IMT SCHOOL

Automotive Industry

LIN Communication

Introduction

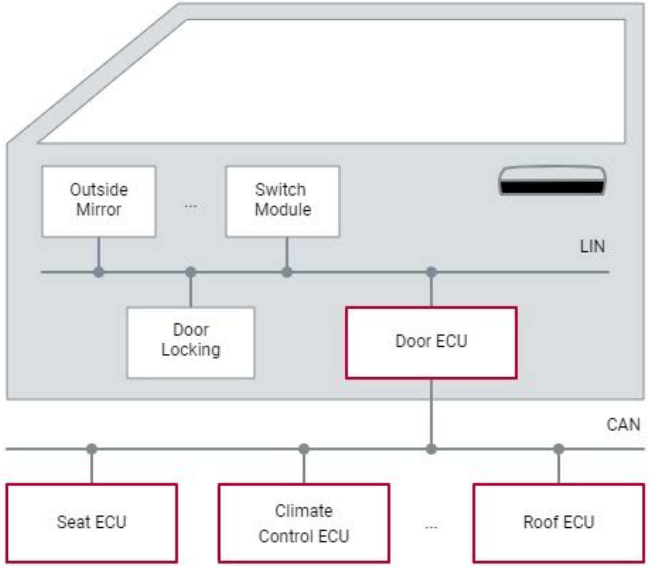
LIN (Local Interconnect Network) is a serial communications technology that was developed for cost-sensitive use areas in the automobile.

Over the past decades, increasing numbers of functions have been developed for motor vehicles, which are intended to make car driving more comfortable, safer and environmentally friendly. In the process, more and more functions are implemented with electronic components, which have a growing need for the exchange of information. These electronic components include both ECUs and sensors and actuators.

Nonetheless, CAN was too expensive for connecting cost-effective sensors and actuators in the convenience area. In the mid-1990s some automotive OEMs and suppliers began to develop more economical solutions. However, since these proprietary bus systems involved low part volumes, they only offered limited cost relief. That is why some OEMs joined together to form the LIN Consortium whose goal was to develop a uniform, standardized and cost-effective communication system.

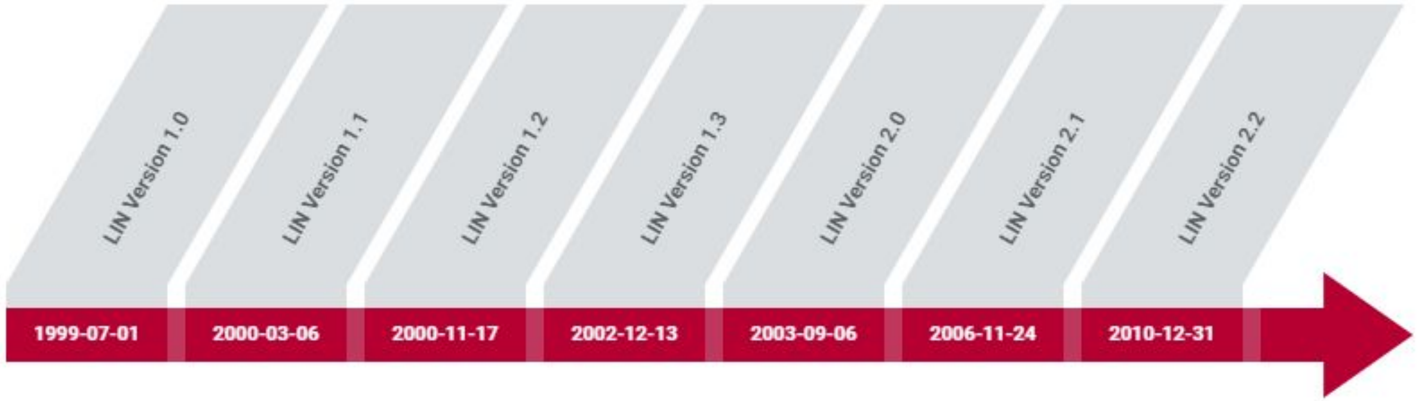
Introduction

At the initiative of BMW, Daimler-Benz (today: Daimler), Motorola (today: Freescale) and VCT (today: Mentor Graphics) began a working group at the end of 1998 to develop the new bus system. The members of this working group, which by then had grown to include Audi, Volvo and VW in addition to the initiators, were the founding members of the LIN Consortium, and they formed its Steering Committee.



LIN has since become established as a subbus and is found in nearly every vehicle. It is especially used in convenience applications such as climate control, seats, doors and mirror control modules. All sensors and actuators are, in contrast to conventional wiring, equipped with a bus interface and are connected to the central ECU over the bus

LIN History

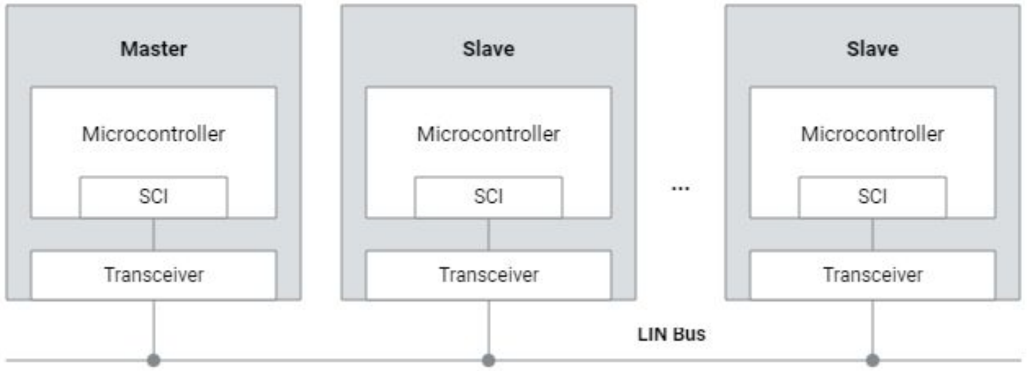


The first specification originates from the year 1999. In the following year, the Consortium introduced Version 1.1 at the SAE Congress in Detroit. The specification was revised in the year 2000, and it was released in November 2000 (LIN 1.2). In the next years, the physical layer in particular was revised, and the LIN 1.3 version was released in November 2002.

Version 2.0 followed in September 2003. Specification 2.1 has been in existence since November 2006. It now consists of eight specification parts. LIN specification 2.2 was released in December 2010, which corrects a number of minor errors and contains slightly weaker rules for bit sampling.

LIN Network

A LIN Cluster consists of a number of nodes that are interconnected via a physical transmission medium. A general distinction is made between two types of nodes. There is always a master that controls bus access, and there are multiple slaves that can send and receive information.



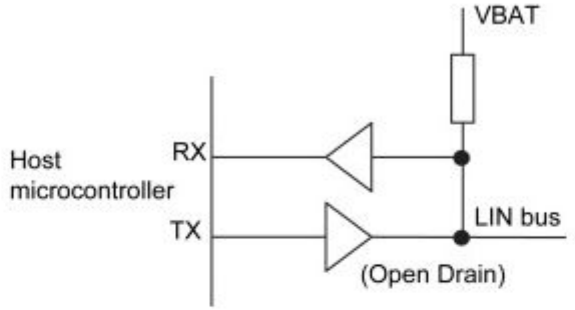
For cost reasons, no dedicated communication controller was implemented. Instead, the protocol must be integrated as a software component in the microcontroller. It should be clearly noted whether a node is to be implemented as the master or a slave. The microcontroller is connected to the transceiver via the serial interface. This interface is referred to as the SCI. The transceiver serves to physically connect the bus to the network. This chip converts the logical bit sequence into transmittable bus levels and in the reverse direction.

LIN Specifications

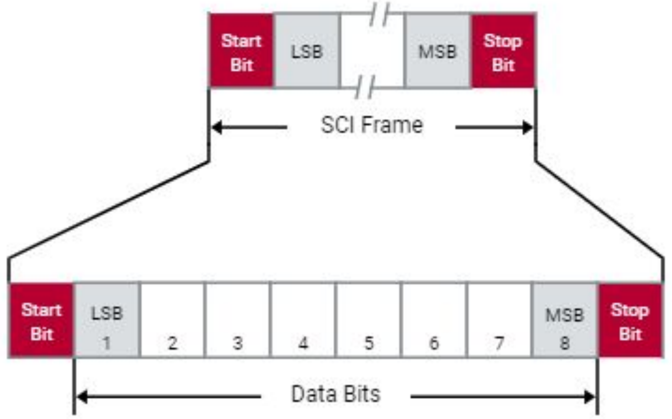
- Serial, Single wire Communication Protocol
- To maintain radiated electrical emissions within limits, the transmission rate is limited to 20 kBit/s
- Single Master Multi Slave
- The maximum recommended number of nodes is 16
- In frame synchronization
- Half Duplex Communication

LIN Transceiver

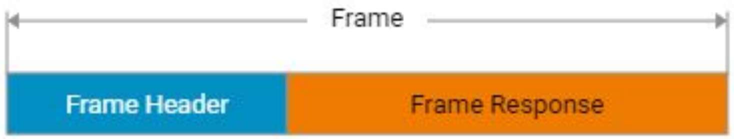
The transceiver is connected via the microcontroller's integrated serial interface. This is known as the SCI. Originally, a UART was used as a serial interface. There is now also microcontroller with Enhanced SCI (ESCI) or LIN SCI.



Serial data transmission in a cluster is byte-oriented. The SCI transmits the least significant bit (lsb) of each byte first, which is framed by a start bit and a stop bit. This combination of ten bits is referred to as an SCI frame. A message is composed of multiple SCI frames.



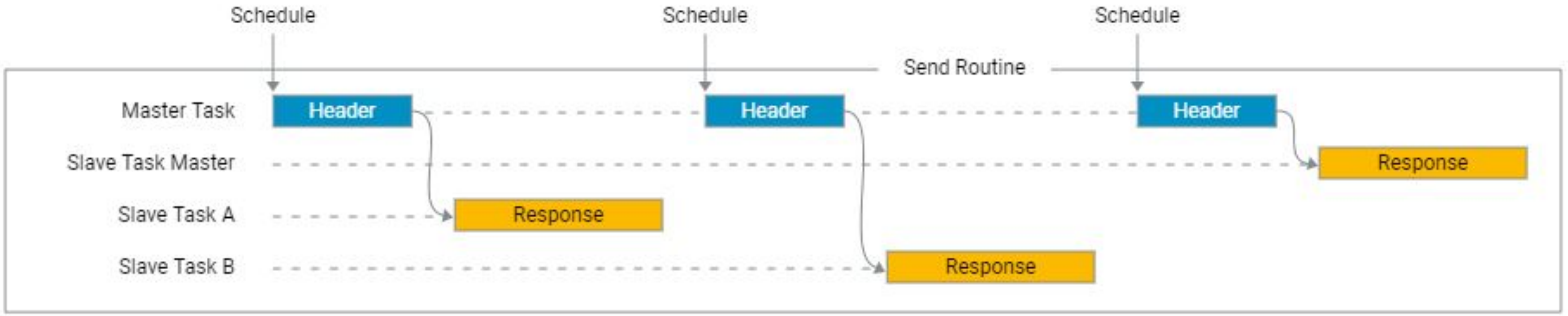
Communication Principle



Communication in a cluster is based on a master-slave architecture. This means that there is always one node which, as master, controls all communication between the various slaves. The control is implemented by only allowing the subordinate slaves to transmit information when they are requested to do so by the privileged master. The master does this by sending a request (frame header) on the bus, which the respective slave supplements with a response (frame response). This combination of request and response is known as a frame.

A disadvantage in central control by a master is that the master can fail, which results in failure of all communication. Therefore, the bus system is not suitable for safety-critical applications in which functionality must be guaranteed. Another weakness is that it is not equipped for event-driven communication. Since a request by the master is always necessary, the individual slaves cannot autonomously access the bus to send messages.

Communication Principle



Since there is no communication controller, the protocol is implemented in the form of software components on the microcontroller. It is through these that the specific node receives a master task or slave task which is needed to carry out the desired communication. In principle, each node has a slave task that serves to receive and send out information. The master has the additional master task which delegates the sending right and controls bus access.

Communication Principle

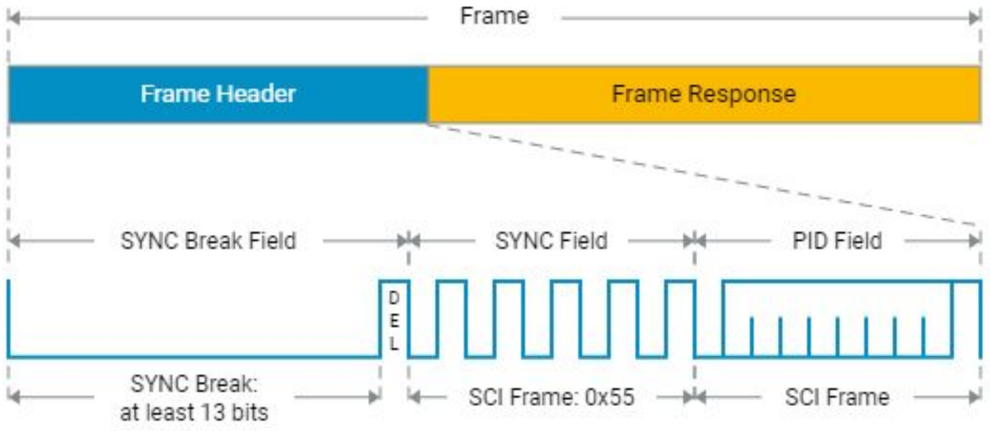
Master Characteristics

As soon as a network begins to operate, the master task is started in the master. It then begins to periodically process the Schedule that represents the desired sending scheme. In the Schedule, slots are defined for the individual messages. These slots must be at least large enough so that the Frame Header and Frame Response can be sent out. A complete frame is always transmittable in a slot.

Slave Characteristics

Response behaviors are defined for the slave tasks. Exactly how a slave task should react to a received header is defined, for example. Possible response behaviors are: sending, receiving or ignoring of a response. A sent response is in principle available for any slave task to receive. The desired response behaviors of the individual nodes are described in the LIN Description File (LDF).

LIN Header



All messages are initiated by the master. It does this by sending the header of a desired frame onto the bus. A frame header always consists of a Sync Break Field, the Sync Field and the PID Field.

The Sync Break Field is a unique sequence of bits by which the slaves recognize the beginning of a header. The master then sends its cycle frequency with the **Sync Field** that follows; the slaves must synchronize to this. The two fields therefore serve to initiate the data transmission and to implement time synchronization to the master.

After the Sync Field, the master transmits **the identifier** of the desired message. The identifiers reflect the communication relationships defined in the LDF. In addition, the response behaviors of the individual slaves are also defined. A slave can either behave passively, or it can send or receive a response. Since the identifier has an important meaning, it is protected with two parity bits P0 and P1. This combination of identifier and associated parity bits is the **Protected Identifier (PID)**.

LIN Header

The Sync Break consists of at least thirteen dominant bits and up to 18. Since cost-effective oscillators are used, the frequency may fluctuate by ±14 %. By transmitting at least 13 dominant bits, it is assured that a slow slave can also detect the start of a data transmission.

While the Sync Break has at least thirteen dominant bits the Sync Break Delimiter has at least one and a maximum of four recessive bits. This separates the Sync Break from the Sync Field that follows it, which begins with a dominant start bit due to the use of the SCI for serial communication.

To ensure that all slaves send and receive at the same clock time, the master transmits the Sync Field after the Sync Break Field. The Sync Field contains the value 0x55. To derive the clock, the slaves measure the time between the first and last falling edge of the Sync Field and divide it by eight. The result corresponds to one bit time.

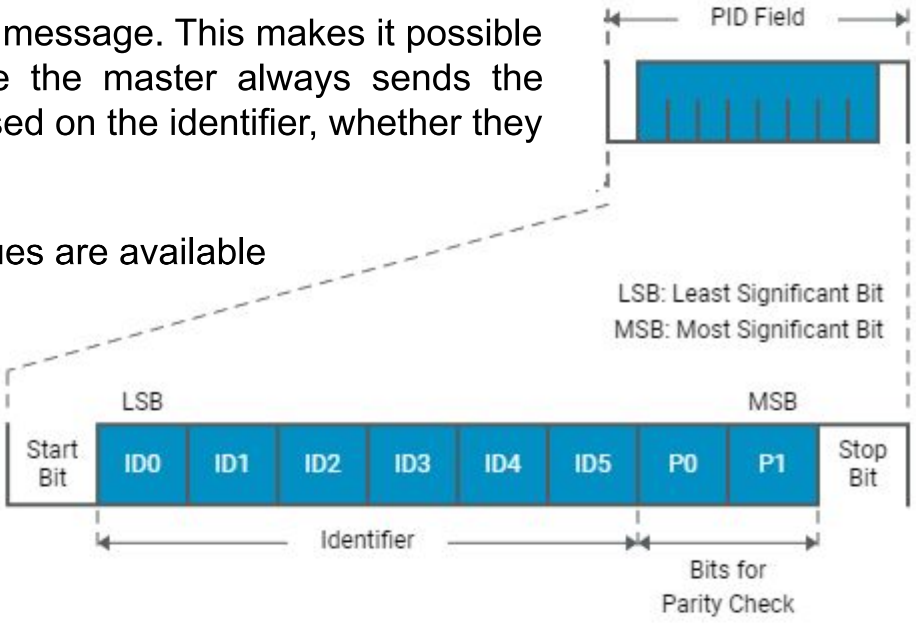
LIN Header

An identifier is used to assign a unique code to a message. This makes it possible to establish communication relationships. Since the master always sends the header on the bus, the slaves can determine, based on the identifier, whether they need to send out a response.

An identifier consists of six bits; therefore, 64 values are available

Four identifiers are used for predefined applications (Reserved Identifiers).

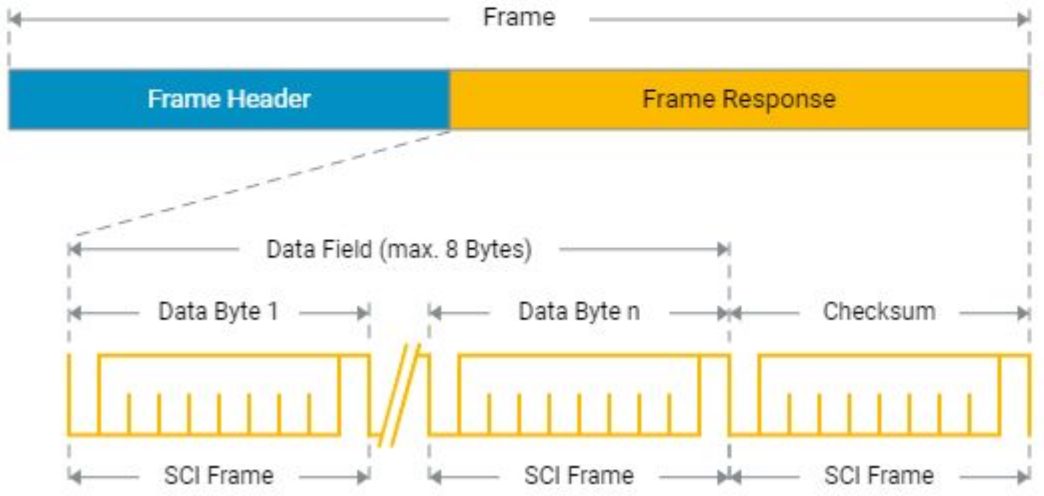
An identifier is protected by two parity bits. This combination is referred to as the PID. The computation of parity bits P0 and P1 is based on exclusive OR logic, where the parity bit P0 represents even parity and parity bit P1 represents odd parity.



P0	$ID0 \oplus ID1 \oplus ID2 \oplus ID4$
P1	$ID1 \oplus ID3 \oplus ID4 \oplus ID5$

LIN Response

A maximum of eight data bytes can be transmitted in a frame response. The transmission is made in ascending order from the least significant to the most significant byte. This also applies to the bits in the bytes. As a rule, a byte is transmitted from the least significant bit to the most significant bit.



A frame response is, in principle, available to be received by any node (broadcasting). Which node actually uses the data it contains is defined in the LDF. The useful data is protected by a checksum.

Note:
Between the header and the response there is a pause time. This is referred to as the response space (RS). During this time period, a node switches from the receiving state to the sending state.

Checksum

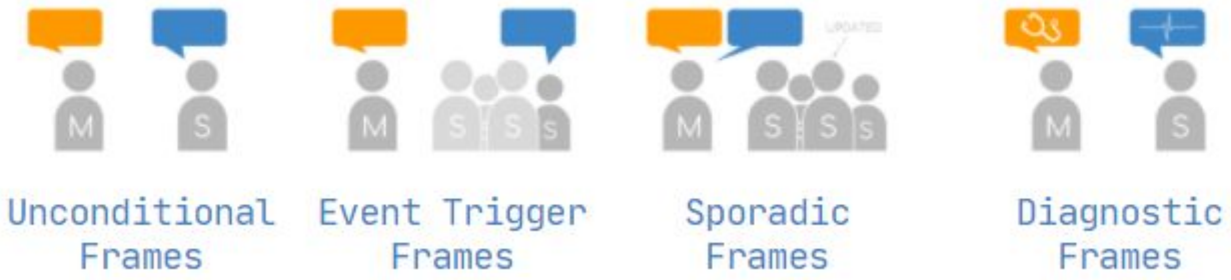
There are two checksum concepts: the **classic checksum** and the enhanced checksum. With the classic checksum, only the useful data is protected. With the enhanced checksum, the useful data and the PID are protected. The **enhanced checksum** is effective with Version 2.0 of the protocol.

The checksums are computed according to the following formula:

$$\boxed{\text{Checksum} = \text{INV}\ (\text{data byte } 1 \oplus \text{data byte } 2 \oplus \ldots \oplus \text{data byte } 8)}$$

This protection method is not particularly powerful. However, it achieves the data integrity required for use in the subbus area.

LIN Frame Types



Unconditional Frames

Event Trigger Frames

Sporadic Frames

Diagnostic Frames

www.imtschool.com

ww.facebook.com/imaketechnologyschool/