# Serial Bus Systems in the Automobile

Part 3:

Simple and cost-effective data exchange in the automobile with LIN

In just a very short time the LIN bus has established itself as the technology of choice for simple and cost-effective data exchange in the automobile. Today, many automotive OEMs are relying on LIN to transmit non-critical signals in the body/convenience area. The following article points out the reasons for the victory of LIN (Local Interconnect Network) in the automobile and explains the underlying technology.
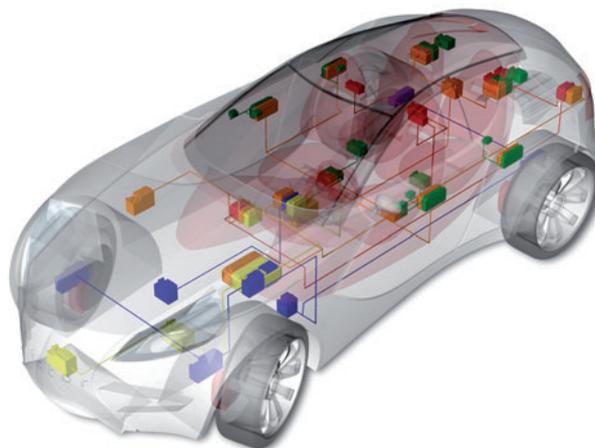
Why another data bus in the automobile?

Growing demands of car users for driving conveniences have led to wide-ranging electronification in this area of vehicle technology. This is reflected in the migration of numerous electronic components into the convenience area. For a long time it was usual practice to interconnect the continuously increasing number of sensors, actuators, stepper motors and DC motors directly to a central control module.

This trend gradually met with criticism: It led to a rapid increase in wiring costs, along with greater space requirements, increasing weight and significantly greater susceptibility to faults. In addition, growing individualization required many different wire harness and connector variants, which in turn made production, installation and maintenance considerably more difficult.

Developers quickly recognized that networking of components over a bus system would be an ideal solution in this automotive application domain too. However, since the CAN bus was not a candidate for use in the cost-sensitive sensor/actuator area, many automotive OEMs and suppliers began to develop their own sensor/actuator bus systems as early as the mid-1990s.

This gradually resulted in the creation of numerous cost-effective and simple yet proprietary sensor/actuator buses. In the year 2000 LIN arrived on the "networking market" as another serial bus system for the sensor/actuator area. This technology has prevailed on a broad front, and LIN can now be found in nearly all vehicles, typically in convenience applications such as climate control, seat adjustment, door controls and mirror adjustment.

LIN Consortium assists in breakthrough

An important reason for the quick establishment of LIN was the founding of the LIN Consortium [1], which prominent automotive OEMs and suppliers as well as semiconductor and tool manufacturers had joined. Its goal was to create a cross-OEM communication standard for the sensor/actuator area. With definition of a simple and cost-effective Physical Layer based on ISO standard 9141, as well as a simple and lean communication protocol, the LIN Consortium has laid the foundation for success. It set the stage for implementation of simple and cost-effective bus nodes.

The LIN Consortium not only focused on LIN communication itself, but also provided a development methodology (LIN Work Flow), and this furthered acceptance of the bus system in the automotive industry significantly. LIN Work Flow makes it is possible to automate the development of a LIN network (LIN Cluster), with resulting time and cost savings. The cornerstones of the development methodology are two data exchange formats used to describe the entire LIN Cluster and individual LIN nodes uniformly (Figure 1).
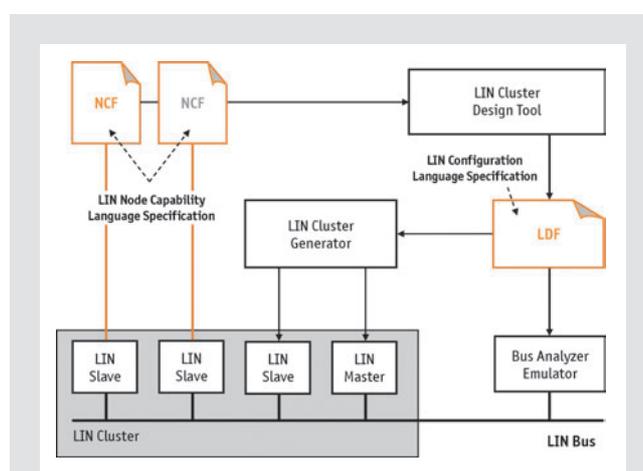

Figure 1:
LIN Work Flow: The standardized and quick path to the LIN Cluster.

Serving to describe an entire LIN Cluster are the uniform syntax (LIN Configuration Language) and the standardized LIN Description File (LDF). Defined in the LDF are all of a LIN Cluster's properties, in particular communication relationships. Generator tools utilize the LDF to generate the software components needed for LIN communication. Additionally, the LDF provides necessary information to analysis, measurement and testing tools and rest-of-bus emulators.

Similarly, the description of individual LIN nodes (LIN Slaves) is given structure by the uniform syntax of the Node Capability Language and standardized Node Capability Files (NCF). The NCF describes the performance characteristics of a LIN Slave (including frame and signal definitions, bit rates, diagnostics) and in the framework of system design it represents the foundation for automated generation of the LDF.

The two date exchange formats and the configuration process defined in the LIN specification let users implement a LIN Slave type (e.g. stepper motor) multiple times in a LIN Cluster or use a LIN Slave in different LIN Clusters (reusability of LIN Slaves).

Making just as important a contribution to the success of LIN is detailed documentation of the specification. LIN specification 2.1 [2], in existence since November 2006, defines the Physical Layer, communication protocol, LIN Work Flow, LIN API as well as diagnostics and configuration of the LIN nodes.

Single-wire communication at rates up to 20 KBit/sec

The goal of creating a low-cost communication protocol for serial data exchange in the non-safety-critical sensor/actuator area primarily affected the design of the Physical Layer. Physical signal transmission in a LIN Cluster does not involve use of the differential signal transmission familiar from CAN, rather a conventional single-wire line is used. To assure sufficient noise immunity in spite of this method, the supply voltage and ground of the ECU electronics are used as reference voltages for the bus level. A LIN transceiver serves as the physical bus interface. A level at least 40 % below the supply voltage is interpreted by the receiver as a logical "0". Receivers interpret a level at least 60 % above the supply voltage as a logical "1".

The maximum data rate is limited to 20 KBit/sec to keep noise emissions within limits. For line lengths up to 40 meters the maximum recommended node count is 16. This takes into account the node and line capacitances as well as the maximum allowable time constant of the LIN Cluster prescribed in the LIN specification.

In terms of circuit technology a LIN Cluster is equivalent to an Open Collector circuit. A pull-up resistor ensures that the bus level remains nearly at the level of the supply voltage (High level) while the Tx transistors of all LIN nodes are inhibited. The bus level is pulled to nearly ground level (Low level) as soon as one of the Tx transistors is enabled. Accordingly, the Low state is referred to as the dominant level and the High state as the recessive level.

Master-Slave communication

Communication in a LIN Cluster is based on a Master-Slave architecture. A cluster consists of a Master node (LIN Master) and at least one Slave node (LIN Slaves). For cost reasons, explicit communication controllers are not used. Instead LIN communication is implemented by software tasks in every node, the so-called Slave tasks. The LIN Master also has a Master task that is used to coordinate cluster communication (Figure 2).

Coordination is achieved by means of periodic execution of the LIN Schedule that is organized in frame slots (Figure 3). At the begin-
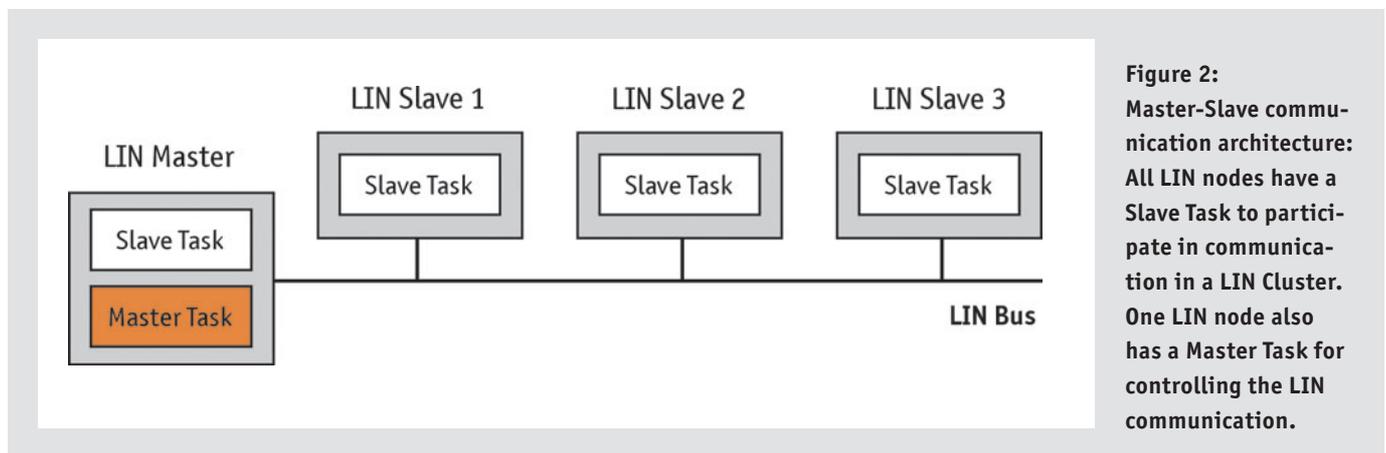

**Figure 2:
Master-Slave communication architecture:
All LIN nodes have a Slave Task to participate in communication in a LIN Cluster. One LIN node also has a Master Task for controlling the LIN communication.**

ning of each frame slot the Master Task transmits a frame header with a Frame Identifier (ID), which all LIN Slaves evaluate in their Slave Task. Immediately after the frame header a LIN Slave transmits the frame response associated with the ID. The LIN Frame, consisting of the frame header and frame response, is available to be received by every LIN node (Broadcasting) due to ID-based message addressing.

Data transmission by LIN frames

Due to the lack of a communication controller, serial data transmission in a LIN Cluster is handled over the microcontroller's serial interface (Serial Communication Interface - SCI) and is performed byte-by-byte. The SCI transmits each byte with the LSB (Least Significant Bit) first, and the byte is framed by a start bit and a stop bit (SCI frame). That is, a LIN frame is composed of a combination of a number of SCI frames distributed between the frame header and frame response (Figure 4).

In transmitting the frame header the LIN Master performs two key communication tasks: It synchronizes the LIN Slaves and delegates communication by assigning a sender and one or more receivers to the frame response.

Due to cost sensitivity issues, the LIN Slaves may use on-chip resonators with a frequency tolerance of up to 14 percent. Therefore the LIN Master transmits a Sync Break first to let all LIN Slaves know that transmission of a LIN Frame is beginning. The Sync Break is made up of at least 13 consecutive dominant bits, and it elicits a SCI error from all LIN Slaves. It is terminated by the Sync Break Delimiter (at least one recessive bit). The LIN Master transmits the communication clock pulse with the subsequent Sync Byte (SCI Frame with the value 0x55).

An ID serves to delegate communication; it is six bits in length and is protected by two parity bits with even parity and odd parity. The ID and two parity bits together are referred to as the PID (Protected Identifier). The first 60 IDs are available for useful data communication. The last four identifiers, ID 60 through 63, are reserved (of which ID 60 and ID 61 are used for diagnostic purposes).

The frame response is composed of up to eight data bytes and a checksum for error detection. A distinction is made between the classic and extended checksum. The classic checksum is the inverted modulo-256 sum of all data bytes. There is a transmission error if result of adding the modulo-256 sum to the arriving data bytes does not equal "0xFF". The extended checksum also considers the PID in forming the inverted modulo-256 sum.

Since the LIN Slaves are usually equipped with very simple and low-cost microcontrollers, not only are they allowed to delay transmission of the frame response a little (Response Space), but they may also insert sending pause (Interbyte Spaces) between transmissions of SCI frames. Overall, this may lengthen the frame response by 40 percent. The same applies to the frame header, primarily because there are different methods for generating the Sync Break. It is absolutely essential to consider the 40 percent time reserve in designing the LIN Schedule.

Sporadic, Event Triggered and Diagnostic Frames

The LIN specification contains provisions for making the communication cycle that is rigidly defined by the LIN Schedule more flexible and economical. The two frame types "Sporadic Frame" and "Event Triggered Frame" are provided for this purpose. In introducing these supplemental frame types it has become common practice to refer to the conventional LIN frame as an Unconditional Frame.
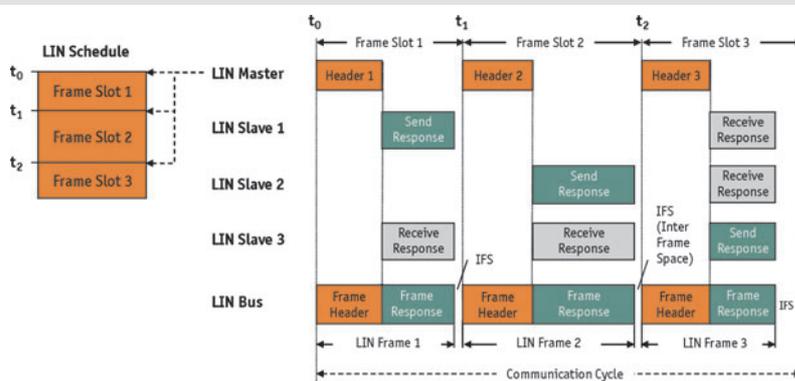

Figure 3:
Central message distribution system: The LIN Master controls sending and receiving of all LIN frames by the Master Task and LIN Schedule. A frame slot must be long enough to transmit the associated LIN frame. The length of the IFS depends, among other things, on the execution cycle (time base) of the Master Task.

A Sporadic Frame is understood as an Unconditional Frame that shares the same frame slot with other Unconditional Frames. Sporadic Frames are transmitted entirely by the LIN Master as necessary, so collisions are impossible. If the LIN Master has no need for any of the frames, the associated frame slot simply remains empty.

The Event Triggered Frame was introduced to communicate sporadic changes or events on the part of the LIN Slaves. It essentially corresponds to an Unconditional Frame, but with the difference that multiple frame responses from different LIN Slaves are allocated to the frame header. The frame response that is used to complete the Event Triggered frame header depends on the needs of the related LIN Slaves. There is a need when there are new data to be transported. The frame response of the Event Triggered Frame is identified by the PID of the associated Unconditional Frame in the first byte.

In contrast to the Sporadic Frame, however, collisions cannot be excluded in the Event Triggered Frame. In case of a collision the LIN Master is responsible for transmission of all Unconditional Frames assigned to the Event Triggered Frame. It does this by activating and processing a "Collision Resolving Schedule".

Both conventional Unconditional Frames and special Diagnostic Frames are suitable for diagnosing the LIN Slaves. Unconditional Frames are used for simple signal-based diagnostics, while Diagnostic Frames are used either for user-defined diagnostics or diagnostics based on a standardized transport protocol [3] and uniform diagnostic services [4] [5].
The LIN specification defines two Diagnostic Frames: The "Master Request Frame" and "Slave Response Frame". The Master Request Frame (ID=0x60) represents the Diagnostic Request. In this case the LIN Master transmits both the frame header and the frame response. For example, a Master Request Frame is transmitted if there is a Diagnostic Request via CAN. The Slave Response Frame (ID=0x61) corresponds to the Diagnostic Response. In this case the LIN Master transmits the header, and the specific LIN Slave transmits the response.

Management functions

The LIN specification defines Status Management and Network Management. Status Management specifies that LIN Slaves must inform the LIN Master of transmission errors that are detected such as parity or checksum errors. This is done by a "Response Error Signal" in an Unconditional Frame ("Status Frame"); however this frame does not contain any further information on the type of error. The LIN specification does not define error handling rather it leaves this task to the user.

The primary task of LIN Network Management is to regulate the transition of all Slaves in a LIN Cluster from the normal communication state (Operational) to the Sleep state and in the other direction (Figure 5). If the LIN Slaves do not detect any bus activity for four seconds, they switch from the Operational state to the Sleep state. The same condition elicits a Sleep command from the LIN Master, which is really a special Master Request Frame.

Conversely, when a LIN Slave detects a "Wake Up Signal" followed by a valid header it switches from the Sleep state via the Initialization state to the Operational state. The Wake Up Signal consists of a
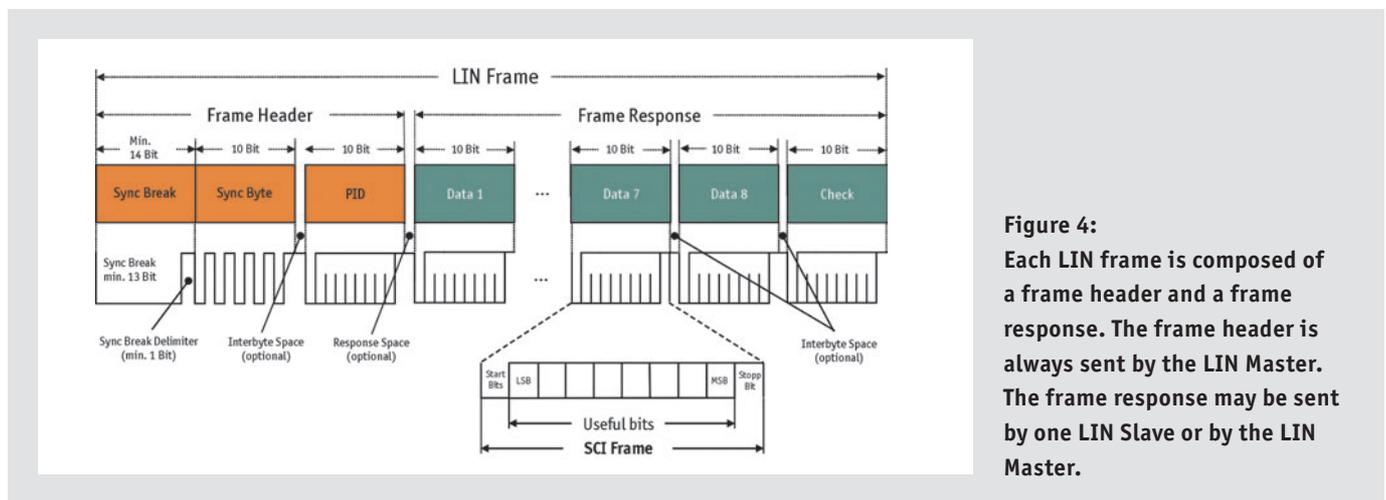

Figure 4:
Each LIN frame is composed of a frame header and a frame response. The frame header is always sent by the LIN Master. The frame response may be sent by one LIN Slave or by the LIN Master.

dominant pulse minimum 250 microseconds and maximum 5 milliseconds in length, and it may be sent by any LIN node. The LIN specification prescribes a maximum Initialization phase of 100 milliseconds, i.e. the LIN Master must begin to execute the LIN Schedule at the latest after this time span. If it remains passive the relevant LIN Slave sends another Wake Up Signal. The number of repetitions and the interval between repetitions, as well as timeouts are defined in the specification.

In case of networking issues: Quick resolution with external expertise

In networking with LIN, CAN, FlexRay and MOST, Vector Informatik [6] supports automotive OEMs and suppliers with a universal tool chain, embedded software components, base software for AUTOSAR control modules and hardware interfaces. Users of the CANoe.LIN tool for example benefit over the entire development process from practice-proven functions for model generation, simulation, functional testing, conformity testing, diagnostics and analysis. Multibus design and maintenance of the communication data of a networked system is supported by DaVinci Network Designer for LIN, CAN and FlexRay for example. Development, calibration and diagnostic tools for in-vehicle ECUs complete Vector's extensive product line-up. Besides consultation the Stuttgart-based company also offers a tool environment for the electronic systems development process. These services are rounded out by a comprehensive training program covering Vector tools, software components and serial bus systems.

For an entry-level introduction to serial data exchange in the automobile the Vector Academy [7] offers the one-day training course "Serial bus systems in the automobile". Training courses on CAN,

LIN, FlexRay and MOST fundamentals convey the necessary basic knowledge needed to quickly become familiar with the wide variety of development activities related to automotive electronics.

The first two parts of this series of articles addressed the topics of serial data exchange in the automobile and CAN. The next two articles will discuss the serial bus systems FlexRay and MOST.

Literature and links:
[1] www.lin-subbus.org
[2] LIN Specification Package Revision 2.1
[3] Road vehicles – Diagnostics on Controller Area Network (CAN) – Part 2: Network layer services, International Standard ISO 15765-2.4, Issue 4, 2002-06-21
[4] Road vehicles – Diagnostics on controller area network (CAN) – Part 3: Implementation of diagnostic services, International Standard ISO 15765-3.5, Issue 5, 2002-12-12
[5] Road vehicles – Diagnostic systems – Part 1: Diagnostic services, International Standard ISO 14229-1.6, Issue 6, 2001-02-22
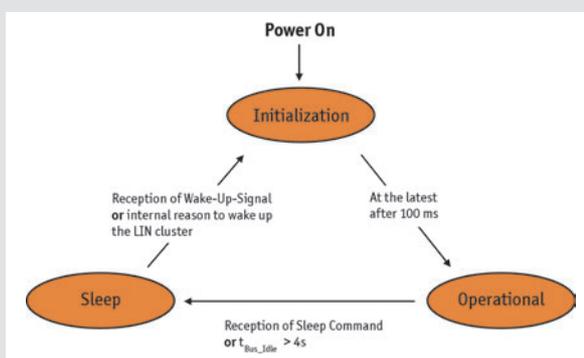[6] www.vector-informatik.com
[7] www.vector-academy.com

Figure 5:
Communication state model of a LIN Slave.

Eugen Mayer
(Graduate Engineer with Technical Teaching Certificate), after completing his vocational training to become a communications technician, studied electronics at the Technical College in Ravensburg/Weingarten, Germany, and studied electrical engineering and vocational teaching at the University of Stuttgart. Since 1999 he has been working at Vector Informatik where he is employed as a Senior Trainer.
Tel. +49-711/80670-574,
Fax +49-711/80670-111,
E-Mail: eugen.mayer@vector-informatik.de