group addresses: each device may also be assigned a group address in the 0x300 to 0x3FF range, making it possible to address all devices in a given group at the same time.
- Broadcast address: the MOST additionally supports a special broadcast address, hexadecimal 0x3C8, which makes it possible to address all devices in the MOST system.

Transmission of control commands

The control channel permits the transmission of control messages with a maximum length of 32 bytes. These have to be shared among consecutive data frames. With MOST 25, the 16 data frames required here are also known as a block. The format of the control messages is defined by the protocol described in the next section.

The sending of messages on the control channel is supported by the NIC. It provides the necessary services for data protection and channel arbitration, i.e. for determining the next opportunity for a message to be sent. The reliable transmission of control messages is safeguarded by the use of a cyclic redundancy code (CRC) and an automatic repeat request.

The control channel's gross bandwidth of 705.6 kBit/s is reduced to a net data rate of 406 kBit/s following the deduction of approximately 3 % for system administration overheads (two of 64 control messages) and less the extra overhead to account for arbitration and data protection. Since arbitration only allows a particular device to populate every third available control message, an individual device has a maximum data rate of 135 kBit/s at its disposal for sending.

To enable the sending of messages of more than 32 bytes, the network service supports the segmentation and desegmentation of application messages up to 65,556 bytes in size on the control channel. This service, also known as the appli-

cation message service (AMS), differs from the control message service (CMS), which only allows 32-byte control messages to be sent.

Transmission of multimedia data

In the MOST system, the transmission of multimedia data takes place on the synchronous channels. The data transfer is controlled by relevant control commands on the control channel. A synchronous channel can be assigned a specific bandwidth, which is achieved with a resolution of one byte of a data frame. A stereo audio channel with a resolution of 16 bits requires four bytes, for example. In the case of MOST 25, a maximum of 60 bytes (depending on the value of the boundary descriptor) are available for the synchronous channels, which equates to 15 stereo audio channels.

The sinks and sources for multimedia data are each assigned to an FBlock, which provides the necessary functions for their management. An FBlock may therefore contain several sources and sinks, which are numbered consecutively with a source and sink number respectively.

An FBlock has functions that supply information about the quantity and type of sources and sinks that it makes available (SyncDataInfo, SourceInfo and SinkInfo). Furthermore, each FBlock with a source has an "Allocate" function with which it requests a synchronous channel and associates its source to it. Its "Deallocate" function, on the other hand, is used to free up the channel again.

Similarly, an FBlock with a sink has a "Connect" function for connecting the sink to a particular synchronous channel, and a "DisConnect" function for its disconnection. Several sinks may also be connected to a particular channel at the same time.

With MOST 25, the channel assignment for synchronous channels is managed by the timing master. A source wishing to occupy a synchronous channel sends an appropriate system message request (ResourceAllocate) to the timing master. A corresponding message (ResourceDeallocate) is used to free the channel.

Transmission of packet data

On the asynchronous channel, data is sent in the form of packets. The asynchronous channel currently supports two modes: a slower 48-byte mode, where 48 bytes are available in each packet for the net data transfer but which places less heavy demands on device implementation, and a 1,014-byte mode that is more complex to implement.

With MOST 25, between 0 and a maximum of 36 bytes of a data frame can be assigned to the asynchronous channel, which corresponds to a maximum gross bandwidth of 12.7 MBit/s. For the 1,014-byte mode, this produces a maximum net bandwidth of almost 11 MBit/s, and a data rate of almost 3 MBit/s for the 48-byte mode. In practice, however, substantially lower data rates are achieved mainly due to restrictions in device implementations.

The transmission of data packets on the asynchronous channel is also directly supported by the NIC. Likewise, a packet on the asynchronous channel contains a CRC to permit detection of simple transmission errors. Unlike the control channel, however, there is no issuing of automatic repeat requests on the data link layer.

To ensure reliable transfer and flow control in the large data volumes typical of the asynchronous channel, it is conventional to use yet another transport protocol that is implemented in a higher driver layer. This may either be the MOST high protocol (MHP), developed specifically for the MOST, or the popular TCP/IP protocol, which is set up on an appropriate adaptation layer known as the MOST asynchronous medium access control (MAMAC).

Administrative functions

The MOST standard defines the following administrative services that are required for the operation of a MOST system.

Configuration status

A valid configuration status in the MOST system is the prerequisite for communication in the application layer. The configuration of the MOST system is managed by a single device, the network master. An applicative communication is only permitted once this device has signaled a conflict-free configuration status.

- OK configuration status: the network master transmits the OK configuration status as soon as a conflict-free configuration is achieved.
- NotOK configuration status: if a conflict arises, e.g. because two devices have the same address, the network master transmits its NotOK configuration status. In response to this, all devices initialize their addresses and communication settings.
- New and Invalid configuration statuses: these configuration statuses indicate that a new application (FBlock) has been registered in the system or that a previously registered application has dropped out.

NetBlock

It is mandatory for each device to implement an FBlock called the NetBlock, which covers a range of different administrative functions (e.g. for address initialization) and supplies information about the device and the FBlocks that it implements.

Network master

The network master is implemented by a special device in a MOST system and is responsible for system configuration. In present-day systems, the network master is usually realized by the headunit (i.e. the control element) of the infotainment system. This device often assumes the role of timing master too, but this does not necessarily have to be the case. The other devices in the MOST system in this relationship are known as network slaves.

The network master manages an image of the current configuration of a MOST system in the central registry. In addition to the addresses of all devices, this also contains all FBlocks implemented by them (down to the system FBlocks such as the NetBlock). A network slave searching for the device address for a particular FBlock may request it from the network master.

If a network slave frequently accesses other devices, it may, for faster access, also store a local copy of the central registry – a decentral registry – which it must update accordingly in the event of any change in the configuration.

To build the central registry, the network master carries out a network scan when the system starts and as soon as a network change event has occurred. During this scan, it queries the NetBlocks of all devices for the FBlocks implemented by this device. If it detects a conflict while doing so, e.g. duplicate device address or duplicate instance of an FBlock, it takes measures to resolve the conflict. To do this, the network master may initiate a recalculation of addresses, convert the InstId of an FBlock or, if necessary, ignore a device.

Connection master

The connection master manages the synchronous connections existent in the MOST system at a given time. It makes available a table containing information about all the connections currently present in the MOST system (SyncConnectionTable). It also makes it possible to query the bandwidth remaining for further synchronous connections (AvailableChannels).

On request, the connection master builds connections between a specific source and sink (BuildSyncConnection) or removes them (RemoveSyncConnection) by sending the necessary commands to the associated FBlocks.

It is even possible to dispense with the connection master completely if a central entity in a MOST system, e.g. the headunit, is responsible for the administration of the synchronous channels.

MOST application layer

The MOST standard defines a suitable protocol in the application layer for the transmission of control commands, status information and events. This protocol makes it possible to address a specific function of an application interface (i.e. of an FBlock) that is provided by any device within the MOST system. For example, it is possible to start playback of a CD on a separate CD player or to query the number of the track currently playing in it. While the MOST protocol is mainly applied to the control channel, it can be transferred across the asynchronous channel if necessary.

Overview

The protocol for MOST control messages provides for the following elements of a control message (Table 1):

- The address of a device in the MOST system (DeviceID)
- An identifier for an FBlock implemented by this device (FBlockID) and its instance in the MOST system (InstID)
- The identifier for the requested function within the FBlock (FktID)
- The type of operation (OpType) that should be applied to this function, e.g. the setting or querying of a property of the FBlock
- A data range containing the parameters of the function call (Data) and a corresponding length value (Length)

1 Elements of a control message

Field	Size	Description
DeviceID	16 bits	Device address
FBlockID	8 bits	FBlock identifier
InstID	8 bits	Instance of the FBlock
FktID	12 bits	Function identifier
OPType	4 bits	Type of operation
Length	16 bits	Length of the data field
Data	0 to 65,535 bytes	Data field

Table 1

Function block (FBlock)

A function block (FBlock) defines the interface of a specific application or system service. An FBlock is assigned an address comprising an 8-bit FBlockID, which specifies the type of FBlock, and an additional 8-bit InstID. The latter is a means of distinguishing between several instances of FBlocks of the same type in a MOST system.

An FBlock that is controlled by an associated application, and the device that implements the FBlock, are known as the slave in this relationship (e.g. an application that addresses an external CD player through the associated FBlock). The application that controls the FBlock is called the controller.

The functions that an FBlock possesses are defined by the function catalog. In the case of system services, the function catalogs are defined by the MOST standard. The MOST standard even defines the interfaces for common applications from the area of vehicle infotainment systems (e.g. for an amplifier or a CD player/changer). However, these interfaces tend to be extended by the system manufacturer. Other proprietary FBlocks are defined in their entirety by the system or device manufacturer.

Functions and operations

A function block is made up of several functions that are addressed with a 12-bit FktID. Depending on the type of function, the function may be assigned differently predefined operations, which are described by a 4-bit OpType identifier. In the case of functions, a further distinction is made between methods and properties.

A method describes an action that an FBlock is able to execute, e.g. starting the station scan of a radio tuner. A method can be initiated by the controller using a Start or StartResult operation. In response, the slave returns a corresponding result message if applicable (for StartResult). If this result is not made available within a specific period, the controller is notified that the method is still being processed by means of a processing message. In the end, a method may even be terminated by the sending of an abort message.

A property describes a particular attribute of the FBlock, such as the number of the track currently playing. A property such as this can be requested (Get operations) or set (Set or SetGet operations) by the controller. As the result of such a query, the slave responds with a status message, if applicable (for Get and SetGet), containing the current contents of the property.

Furthermore, the MOST standard provides for a notification service with which a controller may register with a slave for certain properties. If the value of one of these properties changes, all controllers that have registered for notification are informed of the change by a status message.

Data field

The data field of the control message contains the parameter values for a function call or its results. The data field is interpreted as specified in the definition of the function concerned and may contain one or more parameter values.

The MOST standard defines the following parameter types:

- Boolean and BitField for individual boolean values (1 byte) or sequence of individual bits (1, 2 and 4 bytes).
- Enum for enumerations.
- Unsigned and signed byte, word and long for integer values with or without sign and a size of 1, 2 or 4 bytes.
- String for character strings. These are zero terminated and contain a description of the character coding in the first byte.
- Stream, ClassifiedStream and ShortStream for various types of byte sequences of any length.

A parameter type for real numbers does not exist. The MOST standard provides for real numbers represented as fixed-point values by the integer parameter types. More complex data types are not defined in the standard either. These have to be realized by function classes instead.

Function classes

To standardize the way in which functions are defined, the MOST standard specifies a series of function classes for properties. These determine which properties the function has and which operations are permissible.

In addition to simple function classes (with a single parameter for numerical values or texts), there also exist complex, composite function classes. These are used to represent composite data structures such as records and arrays. They can also be nested at a specific nesting depth, which makes it possible to create two-dimensional arrays or arrays of records, for example. These nested data structures are, for example, used to represent the phone book of a mobile phone.

With multidimensional function classes, the two parameters PosX and PosY are used to address a particular position in the first or second dimension, where 0 always stands for all elements of this dimension. This makes it possible to access not only a specific element but also an entire line of a twodimensional array.

Applications

As well as defining the lower layers necessary for data transfer, the MOST standard defines the interfaces for typical applications from the area of vehicle infotainment systems, e.g. a CD changer, amplifier or radio tuner.

The FBlocks defined by the MOST Cooperation are summarized in a function catalog and listed with their current versions in the next section. The function catalog defines in detail all functions for the FBlocks contained in them as well as the permissible operations and their parameters.

For the function catalog, there is a machine-readable description available in XML format that can be used to import the catalog into various MOST tools.

In addition to the function catalog's description of all functions of an FBlock, developments are underway to specify the dynamic behavior for the use of an FBlock. This involves the use of message sequence charts (MSCs) compliant with the MSC 2000 standard but with minor, MOST-specific enhancements.

Functions required by various applications, e.g. the aforementioned functions for managing sources for multimedia data, are defined in the GeneralFBlock. This is used as a function compilation, i.e. when a new FBlock is defined, all necessary functions are copied from the GeneralFBlock to the new FBlock.

Standardization

The MOST standard is maintained by the MOST Cooperation, which also publishes the corresponding specifications. The specifications are available through the home page of the MOST Cooperation (www.mostcooperation.com).

The following versions of the MOST specifications are currently valid (as at 2006):

- MOST Specification, Version 2.4.
- MOST Dynamic Specification, Version 1.2.
- MOST Specification of Optical Physical Layer, Version 1.1.
- MOST Specification of Advanced Optical Physical Layer, Version 1.0.
- MOST Specification of Electrical Physical Layer, Version 1V1.
- MOST High Protocol Specification, Version 2.2.
- MAMAC Specification, Version 1.1.

The MOST Cooperation defines within the framework of the compliance process requirements placed on MOST devices which must be satisfied by all devices in order to be able to carry the MOST logo. MOST compliance is tested and awarded by test institutes which have received appropriate accreditation from the MOST Cooperation. The compliance requirements are governed by the following specifications, which describe the compliance process itself and the compliance tests for the physical layer, for system mechanisms (Core) and for application interfaces (Profile):

- MOST Compliance Requirements, Version 2.0.
- MOST Compliance Test of Physical Layer Specification, Version 1.0.
- MOST Core Compliance Test Specification, Version 1.1.1.
- MOST Profile Compliance Test Specification, Version 1.0.

The following application interfaces are standardized by the MOST Cooperation. The following versions are currently valid:

- AudioAmplifier (FBlockID: 0x22), Version 2.4.2: a simple amplifier.
- AuxIn (FBlockID: 0x24), Version 2.4: an interface for connecting MP3 players or data carriers with pieces of music.
- MicrophoneInput (FBlockID: 0x26), Version 2.3.1: a microphone input.
- AudioTapePlayer (FBlockID: 0x30), Version 2.3.1: a cassette player.
- AudioDiskPlayer (FBlockID: 0x31), Version 2.4: a CD player or CD changer.
- DVDVideoPlayer (FBlockID: 0x34), Version 2.4.1: a DVD player or DVD changer.
- AmFmTuner (FBlockID: 0x40), Version 2.4.2: a radio receiver for FM/AM.
- TMCTuner (FBlockID: 0x41), Version 2.3.1: a special receiver for traffic-message signals (TMC).
- TVTuner (FBlockID: 0x42), Version 2.3.2: a television receiver.
- DABTuner (FBlockID: 0x42), Version 4.0: a receiver for digital radio (DAB).
- SDARS (FBlockID: 0x44), Version 2.4: a receiver for satellite radio.
- Telephone (FBlockID: 0x50), Version 2.3.2: a telephone module or a connection to a cellular phone.
- GeneralPhonebook (FBlockID: 0x51), Version 2.3.1: access to a phonebook, e.g. that of a cellular phone.
- NavigationSystem (FBlockID: 0x52), Version 1.11: an interface for a navigation system; but only unofficially approved by the MOST Cooperation.
- GraphicDisplay (FBlockID: 0x60), Version 2.3: an independent display screen.

TTP/C

Overview

The Time-Triggered Protocol for SAE Class C networks (TTP/C) is a time-controlled protocol in which the network users transfer data in special time intervals determined in advance. In other words, all network users have a global time definition which is determined by means of a time-synchronization protocol.

TTP/C was developed specially to satisfy real-time requirements in distributed, fault-tolerant systems. It was developed specifically with a view to its use in the automotive field, but has in the meantime also come to be used in other fields, such aviation and railroads.

According to the requirements of the Society of Automotive Engineers (SAE) for Class C protocols, TTP/C is a protocol for high-speed networks (>125 kbit/s bit rate) which can be used for real-time communication in check systems. For this reason, particular emphasis was placed during development on the fields of fault detection and tolerance, robustness against malfunctions, composability, and guaranteed latencies.

TTP/C is based on the principle of Time-Triggered Architecture (TTA), a framework for developing highly reliable, distributed real-time systems and the applications that run on them.

The development of TTP/C dates back to the MARS project (Maintainable Real-Time System), which was conducted in 1979 at the Technical University of Berlin. Further development followed within the framework of further projects, such as TTA (Time-Triggered Architecture) or the Brite-EuRam III project “Safety Related Fault Tolerant Systems in Vehicles (X-by-Wire)”, in which systems for activating safety-relevant automotive components, such as brakes or steering systems, via electronic networks without a mechanical fallback level were examined.

Areas of application

TTP/C was originally developed for use in safety-relevant systems in the automotive field. Because the core functions of TTP/C were formally verified and a TTP/C network can be put together economically compared with other protocols with similar reliability properties, TTP/C has also in the meantime come to be used in aircraft, a field which is subject to very stringent requirements with regard to safeguarding functionality. TTP/C is also used in train control systems.

Fault-tolerance strategy

The fault-tolerance strategy is used to describe which faults can be detected and handled. TTP/C is able to detect and handle each individually occurring fault (Single Fault Hypothesis). In addition, a series of multiple fault scenarios is handled.

However, some important multiple fault scenarios, such as a complete temporary breakdown of communication for example, cannot be detected on the node level. Additional methods of fault detection and handling are required on the application level for this purpose.

The TTP/C network must consist of at least four real member nodes to ensure that all single faults are detected.

The network ensures on the architecture level that a faulty network node cannot prevent any properly functioning network node from transmitting its message. This is done by bus guardians, which permit bus access only at defined points in time.

Time-Triggered Architecture

Time-Triggered Architecture (TTA) forms the basis on which the TTP/C protocol was specified (Fig. 1). The core element of TTA is the network node, which comprises a host processor and a TTP/C controller, referred to in the following as the controller. The network node is the “Smallest Replaceable Unit” (SRU) in a TTP/C system. The host processor processes inputs from

sensors and activates actuators. The Communication Network Interface (CNI) forms the interface between the host processor and the controller and provides memory areas via which the host processor and the controller can exchange information.

The controller is in the end connected to the TTP/C bus, by way of which it connects the node with other nodes. The group of all the nodes connected to a TTP/C bus and the bus forms a cluster.

Host Processor

The host processor executes the application, the actual function of the network node. In order to support the application, an operating system, which also for the most part provides the FT-COM layer, runs on the host processor as well as the application. The FT-COM layer introduces a further abstraction level of the technical realization of communication. The application can provide data for transfer via the interfaces of the FT-COM layer, thanks to which the application designer does not have to deal with deeper concepts, such as the memory areas in the CNI.

Communication Network Interface (CNI)

The CNI functions as a temporary firewall because it makes available a memory area via which data received and to be sent are exchanged between the host processor and the controller. This interface represents the host processor's sole possibility of transferring information via the network. It is therefore not possible for the host processor to influence the communication sequence. In particular, the point in time at which information is provided by the host processor has no influence on the network's communication sequence.

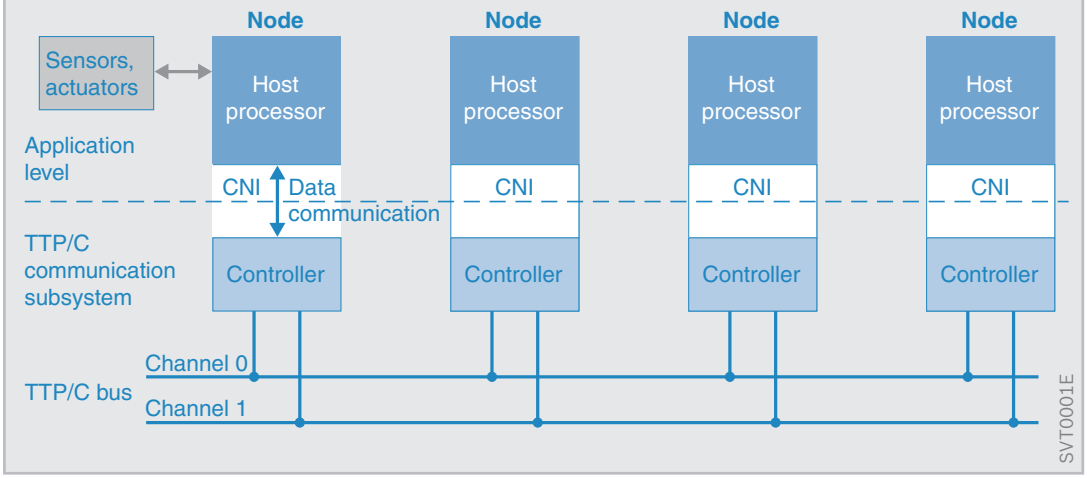
TTP/C Controller

The controller (Fig. 2) operates entirely independently of the host processor. Its most important components are the protocol processor, a local bus guardian, and the Message Descriptor List (MEDL).

The function of the protocol processor is to prepare information provided by the host processor in the CNI in such a way that they can be transferred as TTP/C frames.

The controller is connected via two interfaces with the remaining components of the network node. The connections to the host processor and to the transceivers (drivers) are established via the CNI and the Logical Line Interface respectively.

1 TTA network



Message Descriptor List (MEDL)

The MEDL contains all the check information required by the controller to send and receive data. It is a list adapted for each controller containing, for example, information on the position and size of transmit and receive slots.

The information stored in the MEDL includes the points in time at which the controller frame may be transmitted and at which frames of other controllers are expected which are processed by the host processor.

CNI memory positions in which information from the host processor is to be provided for transfer and in which received information from the controller is to be stored are also defined in the MEDL.

The MEDL also contains further information needed for the operation. For example, the identification of frames used to synchronize the internal clocks and the synchronization times.

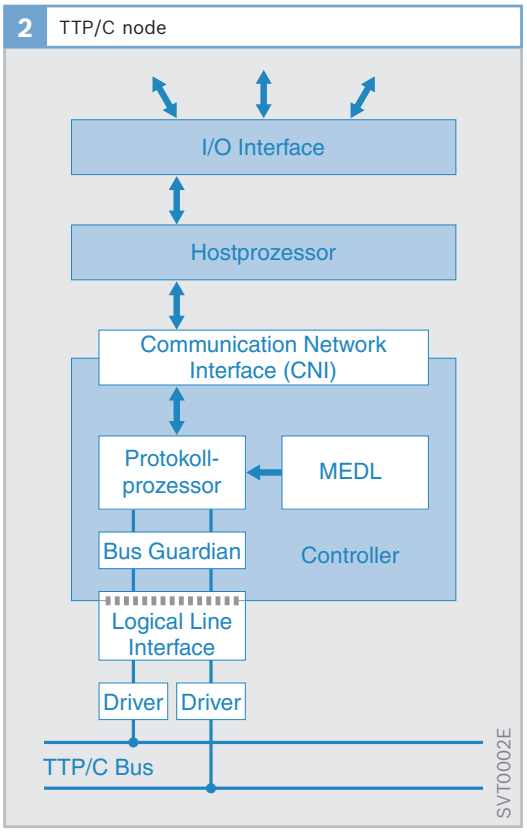
Strictly speaking, the designation Message Descriptor List is incorrect. Information blocks within a frame are designated as messages in TTP/C (Fig. 3). However, this structure exists only from the perspective of the host processor, not from the perspective of the controller, which only processes frames. The designations has persisted, however, because it was introduced in earlier specifications.

Local and Central Bus Guardians

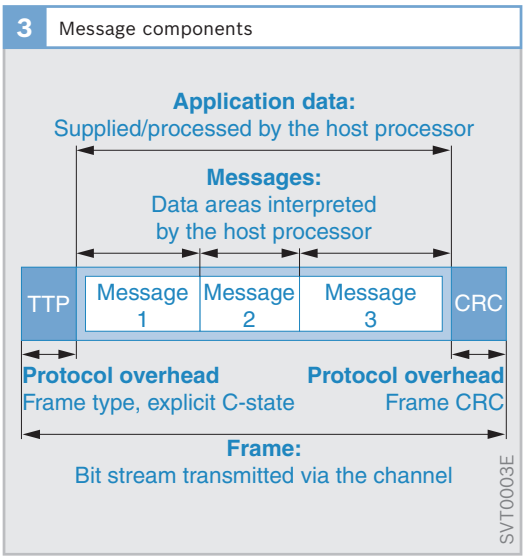
The primary function of the bus guardian is to limit transmissions of a node to the defined time windows. For this purpose, it accesses the information of the MEDL in which the transmit slots of the node are stored.

There are two different types of bus guardians. Local bus guardians are connected between the controller and the TTP/C bus and prevent bus access outside defined transmit slots. Central bus guardians operate in a central coupler. If exclusively central bus guardians are used, however, only faults are excluded for the nodes which are directly connected to the couplers. Connected bus topologies, on the other hand, cannot be completely protected.

2 TTP/C node



3 Message components



In addition to monitoring bus access, the central bus guardian can also assume the function of detecting and containing SOS faults (Slightly-Off Specification). These are faults in which a node continually violates slightly one or more communication parameters, such as the voltages used or the timing of the transmit slots. The upshot of this is that some network nodes which were designed with greater tolerances process these messages, but other reject them as faulty. As a result of SOS faults, it becomes problematic to establish whether data have been correctly transmitted or incorrectly received.

Logical Line Interface

The Logical Line Interface (LLI) is the interface between the controller and the transceiver (driver). The LLI establishes the structure of the logical information in which the driver must accept information to be transmitted and in which it must provide received information. This makes it easier to adapt controllers to a new transmission medium since only the drivers have to be modified; however, the format of the interface to the rest of the system remains unchanged.

Driver

The driver is the transceiver for the transmission medium. It converts the physical states of the TTP/C bus into logic states, which can be processed by the controller.

Network

Network Size

TTP/C is designed for networks of up to 64 nodes. A TTP/C network can consist of a minimum of two nodes, but at least four nodes are required to satisfy the Single Fault Hypothesis. A problem in the data transfer lies in the danger that a transfer is not received uniformly by all the network nodes. If there are at least four real member nodes (nodes with sole access to one transmit slot) in a TTP/C network, TTP/C can identify reliably whether the message transmission was successful or whether faults occurred during transmission or reception.

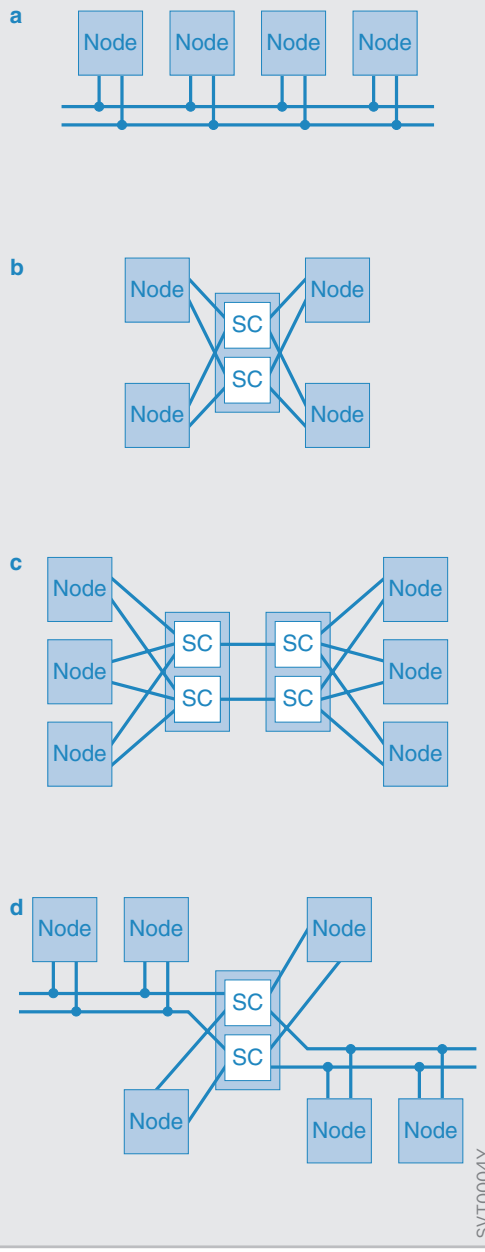
Topology

All TTP/C networks have two independent physical channels via which a network node simultaneously transmits information redundantly. This configuration is required in order that the Single Fault Hypothesis, i.e. the handling of the failure of any component, can be satisfied. However, it is also possible for nodes which do not fulfill any safety-relevant functions to be connected only to one of the two channels.

Since a controller always forwards messages via the LLI to its drivers, it is possible to adapt TTP/C to very different topologies. Bus, star and hybrid topologies from any combinations of bus and star topologies are described (Fig. 4) in the TTP/C Specification (Version 1.1).

Bus topology
In a bus, all the network nodes are connected in series to central cables. If a bus topology is used, the structure dictates that no central bus guardians can be used. However, the overall length of the cabling can be reduced to a minimum for this purpose.

4 Network topologies



Star topology
If a star topology is used, the network nodes are connected to each other via central couplers. Here, each TTP/C channel is brought together in its own coupler.

Star topologies facilitate the use of central bus guardians, which are able to provide additional protection against faults.

Hybrid topologies of bus and star
Even hybrid technologies composed of bus and star topologies can be built. For example, it is possible to cascade several star topologies into a multi-star or to connect a sub-bus to a star topology.

Transmission Media

Within a TTP/C network, communication takes place via two channels. The channels are designated Channel 0 (Ch0) and Channel 1 (Ch1), or Channel A (ChA) and Channel B (ChB).

Because the TTP/C Specification is formulated very openly with regard to the transmission medium, it is possible to build TTP/C networks using very different media. Both electrical and optical media can be used here. In particular, it is also possible to design one channel to be optical and the other to be electrical. Differences in the latencies of the different media can be compensated for by way of the configuration in the MEDL.

The achievable bit rate also varies together with this flexibility; however, the bit rate itself is fixed during operation within a network and cannot be altered.

It is possible with current implementations to achieve bit rates of 5 Mbit/s for asynchronous data transfer during system starting and of up to 25 Mbit/s for synchronous data transfer during normal operation. Prototypes in the laboratory environment has achieved bit rates of up to 1 Gbit/s.

Fig. 4
a Bus
b Star
c Multi-star
d Star/bus combination

SC Coupler

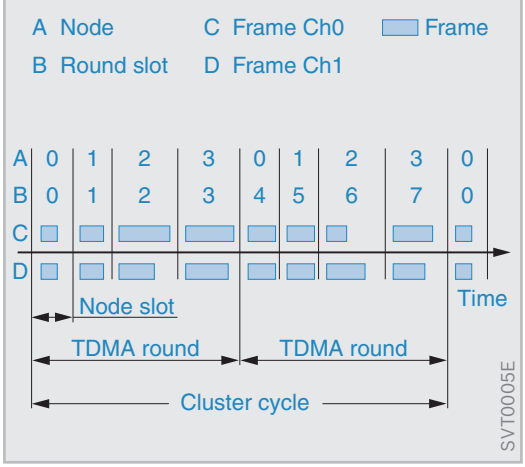
Communication cycle

TTP/C controls bus access by means of a TDMA process (Time Division Multiple Access, Fig. 5). Here, the available bandwidth is divided into time windows (slots) in which in each case an established network node can utilize the network's full bandwidth.

TTP/C distinguishes between rounds and cycles in the sequence of slots. All rounds contain the same sequence of slots whose size can be independently established. A cycle is a defined sequence of rounds in which the individual rounds differ only in the content of the individual slots, but not in the sequence of size of the slots.

In round, each member node of the cluster transmits one frame precisely in one slot. A distinction is made here between two types of member node: real and virtual. Real member nodes are nodes which have their own transmit slot which only this node can access. Virtual member nodes consist of a group of nodes which use a common transmit slot. This structure is used to build up redundant structures (Fault Tolerant Unit, FTU) which make the failure of nodes more tolerable.

5 Schematic bus-access diagram



FTUs are used so that safety-relevant information can also be provided in the event of the failure or one or more network nodes. Here, all the nodes combined in a FTU are able to process a task and use in a cycle alternately the transmit slot of a round allocated to the FTU.

Passive Nodes

A node transfers data only when current data are available for transfer, otherwise it remains silent. A node which does not transfer any data in its transmit slot is a passive node. This is the standard behavior when the controller of the node establishes that its host has not updated the "Host Life Sign", by means of which an application signals that it is active. The controller then assumes that the application cannot generate any valid data and does not transmit.

This mechanism is also used for monitor nodes, which merely monitor the communication of the other nodes, but themselves do not influence the communication.

Time control

The time window of a slot can be subdivided using the performed activities into different phases (Fig. 6). Before a node can transmit information, it needs time in order to read in the transmission parameters or to prepare the transceivers. This phase is called the Pre-Send Phase (PSP). This is followed by the time interval in which information is actually transmitted via the network, the Transmission Phase (TP). After a message has been received, a node needs time in order to prepare the information. This takes pace in the Post-Receive Phase (PRP). Because the time needed for PSP and PRP is not fixed, the PRP is followed by the Idle Phase, a phase through which the length of a node slots is stretched to the time interval defined in the MEDL.

A TDMA slot is fixed at points in time which can be observed using the transmission. But because this is not applicable to all the activities of a controller during a slot, a distinction is made between node slot and TDMA slot.

For a node, the slot begins with the PSP. But because no signal is transmitted in this phase, external observers lack a precise point in time at which the node slot begins.

The first precise point in time which can be observed from an external source is the start of the TP. This point is also called the Action Time (AT). A TDMA slot therefore comprises the time interval which passes between two ATs. Therefore, there exists in a TDMA slot only the TP and the Interframe Gap (IFG), in which the PRP, Idle Phase and PSP are combined.

A node slot, on the other hand, begins with the PSP, which is followed by the TP, the PRP and the Idle Phase.

TTP/C Protocol

TTP/C systems use the TTA framework (Time-Triggered Architecture) as the basis for which – with the exception of the application – finished solutions can be used.

The TTA protocol stack (Fig. 7) can be divided hierarchically into three areas. The top layer is the host layer, the executed application. Under this is the FT-COM layer, also an application, which is processed on the host processor.

The lower layers are implemented in the controller, where data communication takes place between the host processor and the controller via the CNI interface.

In the controller, the top layer is the Protocol Service Layer; under this is the Data Link Layer followed at the lowest level by the Physical Layer.

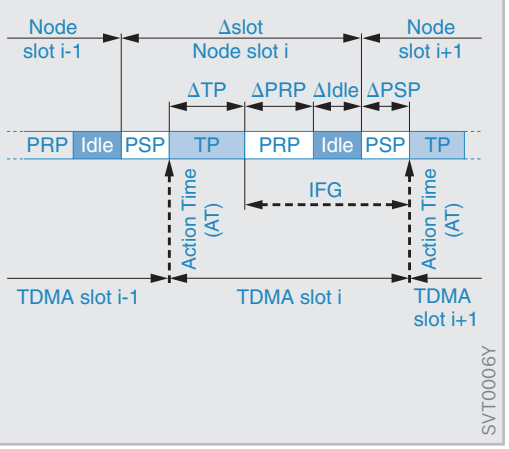
Physical Layer

The Physical Layer comprises the requirements which are placed on the physical transmission of information. Often the connections to be used, the cable types or the voltage levels are established for this purpose in a protocol specification.

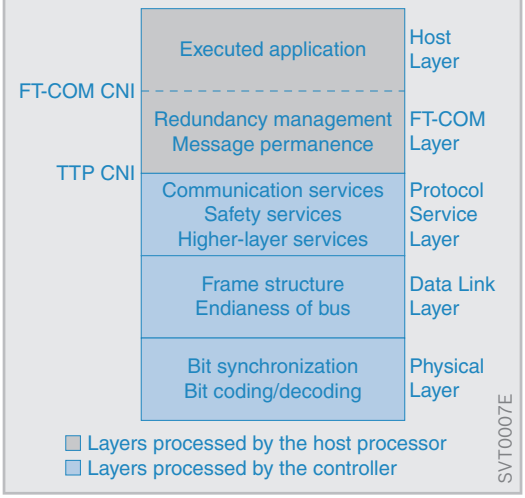
Efforts are indeed underway to formulate a specification for a TTP/C physical layer, but the TTP/C protocol specification merely establishes requirements which must be fulfilled by the physical layer used. This has the advantage that the protocol is very flexible with regard to the networking and bit rate. It is thus possible, for example, for networking to use both optical and electrical media, which can also be combined with each other.

However, the system designer is faced with the problem of initially having to identify and evaluate possible solutions.

6 Slot timing



7 Protocol stack



Essentially, three requirement are placed on the physical layer:

- Two independent physical transmission channels must exist.
- The transmission channel must permit distributed broadcast transmissions, i.e. transmissions which are received by all the network users.
- The delay which occurs during the transmission of information via a medium (Propagation Delay) must be known.

Data Link Layer

The Data Link Layer contains the functions which are needed to transmit frames between nodes. These include the structures which are required for access to the transmission channels and for the transmission of frames. In particular, the Data Link Layer establishes the format of the message frames.

Protocol Service Layer

The Protocol Service Layer is responsible for implementing the protocol services offered. Here, a distinction is made between communication services, safety services and higher-level services.

The functions of the communication services are the safeguarding of reliable data transfer, startup of a cluster, reintegration of member nodes in the cluster, transmission acknowledgment, and fault-tolerant clock synchronization.

The safety services include the management of node membership, avoidance of cliques, and the independent bus guardian.

The higher-level services include functions for delayed mode changes, distributed alarm, external clock synchronization, or reconfiguration of a node.

FT-COM Layer

The FT-COM Layer offers functions for realizing replicated, distributed applications. These include redundancy control and the decision as to whether replicated messages of other nodes are correct.

In order to allow applications standardized access to communication data, the FT-COM Layer offers an interface similar to CNI in which the data for the Host Layer applications are provided.

Host Layer

The Host Layer comprises the applications which run on the host processor. In addition to the actual applications, these include the operating system and the check structures (control loops).

In order to give the communication system as modular a structure as possible, the applications of this layer should access the FT-COM interface and not the CNI. This makes it possible, for example, for an application which previously ran on a real member node to be switched with minimal complexity to a series of virtual member nodes.

Protocol services

Cluster Startup

All the nodes of a cluster act without synchronization during cluster startup; bus access is therefore not yet controlled via TDMA. Because there is always a transmission delay caused by the medium used when messages are distributed in a network, two unsynchronized nodes of a cluster can start simultaneously with the message transmission if the transmission delay is greater than the duration of the startframe. This can occur during system startup if several nodes are authorized to initiate synchronization (coldstart nodes).

The situation can arise where different cliques arise within one cluster of which only the nodes of one clique are synchronized, but not all the nodes of the cluster.

In order to avoid this, all the nodes reject the first correctly received coldstart frame. The coldstart nodes wait for a time interval (startup timeout) before trying again to transmit a coldstart frame. Because the size of the startup timeout must be different for all the coldstart nodes in the cluster and differ at least by the size of the transmission delay, collisions can no longer occur during the second cluster startup. This process is also known as "Big Bang".

Synchronization

In a communication system which controls bus access by means of TDMA, it is important for the individual users to have as precise an idea as possible of a global time.

Synchronization in a TTP/C system uses for the operation time only the offset correction by means of which the onset of a point in time is synchronized to all the internal clocks of the nodes of a cluster. For this purpose, TTP/C uses a variant of the Welch-Lynch algorithm [1]) which was described by Kopetz and Ochsenreiter [2]).

The internal clocks of the network nodes of a cluster can run with different levels of precision. Only nodes which have precise clocks should be used for time synchronization. These nodes receive the flag as "Master Clocks" in the MEDL, whereby the SYF (Synchronization Frame Flag) is set for their frames. Nodes with more imprecise internal clocks act as "Slave Clocks" and use the clock information of the master clocks. There must be at least four master clocks in a cluster to ensure that clock synchronization is Byzantine-fault-tolerant. Byzantine faults are faults in which different nodes interpret the same transmission differently.

The nodes of a cluster are synchronized by measuring the Action Time (AT) in each transmitted slot. If the SYF flag of a frame is set and the received frame was faultlessly received, the AT of the frame is stored in a stack four deep. In other words, the last four correction values determined are stored in the stack.

The actual clock synchronization takes place in frames, for which the ClkSyn flag (Clock Synchronization) is set in the MEDL. If this is the case, the mean value is formed from the stack values, where the highest and lowest values are ignored. The internal clock is now corrected by the calculated mean value.

TTP/C also supports the inclusion of external reference clocks (e.g. GPS) in the system. Time Gateway Hosts are nodes of this type which offer an external time reference. These nodes transmit the external correction value as application data in a frame. The remaining nodes of the cluster store this value in the CNI and include it in the calculation of the AT correction values.

[1]) Jennifer Lundelius-Welch, Nancy A. Lynch

[2]) Hermann Kopetz, Wilhelm Ochsenreiter: "Clock Synchronization in Distributed Real-Time Systems", IEEE Transactions on Computers, 36(8): 933-940, August 1987.

Membership Service

TTP/C uses Membership Service to ensure that all the nodes of a cluster at each point in time have the same consistent view as to which nodes are present in the network and which are not. For this purpose, each node carries a member vector in which it notes whether the last received transmission of a node has run correctly. If a node receives a message incorrectly, it changes the entry of the message sender to "incorrect".

By comparing the member vectors of the following frames, a calibration is made with the remaining nodes of the cluster.

Clique Avoidance

In order to achieve correspondence within the cluster with regard to the current state of the cluster, it is necessary to prevent the formation within the cluster of subgroups or cliques whose view of the current state differs from the real state of the cluster.

To prevent the formation of cliques, each node logs how many transmissions were successfully and how many were incorrect.

Before it can transmit a frame itself, the Clique Avoidance Algorithm is used. The controller checks whether more successful transmissions were received than incorrect transmissions. Only if this is the case does the node set both counters to zero and transmit its frame. If this is not the case, the controller switches to the "Freeze" state, in which the controller is deactivated and no data are transmitted. From this state, it can be placed by the host processor in the initialization mode again.

Transmission acknowledgment

TTP/C offers an implicit transmission acknowledgment. Because the member vector of the next sender in the round is also transmitted as part of the frame, the original sender can check whether its transmission was correctly received.

But because the successor may be faulty, the sender also observes the transmission of the second successor. If the latter identifies in its member vector that it has also received the transmission incorrectly, this means for the sender that a transmit fault has occurred at its end. If the second successor has received the message correctly, the first successor detects that a receive fault has occurred.

Measures can now be taken to handle both fault types.

Controller State (C-State)

In order for the applications executed on the host processors to be able to deliver consistent results, it is necessary that all the nodes of a cluster share common knowledge of the state of the cluster. This knowledge is summarized in the C-state.

Each node may indeed determine its C-state independently, but the cluster agrees by comparison with the other nodes to the view of the majority of the users.

The controller state is influenced by the values for global time, current position in the TDMA round (MEDL position), current cluster mode, requested mode change (DMC), and the member vector.

Cluster Mode
In many real-time systems, operation and check functions can be divided into phases which are mutually exclusive. For example, the operation of a motor vehicle could be divided into the phases "Driving" and "Stationary". In order to support the different requirements placed on the exchange of information in the different phases, it is possible to define in a TTP/C network modes in which the transmitted information in a frame can vary. However, the order of the slot in a TDMA round is the same in all modes.

For the purpose of correct interpretation, it is therefore necessary when modes are used for all the nodes of a cluster to be in the same mode.

Deferred Pending Mode Change (DMC)
A change of cluster mode can be initiated by applications if the controller is permitted in the MEDL to change to the requested new mode.

Because the change of cluster mode comes into force only at the beginning of the following cycle, the information for the deferred pending mode change is recorded in the C-state.

If another change of cluster mode is requested by another application later in the cluster cycle, this request overwrites the previous request. A special case is CPM (Clear Pending Mode change), in which an application prevents the requested change of cluster mode.

Reintegration of a node

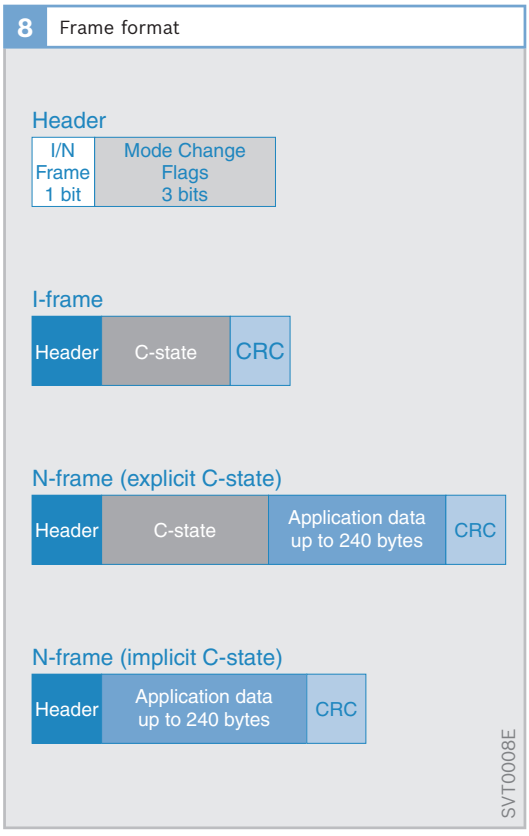
A node must synchronize itself with the cluster if it has failed due to an incorrect transmission from the member vector or if it has been re-initialized. To this end, it waits for a frame in which an explicit C-state was transmitted, e.g. an initialization frame. The controller adopts this C-state and uses it for its transmission.

In order for this to be able to function, a frame with an explicit C-state must be transmitted every two rounds. If this is not the case, the controller assumes that there has been a communication fault and begins with a cluster startup.

Frame format

Because TTP/C regulates bus access via TDMA, all messages are transmitted in broadcast mode and a priori knowledge of the communication sequence and its content is present in the MEDL, it is possible to keep very low the overhead of the protocol, i.e. the volume of information needed to process the frame.

Two different frame types are used in the operation of a cluster (Fig. 8). Cold-start Frames, also known as Initialization Frames or I-Frames, occur exclusively during system startup. Normal Frames, also known as N-Frames, are used for communication during normal operation.



I-Frame

An I-frame begins with the TTP/C header. This is followed by the current controller C-state and finally the CRC value of the frame.

N-Frame

To ensure that Membership Service can function, it is necessary for the C-state to be included in the frame structure. This can occur explicitly or implicitly in the case of N-frames.

If the C-State is explicitly transmitted, the header is followed by the controller C-state. This is followed by application data and then the CRC value of the frame.

If the C-State is implicitly included in the frame, the frame consists of the header, application data and finally the CRC value of the frame, the determination of which was initialized with the current C-state.

Download frames are a special form of the N-frame; these are accepted by all the receiving nodes, regardless of cluster association and C-state. To facilitate this behavior, a constant value which is known to all the nodes is used for these frames to determine the CRC.

Frame Fields

Header

The header of all the TTP/C frames is identical. Firstly, one bit signals the frame type, whether an I-frame or an N-frame is transmitted.

This is followed by three bits, by means of which a change of current cluster mode can be requested. These bits are set to zero if no change has been requested.

Application data

The information of the application is transmitted in this area. The size of this field can be determined for each slot of a round individually in the MEDL and must not exceed 240 bytes per slot.

C-state

The C-state consists of the global time, information on the MEDL and the member vector of the cluster. The MEDL position corresponding to the Round Slot Position in the current cluster mode, the current cluster mode and the requested cluster mode change (DMC) is transmitted as information on the MEDL.

An explicit C-state has a size of 96 bits (six words of two bytes each).

CRC checksum

TTP/C uses the CRC checksum only to identify transmission faults. Transmission faults are not corrected. Transmission faults are detected by a process whereby each node determines a CRC checksum for the received message and compares this with the transmitted checksum.

A polynomial is used to determine the checksum. The TTP/C Specification in this respect does not stipulate a specific polynomial; instead, it requires the polynomial to facilitate a Hamming distance of at least 6. In other words, it must be possible within one transmission for at least five incorrectly transmitted bits to be detected.

The CRC calculation is initialized by the Schedule ID, where the ID is split into two parts. One part is used for initialization for Channel 0 and the other for Channel 1. This prevents a node which does not have the correct MEDL or which was connected crosswise to the channels from transmitting or receiving successfully.

The C-state for initializing the CRC calculation is also used in the transmission of frames with an implicit C-state. Thus, only those frames whose CRC checksum was determined with the same C-state are accepted as valid.

The calculated CRC value is transmitted at the end of a frame in a 3-byte field.

Composability

Networking technology has changed greatly over the years. Where initially stand-alone systems (e.g. activation of a turn-signal lamp via switches and relays) were used, the changeover was quickly made to cooperative systems in which individual components exchange information with each other.

However, the interaction of components is increasingly making it difficult to ensure the correct fulfillment of a function. For this reason, the aspect of composability, i.e. the possibility of combining separately developed subsystems into an overall system without having to verify the function of the overall system again, is becoming increasingly important.

In order to ensure that this functions, it is necessary to ensure that a subsystem autonomously fulfills its functions, but also that this is the case when the subsystem is integrated in an overall system.

TTP/C facilitates the implementation of composability through the communication parameters which are stored in the MEDL.

Verification of the Specification

The functionality of TTP/C has been thoroughly analyzed. The core algorithms of the protocol specification have been formally verified with regard to certain aspects of consistency, stability and safety.

Furthermore, the function of TTP/C systems has been tested using failure-injection experiments. This involved the use of both physical and software faults to check the fault tolerance and fault-detection properties.

Standardization

The development of TTA stretches back over a period of 20 years. Today, development of the Specification is coordinated by the TTAgroup, a cross-industry consortium which was founded by the companies Airbus, Audi, Delphi, Honeywell, PSA Peugeot Citroën, Renault, and TTTech.

The Specification of TTP/C has been influenced in particular by Hermann Kopetz, who between 1979 and 1982 was Professor of Computer Process Control at the Technical University of Berlin and since 1982 has run the Institute for Technical Information Technology at the Technical University of Vienna. Professor Kopetz also directed the MARS project and was substantially involved in each further development of the protocol.

Characteristics

- Support of communication for safety-relevant functions.
- Guaranteed transmission properties.
- Guaranteed detection of all single faults.
- Fault-tolerant communication units (FTU) for fault-tolerant provision of data.
- Local and central bus guardians possible.
- Low protocol overhead.
- Data efficiency of 85 % for transmission.
- Supports composability with regard to time response and ranges of values.
- Functions for the most part formally verified.
- Consistent view by all nodes of the cluster state thanks to Membership Service and Clique Avoidance Algorithm.
- Clear structuring of the protocol interfaces.
- No restriction of the bit rate from the Specification. TTP/C components were tested with a bit rate of up to 1 Gbit/s.
- Experience from product use.

FlexRay

Overview

FlexRay is a field that was designed to support open and closed-loop control technologies in the automotive sector. Development focused on the suitability for use in active safety systems in particular. FlexRay therefore offers transfers with guaranteed compliance with transfer properties, high bit rates and a fault-tolerant design.

The development of FlexRay, managed by the FlexRay Consortium, goes back to cooperation between German automotive manufacturers, BMW and DaimlerChrysler. In 1999 these two companies began to compile requirements for a new communication system. In the process, development would incorporate experience gained from the byteflight system of BMW and the prototype development of DaimlerChrysler.

The aim of FlexRay is to provide a system with high transfer rates that will work in a deterministic and fault-tolerant manner while being as flexible as possible to use and expand. The main fields of application of the FlexRay are drivetrain systems (drive) and active safety systems with no mechanical fall-back level (x-by-wire). However, the areas of passive safety systems and comfort/convenience and body electronics are also supported.

To support these very different domains, FlexRay uses two different types of bus access. For deterministic transfer properties, it offers time-controlled bus access. In the case of applications whose transfers place less demanding requirements on transfer properties, it is desirable to make as effective use of the available transfer capacity as possible. To combine these two approaches, communication takes place in cycles. In each cycle, there is firstly a static transmission component in which bus access is controlled by TDMA (Time Division Multiple Access). There then follows a dynamic transmission component in which bus access is controlled by the use of minislots. Minislots are small time windows in which a defined message can be transferred. This access method is also known as FTDMA (Flexible Time Division Multiple Access).

With messages transmitted in the static component of the cycle, the transfer properties can be assured in accordance with known methods. In the dynamic component, messages are sent as required. In this case, messages are prioritized based on the message ID. The division between the static and dynamic component is freely configurable but it cannot be modified later during system operation. The same applies to the slot lengths in the static range, which, while configurable, must remain constant once operation is underway.

FlexRay systems can be equipped with two transmission channels, with each channel having separate lines. This makes it possible to create a redundant data transfer or increase the available bandwidth for specific applications. The latter can be used both for the parallel transmission of information of two network nodes and for the faster transfer of information of one node.

FlexRay operates at a maximum bit rate of 10 MBit/s, which is achievable in optimal environmental conditions. A rate of up to 20 MBit/s is achievable if two channels are used without redundant access. In addition to the bit rate of 10 MBit/s, which is the only bit rate defined in Specification 2.1 dated 15th December 2005, there are endeavors to support lower bit rates of 2.5 or 5 MBit/s.

Areas of application

The areas of application of the FlexRay are very diverse. By offering redundant, time-controlled and fault-tolerant transfers, not only is FlexRay suitable for use in active safety systems with no mechanical fall-back level and in the area of the drive-train, but its optional, dynamic transmission range means that it is able to support application in the areas of passive safety systems and networking in the body and comfort/convenience domains.

With the high bandwidth of up to 20 MBit/s for non-redundant transfers, it is even conceivable, in theory, that FlexRay could be used for the transfer of audio or highly compressed video.

Topology

Point-to-point

The simplest configuration of a FlexRay system is the direct link between two network nodes. The maximum distance between the two nodes is 24 m in this configuration.

Bus topology

Between 4 and 22 nodes can be connected in a bus topology. The maximum distance between any two nodes in a bus topology may be 24 m.

Star topology

A star topology is another architectural variant of the FlexRay network. Both active and passive couplers are supported.

Passive star topology

3 to 22 nodes can be connected in a passive star. Here, too, the distance between any two nodes cannot be greater than 24 m.

Active star topology

An active star topology can be viewed as a point-to-point link between a number of nodes and an active coupler. In this topology, therefore, only the distance from a node to the coupler is of importance. This cannot be more than 24 m.

Apart from the connection lengths and the number of connectable nodes, the most important difference between an active and a passive star topology is in the achievable bit rates. With the use of passive couplers, the achievable data rate (approx. 1 MBit/s) is below the rate that can be achieved with active couplers (up to 10 MBit/s).

Cascaded star topologies

Cascaded star topologies are the product of several active star topologies linked together. In a FlexRay system, a topology like this is limited to three cascaded star topologies.

Hybrid topologies

Hybrid topologies are a blend of topologies from bus and star topologies, e.g. the connection of several bus topologies to a star topology instead of individual nodes.

Two-channel topologies

Since it is possible to implement both channels of a FlexRay system independently, different topologies can be used for each channel. For example, one channel could be realized as an active star topology, the other as a bus topology.

Hardware

FlexRay controller of a subscriber

A FlexRay node (Fig. 1) comprises the host processor, a communication controller (CC) and one bus driver (BD) for each channel to which the node is connected. Optionally, each node may contain a bus guardian (BG) for monitoring each bus driver and the bus guardian may receive information through an additional host processor.

It is optional whether a network node is connected to one or both channels; here, at the very most, the requirements from the field of application of the network node take effect.

Transmission agents

The transmission agent for a FlexRay system is of a twisted-pair cabling design, where shielded (STP) or unshielded (UTP) cabling may be used.

Each of the two FlexRay channels consists of two wires designated bus plus (BP) and bus minus (BM).

Host processor

The host processor gathers information from the sensors, which is forwarded to the communication controller for transmission. The information received by the communication controller is forwarded to the actuators responsible for processing.

Communication controller

The communication controller (CC) realizes all aspects of the FlexRay system relating to the protocol. Its tasks include scheduling, synchronizing with other network nodes, creating a macrotick signal, creating a bit stream from the information of the host or controlling bus access.

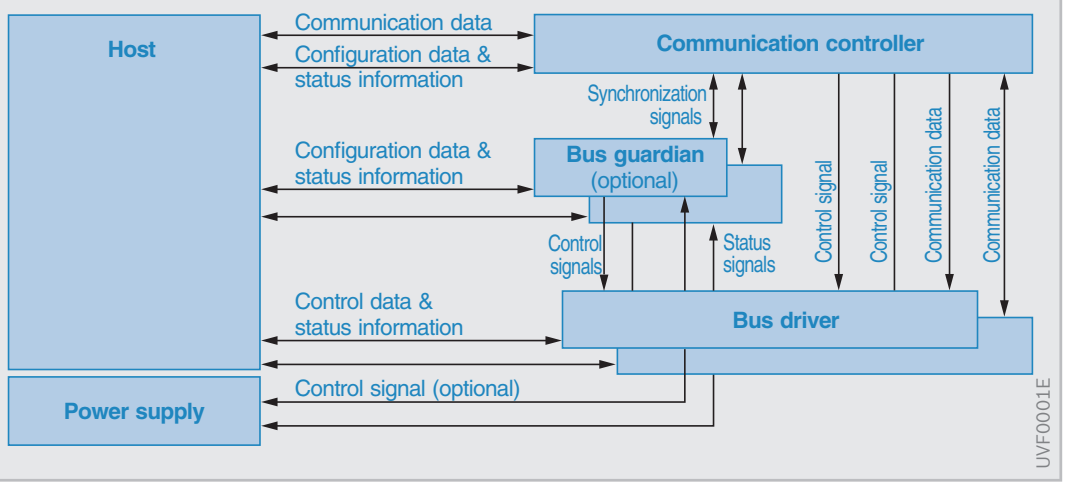
Bus driver

The bus driver (BD) assumes the role of a transceiver. It converts logical information into physical voltages that are carried on copper wires, and vice versa. The role of the BD also includes a protective function against electrostatic discharge (ESD).

Furthermore, the BD monitors the states of the BP and BM and is thereby able to detect whether the bus is affected by physical faults.

The BD also processes wakeup signals and, optionally, is able to control the power management of the network node because functions of the node can be switched off.

1 Network node architecture



Bus guardian

The bus guardian (BG) is a device that will only permit transmissions if the connected network node is authorized to send a message. With this system, it is theoretically possible for faulty network nodes to interfere at the very most with their own transmission but not with the transmissions of other network nodes.

The FlexRay protocol only provides the framework for use of a BG: its use is not a mandatory specification. Specification 2.0 only described a local bus guardian that would operate inside the network node and monitor access to one of the channels. A critical aspect of this concept was that correct operation of the bus guardian in a faulty network node could not be guaranteed as a consequence of the integration of the bus guardian in the node and the permission for the bus guardian to access the oscillator of the network node.

Specification 2.1 contains a refined concept for the local bus guardian and introduces the concept of a central bus guardian.

Local bus guardian

The local bus guardian (node-local bus guardian) is implemented as a stand-alone CC inside the network node that monitors operation of a CC (Fig. 2). Its synchronization process is independent of that of the CC's; in particular, the CC and BG have separate oscillators.

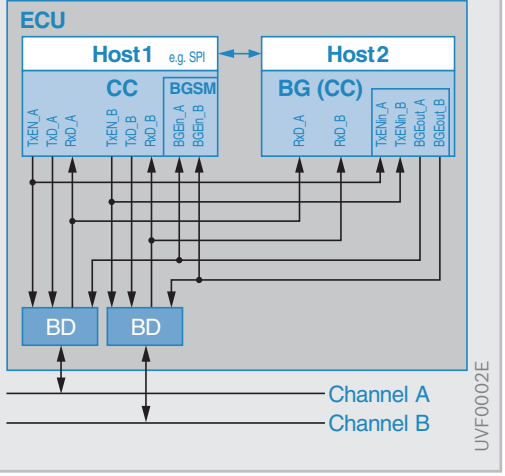
In addition, it is also possible for the BG and CC to be controlled by dedicated host processors that communicate with each other. This is, however, not mandatory.

As a consequence of direct control lines (bus-guardian enable, BGE) between the BG, CC and BD, information can only be transferred from the CC to the BD if the BG permits this transfer to take place.

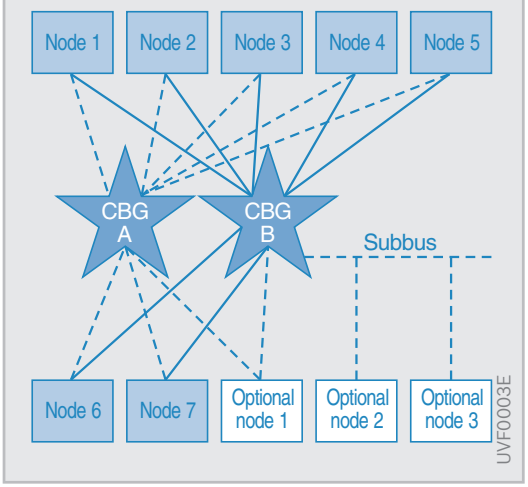
Central bus guardian

The central bus guardian (CBG, Fig. 3) was a new feature introduced with Specification 2.1. It operates in the coupler of a star topology and monitors the operation of all connected nodes or networks. Faults, such as short-circuits on buses or network nodes, that transmit information like a "babbling idiot" with no consideration of the transmission window, and thus interfere with orderly communication, can be detected by a CBG and suitable counter-

2 Local bus guardian



3 Central bus guardian



measures may then be taken. As a result, the effects of these faults can be confined to small areas of the network.

Operating modes

In addition to normal operation, a FlexRay system also supports the complete initialization of the network from sleep to standby (Fig. 4). In order for this system start to function, the BDs of all network nodes have the capability to detect wakeup signals on the bus and to start the remaining other components of the dedicated network node.

Special network nodes that may be connected to the starter motor of a vehicle, for example, generate the wakeup signal.

The system start is split into the wakeup and startup phases. The wakeup phase involves the activation of the network nodes by the wakeup signal. In the startup phase, the network nodes are initialized and synchronized.

If the function of a node is not required for the time being, the node can be set to a standby mode in which all operations of the coding and decoding process are stopped. This reduces the energy consumption of the node. In sleep mode, the power consumption of a network node is at its minimum. In this mode, all functions of the network node are deactivated: only the bus driver looks out for the transmission of a wakeup signal, in response to which the node is set to wakeup mode.

Sending and receiving

A bus driver connects a FlexRay network node to the channel that contains a receiver and transmitter. If a controller is connected to both channels, it follows that there must be two receivers and transmitters available in the network node.

Protocol

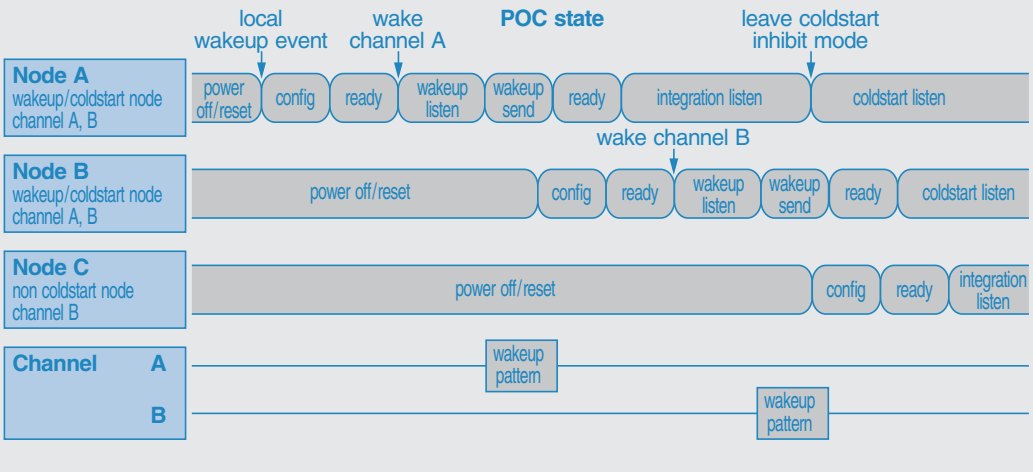
Protocol layers

The FlexRay protocol is built on five core mechanisms:

- Coding and decoding
- Control of bus access (media access control, MAC)
- Processing of frames and symbols
- Clock synchronization and
- Schedule monitoring by the bus guardian

Unlike in other network protocols, with FlexRay there are interfaces between all core mechanisms. This requires a process that coordinates and synchronizes the changes in the core mechanisms. This is the task of the protocol operation control (POC).

4 Wakeup sequence



FlexRay gives the host system the capability of influencing all five core mechanisms directly. To arrange this, communication takes place through an interface – the controller host interface (CHI) – between the network node and host.

Concentrating on the main interfaces, the task areas can be arranged as a protocol stack (Fig. 5). At the top level is the application, which forwards its commands to the CHI. Under the CHI is the POC, which in turn has access to the MAC, clock synchronization and frame & symbol processing. In the next lower layer are the processes for coding and decoding. The lowermost level is represented by the transfer characteristics of the physical layer.

The bottom three layers are in nodes that are connected to both channels: double presence because these functions have to be fulfilled for each channel separately. More precisely speaking, this means that for one node connected to both FlexRay channels there is double availability of synchronization, MAC, frame & symbol processing, coding and decoding, and physical layer.

Coding on the physical layer

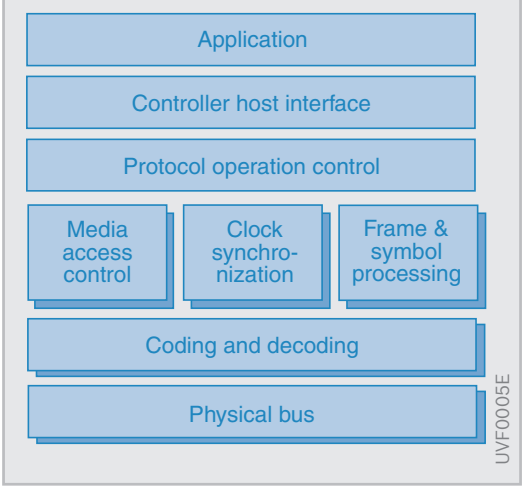
For coding, FlexRay uses a NRZ method (Non-Return to Zero) in which the two identical transmission states are not divided by a range in which the voltage falls back to a zero value. This type of coding makes it necessary to have mechanisms with which a network node can divide the transmitted states. To this end, FlexRay adds a sequence behind each transmitted byte: the byte start sequence (BSS). From this sequence, each node can detect when a byte has been transmitted and, with this information, encode the individual bits of the transmission.

By applying different voltages to the two wires of one channel, it is possible to create four bus states, which are designated Idle_LP, Idle, Data_0 and Data_1 (Fig. 6).

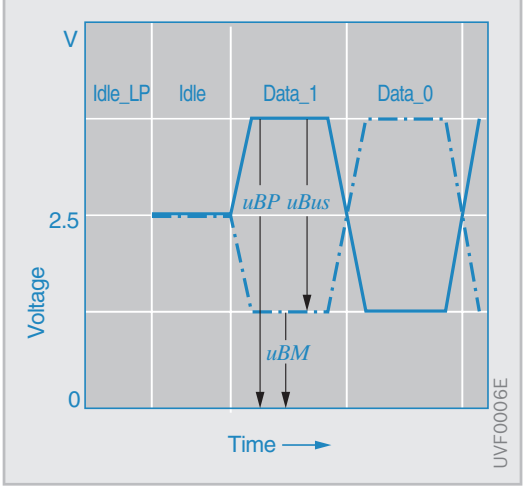
A bus state is identified by measuring the differential voltage. Here, the bus voltage (*uBus*) is made equal to the difference between *uBM* and *uBP* (voltages at BP and BM).

$$uBus = uBP - uBM$$

5 Protocol stack



6 Bus states and voltages



With this method, the data transfer is protected against external electromagnetic interference because these act equally on both wires and are canceled out in the difference.

Idle_LP (LP = low power) is the state in which a very low voltage of -200 mV and 200 mV is present at BP and BM. This state, for example, is used to identify the start of a transmission.

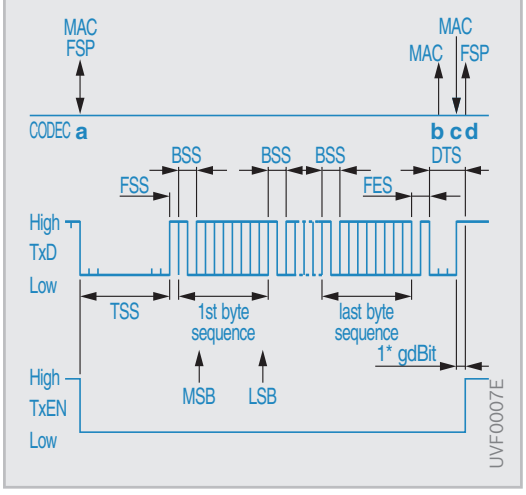
In Idle state, a voltage of 2.5 V with a tolerance of 500 mV is present at BP and BM.

To set the channel to Data_0 state, at least one transmitting node must apply a negative differential voltage of -600 mV to the channel.

For the channel to be set to Data_1 state, at least one transmitting node must apply a positive differential voltage of 600 mV to the channel.

If the transmission of an information signal is neither blocked by the bus guardian nor the communication controller, a HIGH bit is signaled by the Data_1 channel state and a LOW bit by Data_0.

7 Bit stream (dynamic segment)



Generation of a frame bit stream

Before a node can transmit a frame containing the data of the host, the frame is converted into a bit stream. To this end, the frame is first decomposed into individual bytes. The start of a frame is populated with a transmission start sequence (TSS) followed by a frame start sequence (FSS) (Fig. 7). From the bytes of the frames, an extended byte sequence (EBS) is then generated whereby each frame byte is preceded with a byte start sequence (BSS).

The 24-bit checksum (CRC) for this bit sequence (TSS+FSS+EBS) is now calculated and appended to the bit sequence. To finish the bit stream, a frame end sequence (FES) is appended.

If the frame belongs to the dynamic segment, a dynamic trailing sequence (DTS) can be additionally appended to prevent another node from beginning its transmission on the channel prematurely.

Communication cycle

Each FlexRay cycle (Fig. 8) contains a static segment that is transmitted as the first part of the cycle. The static segment contains a fixed number of transmission ranges, the “static slots”.

Optionally, there may be a dynamic segment in the FlexRay cycle and this is transferred in second place. Each dynamic segment contains a fixed, freely configurable number of “minislots”.

Also optional is the symbol window, which is transmitted as the third element of the cycle. It can be used for the transmission of an individual symbol and has the same size as a static slot.

To terminate the cycle, there is a phase – network idle time (NIT) – in which the bus is set to idle state. In general, the length of the NIT corresponds to the remaining macroticks not used by the static and dynamic segment or symbol window. This is not the case if it was detected during synchronization that an offset correction is required through which the length of the NIT can be increased or reduced.

Static segment
The static segment comprises a fixed number of equally sized transmission windows (static slots), which are sent simultaneously on both channels. Exactly one frame can be sent in each static slot. Both the size and quantity of the static slots in a static segment are configured during integration.

Bus access in the static segment is controlled by a TDMA method whereby, in each slot, the frame with the corresponding frame ID is sent. Frame ID 1 is therefore sent in slot 1, frame ID 2 in slot 2 and thus frame ID n in slot n.

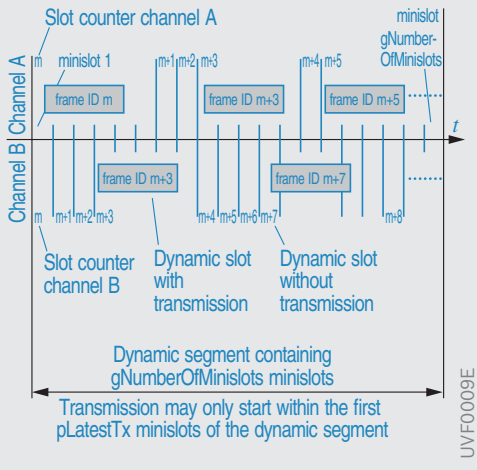
Dynamic segment
Unlike in the static segment, the transmissions in the dynamic segment are not of fixed length. To control bus access, "minislots" are used that hold the same fixed number of macroticks for all nodes connected to the network. Macroticks are ranges in which exactly one frame can be transmitted.

"Dynamic slots" (transmission windows of variable size) are now created based on the minislots. The maximum size of a dynamic slot is limited by a fixed number of minislots specified during configuration.

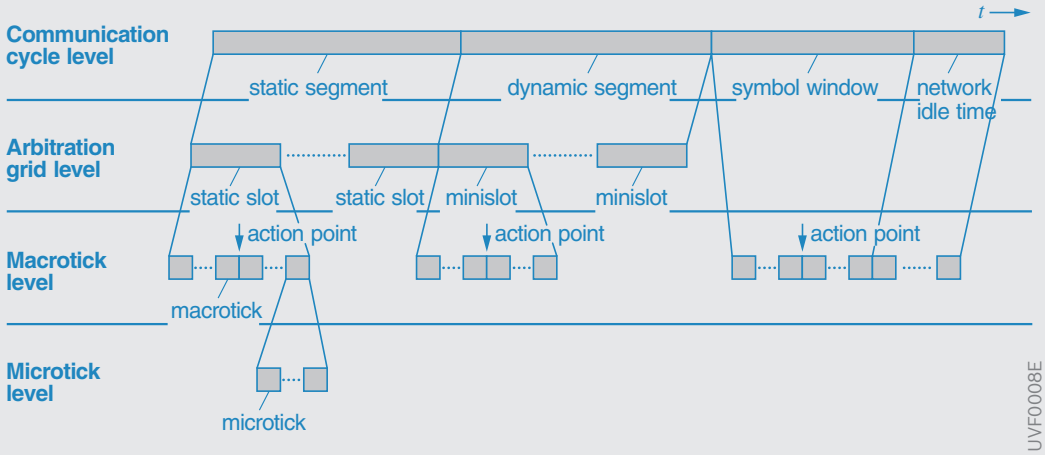
To identify the end of a transmission, a phase in which the channel is idle (dynamic slot idle phase) is added to the end of a dynamic slot. The phase in which the transmission takes place is called the dynamic slot transmission phase.

Another difference between the dynamic and static segment is that the counters of slots transferred are incremented synchronously on both channels in the dynamic segment. In the dynamic segment, the counter of both channels is incremented independently in line with the current transmission status (Fig. 9).

9 Slot counter



8 Communication cycle



Symbol window
The symbol window makes it possible to send a collision avoidance symbol (CAS) or a media access test symbol (MTS). These two symbols are identical and help to prevent collisions during the system-startup phase.

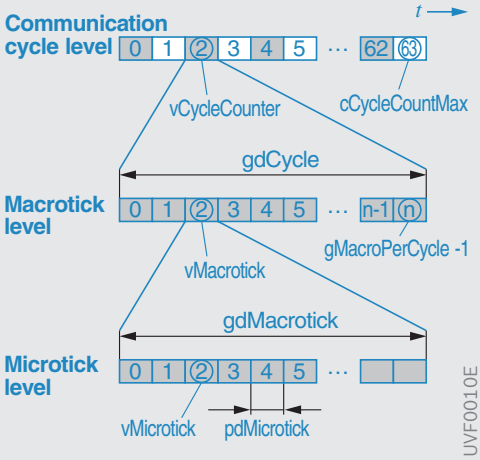
The third symbol defined in the FlexRay protocol – the wakeup symbol (WUS) – is not permitted to be sent in the symbol window. It serves only to generate a wakeup pattern (WUP), which is used during the system startup.

Network idle time (NIT)
During the NIT, a network node corrects the ascertained time deviations of its internal clock. It also gives the network node the opportunity in this phase to make implementation-dependent adjustments and changes to settings relating to the communication cycle.

Clock control

In both the static and the dynamic part of the communication cycle, FlexRay reverts to the identifiers of the transmission window in which a node may transmit. If this method is to be successful, it is necessary for all nodes connected to the network to have synchronous time information.

10 Timing



For this purpose, selected messages from the static part of the cycle are used (sync frames) and are transmitted by the network nodes that are connected to both channels.

In a distributed communication system, each network node generally needs its own internal clock for setting transmission timing correctly. As a consequence of temperature and voltage fluctuations and production-related precision tolerances, these internal clocks can soon deviate from an abstract, system-wide global time. In the case of a system such as FlexRay, which controls bus access by means of time slots, the synchronization of all network nodes plays an especially important role because, without a standardized, global time preset for all nodes, the specified transmission windows cannot be maintained and may break down.

Clock control hierarchy

In a FlexRay system, clocking is controlled on three levels (Fig. 10). The bottom level is represented by the microticks, which are derived directly from the oscillator clock of the network node. A microtick is therefore not a system-wide variable, but only relevant to the node concerned.

At the next level up, the clock is controlled by means of macroticks – ranges in which exactly one frame can be sent. Within fixed tolerances, macroticks form the lowermost system-wide time unit. Each network node determines the duration of a macrotick in microticks. The number of microticks per macrotick does not have to be the same for all macroticks.

The top level is characterized by a clock control based on the communication cycle, which always contains the same, fixed number of macroticks.

Time synchronization

A synchronization method is required in order to guarantee a standardized time across the entire system. The basic parameters for timing are the number of microticks per communication cycle (*pMicroPerCycle*), the duration of a communication cycle (*gdCycle*) and the duration of a microtick (*pdMicrotick*), where the relationship between each is determined by the following equation:

$$pMicroPerCycle = round(\frac{gdCycle}{pdMicrotick})$$

During operation of a FlexRay network, the clocks of the oscillators of various network nodes will not be equal or even simply remain constant. They will fluctuate in response to outside influences. This deviation is balanced by two corrective measures: the gradient correction and the inclusion of an offset in the communication cycle.

Fault-tolerant average algorithm
Since an individual node can only observe the differences between the expected time at which a transmission should have been actioned and the actual time at which the transmission took place, a fault-tolerant average algorithm (FTM), as described by J. L. Welch and N. A. Lynch, is used in both corrective measures. The measured time differences of all transmitted sync frames are sorted by their value. If more than seven sync frames were received, the two highest and two lowest values are removed from the list. If between three and seven sync frames were received, the highest and the lowest value are removed. For fewer than three sync frames, the average is derived from all measured differences.

Gradient correction
With gradient correction, deviations in the transmission frequency of a node are continuously corrected during the communication cycle. The node here checks the transmission frequencies of all other nodes. By means of the FTAA, these values are used to adjust the node's own transmission frequency.

Offset correction
Offset correction is a way to correct deviations in the communication-cycle phase by including an offset in the NIT. With this method, the node similarly employs the FTAA to determine its phase difference in relation to the other nodes of the network and uses it to determine the offset required to shorten or extend the NIT phase. Since all nodes use this approach, all nodes can initiate the transmission of a cycle in a synchronous manner.

Timing
Transmissions in the FlexRay system are timed by action points, which are specially-marked macrotick boundaries. A transmission in a static slot begins once a static-slot action point has been reached. The position of this action point in a macrotick is defined system-wide.

In a dynamic slot, a transmission begins once a minislot action point has been reached and ends at a subsequent minislot action point. This method ensures that there is always an idle period between individual transmissions.

Welch, Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronisation", Information and Computation, vol.77, No.1, April 1988

Frame format

FlexRay uses the same format in both the static and dynamic part of the frame. The format can be broken down into three parts: header, payload segment and trailer segment (Fig. 11).

Header segment

The header segment contains a total of five bytes, at the start of which is a series of indicators.

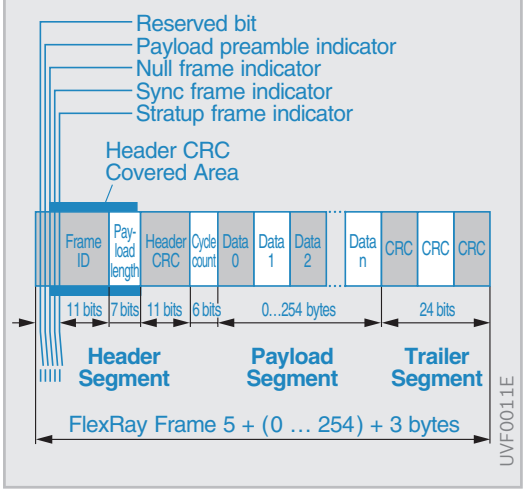
The reserved bit is intended for future modifications to the protocol and is sent as a logical “0”.

The payload preamble indicator indicates whether or not the payload segment contains a network-management vector (NMVector). An NMVector enables the host processor to transmit data directly without first being processed and prepared by the CC.

The null frame indicator indicates a null frame that contains no usable information.

The sync frame indicator indicates that this frame is intended to be used for system synchronization. This indicator may only be set by network nodes that operate as sync nodes.

11 Frame format



The startup frame indicator reveals that this frame is a startup frame. Startup frames are used in the network's starting phase and may only be sent by special coldstart nodes.

The indicators are followed by the frame ID, payload length, header CRC and cycle count.

The frame ID has a length of 11 bits and corresponds to the number of the slot in which the frame is sent. In one cycle, each frame ID exists only once on each channel.

The payload length expresses the size of the information sent in the payload segment. Since the payload in a FlexRay frame is always sent in 2-byte words, a 62 in the payload length field indicates, for example, that 124 bytes of payload are being transmitted in the payload segment.

In the static segment, the payload length field always contains the same value because the size of information here is constant for all frames. Of course, this property does not apply to frames in the dynamic segment.

In the header CRC, an 11-bit CRC value is transmitted. This is formed from the polynomial

$x^{11} + x^9 + x^8 + x^7 + x^2 + 1$.

Since the information in the header segment is constant for all frames in the static segment, the header CRC value can be calculated offline and assigned to the network node through configuration settings.

If a frame in the dynamic segment is always of the same size, it is possible here also for the CRC value to be calculated offline.

The final field in the header segment is the cycle count. This field contains the count number of the cycle in which the sending network node is currently active.

Payload segment

The payload segment has a maximum length of 254 bytes, which are transmitted in 2-byte words.

The payload segment customarily transfers the payload that is further processed by the host processors. Optionally, however, an NMVector may also be sent in the payload segment, or a 16-bit message ID.

With the assignment of message IDs, it is possible to send several data blocks in a frame. For this purpose, the message ID is placed at the front of the other data by the host processor as application data.

Trailer segment

The trailer segment contains a single field in which a 24-bit CRC checksum (frame CRC) is sent. The polynomial used to determine the frame CRC is:

$$x^{24} + x^{22} + x^{19} + x^{18} + x^{16} + x^{15} + x^{14} + x^{13} + x^{11} + x^{10} + x^{8} + x^{7} + x^{6} + x^{3} + x+1$$

Standardization

FlexRay Group

FlexRay is a protocol specified by the FlexRay Group. The FlexRay Group was founded in 2000 by the companies, BMW, DaimlerChrysler, Motorola and Philips Semiconductors (www.flexray-group.com).

The members of the FlexRay Group are organized in a hierarchy. They also have different levels of influence on the further development of FlexRay, depending on the obligations that they have accepted. The uppermost group is represented by the core partners, which first and foremost include the founding members. Alongside the core partners are the premium associates and associate memberships.

By 2004 General Motors, Bosch and Volkswagen had joined the FlexRay Group as core partners. In 2004 Freescale took over Motorola's position in the group of core partners.

Characteristics

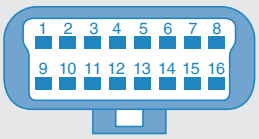
- Deterministic transfers possible
- Optional bus guardian monitors network access and protects against faults
- Differential signal transmission
- Collision-free transmission while operation is in progress
- Redundant transmission of information on two channels
- High transmission speed of up to 10 MBit/s; up to 20 MBit/s with parallel transmission on two channels
- Startup and initialization from an inactive sleep state are directly supported by the protocol
- Support for diverse fields of application
- Event and time-controlled transmission of information possible
- Published standardization by the FlexRay Group

Diagnosis interfaces

Diagnosis legislation (e.g. CARB, California Air Resources Board) demands constant monitoring of emission-related components in various electronic systems (OBD, On-Board Diagnosis). Faults (e.g. electrical short-circuits of sensors, implausible operating states) are stored in a fault memory in the control unit. These faults can be read using a scan tool (official testing station) or workshop tester. For this purpose, there is a socket in the footwell, dashboard or center console of the vehicle to which the tester is connected by means of a standard connector (ISO 15031-3) (Fig. 1).

Using a workshop tester, the after-sales service is able to read the entire fault memory: emission-related faults and vehicle-specific faults (e.g. from Motronic, ABS). With the information stored in the fault memory, it is possible to diagnose a fault and repair it efficiently. The tester can also be used to clear the fault memory. Furthermore, measured values and control-unit identification values can be evaluated. Using the workshop tester, it is also possible to control special diagnostic functions. With actuator diagnosis, for example, individual actuators can be targeted for activation to check them for correct operation. At the end of the line (EOL), the diagnosis interface is used to test the electronic control unit and make changes to configurations (e.g. immobilizer, transmission type). It is even possible to program the entire flash EPROM using end-of-line programming.

1 Diagnosis connector (diagnosis interface)



Pin 2: Bus line (high) from SAE J1850
Pin 10: Bus line (low) from SAE J1850
Pin 7: K line in accordance with ISO 9141-2 and ISO 14230-4
Pin 15: K line in accordance with ISO 9141-2 and ISO 14230-4
Pin 6: CAN_H from ISO 15765-4
Pin 14: CAN_L from ISO 15765-4
Pin 1, 3, 8, 9, 11, 12, 13 Not used by OBD
Pin 4: Vehicle ground
Pin 5: Signal ground
Pin 16: Battery positive

UWT0106-1E

Diagnostics requires an interface for connecting the tester to the electronic control units. For communication, there are basically two options available at the present time:

- K line and
- CAN bus

The K-line network is not a bus in the true sense, which means that collisions may occur. Since most electronic control units have a CAN interface for the exchange of data between electronic systems, this option for communication with the tester is gaining ever more in importance.

Various protocols are implemented on the interfaces. These tend to be the CARB or official protocol and the manufacturer-specific communication, which can be closely related in their properties and functions. It is even possible to use both interfaces: CARB-relevant functions run on the CAN, manufacturer-specific functions on the K line.

Figure 2 provides an overview of the standards and the various layers of the OSI reference model.

Diagnostic protocol

The communication method in diagnostics is characterized by the fact that the tester (client) addresses one or several electronic control units (servers) and requests data output (e.g. from the fault memory) or an action (e.g. actuator diagnosis). These services and the transmission agent are defined in the various protocols.

KWP 71

This protocol was developed in collaboration with Bosch as the first standardized serial interface and is based on a 5-baud initialization and byte handshake. For addressing, the protocol provides for the use of a unidirectional line (L line). The line for data exchange (K line) is bidirectional. Alternatively, if the L line is not used at all in the system architecture, the K line can also be used for addressing because addressing and data exchange are two chronologically sequential states. The way in which communication is established is predefined, the services are standardized.

KWP 2000 (ISO 14230: 1-3)

This diagnostic protocol connects a tester to the electronic control units using a K line. In the KWP 2000 standard, there are various possibilities for the way in which communication is established (fast initialization and 5-baud initialization). The communication services are specified but there is some freedom for customer-specific adaptations.

ISO 15765: 1-3 (CAN)

This standard reproduces the existing protocols, ISO 14230: 1-4, on the CAN. It defines a way to leave untouched the fundamental format of messages exchanged between tester and electronic control unit and to transmit them on the CAN bus.

CARB

The Californian environmental authority (CARB) introduced guidelines to promote support for a standardized interface. The CARB protocol is built on existing standards and specifies the data flow and timing (transport layer). This is established in ISO 14230-4/ISO 9141-2 for the K line and in ISO 15765-4 for the CAN. The actual communication messages (services) in the application layer are precisely defined in ISO 15031-5/SAE J1979 and permit no manufacturer-specific deviations.

In California, every newly registered vehicle must support at least one of these protocols with all emission-related electronic control units. The European counterpart is the EOBD standard.

2 Standards for diagnosis communication

a

Layer	CARB			
	K line			CAN
7	ISO 15031-5	ISO 15031-5	ISO 15031-5	ISO 15031-5
6				
5				ISO 15765-4
4				
3				ISO 15765-2 ISO 15765-4
2	ISO 9141-2	ISO 14230-2 ISO 14230-4	SAE J1850	ISO 11898 ISO 15765-4
1	ISO 9141-2	ISO 14230-1 ISO 14230-4	SAE J1850	ISO 11898 ISO 15765-4

b

Layer	Manufacturer-specific	
	K line: KWP 2000	CAN / UDS
7	ISO 14230-3	ISO 15765-3 ISO 14229-1
6		
5		ISO 15765-3
4		
3		ISO 15765-2
2	ISO 14230-2	ISO 11898-1
1	ISO 14230-1	ISO 11898

SVA0018E

Fig. 2
a CARB communication
b Customer-specific communication

Layers of the OSI reference model
7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data link
1 Physical

UDS Unified Diagnostic Services

Application protocols

The increasing complexity of control-unit functions places high demands on the tuning and optimization of these systems to suit the types of engine and vehicle concerned. This process is known as application engineering. For this purpose, an application system is connected to the electronic control unit. The application protocol for the K line is the McMess protocol, and the CCP protocol for the CAN.

McMess

With the McMess protocol, parameters in the electronic control unit can be modified and operands recorded. In a special application mode, the protocol describes the method and contents of communication between application system and electronic control unit for the K line. With McMess, for example, the measuring device is able to read the contents of the variable memory (e.g. sensor values, measured values) quickly.

CCP (CAN Calibration Protocol)

The CCP protocol describes the contents of communication between an application system and an electronic control unit if these communicate on the CAN bus. CCP allows electronic control units in the network to function at higher payload rates than on the K line.

Communication on the K line

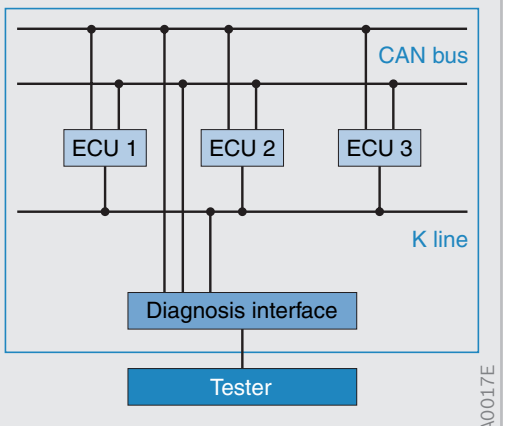
Physical layout

The diagnostic tester connects to one or several electronic control units using the K-line system (Fig. 3). All electronic control units have equal rights. Communication flows bidirectionally between the tester and the electronic control units. This means that data can be received and sent by all subscribers. However, only one subscriber is ever able to send at a given time.

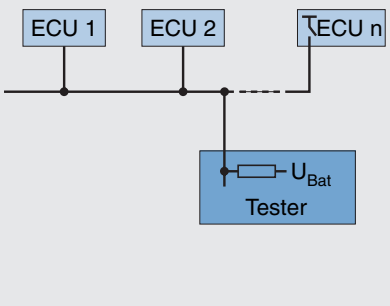
The communication is an asynchronous process. There is no additional line for the transmission of a clock signal. The baud rate on the K line is between 1,200 and 10,400 baud in accordance with ISO standards. In special applications (e.g. manufacturer-specific flash programming), it could even be as high as 250 kBd.

Data transfer on the K line is realized as follows: the resistor in the tester increases the potential of the line – provided no electronic control unit is sending on it – to battery-voltage level (Fig. 4). If an electronic control unit is activated, it connects its K-line connection to the shared ground and therefore switches the potential of the K line to ground. Each of the connected electronic control units is able to detect that one of the bus subscribers is actively accessing the K line.

3 System architecture of bus systems



4 Circuit diagram of an electronic-control-unit network



Message format

The message format of the KWP 2000 protocol comprises three parts (Fig. 5):

- Header
- Data bytes and
- Checksum

Header

The header is made up of no more than four bytes.

- The format byte (Fmt) contains information about the form and composition of the message. Two bits (A0 and A1) provide address information, while six bits (L0 to L5) provide the quantity of data bytes. With this coding, there can be up to 63 data bytes. If L0 to L5 are set to zero, the number of payload bytes must be coded using the length byte (Len).
- The target address (Tgt) indicates the communication partner for which the information is destined.
- The source address (Src) indicates the communication partner that sent the information.
- The length byte (Len) specifies the quantity of payload bytes (max. 255 bytes). Up to a value of 63, the coding can take place in Fmt. For more than 63, the length must be specified using this byte.

The existence of a target and source address as well as the length byte is dependent on the parameters from the format byte.

Data bytes

Up to 255 bytes of payload can be sent per message. The first byte of the payload is always a service-identification byte (SId). The subsequent data bytes are data that differ depending on the service concerned.

Checksum

The 1-byte checksum is always positioned at the end of a message. It contains the value of the modulo 256 sum across all bytes in the message – apart from the checksum itself.

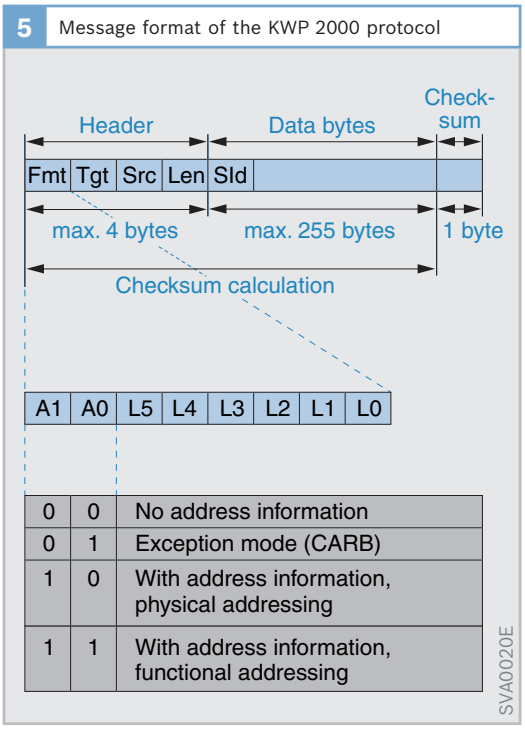
Initialization

To be able to send messages, each electronic control unit must support a subset of the possible message formats. At the beginning of communication, the format supported by the electronic control unit is communicated to the tester by means of key bytes as part of initialization.

The ISO standards specify which of the possible initialization procedures are customary for ECU diagnostics. The following initialization procedures are supported:

- 5-baud initialization and
- Fast initialization

5 Message format of the KWP 2000 protocol



These different initialization methods are used either for CARB diagnostics or for after-sales diagnostics. Furthermore, a distinction can be made between physical initialization and functional initialization, which introduces the topic of "point-to-point" or "point-to-multi" communication. With functional initialization, a group of electronic control units is addressed and initialized. A function initialization can only be successful if all the electronic control units of this group support the same baud rate, the same transmission timing and the same protocol. By contrast, only one electronic control unit is initialized in the case of physical initialization.

5-baud initialization

Communication is initiated by means of a 5-baud address sent by the tester. It is thus possible for each electronic control unit to be addressed individually and addressing is clearly separate from the data transfer.

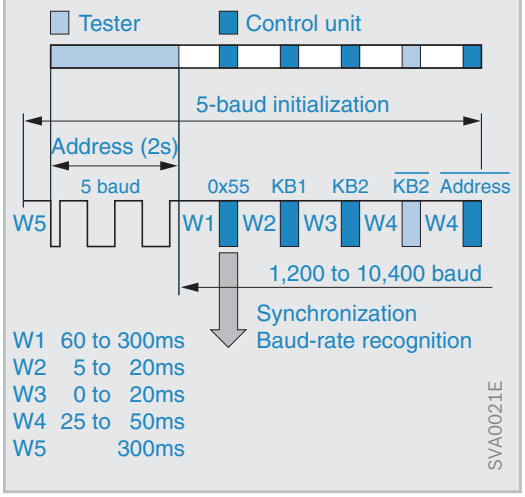
The tester outputs the address onto the K line at a transfer rate of 5 baud (Fig. 6). 10 bits (eight data bits, one start and one stop bit) take approx. 2 seconds.

On receiving the address, the electronic control unit responds with the baud-rate recognition pattern. From the bit sequence 01010101 (binary format), the tester determines the baud rate of the electronic control unit, which may be between 1,200 and 10,400 baud. The electronic control unit then transmits the two key bytes, which communicate the header format and timing method that the electronic control unit supports. As acknowledgment that communication has been successfully established, the tester sends the complement of the second key byte back to the electronic control unit and, in return, receives from the electronic control unit the complement of the address. Initialization is then complete and the regular data transfer may begin.

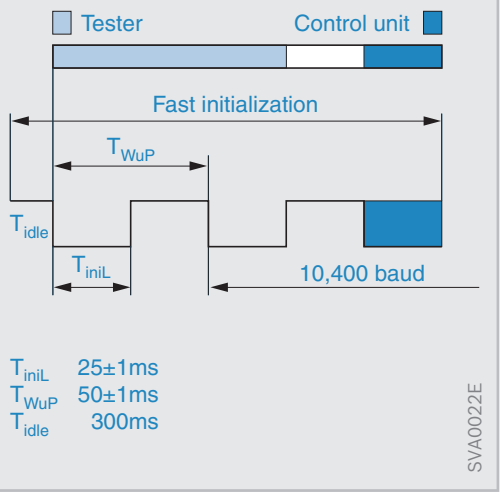
Fast initialization

Fast initialization establishes communication in a shorter time than 5-baud initialization. With fast initialization, the tester sends a wakeup pattern (WuP) comprising a low and high phase, each lasting 25 ms (Fig. 7). At the end of the WuP, the tester sends the StartCommunication Service at a rate of 10,400 baud. The electronic control unit returns a positive reply containing the key bytes. Initialization is then complete and regular communication may begin.

6 5-baud initialization



7 Fast initialization



CARB initialization
CARB initialization is a special case of 5-baud initialization, although CARB may also be activated by fast initialization. A functional initialization of all connected electronic control units is achieved with the fixed address 0x33. The baud rate is permanently set at 10,400 Bd. The requirements are described in more detail in ISO 14230-4 and ISO 9141-2.

One of the outcomes of these standards is that all emission-related electronic control units fitted in a vehicle may only support one protocol: ISO 14230-4, ISO 9141-2 or SAE J1850. A mixture of protocols in the same vehicle is not permitted.

Key bytes
The electronic control unit uses key bytes to inform the tester of which header bytes, length bytes and timing method are supported. The key-byte decoding process is defined in ISO 9141 and ISO 14230-2.

Arbitration

Arbitration is a means of preventing or detecting collisions of data sent by different electronic control units at the same time. Arbitration is necessary if a correct response to the tester request is to be received in the case of functional communication (e.g. reading the fault memory of all emission-related electronic control units using a scan tool). Collisions can be expected because more than one electronic control unit is permitted to send following initialization by the tester.

Arbitration is only relevant to data transfer on the K line.

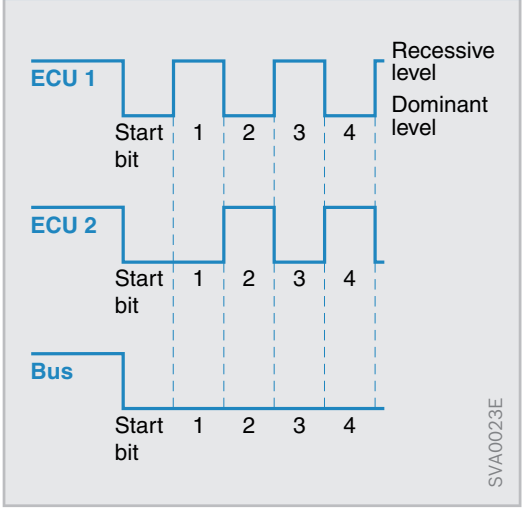
Collision prevention
To prevent collisions following a tester request, each electronic control unit checks the K line for a falling edge after a time of P2min. A falling edge may be interference, or another electronic control unit has already been sending data.

If a falling edge was detected, the electronic control unit delays sending its own data until the bus has returned to idle state and the P2min time period has again elapsed. Only then may it attempt to send its data again.

Collision detection
A collision cannot be prevented if two electronic control units send at the same time. In this event, the collision still has to be detected. This is possible by monitoring the K line. To this end, the electronic control units reread their own transmitted byte through the serial interface.

In Figure 8, two electronic control units (ECU 1 and ECU 2) send a start bit at the same time. ECU 2 then sends a 0 (dominant bit) and ECU 1 a 1 (recessive bit). The 0 appears on the K line. With bit 2, the exact opposite is true. Here, ECU 1 persists with the dominant level. When the complete byte has been transferred, the electronic control units compare the byte sent on the K line with the byte they have read. In the example, a recessive level was overwritten by a dominant level for both electronic control units. From this, both electronic control units deem the transfer to have been faulty and retreat from sending.

8 Sequence of unsuccessful communication



To prevent cyclical collisions, the electronic control unit reports after a time of P2min, which is calculated in accordance with a specific algorithm.

Let us assume that ECU 2 in the previous example sent a 1 as its first data bit. ECU 1 would then receive nothing from the data transfer of ECU 2 and would continue to establish communication. ECU 2, however, detects the collision in bit 2 and retreats. The collision in this event is non-destructive.

Communication on the CAN

Physical layout

A CAN interface containing the necessary hardware already exists in many electronic control units. This bus system can also be used for diagnostics. It is possible to communicate with the electronic control units using a diagnostic tester connected to the bus by a diagnosis connector.

Baud rates

The typical baud rates associated with CAN are 500 kBd or 1 MBd.

Addressing and message types

Tester communication on the CAN bus is defined by ISO 15765. The communication services of ISO 15765-3 or 14 229-1 are defined in a similar way to those of ISO 14230-3 (Fig. 2).

The fundamental difference between these protocols is in the format of messages and how they are transmitted. While up to 255 data bytes can be transmitted in a message on the K line, only eight data bytes are possible with the CAN. To reproduce the service messages on the CAN, the contents (data bytes) of the header and the checksum are separated and embedded in a similar, new message format. The address of the electronic control unit is now the CAN identifier itself. With different CAN identifiers, it is now possible to support functional or physical addressing.

Different addressing methods exist: normal and extended addressing, whereby normal addressing is the regular addressing method and is analyzed below. Longer messages that do not fit into a CAN frame (seven bytes with normal addressing, six with extended addressing) are segmented into several CAN frames and recomposed by the recipient.

Unsegmented messages

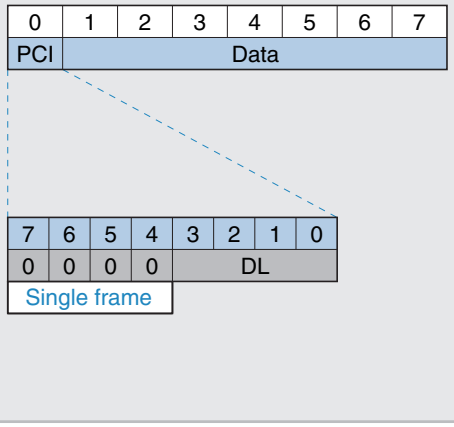
If the quantity of data bytes to be transmitted is no greater than seven bytes (for normal addressing), a single frame is transferred. The first byte to be sent is the PCI (Protocol Control Information). It identifies the frame in the high nibble (top four bits of a byte) as being a single frame and indicates the data length (DL) in the low nibble (Fig. 9).

The data is sent by the transmitter to the receiver in a single message (Fig. 10a).

Segmented messages

If more than seven data bytes (for normal addressing) are to be delivered, the diagnosis tester (client) sends a first frame on the CAN data bus (Fig. 10b) first of all. The electronic control unit (server) acknowledges with a flow-control frame. The other data is subsequently sent in consecutive frames.

9 Format of an unsegmented message



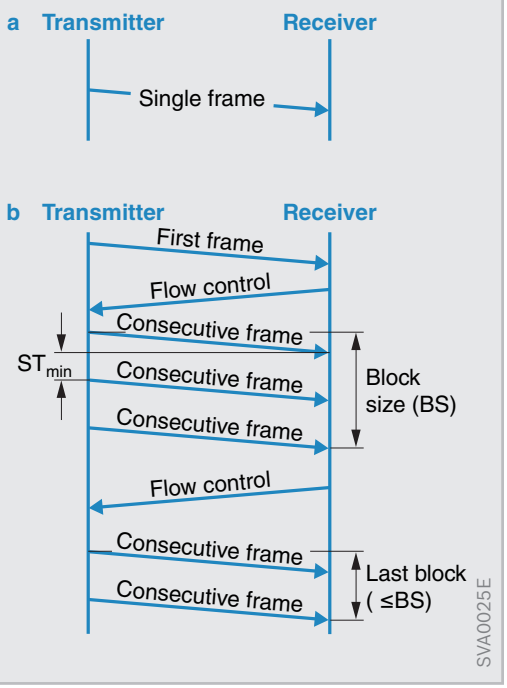
The first frame contains the PCI, an additional length byte (DL, Data Length) and the first six data bytes (Fig. 11). In the high nibble, the PCI contains the information for identifying the frames as the first frame. The extended data length (XDL) is stored in the low nibble. Together with the DL, a 12-bit data word is formed with which it is possible to express values from 0 to 4,095.

The transmitter sends the first frame and waits for an acknowledgment of receipt from the receiver in the form of a flow-control frame. This message again contains the PCI, the high nibble of which identifies the frame as the flow-control frame. The low nibble contains the flow status (FS), which can be used to authorize or delay the sending of further frames (consecutive frames).

BlockSize (BS) indicates how many consecutive frames can be received without the need for a further flow-control frame to be sent. The values expressed in ST_{min} (Separation Time) specifies the interval to be maintained between consecutive frames.

The flow-control frame is followed by the consecutive frames. Again, the PCI in the high nibble contains the identifier, while the low nibble contains the sequence number (SN). With the first consecutive frame to be sent, the SN is set to 1. The SN is incremented with each subsequent consecutive frame; after 15, the SN restarts from 0. By evaluating the sequence number, the receiver is able to detect whether or not all the frames have arrived.

10 Diagram of message transfer



11 Format of a segmented message

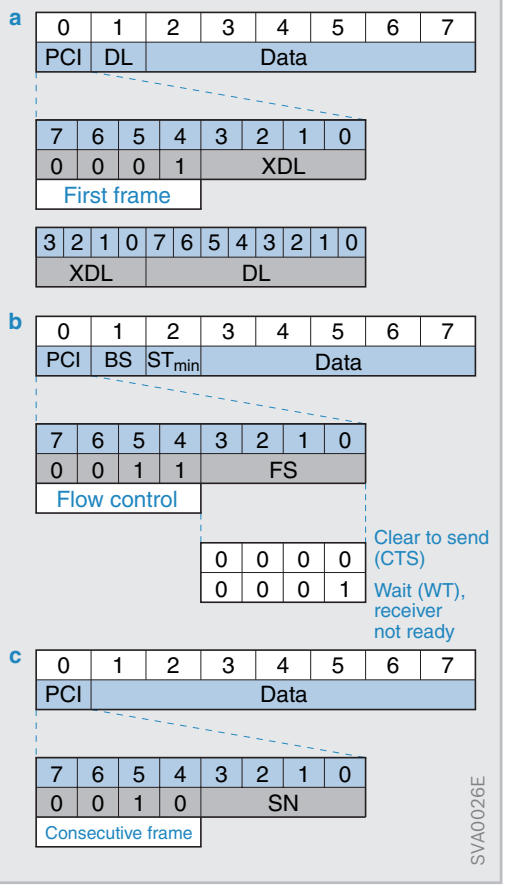


Fig. 10
a Message transfer in a single frame (unsegmented message)
b Segmented message transfer

Fig. 11
a First frame
b Flow control
c Consecutive frame

Architecture of electronic systems

Electronic systems in motor vehicles are today characterized by a high level of networking and complexity. The latest processes, methods and tools of system architecture are required to keep on top of this structure in the future too.

Overview

History

Over the many decades of automobile history, there has been a manageable number of electrical systems in motor vehicles: ignition, lighting, windshield wipers, horn, fuel gauge, various indicator lamps, and a vehicle radio. Semiconductors were used initially only for rectification (direct-current generator replaced by alternator from approximately 1963) and then later for electronic control (transistorized ignition from 1965).

Certain in-vehicle functions were realizable with electromechanical means or with discrete electronic components either not at all or only with disproportionately high complexity. Thus, for example, the first electronic ABS had already been developed in 1970, but was never ready for series production or the market on account of its size, weight and cost (Fig. 1).

By the mid-1970s the development of integrated circuits for a broad range of applications had also reached and revolutionized automotive engineering.

One of the first instances of the networking of electronic systems came about during the development of TCS (Traction Control System). This networking was initially realized by purely mechanical means. The throttle valve in the air-intake system of the internal-combustion engine was fitted with a device which could be activated directly by the traction control system. It was not discernible to the engine management whether the driver or the TCS was moving the throttle valve.

The next stage involved the realization of an electronic connection to the engine control unit via a PWM interface (pulse-width modulation) to improve dynamic response. This could be used to transfer the signal to the engine control unit for reducing the drive torque. This was then implemented in the form of an air-supply throttling, an injection blank-out or an ignition-timing advance.

On account of the ever more stringent exhaust-emission regulations, the previously depicted ways of coupling could no longer be maintained. It had to be left to

1 Comparison of different-generation antilock braking systems

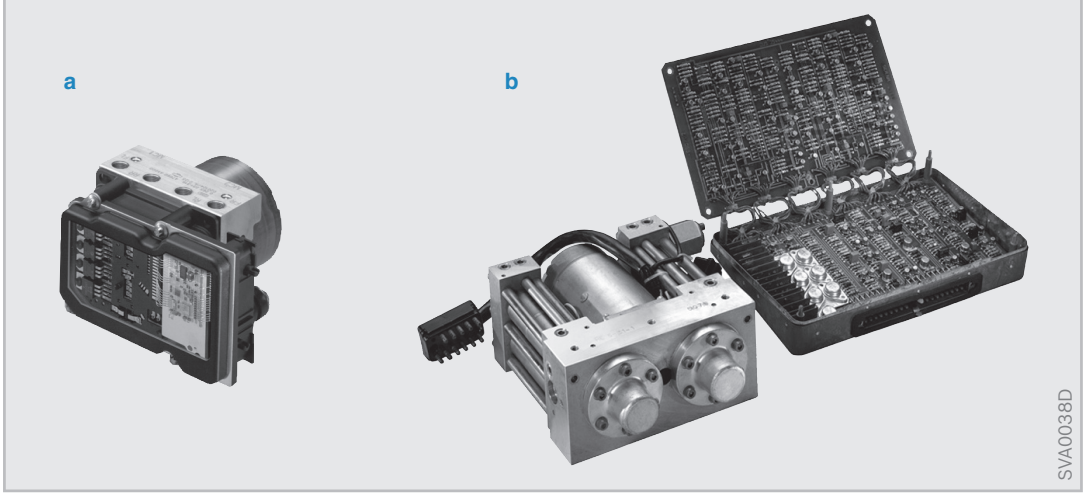


Fig. 1
a ABS8 (current system)
b ABS1 from 1970 (transistor-technology device, did not go into series production)

the engine management as to how a TCS-requested reduction of the drive torque was effected (air path, fuel path or ignition path). It was therefore necessary to come up with a more powerful interface via which a desired torque and a dynamic-response request could be transmitted. By contrast, the actual torque, the engine speed and the current setting reserve were to be transmitted to the TCS control unit. It proved complex and expensive in terms of the number of cables required to transfer these different data via discrete and, for example, pulse-width modulated interfaces. The CAN bus system (Controller Area Network) was introduced in 1991 as an alternative to discrete cabling. In this way, the foundations for the modern networking of systems in motor vehicles were laid.

Technology of the present day

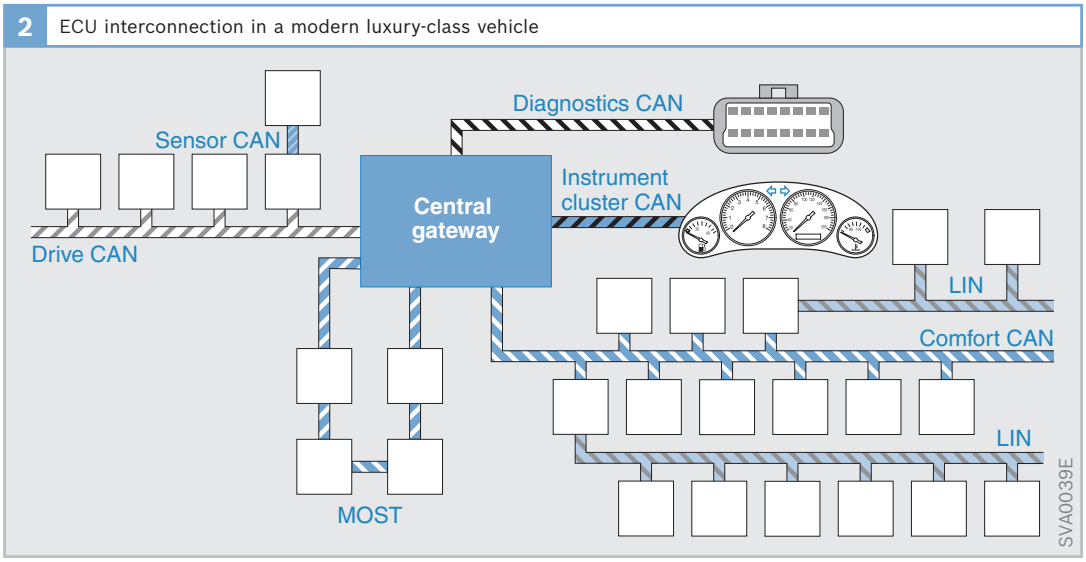
In today's mid-size and luxury-class vehicles, virtually all the ECUs are networked directly or indirectly (e.g. via gateways) with each other (Fig. 2). Networking goes so far to some extent that 60 or more ECUs communicate via several CAN buses and further communication systems, such as MOST (Media Oriented Systems Transport) or LIN (Local Interconnect Network), communicate with each other.

New systems, such as automatic ranging or parking-aid assistant for example, have their own ECU with the determining sensors and actuators. However, they increasingly rely on being supplied with further information by the network. Thus, the ESP control unit (Electronic Stability Program) supplies the network with the information on the vehicle speed. The vehicle radio can use this information, for example, to adapt the volume to the vehicle speed.

To keep on top of the complexity, it is necessary when developing new systems to specify jointly with the creator all the variables obtained from the network. This relates in particular to signal quality and signal availability. The theme of in-vehicle safety always play an important role here. The receiving system provides a limp-home function if the signals obtained from the network are unavailable.

Because of the powerful networking between the ECUs, a good many new performance features can even be achieved completely without additional hardware, i.e. purely by means of data communication and software. One example of this is the opening of the side windows through longer actuation of the radio remote control for the central-locking system. Thus, for example, the vehicle can be uniformly

2 ECU interconnection in a modern luxury-class vehicle



ventilated in the summer months when the doors are opened. The power-window units and the central-locking system exchange the necessary information. The software required for this purpose runs either on the ECU for the central-locking system or the ECU for the power-window units. In many vehicles the two systems share a common ECU so that new software-based performance features can be integrated even more easily.

This demonstrates a trend which is already encountered in body electronics: the integration of individual ECUs to form central ECUs (Fig. 3). These central ECUs are connected with the sensors and actuator either via discrete, analog cables or via buses. The latter reduce significantly the number of pins in the ECU plug and thereby also reduce the cabling costs. Sensors and actuators connected via buses are also known as "intelligent" sensors and actuators. These must for the purpose of bus connection have on board electronic circuitry, which in many cases also contains the sensor-signal conditioning or actuator driver functions. At the same time, however, the use of electronic circuitry gives rise to higher costs in the sensors or actuators. Minimizing the overall costs of electronics and cabling thus represents an important task when defining new networking concepts.

Thus, for example, the logic circuit for the finger-protection function of the power-window units is located in many a design variations directly in the ECU on the power-window motor. The activation signal for normal operation, e.g. the mentioned window opening by radio remote control, is transmitted via a LIN bus from a central ECU of the body electronics (body computer). A client-server architecture is referred to in this respect. Finger protection is a local function which for reasons of safety and on account of its time requirements cannot usually be realized on a distributed basis.

Development trends

The above-mentioned centralization and the use of intelligent sensors and actuators in the field of body electronics are expected to extend in the coming vehicle generations to the other electronics fields. "Domain master computers" will be used to perform central functions in the areas of comfort and convenience, safety, driver assistance, infotainment, and energy supply (Fig. 4). The ECUs of the intelligent sensors and actuators distributed in the vehicle are dependent on these master computers.

3 Comparison of decentralized control with centralized control

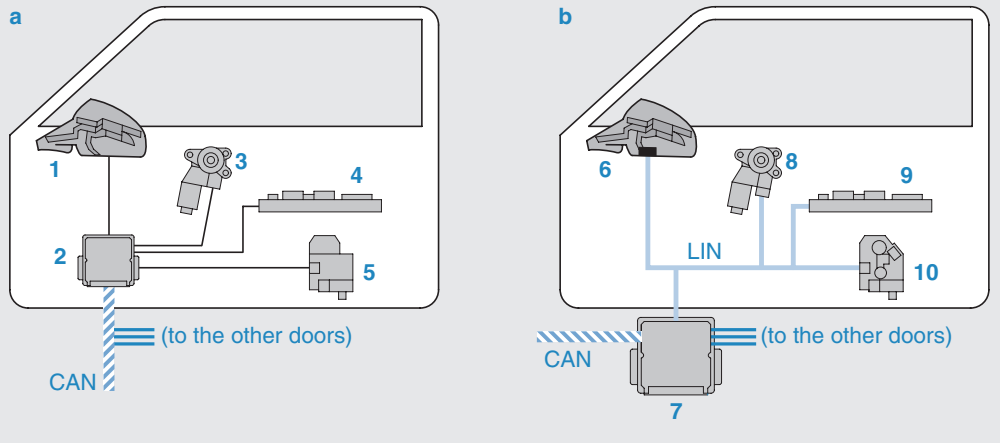


Fig. 3
1 Mirror
2 Door ECU
3 Power-window unit
4 Check unit
5 Lock
6 Mirror (LIN)
7 Central ECU
8 Power-window unit (LIN)
9 Check unit (LIN)
10 Lock (LIN)

Functions which require a high degree of networking of information control commands are predominantly reproduced on these central computers in software. A standard software architecture is required to enable these functions also to run on different ECU platforms and thus to be reused. This is to be achieved by means of the AUTOSAR Initiative (see "AUTOSAR" section).

Here, the master computers are to be networked with each other via a powerful data backbone. Central network access for diagnostics and software downloading is also connected to this backbone. It will also include a firewall to prevent the permeation of undesirable software (viruses).

Architecture methods of electronic systems

Architecture

As the amount of electronics and networking in the vehicle increases, so too the demand for powerful development processes and their description methods takes center stage: the architecture of electronic systems.

The term "architecture" generally refers to the art of building. In the construction industry, the architect designs a building by drawing up plans for the different views and contractual work based on the client's wishes and boundary conditions. A plan abstracts the reality with regard to a particular aspect (e.g. geometric conditions or electric cabling). The building can finally be erected on the basis of the plans of all the necessary aspects.

When carried over to a motor vehicle, this is referred to as the "E/E architecture". "E/E" denotes the electrical and electronic aspects of the motor vehicle. The "plans" of the E/E architect are referred to in the following with the general term of "model".

Automobile manufacturers and suppliers have different views on how many models of which type are needed to

4 Possible scenario for a future luxury-class vehicle

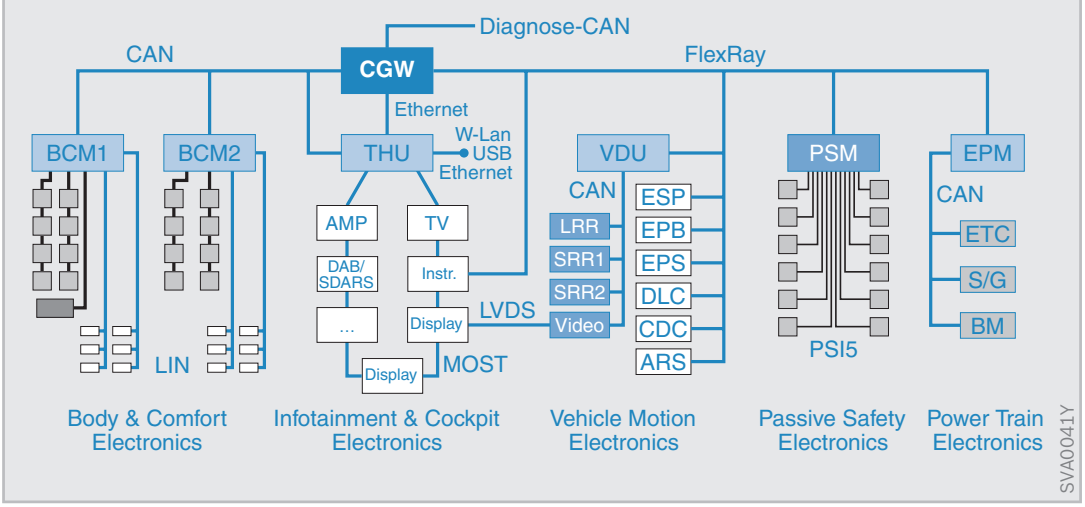


Fig. 4
AMP Audio Amplifier
ARS Active Roll Stabilizer
BCM Body Computer
BM Battery Management
CDC Continuous Damping Control
CGW Central Gateway
DAB Digital Audio Broadcasting
DLC Differential Locks Control
EPB Electric Parking Brake
EPS Electric Power Steering
ESP Electronic Stability Program
ETC Electronic Transmission Control
LRR Long Range Radar
LVDS Low Voltage Differential Signaling
PSI Peripheral Sensor Interface
PSM Passive Safety Manager
SDARS Satellite Digital Audio Radio Services
S/G Starter Generator
SRR Short Range Radar
THU Telematics Head Unit
TV Television
VDU Vehicle Dynamics Unit

describe completely the electrical and electronic systems in the vehicle. The models presented in the following have proven successful in practice and are a necessary framework for describing the E/E scope.

Note: The term architecture is often used in the literature and in publications to denote the models themselves. Here, a clear distinction is made between work operation (= architecture development) and presentation of the result (= model).

Models of E/E architecture

The models of E/E architecture reflect the results of the different integration stages of the electronic systems in the vehicle. These steps are usually dealt with simultaneously because both the geometry (the "body structure") and new systems are addressed in the concept phase. In the course of vehicle development, the situation may arise where an electronic system in the chosen technology does not fit into the available space. Compromises must be found in this case.

Function model, Function network

Function models are the preliminary stage of concrete technical systems. They describe the transfer elements which are needed to realize the required performance features without going into their concrete technology.

For the example of active front steering, this means an analysis of logically functional transfer elements, such as

- Variable steering ratio
- Stabilization control
- Vehicle model
- Actuator
- Vehicle
- Driver

The function models are usually created using block diagrams in accordance with DIN 19 226 (Fig. 5).

There have in the past been several approaches to integrating the active function chains in a standardized hierarchical structure for the complete vehicle ("function network"). This should, for example, distinguish between functions on the vehicle level and functions on the subsystem level (e.g. in the drivetrain). The CARTRONIC® concept from Bosch is an example of this. Unfortunately, the

5 Signal-flow diagram with standard elements

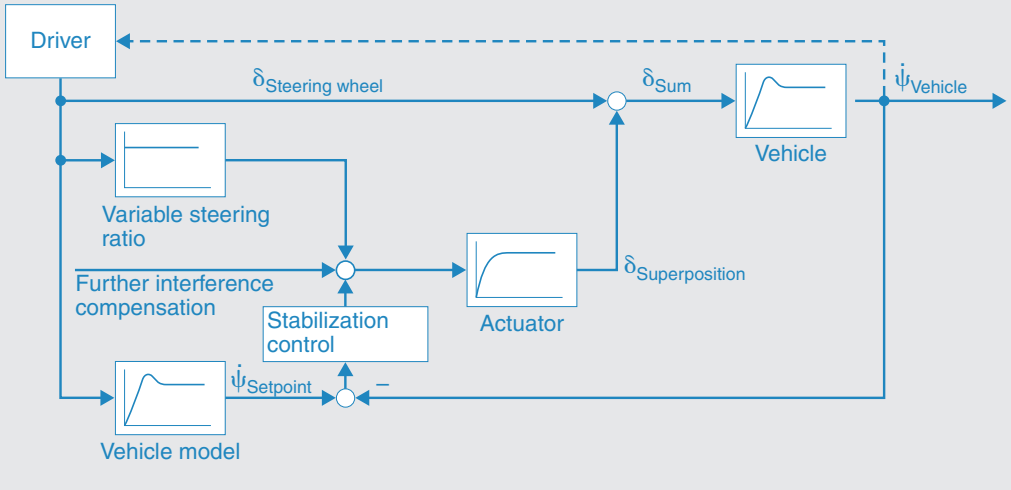


Fig. 5
Signal-flow diagram showing active front steering as an example

automotive industry has up to now failed to agree to a standardized functional structure of the vehicle. However, many promising approaches are contained in the AUTOSAR Initiative, to which Bosch is actively contributing ideas from CARTRONIC®.

Technology model, Technological network

The technology model describes what technical realization is used for the specified function blocks without already combining these into modules, such as e.g. electronic control units (ECUs). "Technology blocks" are created.

Thus, the "signal filtering" function can be realized with discrete components by means of a digital circuit or filter software on a microprocessor. Even a controller function can be executed with a discrete electronic circuit or a microprocessor. Voltage stabilization can be achieved by either a smoothing capacitor or an electronic DC/DC converter.

The decision as to which realization technology to opt for is dictated on the one hand by the function and on the other hand by the costs. Another important dictating factor is reuse. Before the technology blocks are combined into modules in the form of ECUs, the first step is to look for synergy with the further technology blocks to be integrated. A technological network is created (Fig. 6). If, for example, a specific sensor technology is available for an active-chain link whose signal is required by another active chain, this will also be used. This occurs even if this sensor is overspecified for the additional user.

It is nevertheless important to store the original requirement in a database as this synergy may no longer be present in another vehicle.

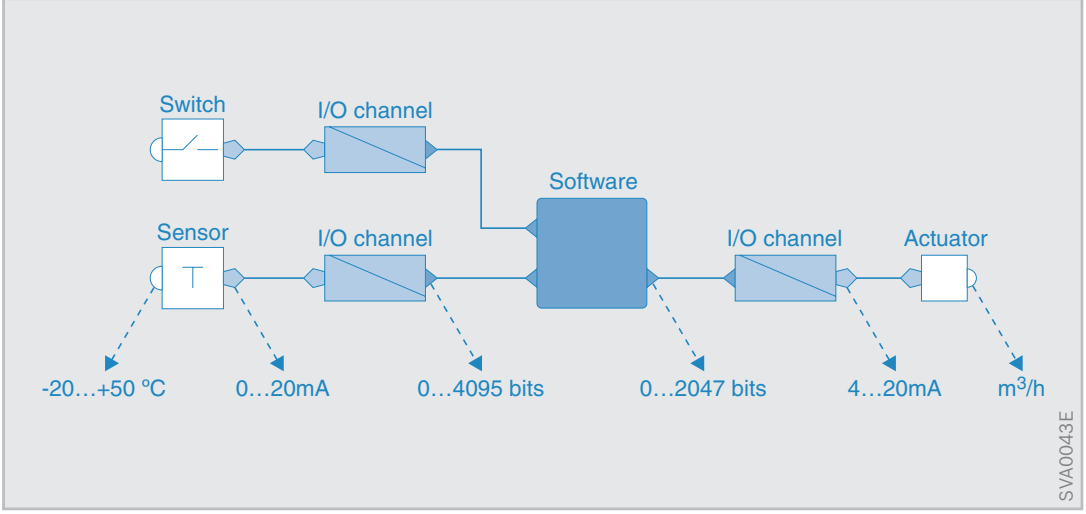
The automotive industry usually uses the nomenclature e.g. according to DIN 19 227, IEC 617-4 and other standards to describe the hardware.

Integration models

Node model

The links of the technological active chains are combined into groups at different locations, the "nodes". Here, strict adherence to the optimum cost of integrating the technology blocks is maintained. Thus, attempts are made, for example, to integrate the software parts of several technological active chains on a common microprocessor. Sensor signals are used repeatedly where possible and actuators used commonly where possible. But the history

6 Example of a technological active chain



shows that there are remarkable synergies even in the mechanical field (e.g. vacuum supply of a pneumatic brake booster by the intake port of the spark-ignition engine).

ECU hardware model

This model represents the structure of the electronic hardware of an individual ECU. It is created by allocating specific electronic components from the technological active chains to an electronic module in a node. An ECU is therefore, generally speaking, a collecting point for electronic components of different systems, an "integration platform".

Software for controlling different systems from different sources (automobile manufacturers or suppliers) is also integrated on the microprocessors located in the ECU. By networking the ECUs, it is possible to represent complex distributed functions, which utilize sensors and actuators for different installation locations in the vehicle.

During development, initially customary circuit diagrams will be used for the electrical or electronic part of the ECU. Then the ECU mechanics and the design and connection technology will be established. The model is confined to a very rough representation in the early concept phase.

ECU software model

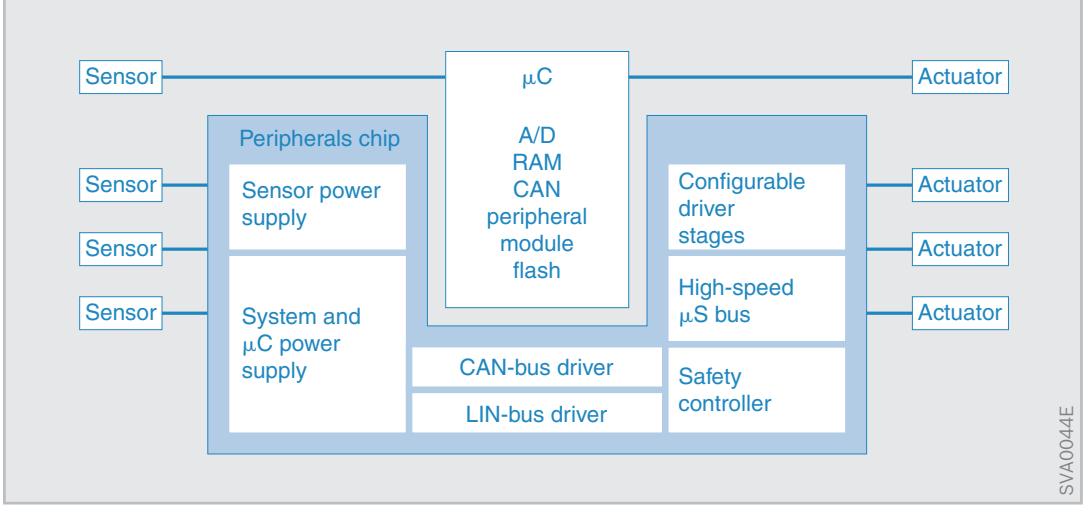
If the software parts of several technological active chains have been assigned to an ECU, it is necessary to plan and model the software required for the respective processor.

Classical IT (PC systems) have some recognized methods for software-architecture development with associated models. However, no standard has developed up to now in the automotive industry. The most widely spread method is that of modeling with the aid of UML (Unified Modeling Language).

AUTOSAR is currently in the process of defining a standard for structuring software along hardware lines and its interface to the application functions in the ECUs of motor vehicles (Fig. 8). A standardized form of representation is also in progress. Some manufacturers of UML tools already offer "AUTOSAR profiles" on the basis of the previously achieved standardization.

Typically, a distinction is made between basic and application software (the "useful load"). Blocks of basic software are, for example, device-driver software, communication software, operating system, and hardware abstraction.

7 ECU model



Network model of communication
Because all the technology blocks of a vehicle have been allocated to ECUs in the previous steps, a network of these ECUs is now in place with their communication relationships. The network model of communication represents all the ECUs in the motor vehicle which have bus communication and are thus directly or indirectly networked with each other.

Each signal which is exchanged between two or more ECUs is assigned to a suitable bus system.

There is currently still no standardized notation for drafting and graphic representation. The models are for the most part created in standard PC-graphics tools.

Network model of energy supply
The allocation of the technology blocks to ECUs and sensor and actuator modules has also given rise to a network of electrical loads/consumers which requires a suitable energy supply.

On the one hand, it is important to fuse individual electric circuits so that a short-circuit does not affect the entire network. On the other hand, not all circuits should be supplied with electrical energy in each operating state. The principle of "terminals" was therefore introduced for this purpose. Thus, for example, terminal 15 is only supplied with electrical energy when the ignition is switched on.

The electrical circuit diagram (Figures 9 and 10) shows the electrical networking and fusing of the individual modules without the installation position being taken into consideration. Here the colors of the cables and the matching with a terminal or fuse can be seen. The terminal designations comply with DIN 72 552.

The positive pole of the supply voltage is usually featured in the top half of the representation while the negative pole (ground) is featured in the bottom half.

8 Simplified representation of the AUTOSAR software model

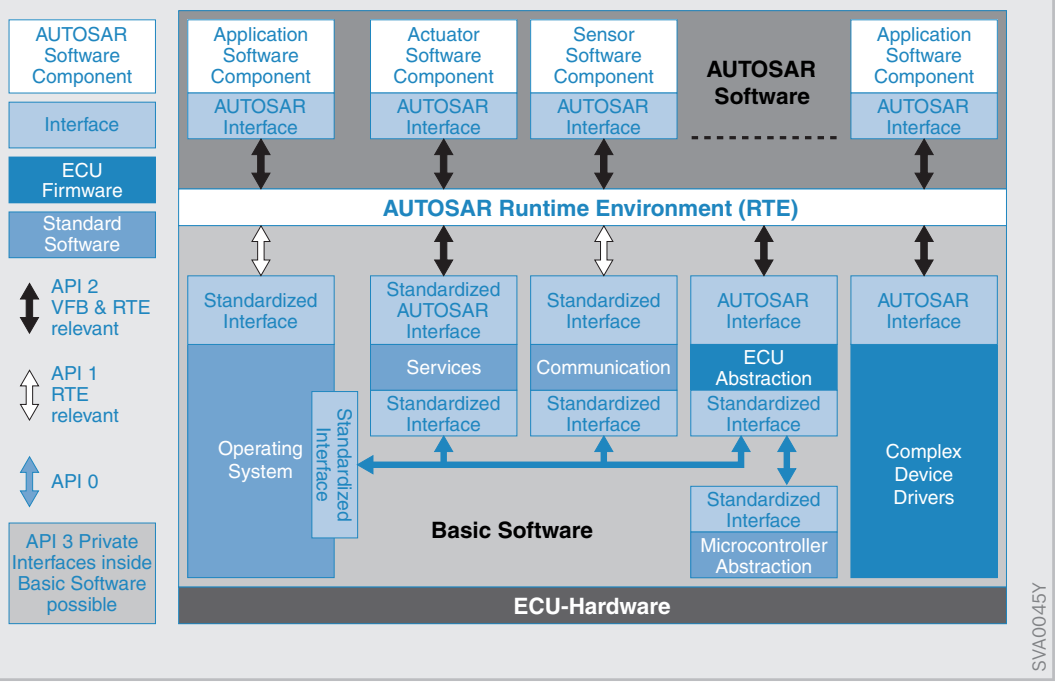


Fig. 8
API Application Program Interface
ECU Electronic Control Unit
RTE Real Time Environment
VFB Virtual Function Bus

Space model and wiring harness

This model groups electronic modules in a specific location in the vehicle (Fig. 11). In this way, the connecting cables between the ECUs or the energy-supply leads of the electrical loads/consumers are brought together in cable looms. This creates the backbone of every vehicle, the wiring harness. Many different boundary conditions must be observed here, such as, for example:

- The manufacturing concept (one-part or multi-part wiring harness)
- Loom cross-sections (flexibility)
- Electromagnetic compatibility (EMC)
- Heat dissipation
- Weight
- Costs (copper)

Two-dimensional models are usually sufficient in the concept phase of a vehicle; detailed three-dimensional models are used in the later development phase.

10 Circuit diagram: vehicle radio

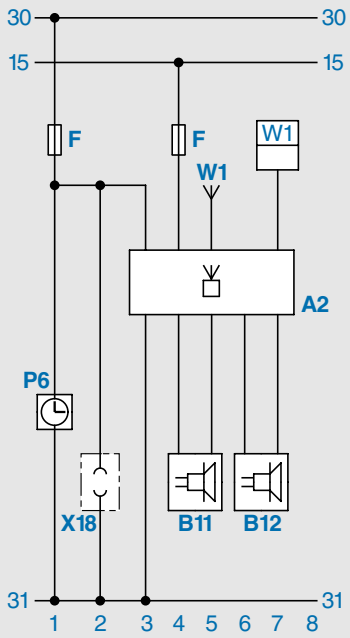


Fig. 10
A2 Vehicle radio
B11 Speaker
B12 Speaker
F Fuse
P6 Clock/timer
W1 Vehicle antenna
X18 Diagnosis socket

9 Circuit diagram: power supply

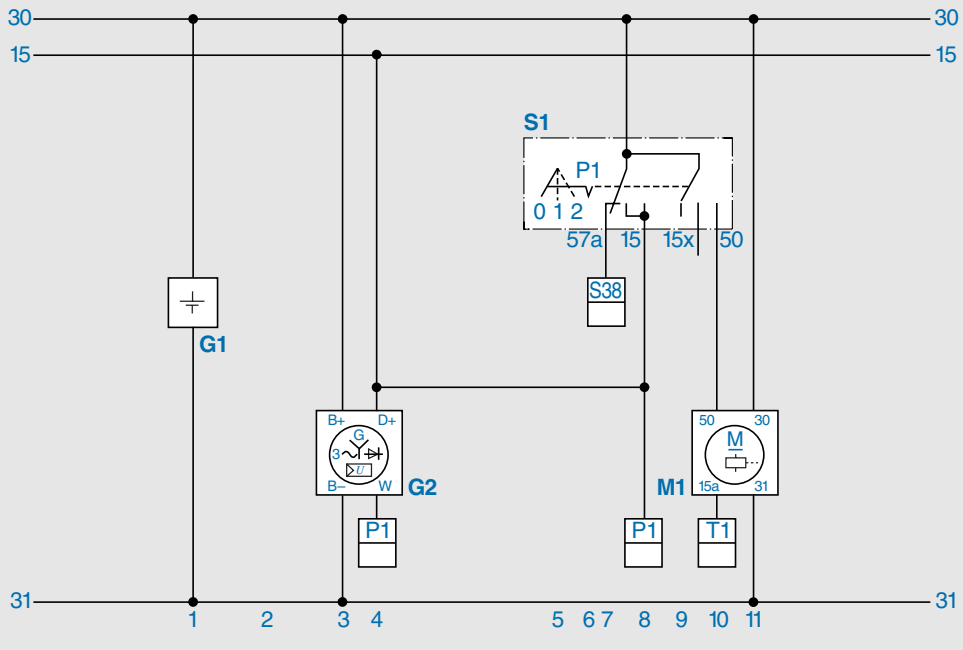


Fig. 9
G1 Starter battery
G2 Alternator with voltage regulator
M1 Starter motor
P1 Display
S1 Ignition switch
T1 Ignition coil

E/E development process

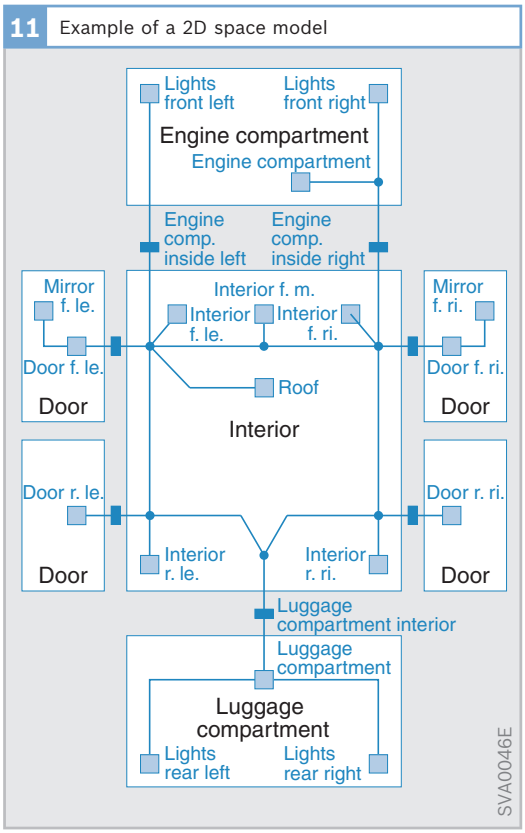
The E/E development process links the individual draft stages with each other on a logic and time basis and provides quality criteria at the beginning and the end of a draft stage.

Because E/E architecture for automotive applications is still a young discipline, the processes at automobile manufacturers and suppliers still differ greatly. This relates both to the number and sequence of the draft stages and to the quality criteria.

Requirement management

The requirements decisively determine the decisions of the E/E architect. It is advisable to distinguish between functional and non-functional requirements. Functional requirements refer to the desired performance features when the vehicle is being used. Non-functional requirements refer to the technical solution and are therefore also known as draft restrictions.

11 Example of a 2D space model



Such a restriction can be, for example, the space available in the center console for installing ECUs. Another restriction can be the maximum permissible heat dissipation in a location which influences the power electronics positioned there. Thus, for example, the audio amplifier in vehicles is frequently installed in the luggage-compartment areas since the heat in the cockpit area cannot be adequately dissipated.

Evaluation of models

The following must be observed for all approaches: During the transition from one model hierarchy to the next (e.g. from the function model to the technology model), a list of evaluation criteria (e.g. reuse or testability) is compared with a portfolio of specimen solutions (e.g. bus technologies). Evaluation of the specimen solutions using the criteria allows a solution to take shape on the basis of the purely functional requirements and irrefutable boundary conditions (“MUST criteria”). This procedure is also known as QFD (Quality Function Deployment).

An alternative procedure consists in comparing a reference solution (e.g. the previous networking model) using the evaluation criteria with alternative solutions. This does indeed deliver fast results, but possibly not the global optimum.

Because the evaluation criteria are generally weighted differently by the automobile manufacturers, the electronic systems of the vehicles sometimes differ considerably from each other.

E/E development tools

It would be desirable to have a tool that can represent the different models of E/E architecture work and network them with each other. It should furthermore be possible to record the modeling properties numerically in order to be able to provide them with an evaluation.

However, this requires a certain degree of standardization of the models and their data formats. Only this facilitates competition between the tool manufacturers and opens up to the different disciplines involved the opportunity of joining the process at different points.

Many automobile manufacturers and suppliers are working intensively on such concept tools or formats. Unfortunately, it is not currently possible to foresee if and when standard formats for describing the models will develop.

AUTOSAR opens up this standard at least for describing software and ECU networks. AUTOSAR will therefore be discussed in more detail in the following.

AUTOSAR

The AUTOSAR Partnership (AUTomotive Open Systems ARchitecture) was founded in July 2003 by vehicle manufacturers and suppliers - Bosch among them. Its global objective is the joint development of an open software architecture for future automotive applications. The Partnership's objectives include the standardization of a fundamental ECU infrastructure (basic software), exchange formats and functional interfaces. These are intended to replace the previous company-specific individual solutions. Model-based concepts and methods make it possible to keep on top of the continuous rise in complexity, brought about by new functions. The demands for quality and reliability are fulfilled by the multiple use of proven standards. AUTOSAR concerns itself in stages with all vehicle domains.

On the basis of the standardized infrastructure software, which consists primarily of standard modules, each vehicle manufacturer can implement its specific content (application software).

Objectives and concepts

In concrete terms, the following objectives are pursued at AUTOSAR:

- Reuse of basic software for different vehicle platforms and vehicle manufacturers
- Support in the integration of third-party software in the field of both basic and application software during ECU development
- Support in the shifting of application software between ECUs during development
- Replacement of standard hardware blocks without this result in changes in the application software
- Model-based concepts for early validation of the system draft

The ECU-software mode defined by AUTOSAR (Fig. 8) supports a horizontalization of the software, whereby a clear separation between the basic software and the application software is created. This is achieved by several abstraction levels in the basic software – from hardware drivers through to complex infrastructure services and the AUTOSAR Runtime Environment (RTE). Because the interfaces of most basic-software modules within these layers are standardized, standard hardware blocks and the associated drivers can be replaced without this incurring changes to the application software.

Conversely, application software which limits itself to the use of these standard interfaces can during development be inserted more easily in an ECU or even shifted to another ECU. It is thus possible, for example, for the same application software of a vehicle-speed controller

depending on the vehicle platform to run on the engine control unit, the transmission control unit or another ECU without the application software having to be changed. This naturally requires the ECU and the networking to be sufficiently powerful.

Standardization of the basic software and its configurability depending on the requirements of the application software enable basic-software modules to be reused for different vehicle platforms and manufacturers. This increases the quality of the software since there are no product-specific changes, reduces the development costs through reuse and forms a stable basis for the constantly increasing complexity and networking of the application functions.

AUTOSAR is also developing model-based concepts for early validation of the system draft. A check is made on the basis of a formalized description as to whether application-software interfaces are consistent with respect to each other without the application software having to be in place as a full program.

Summary and outlook

As a result of the increasing scope and the increasing networking of electronic systems, it is necessary to invest in suitable processes, methods and tools for E/E architecture. E/E architecture has taken shape as an independent function in the automotive industry. If the procedures described in this publication are consistently applied and developed further, electronics in motor vehicles will also be manageable in the future and will continue to make significant contributions to improving traffic flow, traffic safety, driving comfort, and economical fuel utilization.

Index of technical terms

Abbreviations

A

ABC: Active Body Control
ABS: Antilock Braking System
ACC: Adaptive Cruise Control
ADC: Analog Digital Converter
AMA: Active Member Address
AMP: Audio Amplifier
AMS: Application Message Service
ARS: Active Roll Stabilizer
ASC: Active Suspension Control
TCS: Traction Control System
AT: Action Time
AUTOSAR: Automotive Open Systems Architecture

B

BCM: Body Computer
BD: Bus Driver
BG: Bus Guardian
BGE: Bus Guardian Enable
Bit: Binary Digit
BM: Battery Management
BM: Bus Minus
BNEP: Bluetooth Network Encapsulation Protocol
BP: Bus Plus
BSS: Byte Start Sequence

C

CAC: Channel Access Code
CAN: Controller Area Network
CAS: Collision Avoidance Symbol
CBG: Central Bus Guardian
CDC: Continuous Damping Control
CDM: Code Division Multiplex
CDMA: Code Division Multiple Access
CGW: Central Gateway
CHI: Controller Host Interface
CMS: Control Message Service
CNI: Communication Network Interface
CPM: Clear Pending Mode Change
CRC: Cyclic Redundancy Checksum

D

DAB: Digital Audio Broadcasting
DAC: Device Access Code
DI: Direct Injection
DLC: Differential Locks Control
DMA: Direct-Memory-I/O-Access
DMC: Deferred Pending Mode Change

E

EBS: Extended Byte Sequence
ECU: Electronic Control Unit
EDC: Electronic Diesel Control
EDR: Enhanced Data Rate
EEM: Electric Energy Management
EMC: Electromagnetic Compatibility
EPB: Electric Parking Brake
EPS: Electric Power Steering
ESD: Electrostatic Discharge
ESP: Electronic Stability Program
ETC: Electronic Transmission Control

F

FEC: Forward Error Correction
FES: Frame End Sequence
FOT: Fiber Optic Transceiver
FSK: Frequency Shift Keying
FSR: Force Sensitive Resistance
FSS: Frame Start Sequence
FTDMA: Flexible Time Division Multiple Access
FTU: Fault Tolerant Unit
FTM: Fault Tolerant Mean Value

G

GAP: Generic Access Profile
GFSK: Gaussian Frequency Shift Keying
GOEP: Generic Object Exchange Profile
GPS: Global Positioning System

H

HCI: Host Controller Interface
HDL: Hardware Description Language
HFM: Hot-film air-mass meter (German: Heißfilm-Luftmassenmesser)
HHC: Hill Hold Control
HUD: Headup Display

I

I/O Ports: In/Out Ports
IAC: Inquiry Access Code
IC: Integrated Circui
IEEE: Institute of Electrical and Electronics Engineers
IFG: Interframe Gap
INIC: Intelligent Network Interface Controller

L

L2CAP: Logical Link Control and Adaptation Protocol
LAN: Local Area Network
LAP: Lower Address Part
LCD: Liquid Crystal Display
LDF: LIN Description File
LED: Light Emitting Diode
LIN: Local Interconnect Network
LLI: Logical Line Interface
LRR: Long Range Radar
LSB: Least Significant Bit
LVDS: Low Voltage Differential Signaling

M

MAC: Media Access Control
MAMAC: MOST Asynchronous Medium Access Control
MARS: Maintainable Real Time System
MC, µC: Microcontroller
MEDL: Message Descriptor List
MHP: MOST High Protocol
MOST: Media Oriented Systems Transport
MSB: Most Significant Bit
MSC: Message Sequence Chart

N

NAP: Non-significant Address Part
NIC: Network Interface Controller
NIT: Network Idle Time
NM: Network Management
NRZ: Non-Return to Zero

O

OBEX: Object Exchange
OSI: Open System Interconnection

P

PAN: Personal Area Network
PC: Personal Computer
PCM: Pulse Code Modulation
PD2: Partition Design 2
PDA: Personal Digital Assistant
PLC: Powerline Communication
PMA: Parked Member Address
POC: Protocol Operation Control
POF: Plastic Optical Fiber
ppm: parts per million
PPP: Point-to-Point Protocol
PRP: Post-Receive Phase
PSI: Peripheral Sensor Interface
PSK: Phase Shift Keying
PSM: Passive Safety Manager
PSP: Pre-Send Phase

Q

QFD: Quality Function Deployment

R

RADAR: Radiation Detecting and Ranging
RFCOMM: Radio Frequency Communication

S

SAE: Society of Automotive Engineers
SDAP: Service Discovery Application Protocol
SDARS: Satellite Digital Audio Radio Service
SDP: Service Discovery Protocol
SEI: Software Engineering Institute
SIG: Special Interest Group (Bluetooth)
SOS: Slightly-Off Specification
SPP: Serial Port Profile
SRR: Short Range Radar
SRU: Smallest Replaceable Unit
STN: Super Twisted Nematic
STP: Shielded Twisted Pair

T

TR: Technical Requirements
TCS BIN: Telephony Control Protocol Specification - Binary
TDD: Time Division Duplex
TDMA: Time Division Multiple Access
THU: Telematic Head Unit
TP: Transmission Phase
TSS: Transmission Start Sequence
TTA: Time Triggered Architecture
TTP: Time Triggered Protocol
TV: Television

U

UAP: Upper Address Part
UART: Universal Asynchronous Receiver Transmitter
UTP: Unshielded Twisted Pair

V

VDU: Vehicle Dynamics Unit

W

WAN: Wide Area Network
WLAN: Wireless Local Area Network
WPAN: Wireless Personal Area Network
WUP: Wakeup Pattern
WUS: Wakeup Symbol