INTERNATIONAL STANDARD

ISO 17987-3

First edition
2016-08-15

Road vehicles — Local Interconnect Network (LIN) —

Part 3: Protocol specification

Véhicules routiers — Réseau Internet local (LIN) —

Partie 3: Spécification du protocole

Reference number
ISO 17987-3:2016(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

A list of all parts in the ISO 17987 series can be found on the ISO website.

Introduction

ISO 17987 (all parts) specifies the use cases, the communication protocol and physical layer requirements of an in-vehicle communication network called Local Interconnect Network (LIN).

The LIN protocol as proposed is an automotive focused low speed UART-based network (Universal Asynchronous Receiver Transmitter). Some of the key characteristics of the LIN protocol are signal-based communication, schedule table-based frame transfer, master/slave communication with error detection, node configuration and diagnostic service communication.

The LIN protocol is for low cost automotive control applications, for example, door module and air condition systems. It serves as a communication infrastructure for low-speed control applications in vehicles by providing the following:

— signal-based communication to exchange information between applications in different nodes;

— bit rate support from 1 kbit/s to 20 kbit/s;

— deterministic schedule table based frame communication;

— network management that wakes up and puts the LIN cluster into sleep mode in a controlled sanner;

— status management that provides error handling and error signalling;

— transport layer that allows large amount of data to be transmitted (such as diagnostic services);

— specification of how to handle diagnostic services;

— electrical physical layer specifications;

— node description language describing properties of slave nodes;

— network description file describing behaviour of communication;

— application programmer's interface.

ISO 17987 (all parts) is based on the Open Systems Interconnection (OSI) Basic Reference Model as specified in ISO/IEC 7498-1 which structures communication systems into seven layers.

The OSI model structures data communication into seven layers called (top down) *application layer* (layer 7), *presentation layer*, *session layer*, *transport layer*, *network layer*, *data link layer* and *physical layer* (layer 1). A subset of these layers is used in ISO 17987 (all parts).

ISO 17987 (all parts) distinguishes between the services provided by a layer to the layer above it and the protocol used by the layer to send a message between the peer entities of that layer. The reason for this distinction is to make the services, especially the application layer services and the transport layer services, reusable also for other types of networks than LIN. In this way, the protocol is hidden from the service user and it is possible to change the protocol if special system requirements demand it.

ISO 17987 (all parts) provides all documents and references required to support the implementation of the requirements related to.

— ISO 17987-1: This part provides an overview of ISO 17987 (all parts) and structure along with the use case definitions and a common set of resources (definitions, references) for use by all subsequent parts.

— ISO 17987-2: This part specifies the requirements related to the transport protocol and the network layer requirements to transport the PDU of a message between LIN nodes.

— ISO 17987-3: This part specifies the requirements for implementations of the LIN protocol on the logical level of abstraction. Hardware related properties are hidden in the defined constraints.

— ISO 17987-4: This part specifies the requirements for implementations of active hardware components which are necessary to interconnect the protocol implementation.

— ISO/TR 17987-5: This part specifies the LIN API (Application Programmers Interface) and the node configuration and identification services. The node configuration and identification services are specified in the API and define how a slave node is configured and how a slave node uses the identification service.

— ISO 17987-6: This part specifies tests to check the conformance of the LIN protocol implementation according to ISO 17987-2 and ISO 17987-3. This comprises tests for the data link layer, the network layer and the transport layer.

— ISO 17987-7: This part specifies tests to check the conformance of the LIN electrical physical layer implementation (logical level of abstraction) according to ISO 17987-4.

Road vehicles — Local Interconnect Network (LIN) —

Part 3:
Protocol specification

1 Scope

This document specifies the LIN protocol including the signal management, frame transfer, schedule table handling, task behaviour and status management and LIN master and slave node. It contains also OSI layer 5 properties according to ISO 14229-7 UDSonLIN-based node configuration and identification services (SID: $B0_{16}$ to $B8_{16}$) belonging to the core protocol specification.

A node (normally a master node) that is connected to more than one LIN network is handled by higher layers (i.e. the application) not within the scope of this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 17987-2:2016, *Road vehicles — Local Interconnect Network (LIN) — Part 2: Transport protocol and network layer services*

ISO 17987-4:2016, *Road vehicles — Local Interconnect Network (LIN) — Part 4: Electrical Physical Layer (EPL) specification 12V/24V*

ISO 17987-6:2016, *Road vehicles — Local Interconnect Network (LIN) — Part 6: Protocol conformance test specification*

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 17987-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
big-endian
method of storage of multi-byte numbers with the most significant bytes at the lowest memory addresses

3.1.2
broadcast NAD
addressing every slave node on LIN

3.1.3
bus interface
hardware (transceiver, UART, etc.) of a node that is connected to the physical bus wire in a *cluster* (3.1.6)

3.1.4
bus sleep state
state in which a node only expects an internal or external wake up

Note 1 to entry: The nodes switch output level to the recessive state.

[SOURCE: ISO 17987-2:2016, 6.1.2]

3.1.5
classic checksum
used for *diagnostic frames* (3.1.10) and legacy LIN slave nodes frames of protocol version 1.x

Note 1 to entry: The classic checksum considers the frame response data bytes only.

3.1.6
cluster
communication system comprising the LIN wire and all connected nodes

3.1.7
cluster design
process of designing the LDF

[SOURCE: ISO 17987-2:2016, 11.2.3]

3.1.8
data
response (3.1.27) part of a frame carrying one to eight *data bytes* (3.1.9)

3.1.9
data byte
one of the bytes in the data

3.1.10
diagnostic frame
master request frame (3.1.21) and the *slave response frame* (3.1.29)

3.1.11
diagnostic trouble code
DTC
value making reference to a specific fault in a system implemented in the server

3.1.12
enhanced checksum
checksum model used for all non-diagnostic frames and legacy 1.x LIN slave nodes

3.1.13
event-triggered frame
allowing multiple slave nodes to provide their *response* (3.1.27) to the same header

3.1.14
frame
communication entity consisting of a header and *response* (3.1.27)

3.1.15
frame identifier
value identifier uniquely a frame

3.1.16
frame slot
time period reserved for the transfer of a specific frame on LIN

Note 1 to entry: This corresponds to one entry in the schedule table.

3.1.17
go-to-sleep command
special *master request frame* (3.1.21) issued to force slave nodes to *bus sleep state* (3.1.4)

[SOURCE: ISO 17987-2:2016, 6.1.4]

3.1.18
header
first part of a frame consists of a break field, sync byte field and protected identifier; it is always sent by the LIN master node

3.1.19
LIN product identification
number containing the supplier and function and variant identification in a LIN slave node

3.1.20
little-endian
method of storage of multi-byte numbers with the least significant bytes at the lowest memory addresses

3.1.21
master request frame
diagnostic frames (3.1.10) issued by the master node *frame identifier* (3.1.15)

3.1.22
node capability file
NCF
file format describes a slave node as seen from the LIN network

3.1.23
operational state
slave node may transmit/receive frames in this state

[SOURCE: ISO 17987-2:2016, 5.1.2]

3.1.24
protected identifier
PID
8-bit value consisting of a unique 6-bit *frame identifier* (3.1.15) and 2-bit parity

3.1.25
publisher
node providing a frame response containing *signals* (3.1.30)

3.1.26
request
diagnostic frame (3.1.10) transmitted by the master node requesting data from a slave nodes

3.1.27
response
answer sent by a slave node on a diagnostic *request* (3.1.26)

3.1.28
service
combination of a diagnostic *request* (3.1.26) and *response* (3.1.27)

3.1.29
slave response frame
frame used for diagnostic communication sent by one of the slave nodes

3.1.30
signal
value or byte array transmitted in the *cluster* (3.1.6) using a *signal carrying frame* (3.1.31)

3.1.31
signal carrying frame
unconditional frames (3.1.34), *sporadic frames* (3.1.32), and *event-triggered frames* (3.1.13) containing *signals* (3.1.30)

3.1.32
sporadic frame
group of *unconditional frames* (3.1.34) assigned to a schedule slot published by the master node

3.1.33
subscriber
master or slave node that receives the data within a LIN frame

3.1.34
unconditional frame
frame carrying *signals* (3.1.30) that is always sent in its allocated *frame slot* (3.1.16)

3.2 Symbols

⊕ exclusive disjunction (exclusive or)

EXAMPLE a ⊕ b; exclusive or. True if exactly one of 'a' or 'b' is true.

¬ negation

∈ element of

EXAMPLE f ∈ S; belongs to. True if 'f' is in the set 'S'.

3.3 Abbreviated terms

API	application programmers interface
ASIC	application specific integrated circuit
DTC	diagnostic trouble code
LDF	LIN description file
LIN	Local Interconnect Network
LSB	least significant bit
MRF	master request frame
MSB	most significant bit
NAD	node address
NCF	node capability file
NRC	negative response code

NVRAM	non-volatile random access memory
OSI	Open Systems Interconnection
PCI	protocol control information
PDU	protocol data unit
PID	protected identifier
ROM	read only memory
RSID	response service identifier
SID	service identifier
SRF	slave response frame
TL	transport layer
VRAM	volatile RAM

4 Node concept

4.1 General

The LIN specifications assumes a software implementation of most functions, alternative realizations are promoted.

A node in a cluster interfaces to the physical bus wire using a frame transceiver. The frames are not accessed directly by the application; a signal-based interaction layer is added in between. As a complement, a transport layer (TL) interface exists between the application and the frame handler; see Figure 1.

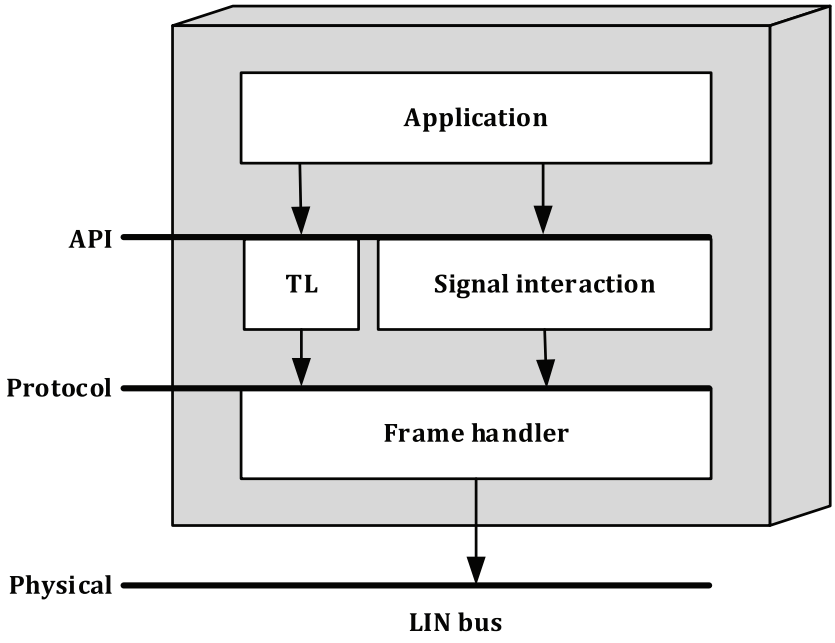


Figure 1 — Node concept

4.2 Concept of operation

4.2.1 Master and slave

A cluster consists of one master node and several slave nodes. A master node contains a master task as well as a slave task. All slave nodes contain a slave task only.

NOTE The slave task in the master node and in the slave node is not identical due to differences in PID handling.

A master node may participate in more than one cluster with one dedicated bus interfaces for each cluster. A sample cluster with one master node and two slave nodes is shown in Figure 2.

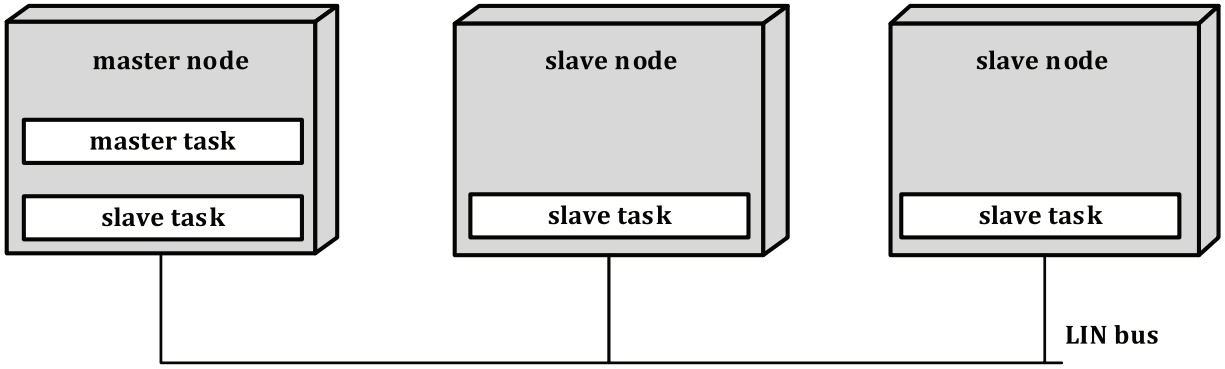


Figure 2 — Master and slave tasks

The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transmitted by each frame. Both, the master task and the slave task are parts of the frame handler.

4.2.2 Frames

A frame consists of a header (provided by the master task) and a response (provided by a slave task).

The header consists of a break field and sync byte field followed by a protected identifier. The protected identifier uniquely defines the purpose of the frame and node providing the response. The slave task of the node assigned as response transmitter provides the response. In case of diagnostic frames, not only the frame identifier but also the NAD assigns the transmitting node.

The response consists of a data field and a checksum field. The slave tasks interested in the data associated with the frame identifier receives the response, verifies the checksum and uses the data transmitted.

Figure 3 shows the LIN frame header and response fields.

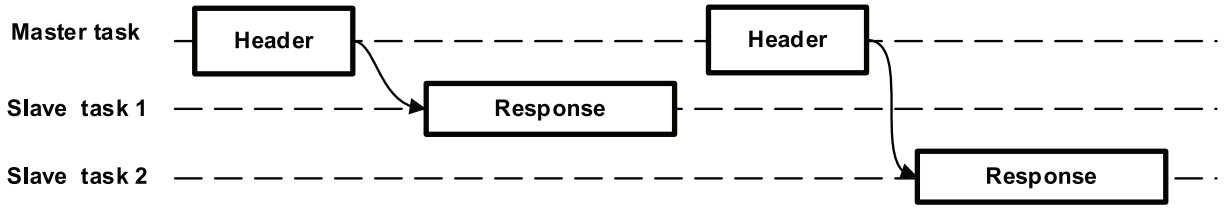


Figure 3 — LIN frame header and response fields

This results in the following desired features.

— System flexibility: Nodes can be added to the LIN cluster without requiring hardware or software changes in other slave nodes.

— Message routing: The content of a message is defined by the frame identifier (similar to CAN).

— Multicast: Any number of nodes can simultaneously receive and act upon a single frame.

4.2.3 Data transport

The following two types of data may be transmitted in a frame.

— Signals:

  Signals are scalar values or byte arrays that are packed into the data field of a frame. A signal is always present at the same position of the data field for all frames with the same frame identifier.

— Diagnostic messages:

  Diagnostic messages are transmitted in frames with two reserved frame identifiers. The interpretation of the data field depends on the data field itself as well as the state of the communicating nodes.

5 Protocol requirements

5.1 Signal

5.1.1 Management

A signal is transmitted in the data field of a frame.

5.1.2 Types

A signal is either a scalar value or a byte array.

A scalar signal is between 1 bit and 16 bit long. Scalar signals are treated as unsigned integers. A 1-bit scalar signal is called a Boolean signal.

A byte array is an array of one up to eight bytes.

Each signal shall have one publisher, i.e. it shall be always sent by the same node in the cluster. Zero, one or multiple nodes may subscribe to the signal.

All signals shall have initial values. The initial value for a published signal should be valid until the node writes a new value to this signal. The initial value for a subscribed signal should be valid until a new updated value is received from another node.

5.1.3 Consistency

Scalar signal writing or reading shall be atomic operations, i.e. it shall not be possible for an application to receive a signal value that is partly updated. This shall also apply to byte arrays. However, no consistency is guaranteed between any signals.

5.1.4 Packing

There is no restriction on packing scalar signals over byte boundaries. Each byte in a byte array shall map to a single frame byte starting with the lowest numbered data byte, see 5.2.2.6.

Several signals may be packed into one frame as long as they do not overlap each other.

NOTE Signal packing/unpacking is implemented more efficient in software-based nodes if signals are byte aligned and/or if they do not cross byte boundaries.

The same signal may be packed into multiple frames as long as the publisher of the signal is the same. If a node is receiving one signal packed into multiple frames, the latest received signal value is valid. Handling the same signal packed into frames on different LIN clusters is out of the scope.

5.1.5 Reception and transmission

The point in time when a signal is transmitted/received needs to be defined to help design tools and testing tools to analyse timing of signals. This means that all implementations behave in a predictable way.

The definitions below do not contain factors such as bit rate tolerance, jitter, buffer copy execution time, etc. These factors needs be taken into account to get a more detailed analysis. The intention for the definitions below is to have a basis for such analysis.

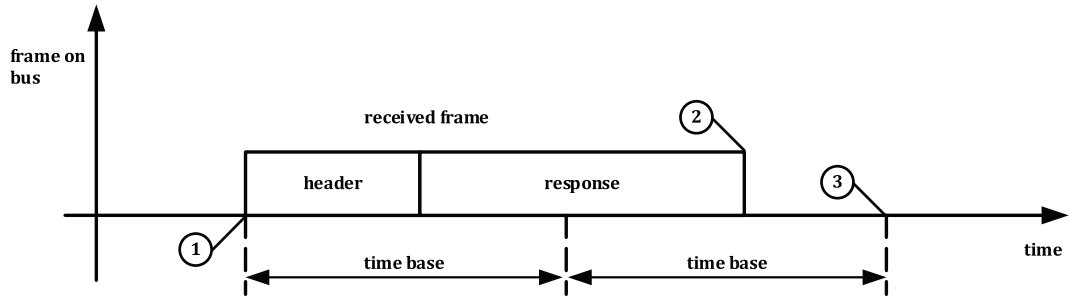
The timing is different for a master node and a slave node. The reason is that the master node controls the schedule and is aware of the due frame. A slave node gets this information first when the header is received.

The time base and time base tick is defined in 5.3.

A signal is considered received and available to the application as shown in Figure 4.

— Master node, at the next time base tick after the maximum frame length. The master node updates its received signals periodically at the time base start, i.e. at task level.

— Slave node, when the checksum for the received frame is validated. The slave node updates its received signals directly after the frame is finished, i.e. at interrupt level.



Key

1 time base tick

2 slave node – signal is available to the application

3 master node – time the signal is available to the application

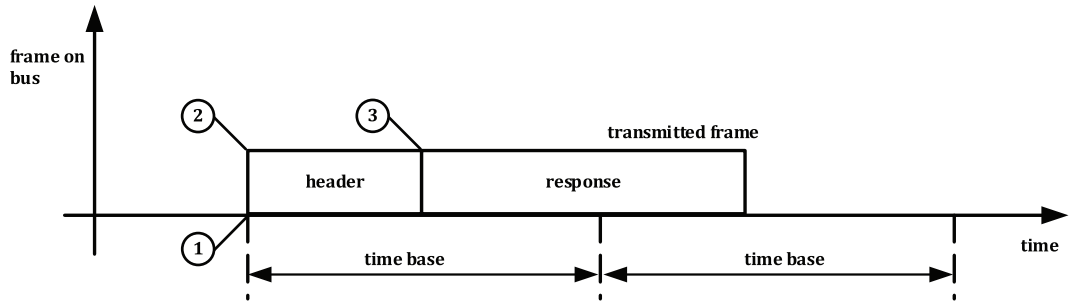
Figure 4 — Timing of signal reception

A signal is considered transmitted (latest point in time when the application may write to the signal).

— Master node – before the frame transmission is initiated.

— Slave node – when the ID for the frame is received.

Figure 5 shows the timing of signal transmission.



Key

1 time base tick

2 master – latest point the application can update the signal

3 slave – latest point the application can update the signal

Figure 5 — Timing of signal transmission

5.2 Frame

5.2.1 Transfer

The entities that are transferred on the LIN network are frames.

5.2.2 Structure

5.2.2.1 Definition of fields

The frame consists of a number of fields, one break field followed by four to eleven byte fields, labelled as in Figure 6. The time it takes to send a frame is the sum of the time to send each byte plus the response space and the inter-byte spaces.

The header starts at the falling edge of the break field and ends after the end of the stop bit of the protected identifier (PID) field. The response starts at the end of the stop bit of the PID field and ends at the after the stop bit of the checksum field.

The inter-byte space is defined to be the time between the end of the stop bit of the preceding field and the start bit of the following byte. The response space is defined to be the inter-byte space between the PID field and the first data field in the data. Both of them shall be non-negative.

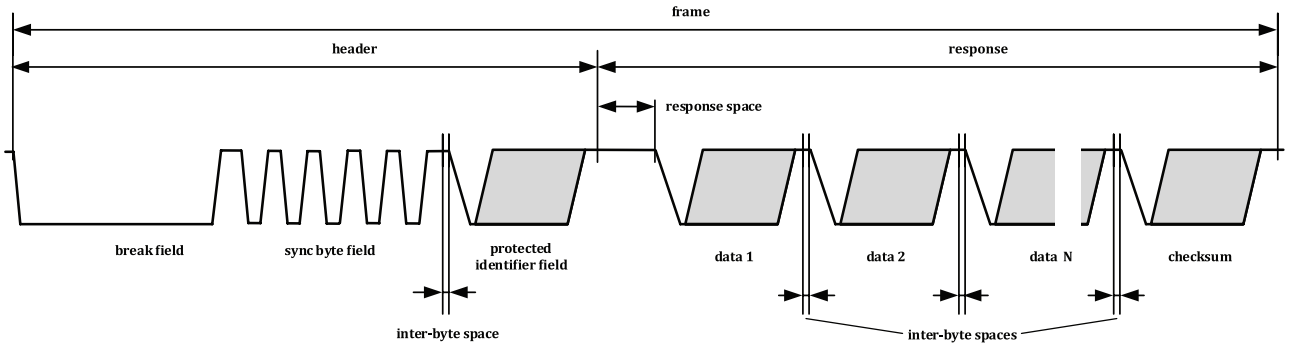


Figure 6 — Structure of a frame

5.2.2.2 Byte field

Each byte field, except the break field, is transmitted as the byte field as specified in Figure 7. The LSB of the data is sent first and the MSB last. The start bit shall be encoded as a bit with value zero (dominant) and the stop bit shall be encoded as a bit with value one (recessive).

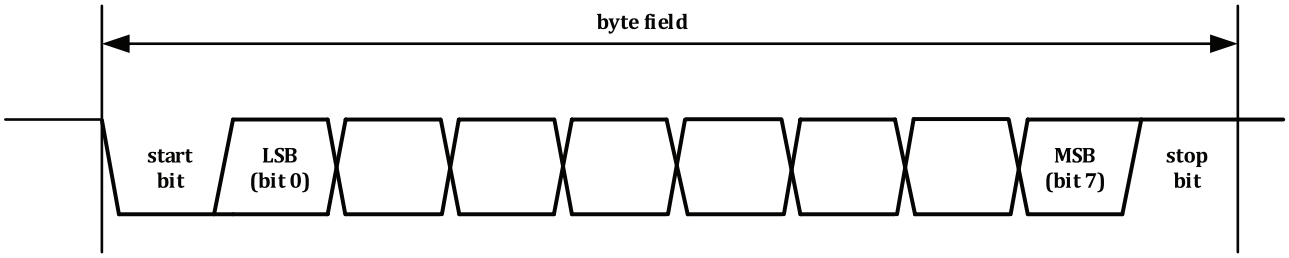


Figure 7 — Structure of a byte field

5.2.2.3 Break field

The break field is used to signal the beginning of a new frame. It is the only field that does not comply with Figure 7. A break field is always generated by the master task (in the master node) and it shall be at least 13 nominal bit times of dominant value, followed by a break delimiter, as shown in Figure 8. The break delimiter shall be at least one nominal bit time long. It should be sent with 2 bit length in order to avoid misinterpretation at receiving slave nodes.

NOTE A UART can only handle complete bits, so it can occur on the physical layer that the break delimiter is shorter than one bit time.

A slave node shall use a break detection threshold of 11 dominant local slave bit times. However, slave nodes without specific break field detection capabilities use a detection threshold of 9,5 T_{BIT} on the Rx pin. A slave node shall not check that the break delimiter is at least one nominal bit time long. A slave node shall be capable of detecting a break delimiter of at least 9/16 of a bit in length on the Rx line.

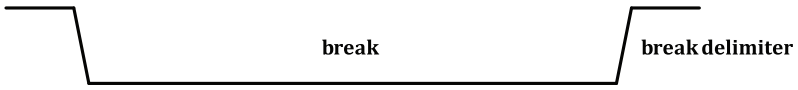


Figure 8 — Break field

5.2.2.4 Sync byte field

A slave task shall always be able to detect the break/sync byte field sequence, even if it expects a byte field (assuming the byte fields are separated from each other). When a break/sync byte field sequence occurs, the transfer in progress shall be aborted and processing of the new frame shall commence.

Sync byte field is a byte field with the data value 55_{16}, as shown in Figure 9.

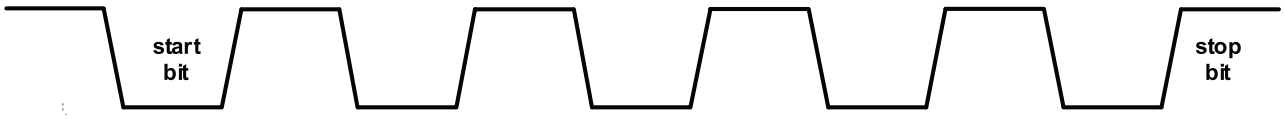


Figure 9 — Sync byte field

5.2.2.5 PID field

5.2.2.5.1 General

A protected identifier field consists of two sub-fields:

— the frame identifier;

— the parity.

Bit 0 to bit 5 are the frame identifier and bit 6 and bit 7 are the parity.

5.2.2.5.2 Frame identifier

Six bits (ID0 to ID5) are reserved for the frame identifier and values in the range of 0_{10} to 63_{10} shall be used. The frame identifiers are split into three categories:

— Values 0_{10} to 59_{10} are used for signal carrying frames;

— Values 60_{10} ($3C_{16}$) and 61_{10} ($3D_{16}$) are used to carry diagnostic and node configuration data;

— Values 62_{10} ($3E_{16}$) and 63_{10} ($3F_{16}$) are reserved for future protocol enhancements.

5.2.2.5.3 Parity

The parity (P0 and P1) is calculated on the frame identifier bits as shown in Formula (1) and Formula (2):

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \quad (1)$$

$$P1 = \neg(ID1 \oplus ID3 \oplus ID4 \oplus ID5) \quad (2)$$

5.2.2.5.4 Mapping

The mapping of the bits (ID0 to ID5 and P0 and P1) is shown in Figure 10.

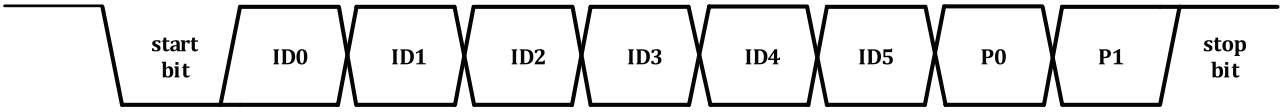


Figure 10 — Mapping of frame identifier and parity to the protected identifier field

5.2.2.6 Data

A frame carries between one and eight bytes of data. The number of data contained in a frame with a specific frame identifier shall be agreed by the publisher and all subscribers. A data byte is transmitted as part of a byte field, see Figure 7.

The data fields are labelled Data 1, Data 2, … up to maximum Data 8, see Figure 11.

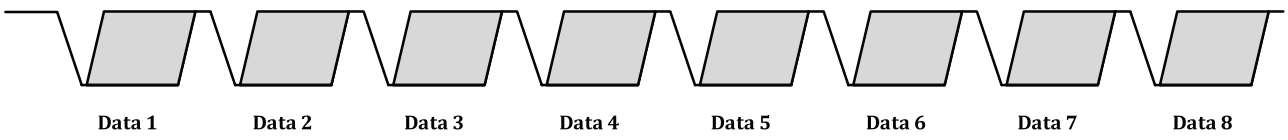


Figure 11 — Numbering of the data bytes in a frame with eight data bytes

How signals are mapped into frames is a decision that shall affect at least the whole LIN cluster.

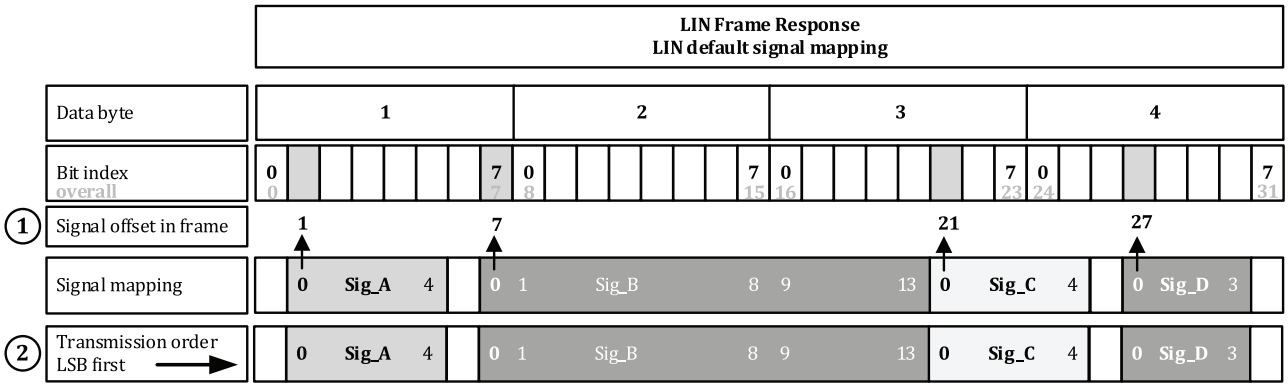
Beside the default signal mapping used in LIN clusters since its founding, an optional signal mapping variant is specified that allows to keep the big-endian signal encoding when routing frames from a back bone bus to a LIN cluster.

The two possible encodings are described in the following subclauses.

5.2.2.6.1 LIN default signal mapping to data bytes

For signals longer than one byte, the LSB signal entity is contained in the byte sent first and the MSB signal entity in the byte sent last (little-endian).

Figure 12 provides an example of the LIN default signal mapping into frame response data bytes with four signals.



Key

1 see parameter `signal_offset` in ISO 17987-2:2016, 12.3.3.2

2 transmission order is starting from least significant bit in least significant data byte

Figure 12 — LIN default signal mapping into frame response data bytes

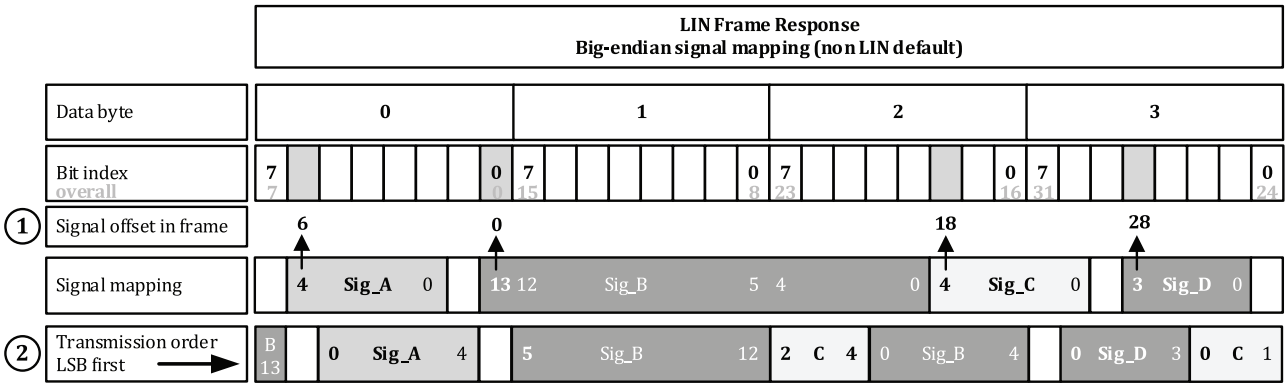
The LSB of each signal is referenced in the LDF and NCF frame definition as offset of the signal position. Key 1 values (Sig_A: 1, Sig_B: 7, Sig_C: 21 and Sig_D: 27) mark the offset for the four signals in the example.

5.2.2.6.2 Optional big-endian LIN signal mapping to data bytes variant

For signals longer than one byte, the MSB signal entity is contained in the byte sent first and the LSB signal entity in the byte sent last (big-endian).

Data byte transmission order, as well as the bit transmission order on LIN, is not changed due to big-endian signal encoding.

Figure 13 provides an example of the LIN big-endian signal mapping into frame response data bytes with four signals.



Key

1 see parameter `signal_offset` in ISO 17987-2:2016, 12.3.3.2

2 transmission order is starting from least significant bit in least significant data byte

Figure 13 — LIN big-endian signal mapping into frame response data bytes

The most significant bit of each signal is referenced in the LDF and NCF frame definition as offset of the signal position. Key 1 values (Sig_A: 6, Sig_B: 0, Sig_C: 18 and Sig_D: 28) mark the offset for the four signals in the example.

LIN node configuration commands AssignNAD, ReadByIdentifier and AssignFrameIdentifier use the 16-bit product identifier signals `supplier_id` and `function_id` in the request. As those commands have a LIN specific fixed format they are not affected by this big-endian signal encoding variant definition.

5.2.2.7 Checksum

The last field of a frame is the checksum. The checksum contains the inverted eight bit sum with carry over all data bytes (classic checksum) or all data bytes and the protected identifier (enhanced checksum).

Eight bit sum with carry is equivalent to the sum of all values and subtract 255_{10} every time the sum is greater or equal to 256_{10}. See A.3 for examples how to calculate the checksum.

Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for non-diagnostic communication.

The checksum is transmitted in a byte field, see Figure 7.

Use of classic or enhanced checksum is managed by the master node and it is determined per frame identifier.

Frame identifiers 60_{10} ($3C_{16}$) to 61_{10} ($3D_{16}$) shall always use classic checksum.

5.2.3 Frame length

The nominal value for transmission of a frame exactly matches the number of bits sent (no response space and no inter-byte spaces). The nominal break field is 14 nominal bits long (break is 13 nominal bits and break delimiter is 1 nominal bit).

Formula (3) defines T_{HEADER_MIN}.

$$T_{HEADER\_MIN} = 34\ T_{BIT} \quad (3)$$

where

T_{BIT} nominal time required to transmit a bit.

Formula (4) defines the $T_{RESPONSE\_MIN}$.

$$T_{RESPONSE\_MIN} = 10 * (N_{Data} + 1) * T_{BIT} \quad (4)$$

Formula (5) defines T_{FRAME_MIN}.

$$T_{FRAME\_MIN} = T_{HEADER\_MIN} + T_{RESPONSE\_MIN} \quad (5)$$

The break field is 14 nominal bits or longer, see 5.2.2.3. This means that T_{HEADER_MAX} puts a requirement on the maximum length of the break field.

The maximum space between the bytes is additional 40 % duration compared to the nominal transmission time. The additional duration is split between the header (the master task) and the frame response (a slave task).

Formula (6) defines T_{HEADER_MAX}:

$$T_{HEADER\_MAX} = 1{,}4 * T_{HEADER\_MIN} = 47{,}6\ T_{BIT} \quad (6)$$

Formula (7) defines $T_{RESPONSE\_MAX}$:

$$T_{RESPONSE\_MAX} = 1{,}4 * T_{RESPONSE\_MIN} \quad (7)$$

Formula (8) defines T_{FRAME_MAX}:

$$T_{FRAME\_MAX} = T_{HEADER\_MAX} + T_{RESPONSE\_MAX} \quad (8)$$

The maximum length of the header, response and frame is based on the nominal time for a frame (based on the F_{Nom} as defined in ISO 17987-4:2016, 5.1). Therefore, the bit tolerances are included in the maximum length.

EXAMPLE A master node that is 0,5 % slower than F_{Nom} is within 1,4 * T_{HEADER_MIN}.

All subscribing nodes shall be able to receive a frame that has a zero overhead, i.e. which is T_{FRAME_MIN} long.

Tools and tests shall check the T_{FRAME_MAX}. Nodes shall not check this time. Reception of a frame response after T_{FRAME_MAX} is not guaranteed. A receiving node can reject the frame.

5.2.4 Frame types

5.2.4.1 General

The frame type refers to the preconditions that shall be valid to transmit the frame. Some of the frame types are only used for specific purposes, which are also defined in the following subclauses. Not all frame types specified need to be used in a network.

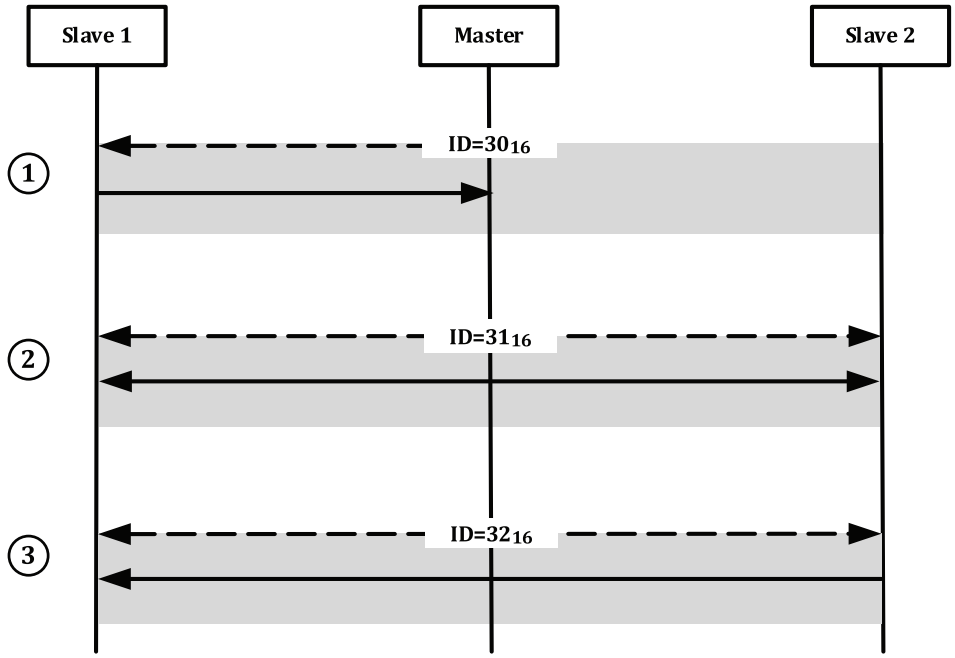
All bits not used/defined in a frame shall be recessive (ones).

5.2.4.2 Unconditional frame

Unconditional frames carry signals and their frame identifiers are in the range 0_{10} to 59_{10} ($3B_{16}$).

The header of an unconditional frame is always transmitted when a frame slot allocated to the unconditional frame is processed (by the master task). The publisher of the unconditional frame (the slave task) shall always provide the response to the header. All subscribers of the unconditional frame shall receive the frame and make it available to the application (assuming no errors were detected).

Figure 14 shows a sequence of three unconditional frames. A transfer is always initiated by the master. It has a single publisher and one or multiple subscribers.



Key

1 master requests a frame from slave 1
2 master sends a frame to both slaves
3 slave 2 sends a frame to slave 1

Figure 14 — Three unconditional frame transfers

5.2.4.3 Event-triggered frame

5.2.4.3.1 General

The purpose of an event-triggered frame is to lower the reaction time of the LIN cluster without assigning too much of the bus bandwidth to the polling of multiple slave nodes with seldom occurring events.

All subscribers of the event-triggered frame shall receive the frame and use its data (if the checksum is validated) as if the associated unconditional frame was received.

If the unconditional frame associated with an event-triggered frame is scheduled as an unconditional frame, the response shall always be transmitted (i.e. behave as a scheduled unconditional frame).

5.2.4.3.2 Unconditional frames associated with the event-triggered frame

Event-triggered frames carry the response of one or more unconditional frames. The unconditional frames associated with an event-triggered frame shall

— have equal length,

— use the same checksum type classic checksum or enhanced checksum,

— reserve the first data field to its protected identifier (even if the associated unconditional frame is scheduled as an unconditional frame in the same or another schedule table),

— be published by different slave nodes, and

— not be included directly in the same schedule table as the event-triggered frame is scheduled.

5.2.4.3.3 Transmission of the event-triggered frame

The header of an event-triggered frame is transmitted when a frame slot allocated to the event-triggered frame is processed. The publisher of an associated unconditional frame shall only transmit the response if at least one of the signals carried in its unconditional frame is updated. If the response is successfully transmitted, the signal is no longer considered to be updated.

If none of the slave nodes respond to the header, the rest of the frame slot is silent and the header is ignored.

If more than one slave node responds to the header in the same frame slot, a collision occurs.

5.2.4.3.4 Collision resolving

The master node shall resolve the collision in a collision resolving schedule table. Each event-triggered frame has an associated collision resolving schedule table. The switch to the collision resolving schedule is made automatically by the driver in the master node (i.e. not by the application). The collision resolving schedule shall be activated at the start of the subsequent frame slot after the collision.

At least all the associated unconditional frames shall be listed in this collisionresolving schedule table. The collision resolving schedule may contain other unconditional frames than the associated frames. These other unconditional frames may be of different length.

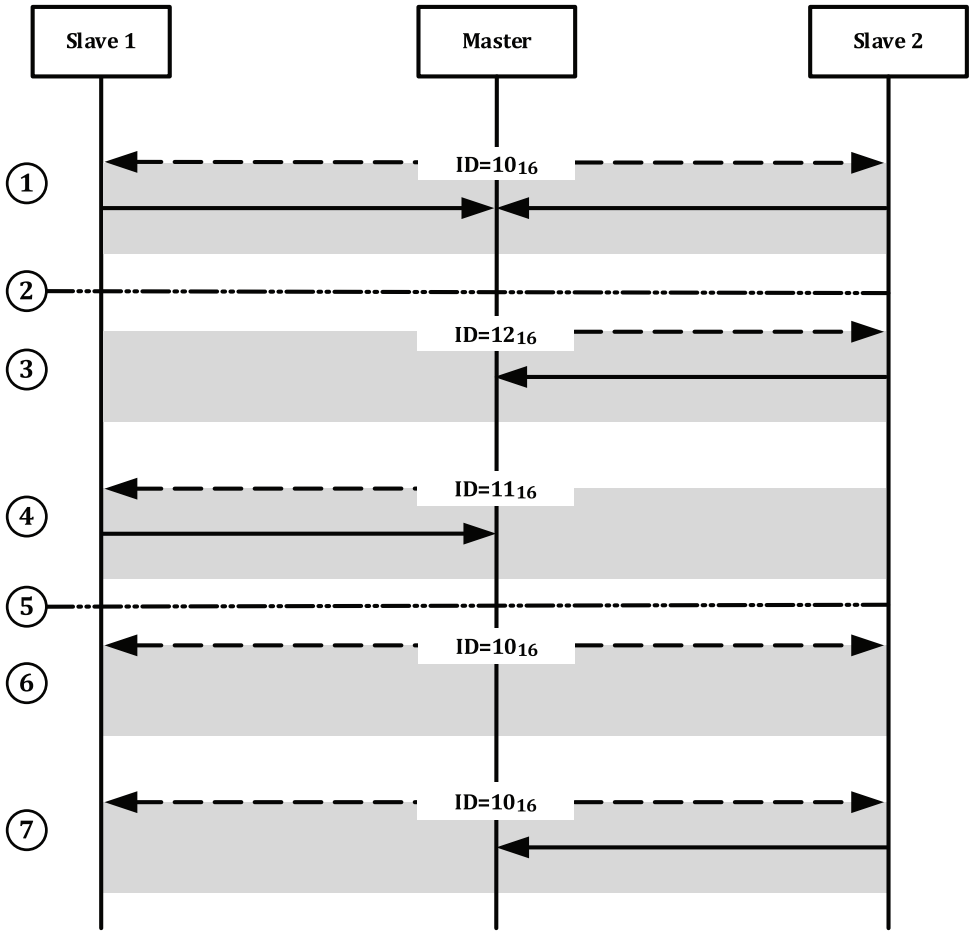
After the collision schedule table has been processed once, the driver in the master node shall switch back to the previous schedule table. It shall continue with the schedule entry subsequent to the schedule entry where the collision occurred (or first schedule entry in case the collision occurred in the last entry).

If one of the colliding slave nodes withdraws without corrupting the transfer, the master node does not detect this. A slave node that has withdrawn its response shall therefore retry transmitting its response in the next occurrence of the appropriate PID until successful; otherwise, the response is lost.

In case the master node application switches the schedule table before the collision is resolved, the collision resolving is lost. The new schedule table is activated as described in 5.3.4.

NOTE The colliding slave nodes still have their responses pending for transmission.

EXAMPLE 1 A schedule table contains only one event-triggered frame (ID = 10_{16}). The event-triggered frame is associated with two unconditional frames from slave 1 (ID = 11_{16}) and slave 2 (ID = 12_{16}). The collision resolving schedule table contains the two unconditional frames. See Figure 15 for the behaviour on the bus.



Key

1 request for event-triggered frame cause a collision, because both, slave 1 and slave 2 responded
2 automatic switch to collision resolving schedule table
3 frame from slave 2 is requested
4 frame from slave 1 is requested
5 automatic switch back to normal schedule table
6 none of the slave nodes has a new response to send
7 one of the slave nodes has a new response

Figure 15 — Event-triggered frame example

EXAMPLE 2 A typical use case for the event-triggered frame is to monitor the door knobs in a four door central locking system. By using an event-triggered frame to poll all four doors, the system shows good response times, while still minimizing the bus load. In the rare occasion that multiple passengers press a knob each, the system does not lose any of the pushes, but it takes additional time.

5.2.4.4 Sporadic frame

The purpose of sporadic frames is to blend some dynamic behaviour into the deterministic and real-time focused schedule table without losing the determinism in the rest of the schedule table.

A sporadic frame is a group of unconditional frames that share the same frame slot. When the sporadic frame slot is due for transmission, the unconditional frames are checked if they have any updated signals. If no signals are updated, no frame shall be transmitted and the frame slot is empty. If one signal (or more signals packed in the same frame) has been updated, the corresponding frame shall be transmitted. If more than one signal (packed in different frames) has been updated, the highest prioritized (see below) frame shall be transmitted. The candidate frames not transmitted shall not be

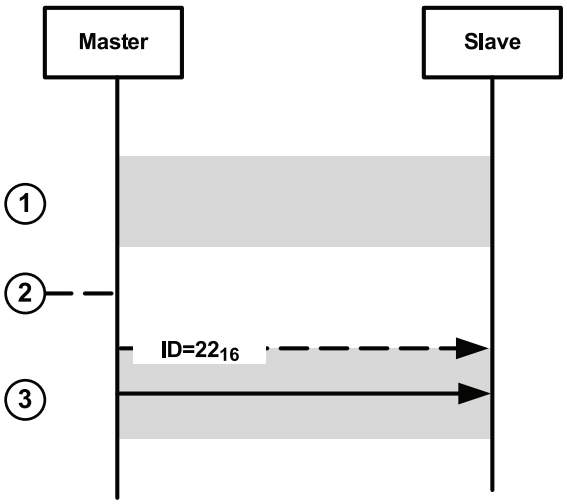
lost. They are candidates to be transmitted every time the sporadic frame is due, as long as they have not been transmitted.

If the unconditional frame was successfully transmitted, the unconditional frame shall no longer be pending for transmission until a signal is updated in the unconditional frame again.

Normally, multiple sporadic frames are associated with the same frame slot, the most prioritized of the pending unconditional frames shall be transmitted in the frame slot. If none of the unconditional frames is pending for transmission, the frame slot shall be silent. How the sporadic frames are prioritized is described in the LDF sporadic frame definition, see ISO 17987-2:2016, 12.3.3.3.

The master node is the only publisher of the unconditional frames in a sporadic frame.

EXAMPLE A sporadic frame is the only frame in the active schedule table. The sporadic frame has a number of associated unconditional frames, where one has ID 22_{16}. Normally, sporadic frame slots are empty. In the second slot (see Figure 16), at least one signal of the associated frame with ID 22_{16} is updated.



Key

1 master has nothing to send

2 something happens that updates a signal in frame 22_{16}

3 associated frame 22_{16} has an updated signal and is sent by the master

Figure 16 — Sporadic frame example

An unconditional frame associated with a sporadic frame may not be allocated in the same schedule table as the sporadic frame.

5.2.4.5 Diagnostic frames

Diagnostic frames always carry transport layer data and they shall always contain eight data bytes. The frame identifier shall be either $3C_{16}$ and called master request frame, or $3D_{16}$ and called slave response frame. The interpretation of the transport layer data bytes is described in ISO 17987-2:2016, Clause 8.

Before transmitting a master request frame, the master task queries its diagnostic module if it shall be transmitted or if the bus shall be silent for this schedule slot. A slave response frame header shall be sent unconditionally.

The slave task in a master node sends the response of a master request frame after the master request header has been transmitted.

The slave task in a slave node always receives a master request frame.

The slave task in a slave node sends the response of a slave response frame depending on the request received before and the state of the diagnostic module.

5.2.4.6 Reserved frames

Reserved frames shall not be used in a LIN cluster. Their frame identifiers are $3E_{16}$ and $3F_{16}$.

5.3 Schedule tables

5.3.1 General

A key property of the LIN protocol is the use of schedule tables. Schedule tables make it possible to assure that the bus is never overloaded. They are also the key component to guarantee the periodicity of signals.

Deterministic behaviour is made possible by the fact that all transfers in a LIN cluster are initiated by the master task. It is the responsibility of the master node to assure that all frames relevant in a mode of operation are given enough time to be transferred.

This subclause identifies all requirements that a schedule table shall adhere. The rationale for most of the requirements is to provide a conflict-free standard or to provide for a simple and efficient implementation of the LIN protocol.

5.3.2 Time definitions

The minimum time unit that is used in a LIN cluster is the time base (T_{BASE}). The time base is implemented in the master node and is used to control the timing of the schedule table. This means that the timing for the frames in a schedule table is based upon the time base. Usually, a time base is 5 ms or 10 ms.

The starting point of the time base is defined as the time base tick. A frame slot always starts at a time base tick.

The jitter specifies the differences between the maximum and minimum delay from time base tick to the header sending start point (falling edge of break field), see Figure 17.

The inter-frame space is the time from the end of the frame until start of the next frame. The inter-frame space shall be non-negative, see Figure 17.

5.3.3 Frame slot

Formula (9) is the time that is controlling the schedule table timing. It is the time from when a schedule table entry is due (a frame transmission is initiated) until the subsequent schedule entry is due. It is defined as an integer multiple of the time base. The integer multiple is normally different for each frame slot.

$$T_{FRAME\_SLOT} = T_{BASE} * n \qquad (9)$$

T_{FRAME_SLOT} [see Formula (9)] shall have a duration as defined in Formula (10) which is long enough to allow for the jitter introduced by the master task and the T_{FRAME_MAX} defined in Formula (10).

$$T_{FRAME\_SLOT} > jitter + T_{FRAME\_MAX} \qquad (10)$$

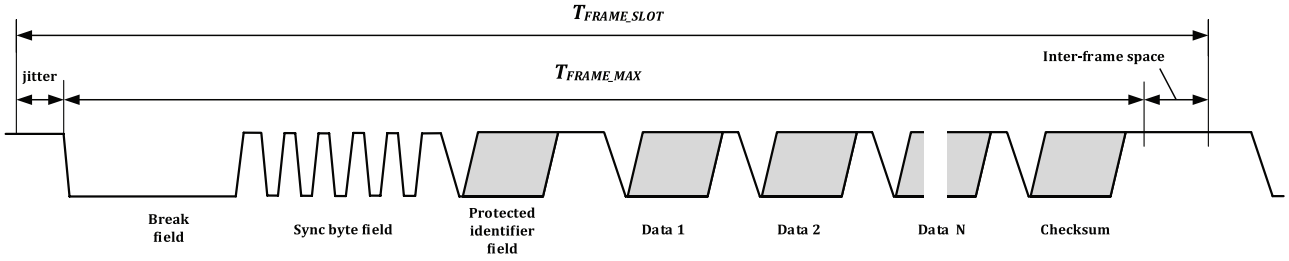


Figure 17 — Frame slot

5.3.4 Schedule table handling

The active schedule table shall be processed until another requested schedule table is selected. When the end of the current schedule is reached, the schedule is started again at the beginning of the schedule. The actual switch to the new schedule is made at start of a frame slot. This means that a schedule table switch request shall not interrupt any on-going transmission on the bus.

5.4 Task behaviour model

5.4.1 General

This subclause defines a behaviour model for a LIN node. The behaviour model is based on the master task/slave task concept.

5.4.2 Master task state machine

The master task is responsible for generating correct headers, i.e. deciding which frame shall be sent and for maintaining the correct timing between frames, all according to the schedule table. The master task state machine is shown in Figure 18.

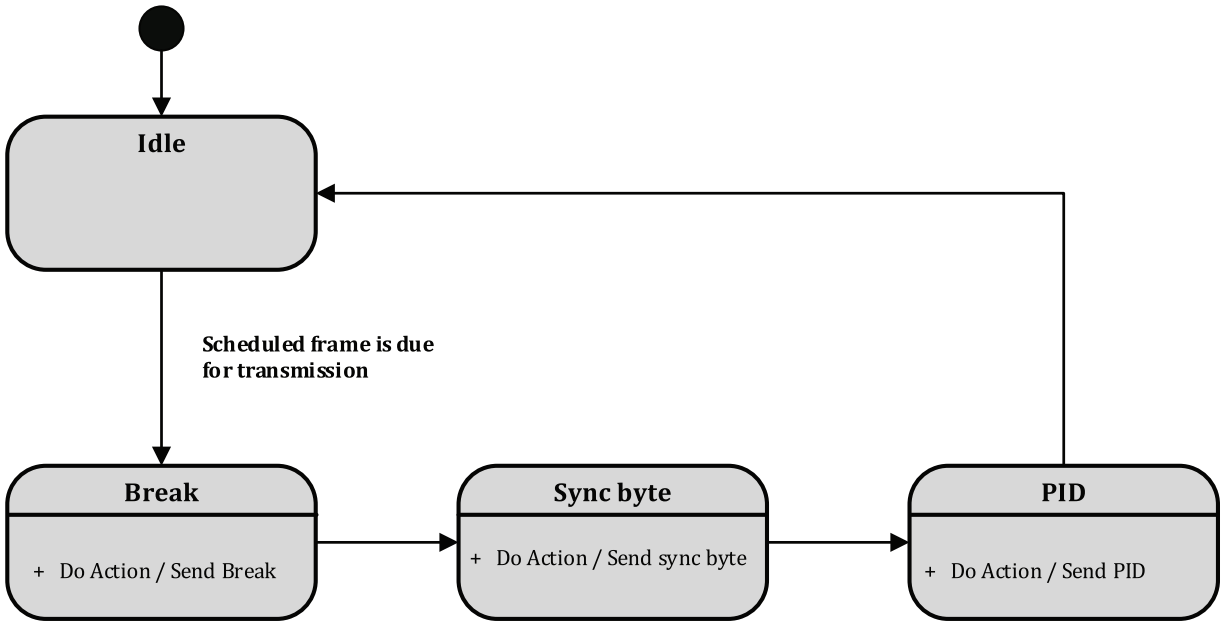


Figure 18 — Complete state machine for the master task

5.4.3 Slave task state machine

5.4.3.1 General

The slave task is responsible for transmitting the frame response when it is the publisher and for receiving the frame response when it is a subscriber. The slave task is modelled with two state machines:

— break/sync byte field sequence detector;

— frame processor.

5.4.3.2 Break/sync byte field sequence detector

A slave task shall be synchronized at the beginning of the protected identifier field of a frame, i.e. it shall be able to receive the protected identifier field correctly. It shall stay synchronized within the required bit-rate tolerance throughout the remainder of the frame, as specified in ISO 17987-4:2016, 5.1. For this purpose, every frame starts with a sequence starting with break field followed by a sync byte field. This sequence is unique in the whole LIN communication and provides enough information for any slave task to detect the beginning of a new frame and to be synchronized at the start of the identifier field.

5.4.3.3 Frame processor

The frame processing consists of two states: Idle and Active. Active contains five sub-states. As soon as a break/sync byte field sequence is received (from any state or sub-state), the Active state is entered in the PID sub-state. This implies that processing of one frame is aborted by the detection of a new break/sync byte field sequence. The frame processor state machine is shown in Figure 19.

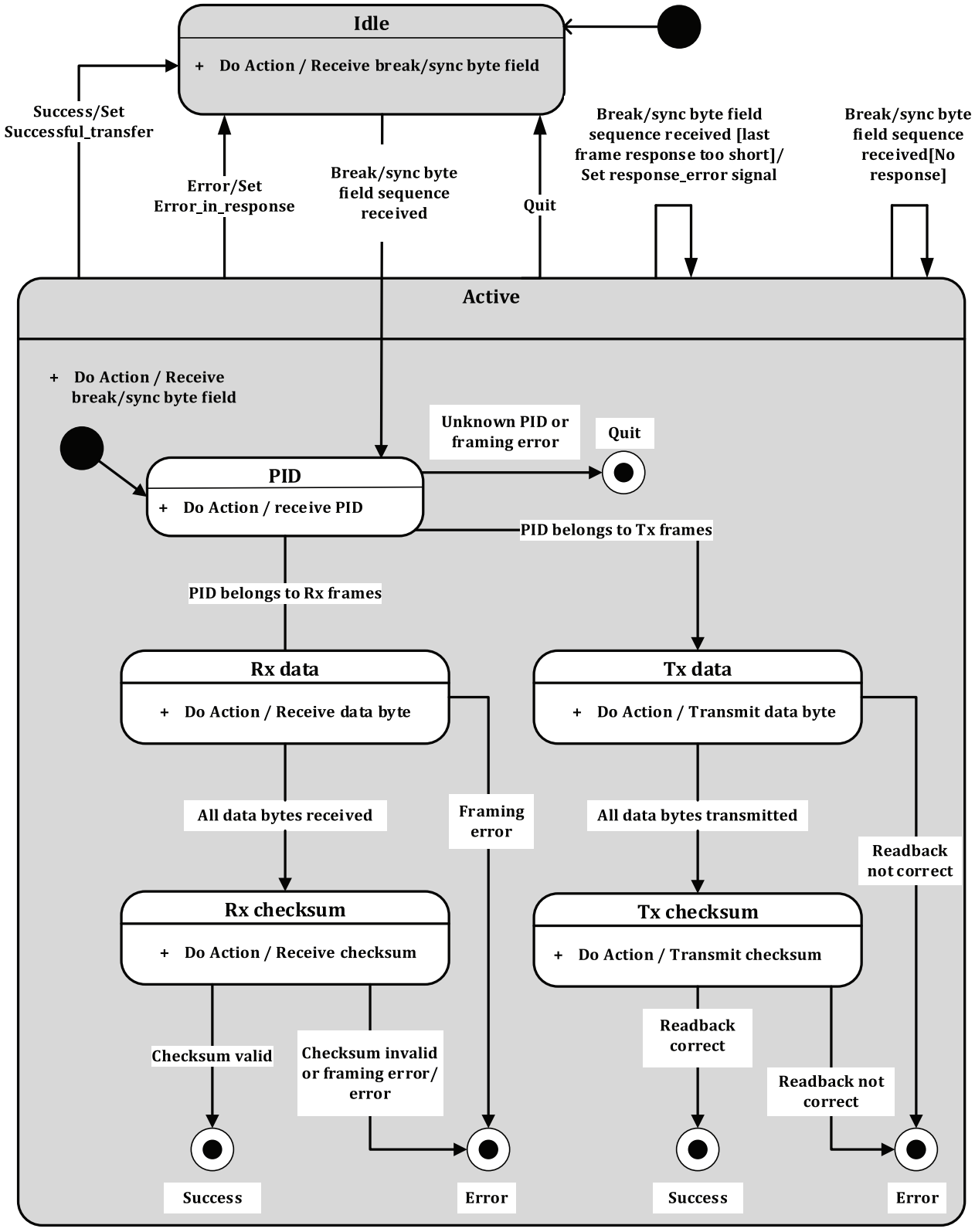


Figure 19 — Frame processor state machine

"Error" and "Success" refers to the status management described in 5.5.

The last frame response too short means that the last frame contained at least one field (correct data byte or even framing error) in the response. This is to distinguish between error in response and no response.

A mismatch between read back and sent data shall be detected not later than after completion of the byte field containing the mismatch. When a mismatch is detected, the transmission shall be aborted latest at the next byte boundary. A mismatch shall be detected in the data bits and also in the surrounding start and stop bit.

5.5 Status management

5.5.1 General

The purpose of status management is to detect errors during operation. The purpose of detecting errors is twofold:

— to provide means to easily replace faulty units;

— to provide for nodes to enter a limp home mode when problems occur.

In addition to the status management function mandated, a node may provide further detailed error information, which is not within the scope of this document.

5.5.2 Concept

Central cluster status management is made in the master node. The master node monitors status reports from each node and filters/integrates the reports to conclude if one or more nodes are faulty.

Each node application may also monitor its interaction with the LIN network. This may be used to enter a limp home mode, if applicable.

5.5.3 Event-triggered frames

Event-triggered frames, see 5.2.4.3, are defined to allow collisions. Therefore, a bus error, i.e. framing error, shall not affect the `response_error` signal (it is neither a successful transfer, nor an error in response). Of course, if an error in the associated unconditional frame occurs, this shall be counted as an error.

5.5.4 Reporting to the cluster

The master node may monitor the status on the cluster by checking the behaviour of a specific signal published by all slave nodes.

Each slave node shall publish a one bit scalar signal, named `response_error`, to the master node in one of its transmitted unconditional frames. In case the unconditional frame is associated with an event-triggered frame, the frame should additionally be scheduled as unconditional.

The `response_error` signal shall be set whenever a frame (except for event-triggered frame responses) that is transmitted or received by the slave node contains an error in the frame response.

The `response_error` signal shall be cleared when the unconditional frame containing the `response_error` signal is successfully transmitted.

The `response_error` signal shall not be set in case there is no response received (this refers to data and checksum). This frame is not considered to be received by any slave.

Based on this single bit, the master node shall make the conclusions as specified Table 1.

Table 1 — Interpretation of the `response_error` signal

`response_error`	Interpretation
false	the slave node is operating correctly
true	the slave node has intermittent problems
The slave node did not answer	the slave node, bus or master node has serious problem

It is the responsibility of the master node application to integrate and filter the individual status reports, as well as to do a synthesis of the reports from different slave nodes.

The `response_error` signal is enough to perform a conformance test of the frame transceiver (the protocol engine) independent of the application and the signal interaction layer.

A slave node may provide more status information, if desired, but the single `response_error` signal shall always be present.

5.5.5 Reporting within own node

This subclause applies to software-based nodes; however, ASIC-based state machine implementations are recommended to use the same concepts. See ISO 17987-5 `l_ifc_read_status()` for further information of this reporting.

The node provides two status bits for status management within the own node; `error_in_response` and `successful_transfer`. The own node application also receives the protected identifier of the last frame recognized by the node.

— `Error_in_response`:

  is set whenever a frame received by the node or a frame transmitted by the node contains an error in the response field, i.e. by the same condition as the `response_error` signal. It shall not be set in case there is no response.

— `Successful_transfer`:

  shall be set when a frame has been successfully transferred by the node, i.e. a frame has either been received or transmitted.

The reporting within the own node is described in ISO/TR 17987-5 and can be used to automatically generate applications that perform an automatic conformance test of the complete LIN driver module, including the signal interaction layer.

6 Node configuration and identification

6.1 General

The node configuration and identification services define how a slave node can be configured and how certain slave parameters are requested using the identification service.

The node configuration and identification services are transmitted by the transport layer as specified in ISO 17987-2.

Node configuration is used to set up slave nodes in a cluster. It is a set of services to avoid conflicts between slave nodes within a cluster built out of off-the-shelf slave nodes. Identification is used to identify a slave node.

Node configuration is done by having an address space, consisting of a LIN product identification and an initial NAD per slave node. It is possible to map unique frame identifiers to all frames transmitted in the cluster using these values.

6.2 LIN product identification

6.2.1 Supplier ID, function ID and variant ID

Each slave node shall have a LIN product identification, as specified in Table 2.

Table 2 — LIN product identification

D1	D2	D3	D4	D5
Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	Variant ID

The supplier ID is a 16-bit value, with the most significant bit equal to zero. Most significant bit set to one is reserved for future extended numbering systems. The supplier ID is assigned to each supplier by the Registration authority (http://www.iso.org/iso/maintenance_agencies) to maintain the supplier ID list. The supplier ID shall represent the supplier of the fully operational slave node.

The function ID is a 16-bit value assigned by each supplier. If two products differ in function, i.e. LIN communication or physical world interaction, their function ID shall differ. For absolutely equal function, however, the function ID shall remain unchanged.

The variant ID is an 8-bit value. It shall be changed whenever the product is changed but with unaltered function. The variant ID is a property of the slave node and not the LIN cluster.

6.2.2 Serial number

A slave node may have a serial number to identify a specific instance of a slave node product. The serial number is 4 bytes, as specified in Table 3.

Table 3 — Serial number

D1	D2	D3	D4
LSB	...	...	MSB

6.2.3 Wildcards

To be able to leave some information unspecified, the wildcard values specified in Table 4 may be used in node configuration requests. All slave nodes shall support the wildcards in requests.

Table 4 — Wildcards

Property	Wildcard value
NAD	$7F_{16}$
Supplier ID	$7FFF_{16}$
Function ID	$FFFF_{16}$

6.3 Slave node model

6.3.1 Memory model

The memory layout of a slave node is shown in Figure 20.

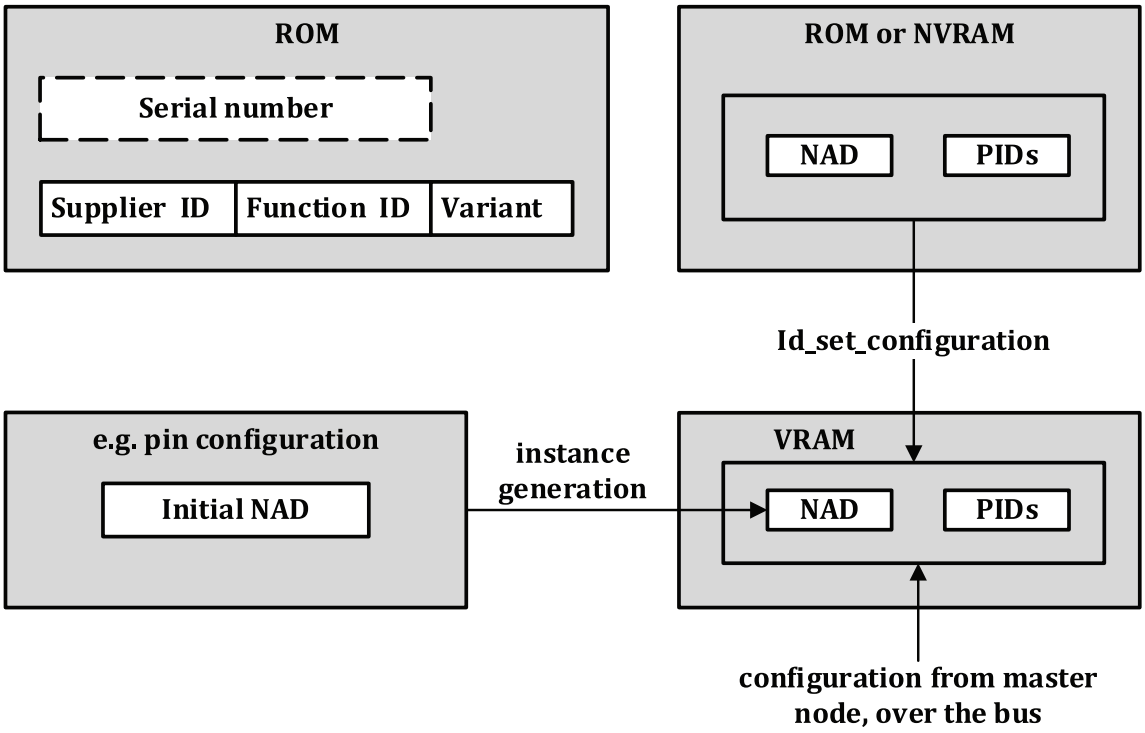


Figure 20 — Slave node memory model

Volatile RAM (VRAM) is considered a memory that is invalid after reset. The Non-Volatile RAM (NVRAM) is memory that is maintained after reset and can be modified with internal processes (i.e. the application). Read Only Memory (ROM) is considered as constant memory that cannot be modified with internal processes (i.e. application).

6.3.2 Slave node configuration variants

Three slave node configuration variants are defined.

a) Unconfigured slave node – After reset, the slave node does not contain a valid configuration. Therefore, the master shall configure the slave node. The configuration is stored in VRAM.

b) Preconfigured slave node – This slave node has a valid configuration after reset [after `l_ifc_init()` is called]. The configuration is normally stored in ROM, but reconfigured data are lost after reset.

c) Full configured slave node – The slave node stores the configuration in NVRAM, so it is still active after reset.

All variants of the slave nodes above shall understand at least the mandatory configuration services.

When a slave node enters operational state (see ISO 17987-2:2016, 5.1.2), it shall fulfill the following requirements for node configuration.

— It is up to the network designer and the LIN master to guarantee the supplier ID, function ID and configured NAD is unique. In case this can't be achieved ahead of communication, node configuration commands are mandatory to resolve conflicts.

— If the slave node does not contain a configuration, all frames (except the master request frame and slave response frame) in the slave node are marked as invalid.

— Understand and be able to process all supported configuration requests.

6.3.3 Initial node address

Each slave node has an initial NAD list, defined in the NCF, see ISO 17987-2:2016, 11.3.2. For slave nodes that have no instance generation of the initial NAD, the list contains only one entry. The instance generation shall set the initial NAD based on the initial NAD list. The instance generation of the initial NAD is not part of this specification.

The configuration, using `ld_set_configuration()` API or assign NAD request, set the NAD to the configured NAD. If the initial NAD is already equal to the configured NAD, then no action is taken.

Figure 21 shows the relationship between the initial NAD list, the initial NAD and the configured NAD.

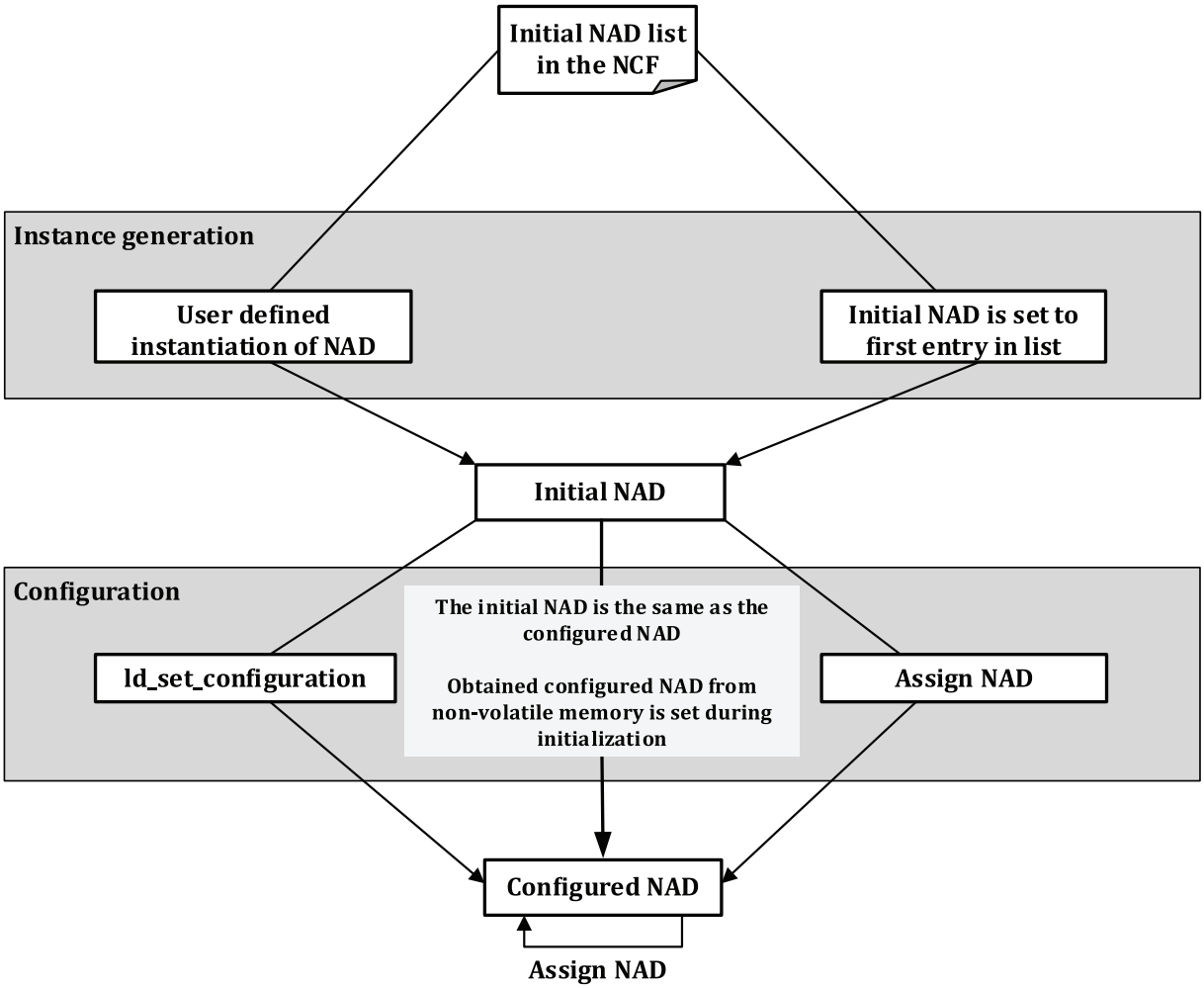


Figure 21 — NAD instantiation and configuration process

6.3.4 PDU structure

6.3.4.1 General

Requests are always sent in master request frames and responses are always sent in slave response frames.

The node configuration and identification services are transmitted as defined in ISO 17987-2 using the transport protocol and network layer services. Single frame (SF) shall be used for all requests and responses during the node configuration and identification services.

6.3.4.2 Node address

NAD is the address of the slave node being addressed in a request, i.e. only slave nodes have an address. NAD is also used to indicate the source of a response.

Table 5 specifies the NAD values.

Table 5 — NAD values

NAD value		Description
decimal	hex	
0	00_{16}	Reserved for go-to-sleep command, see ISO 17987-2:2016, 6.1.4.
1 to 125	01_{16} to $7D_{16}$	Slave node addresses (NAD).
126	$7E_{16}$	Functional node address (functional NAD), only used for diagnostics (using the transport layer).
127	$7F_{16}$	Slave node address broadcast (broadcast NAD).
128 to 255	80_{16} to FF_{16}	Diagnostic frames with the first byte in the range 80_{16} to FF_{16} are allocated for free usage.

There is a one-to-many mapping between a physical slave node and a logical slave node and it is addressed using the NAD. This means that one physical slave node may be composed of several logical slave nodes.

Functional addressing during configuration should not be used.

6.3.4.3 Protocol control information

The Protocol Control Information (PCI) is a transport protocol parameter and specified in ISO 17987-2. It contains the transport layer control information.

6.3.4.4 Service identifier

The Service Identifier (SID) specifies the request that shall be performed by the slave node addressed. The LIN protocol SID numbering range ($B0_{16}$ to $B8_{16}$) for node configuration and identification services is consistent with ISO 14229-7 defined SID ranges, see Table 6.

Table 6 — Node configuration and identification services

SID	Service name	Convention	Reference
$B0_{16}$	AssignNAD	optional	See 6.3.6.1
$B1_{16}$	AssignFrameIdentifier	optional	This service is mandatory for LIN master nodes with backward compatibility to LIN 2.0 slave nodes. For slave nodes according to LIN 2.1 or later (including ISO 17987), this service is obsolete.
$B2_{16}$	ReadByIdentifier	mandatory	See 6.3.6.6
$B3_{16}$	ConditionalChangeNAD	obsolete	None
$B4_{16}$	DataDump	optional	See 6.3.6.3
$B5_{16}$	Reserved for SAE J2602 TargetedReset	optional	SAE J2602 targeted reset. This service was used for auto addressing/slave node position detection. For legacy auto addressing nodes $B5_{16}$ can be used but the definition of use will be under $B8_{16}$.
$B6_{16}$	SaveConfiguration	optional	See 6.3.6.4

Table 6 *(continued)*

SID	Service name	Convention	Reference
$B7_{16}$	AssignFrameIdentifierRange	mandatory	See 6.3.6.5
$B8_{16}$	AutoAddressingSlave	optional	Auto addressing/slave node position detection, see Annex C.
$B9_{16}$ to BF_{16}	not applicable	reserved	Reserved for future use

6.3.4.5 Response service identifier

The Response Service Identifier (RSID) specifies the content of the response.

The RSID for a positive response is always SID + 40_{16}. The sending of positive responses is dependent on the definition for each service. By default, the positive response is assumed if all conditions are met (for example, supplier ID, function ID, PCI.length match).

The RSID for a negative response is always $7F_{16}$. The sending of negative responses is dependent on the definition for each service. By default, no negative response is assumed. The support of a specific service shall be listed in the NCF, see ISO 17987-2.

A slave shall process the configuration request immediately and be able to respond in the next schedule slave response frame (ST_{min} and P2, see ISO 17987-2, are not used for node configuration).

6.3.4.6 Data bytes D1 to D5

The interpretation of the data bytes (up to five in a node configuration PDU) depends on the SID or RSID.

If a PDU is not completely filled, the unused bytes shall be recessive, i.e. their value shall be FF_{16}. This is necessary since a diagnostic frame is always eight bytes in length.

6.3.5 Node configuration handling

All requests are carried in master request frames and all responses are carried in slave response frames. All requests and responses are using single frames only.

The slave node shall cancel a pending response after

— reception of a valid master request (except when NAD is the functional NAD), and

— if a N_As_{max} timeout occurs as described in ISO 17987-2.

The slave node shall proceed with a previous configuration request after the reception of an invalid master request (header error, checksum error, framing error and response time out).

A slave node shall remember the response for a request until a new request with a NAD or broadcast NAD (i.e. any NAD except a functional NAD) from the master node. If the master node transmits an unconditional frame between the request and the response, the slave node shall not forget the response.

A master node may choose not to ask for the response from the slave node, i.e. after transmitting a broadcast request ($7F_{16}$).

All services shall support the use of the wildcards, as defined in 5.2.3.

6.3.6 Node configuration services

6.3.6.1 AssignNAD service

The AssignNAD service is used to resolve conflicting NADs in LIN clusters built using off-the-shelf slave nodes or reused slave nodes. This request uses the initial NAD (or the NAD wildcard); this is to avoid

the risk of losing the address of a slave node. The NAD used for the response shall be the same as in the request, i.e. the initial NAD.

Table 7 specifies the AssignNAD request.

Table 7 — AssignNAD request message

NAD	PCI	SID	D1	D2	D3	D4	D5
Initial NAD	06_{16}	$B0_{16}$	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	New NAD

The service shall only be processed by the slave node if the service is supported and the PCI.length, the supplier ID, and the function ID match. The slave is not responsible for checking if the new NAD is within the valid range. A positive response shall only be sent if the requests is processed. The positive response is specified in Table 8.

Table 8 — AssignNAD positive response message

NAD	PCI	RSID	Unused bytes				
Initial NAD	01_{16}	$F0_{16}$	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

NOTE The response is using the initial NAD and not the new NAD.

A negative response shall never be sent by the slave node.

6.3.6.2 AssignFrameIdentifier service (LIN 2.0 specification)

Assign frame id is used to set a valid protected identifier to a frame specified by its message ID. Table 9 specifies the AssignFrameIdentifier request.

It is important to notice that the request provides the protected identifier, i.e. the identifier and its parity. Furthermore, frames with identifier 60_{10} ($3C_{16}$) and up cannot be changed (diagnostic frames, user-defined frames and reserved frames).

Table 9 — AssignFrameIdentifier request message

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	06_{16}	$B1_{16}$	Supplier ID LSB	Supplier ID MSB	Message ID LSB	Message ID MSB	Protected ID

The service shall only be processed by the slave node if the service is supported and the PCI.length, the supplier ID, and the message ID match. The slave shall not validate the assigned PID. An optional positive response shall only be sent if the request is processed.

The positive response frame is specified in Table 10.

Table 10 — AssignFrameIdentifier positive response message

NAD	PCI	RSID	Unused bytes				
NAD	01_{16}	$F1_{16}$	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

A negative response shall never be sent by the slave node.

6.3.6.3 DataDump service

The DataDump service is reserved for initial configuration of a slave node by the slave node supplier and the format of this message is supplier specific. This service shall only be used by supplier diagnostics and not in a running cluster, i.e. when LIN node is implemented in a vehicle.

Table 11 specifies the DataDump request.

Table 11 — DataDump request message

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	06_{16}	$B4_{16}$	User defined	User defined	User defined	User defined	User defined

The service shall only be processed by the slave node if the service is supported and the PCI.length matches. Table 12 specifies the positive response message from the slave.

Table 12 — DataDump positive response message

NAD	PCI	RSID	D1	D2	D3	D4	D5
NAD	06_{16}	$F4_{16}$	User defined	User defined	User defined	User defined	User defined

A negative response shall never be sent by the slave node.

6.3.6.4 SaveConfiguration service

This service is used to notify slave nodes to store the current configured PIDs and PIDs located in RAM to NVRAM. The slave application gathers the current configured NAD and PIDs from the data link layer and trigger the NVRAM write routine. The API `ld_read_configuration()` described in ISO 17987-5 may be used for this purpose. A configuration in the slave node may be valid even without the master node using this request, i.e. the slave node does not need to wait for this request to have a valid configuration.

Table 13 defines the SaveConfiguration request message.

Table 13 — SaveConfiguration request message

NAD	PCI	SID	Unused bytes				
NAD	01_{16}	$B6_{16}$	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

The service shall only be processed by the slave node if the service is supported and the PCI.length matches. A positive response shall only be sent if the request is processed. The slave shall not wait until the configuration is saved before a positive response is sent, i.e. the request is accepted by the slave node.

Table 14 specifies the positive response message from the slave.

Table 14 — SaveConfiguration positive response message

NAD	PCI	RSID	Unused bytes				
NAD	01_{16}	$F6_{16}$	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

A negative response shall never be sent by the slave node.

6.3.6.5 AssignFrameIdentifierRange service

The AssignFrameIdentifierRange service is used to set or disable PIDs up to four frames. Table 15 specifies the AssignFrameIdentifierRange request message.

It is important to notice that the request message provides the protected identifier, i.e. the frame identifier and its parity. Furthermore, frames with frame identifiers 60_{10} ($3C_{16}$) to 63_{10} ($3F_{16}$) shall not be changed (diagnostic frames and reserved frames).

Table 15 — AssignFrameIdentifierRange request message

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	06_{16}	$B7_{16}$	Start index	PID (index)	PID (index+1)	PID (index+2)	PID (index+3)

The service shall only be processed by the slave node if the PCI.length matches and the start index is consistent with the amount of PIDs supported in the node. A positive response shall only be sent if the request is processed.

The positive response message is specified in Table 16.

Table 16 — AssignFrameIdentifierRange positive response message

NAD	PCI	RSID	Unused bytes				
NAD	01_{16}	$F7_{16}$	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

The start index specifies which is the first frame to assign a PID. The order of the list is specified in the node attributes subclause in the NCF and LDF of the slave node, see ISO 17987-2:2016, 12.2.3 and ISO 17987-2:2016, 13.2.4.3. The first frame in the list shall have index 0_{10} (zero).

The PIDs are an array of four PID values that shall be used in the configuration request. Valid PID values here are the PID values for signal carrying frames, the "unassign PID" value 0_{10} (zero) and the "do not care" value FF_{16}. The "unassign PID" value is used to invalidate this frame for transmission on the bus. The "do not care" is used to keep the previous assigned value of this frame.

In case the slave cannot fulfill all "set PID" and "unassign PID" requests, the slave shall reject the request message and shall not sent a response. The "do not care" is always accepted by the receiving slave node.

The slave node does not validate the payload PIDs (i.e. validating the parity flags) beyond "do not care" pattern FF_{16}, the slave node relies on that the master sets the correct PIDs.

It is not necessary to unassign an already set PID in a slave node to be able to set a new PID for the same frame.

EXAMPLE 1 A slave node has five frames {power_status, IO_1, IO_2, IO_3, IO_4}. The master node application setup an AssignFrameIdentifer request message with the parameters.

— start index set to 1,

PID (index 1..4) set to {80_{16}, $C1_{16}$, 42_{16}, 00_{16}}.

When the slave node receives the request message, it sets the PIDs to {IO_1=80_{16}, IO_2=$C1_{16}$, IO_3=42_{16}, IO_4=unassigned}. The power_status frame is not affected. The slave responds with a positive response if requested.

EXAMPLE 2 A slave node has only two frames {status_frame, response_frame}. To assign PIDs to these two frames, the master application setups the following request.

— start index set to 0,

PID (index 0..3) set to {$C4_{16}$, 85_{16}, FF_{16}, FF_{16}}.

Since the slave node has only two frames the last two shall be set to do not care; otherwise, the request message fails.

A negative response shall never be sent by the slave node.

6.3.6.6 ReadByIdentifier service — Identification

It is possible to read the LIN product identification and other properties from a slave node using the ReadByIdentifier request message in Table 17.

Table 17 — ReadByIdentifier request message

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	06_{16}	$B2_{16}$	Identifier	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB

The service shall only be processed by the slave node if the PCI.length, the supplier ID, and the function ID match. A positive or negative response shall only be sent if the request is processed.

Table 18 defines the supported identifiers using the ReadByIdentifier request message.

Table 18 — Supported identifiers using the ReadByIdentifier request message

Identifier	Interpretation	Length of response
0_{10}	LIN product identification	RSID + 5
1_{10}	Serial number	RSID + 4
2_{10}	Used for bit timing test. Triggers a negative response defined in Table 20. Service is defined in the ISO 17987-6:2016, 8.15.	RSID + 2
3_{10}	Support of identifier 3_{10} (three) is mandatory if the optional NCF/LDF version is defined. If no NCF/LDF definition is available for the slave node, a negative response is provided. Legacy slave nodes don't support this identifier and do not provide a positive or negative response.	RSID + 5
4_{10} to 15_{10}	Reserved	—
16_{10} to 31_{10}	Supports 'Message ID' parameter of legacy LIN 2.0 nodes otherwise reserved.	—
32_{10} to 63_{10}	user defined	user defined
64_{10} to 255_{10}	reserved	—

Support of identifier 0_{10} (zero) is the only mandatory identifier, i.e. the serial number is optional.

If the slave successfully processed the ReadByIdentifier request message, it shall respond according to Table 19. Each row represents one possible response.

Table 19 — Possible ReadByIdentifier positive response message

lD	NAD	PCI	RSID	D1	D2	D3	D4	D5
0_{10}	NAD	06_{16}	$F2_{16}$	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	Variant
1_{10}	NAD	05_{16}	$F2_{16}$	Serial0, LSB	Serial1	Serial2	Serial3, MSB	FF_{16}
3_{10}	NAD	06_{16}	$F2_{16}$	Major Version – 8-bit int_major of LDF/NCF defini-tion (see ISO 17987-2)	Minor Version – 8-bit int_minor of LDF/NCF defini-tion (see ISO 17987-2)	Sub Version – 8-bit int_sub of LDF/NCF defini-tion (see ISO 17987-2)	Source: 01_{16}: LDF – the slave node was configured based on a LDF 02_{16}: NCF – the slave node was configured based on a NCF 00_{16}, 03_{16} to FF_{16} other values are reserved for future use.	Reserved, value is re-served for future use and trans-mitted with value 00_{16}
32_{10} to 63_{10}	NAD	02_{16}–06_{16}	$F2_{16}$	user defined	user defined	user defined	user defined	user defined

If the identifier in D1 is unknown or not supported, then a negative response shall be sent according to Table 20.

Table 20 — ReadByIdentifier negative response message

NAD	PCI	RSID	D1 (echo of SID)	D2	Unused bytes		
NAD	03_{16}	$7F_{16}$	Requested SID ($=B2_{16}$)	NRC = 12_{16} with sub-functionNotSupported	FF_{16}	FF_{16}	FF_{16}

Annex A
(normative)

Definition of properties, frame identifiers and various examples

A.1 Definition of numerical properties

Table A.1 defines the numerical properties.

Table A.1 — Defined numerical properties

Property	Min	Max	Unit	Reference
Scalar signal size	1	16	bit	see 5.1.2
Byte array size	1	8	byte	see 5.1.2
Break field length (dominant + delimiter)	14	27,6	T_{BIT}	see 5.2.2.3
Break detect threshold	11	11	T_{BIT}	see 5.2.2.3
Wake up signal duration	0,25	5	ms	see ISO 17987-2:2016, 5.1.3
Slave initialization time	—	100	ms	see ISO 17987-2:2016, 5.1.3
Silence period between wake up signals	150	250	ms	see ISO 17987-2:2016, 5.1.3
Silence period after three wake up signals	1,5	defined by application	s	see ISO 17987-2:2016, 5.1.3

A.2 Definition of valid frame identifiers

Table A.2 defines the valid frame identifiers.

Table A.2 — Valid frame identifiers

ID[0..5]		P0 =	P1 =	PID-Field								PID-Field	
Dec	Hex	ID0⊕ID1⊕ID2⊕ID4	¬ID1⊕ID3⊕ID4⊕ID5	P1	P0	5	4	3	2	1	0	Dec	Hex
0	00_{16}	0	1	1	0	0	0	0	0	0	0	128	80_{16}
1	01_{16}	1	1	1	1	0	0	0	0	0	1	193	$C1_{16}$
2	02_{16}	1	0	0	1	0	0	0	0	1	0	66	42_{16}
3	03_{16}	0	0	0	0	0	0	0	0	1	1	3	03_{16}
4	04_{16}	1	1	1	1	0	0	0	1	0	0	196	$C4_{16}$
5	05_{16}	0	1	1	0	0	0	0	1	0	1	133	85_{16}
6	06_{16}	0	0	0	0	0	0	0	1	1	0	6	06_{16}
7	07_{16}	1	0	0	1	0	0	0	1	1	1	71	47_{16}
8	08_{16}	0	0	0	0	0	0	1	0	0	0	8	08_{16}
9	09_{16}	1	0	0	1	0	0	1	0	0	1	73	49_{16}
10	$0A_{16}$	1	1	1	1	0	0	1	0	1	0	202	CA_{16}
11	$0B_{16}$	0	1	1	0	0	0	1	0	1	1	139	$8B_{16}$
12	$0C_{16}$	1	0	0	1	0	0	1	1	0	0	76	$4C_{16}$

[a] Frame identifier 60_{10} ($3C_{16}$) is reserved for the master request frame (see 5.2.4.5).

[b] Frame identifier 61_{10} ($3D_{16}$) is reserved for the slave response frame (see 5.2.4.5).

[c] Frame identifier 62_{10} ($3E_{16}$) and 63_{d} ($3F_{16}$) are reserved for a future LIN extended format (see 5.2.4.6).

Table A.2 *(continued)*

ID[0..5]		P0 =	P1 =	PID-Field								PID-Field	
Dec	Hex	ID0⊕ID1⊕ID2⊕ID4	¬ID1⊕ID3⊕ID4⊕ID5	P1	P0	5	4	3	2	1	0	Dec	Hex
13	$0D_{16}$	0	0	0	0	0	0	1	1	0	1	13	$0D_{16}$
14	$0E_{16}$	0	1	1	0	0	0	1	1	1	0	142	$8E_{16}$
15	$0F_{16}$	1	1	1	1	0	0	1	1	1	1	207	CF_{16}
16	10_{16}	1	0	0	1	0	1	0	0	0	0	80	50_{16}
17	11_{16}	0	0	0	0	0	1	0	0	0	1	17	11_{16}
18	12_{16}	0	1	1	0	0	1	0	0	1	0	146	92_{16}
19	13_{16}	1	1	1	1	0	1	0	0	1	1	211	$D3_{16}$
20	14_{16}	0	0	0	0	1	0	0	1	0	0	20	14_{16}
21	15_{16}	1	0	0	1	0	1	0	1	0	1	85	55_{16}
22	16_{16}	1	1	1	1	0	1	0	1	1	0	214	$D6_{16}$
23	17_{16}	0	1	1	0	0	1	0	1	1	1	151	97_{16}
24	18_{16}	1	1	1	1	0	1	1	0	0	0	216	$D8_{16}$
25	19_{16}	0	1	1	0	0	1	1	0	0	1	153	99_{16}
26	$1A_{16}$	0	0	0	0	0	1	1	0	1	0	26	$1A_{16}$
27	$1B_{16}$	1	0	0	1	0	1	1	0	1	1	91	$5B_{16}$
28	$1C_{16}$	0	1	1	0	0	1	1	1	0	0	156	$9C_{16}$
29	$1D_{16}$	1	1	1	1	0	1	1	1	0	1	221	DD_{16}
30	$1E_{16}$	1	0	0	1	0	1	1	1	1	0	94	$5E_{16}$
31	$1F_{16}$	0	0	0	0	0	1	1	1	1	1	31	$1F_{16}$
32	20_{16}	0	0	0	0	1	0	0	0	0	0	32	20_{16}
33	21_{16}	1	0	0	1	1	0	0	0	0	1	97	61_{16}
34	22_{16}	1	1	1	1	1	0	0	0	1	0	226	$E2_{16}$
35	23_{16}	0	1	1	0	1	0	0	0	1	1	163	$A3_{16}$
36	24_{16}	1	0	0	1	1	0	0	1	0	0	100	64_{16}
37	25_{16}	0	0	0	0	1	0	0	1	0	1	37	25_{16}
38	26_{16}	0	1	1	0	1	0	0	1	1	0	166	$A6_{16}$
39	27_{16}	1	1	1	1	1	0	0	1	1	1	231	$E7_{16}$
40	28_{16}	0	1	1	0	1	0	1	0	0	0	168	$A8_{16}$
41	29_{16}	1	1	1	1	1	0	1	0	0	1	233	$E9_{16}$
42	$2A_{16}$	1	0	0	1	1	0	1	0	1	0	106	$6A_{16}$
43	$2B_{16}$	0	0	0	0	1	0	1	0	1	1	43	$2B_{16}$
44	$2C_{16}$	1	1	1	1	1	0	1	1	0	0	236	EC_{16}
45	$2D_{16}$	0	1	1	0	1	0	1	1	0	1	173	AD_{16}
46	$2E_{16}$	0	0	0	0	1	0	1	1	1	0	46	$2E_{16}$
47	$2F_{16}$	1	0	0	1	1	0	1	1	1	1	111	$6F_{16}$
48	30_{16}	1	1	1	1	1	1	0	0	0	0	240	$F0_{16}$
49	31_{16}	0	1	1	0	1	1	0	0	0	1	177	$B1_{16}$
50	32_{16}	0	0	0	0	1	1	0	0	1	0	50	32_{16}
51	33_{16}	1	0	0	1	1	1	0	0	1	1	115	73_{16}
52	34_{16}	0	1	1	0	1	1	0	1	0	0	180	$B4_{16}$

[a] Frame identifier 60_{10} ($3C_{16}$) is reserved for the master request frame (see 5.2.4.5).

[b] Frame identifier 61_{10} ($3D_{16}$) is reserved for the slave response frame (see 5.2.4.5).

[c] Frame identifier 62_{10} ($3E_{16}$) and 63_{d} ($3F_{16}$) are reserved for a future LIN extended format (see 5.2.4.6).

Table A.2 *(continued)*

ID[0..5]		P0 =	P1 =	PID-Field								PID-Field	
Dec	Hex	ID0⊕ID1⊕ID2⊕ID4	¬ID1⊕ID3⊕ID4⊕ID5	P1	P0	5	4	3	2	1	0	Dec	Hex
53	35_{16}	1	1	1	1	1	1	0	1	0	1	245	$F5_{16}$
54	36_{16}	1	0	0	1	1	1	0	1	1	0	118	76_{16}
55	37_{16}	0	0	0	0	1	1	0	1	1	1	55	37_{16}
56	38_{16}	1	0	0	1	1	1	1	0	0	0	120	78_{16}
57	39_{16}	0	0	0	0	1	1	1	0	0	1	57	39_{16}
58	$3A_{16}$	0	1	1	0	1	1	1	0	1	0	186	BA_{16}
59	$3B_{16}$	1	1	1	1	1	1	1	0	1	1	251	FB_{16}
60[a]	$3C_{16}$	0	0	0	0	1	1	1	1	0	0	60	$3C_{16}$
61[b]	$3D_{16}$	1	0	0	1	1	1	1	1	0	1	125	$7D_{16}$
62[c]	$3E_{16}$	1	1	1	1	1	1	1	1	1	0	254	FE_{16}
63[c]	$3F_{16}$	0	1	1	0	1	1	1	1	1	1	191	BF_{16}

[a] Frame identifier 60_{10} ($3C_{16}$) is reserved for the master request frame (see 5.2.4.5).

[b] Frame identifier 61_{10} ($3D_{16}$) is reserved for the slave response frame (see 5.2.4.5).

[c] Frame identifier 62_{10} ($3E_{16}$) and 63_{d} ($3F_{16}$) are reserved for a future LIN extended format (see 5.2.4.6).

A.3 Example of checksum calculation

Below the checksum calculation of four bytes (Data = $4A_{16}$, 55_{16}, 93_{16}, $E5_{16}$) is shown. If the frame has four data bytes or the protected identifier and three data bytes, the calculation is the same.

Table A.3 defines an example of checksum calculation.

Table A.3 — Example of checksum calculation

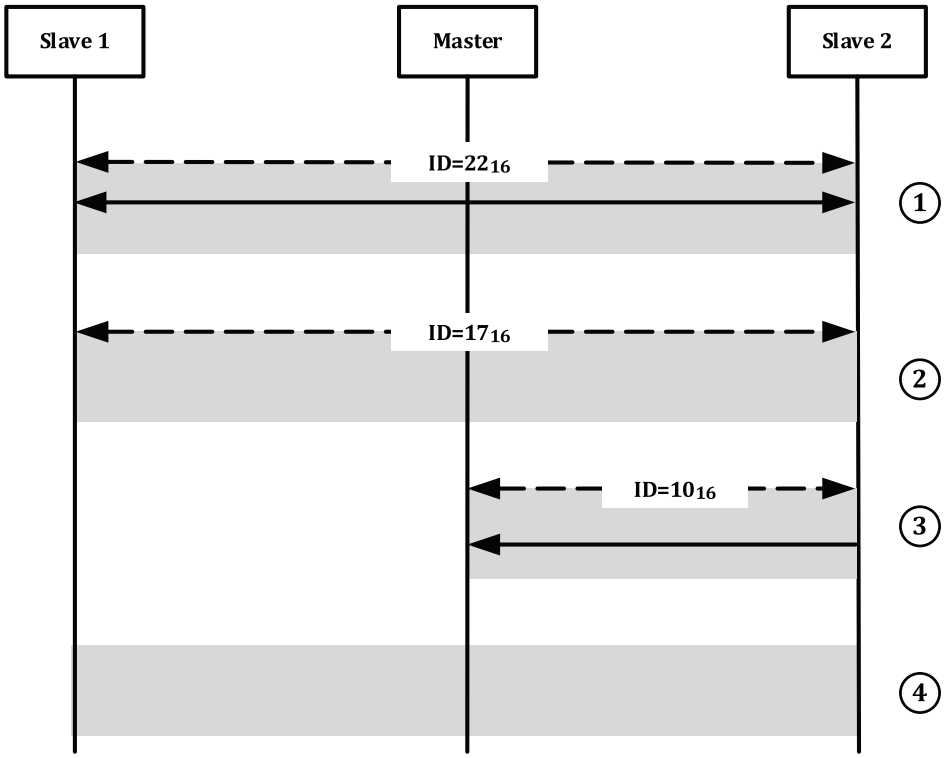
Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
$4A_{16}$	$4A_{16}$	0	0	1	0	0	1	0	1	0
$+55_{16}$ = add Carry	$9F_{16}$ $9F_{16}$	0	1 1	0 0	0 0	1 1	1 1	1 1	1 1	1 1
$+93_{16}$ = add Carry	132_{16} 33_{16}	1	0 0	0 0	1 1	1 1	0 0	0 0	1 1	0 1
$+E5_{16}$ = add Carry	118_{16} 19_{16}	1	0 0	0 0	0 0	1 1	1 1	0 0	0 0	0 1
Invert	$E6_{16}$	n/a	1	1	1	0	0	1	1	0
$19_{16} + E6_{16}$ =	FF_{16}	**n/a**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

The resulting sum is 19_{16}. Inversion yields the final result: checksum = $E6_{16}$.

The receiving node can easily check the consistency of the received frame by using the same addition mechanism. When the received checksum ($E6_{16}$) is added to the intermediate result (19_{16}), the sum shall be FF_{16}.

A.4 Example of sequence diagrams

To visualize the implications of this document, sequence diagrams are used when appropriate. The syntax used in these diagrams are exemplified in Figure A.1.The shaded areas represent the frame slots (with gaps added to clarify the drawing). Dotted/hollow arrows represent the headers and solid arrows represent responses.



Key

1 frame published by the master and subscribed to by both slave 1 and slave 2
2 header that nobody responded to
3 frame that slave 2 responded to, i.e. published and the master subscribed to
4 silent frame slot (master did not transmit the header)

Figure A.1 — Frame sequence example

Annex B
(informative)

LIN history and version compatibility

B.1 LIN history and background

LIN revision 1.0 was released in July 1999 and it was heavily influenced by the VLITE bus used by some automotive companies. The LIN standard was updated twice in year 2000, resulting in LIN 1.2 in November 2000. In November 2002, the LIN Consortium released the LIN 1.3 standard. Changes have mainly been made in the physical layer and those where targeted at improving compatibility between nodes.

The LIN 2.0 represents an evolutionary growth from its predecessor, LIN 1.3. Nodes designed for LIN 2.0 and LIN 1.3 communicate with each other with a few exceptions, as described in B.2.1.

At the same time, the LIN 2.0 specification was completely reworked and areas where problems have been found were clarified and, when needed, reworked.

LIN 2.0 was an adjustment of the LIN specification to reflect the latest trends identified; especially the use of off-the-shelves slave nodes. Three years of experience with LIN and inputs from the SAE J2602 Task Force have contributed to this major revision. LIN 2.0 also incorporates new features, mainly standardized support for configuration/diagnostics and specified node capability files, both targeted at simplifying use of off-the-shelves slave nodes.

Practical experience with LIN 2.0 has led to some findings in the specification. At the start of the work to update to LIN 2.1, it was decided that backwards compatibility is a major goal, see B.2 to B.3. The LIN 2.1 contains mostly clarifications of the functionality. Some restrictions have been introduced to make different implementations more in line with each other. Some functionality has been deleted and some has been added, see B.3.

The issues found in the LIN 2.1 specification have been collected in an LIN errata sheet. Now, because the LIN errata sheet has been unchanged for some time, all the findings have been introduced in the LIN 2.2 specification.

Following LIN 2.2, the next version is ISO 17987.

B.2 LIN version compatibility

B.2.1 Compatibility of this version with LIN 1.3

ISO 17987 is a superset of LIN 1.3.

An ISO 17987 master node can handle clusters consisting of all slaves node types (LIN 1.3, LIN 2.x, ISO 17987). The master avoids then to requesting new LIN 2.x and ISO 17987 features from a LIN 1.3 slave node:

— enhanced checksum;

— reconfiguration and diagnostics;

— automatic baudrate detection;

— response_error status monitoring;

— ISO 17987 and LIN 2.x slave nodes are not recommended to operate with a LIN 1.3 node because it would require classic checksum for unconditional frames interchanged by the slave nodes. It is recommended that communication is gated via the LIN master.

The ISO 17987 physical layer is backwards compatible with the LIN 1.3 physical layer but not vice versa. The ISO 17987 physical layer has more stringent requirements, i.e. a node using the ISO 17987 physical layer can operate in a LIN 1.3 cluster.

B.2.2 Compatibility of this version with LIN 2.0

An ISO 17987 master node may handle a LIN 2.0 slave node if the master node also contains all functionality of a LIN 2.0 master node, e.g. obsolete functions like AssignFrameIdentifier service.

An ISO 17987 slave node can be used in a cluster with a LIN 2.0 master node if the ISO 17987 slave node is preconfigured, i.e. the ISO 17987 slave node has a valid configuration after reset or if the LIN 2.0 master node uses FreeFormat schedule table commands to assign PIDs in the AssignFrameIdRange format.

A LIN 2.0 slave node shall not use NAD $7E_{16}$ since it is reserved as functional address for diagnostics in LIN 2.1/2.2 and ISO 17987. The ISO 17987 slave node considers NAD $7E_{16}$ as a functional NAD and a LIN 2.0 slave node as a configured NAD.

B.2.3 Compatibility of this version with LIN 2.1

An ISO 17987 node is compatible with a LIN 2.1 node. See B.3.3 and B.3.4.

B.2.4 Compatibility of this version with LIN 2.2

An ISO 17987 node is compatible with a LIN 2.2 node. See B.3.4.

As ISO 17987 won't match word by word to the LIN 2.2A, it can't be considered as LIN 2.2A, but as an upper version.

B.3 Changes between LIN versions

B.3.1 Changes between LIN 1.3 and 2.0

The items listed below are changed between LIN 1.3 and LIN 2.0. Renaming and clarifications are not listed in this subclause.

— Byte array signals are supported, thus allowing signals sizes up to eight bytes.

— Signal groups are deleted (replaced by byte arrays).

— Automatic bit rate detection is incorporated in the specification.

— Enhanced checksum (including the protected identifier) as an improvement to the LIN 1.3 classic checksum.

— Sporadic frames are defined.

— Network management timing is defined in seconds, not in bit times.

— Status management is simplified and reporting to the network and the application is standardized.

— Mandatory node configuration commands are added, together with some optional commands.

— Diagnostics is added.

— A LIN Product Identification for each slave node is standardized.

— The API is made mandatory for micro controller based nodes programmed in C.

— The API is changed to reflect the changes; byte array, go-to-sleep, wake up and status reading.

— A diagnostics API is added.

— A node capability language specification is added.

— The configuration language specification is updated to reflect the changes made; node attributes, node composition, byte arrays, sporadic frames and configuration are added.

B.3.2 Changes between LIN 2.0 and 2.1

The major work has been to extend descriptions for better understanding.

The following functional changes have been done:

— message ID for slave node frames are removed;

— assign frame ID configuration service is removed;

— assign frame ID range configuration service is added;

— save configuration service is added;

— status reporting to application is enhanced;

— event-triggered frame collision handling modified;

— the IDs 2 to 31 of the service Read by Identifier are reserved;

— implementation of Diagnostic Classes 1 to 3 and respective Diagnostic Services;

— transport layer enhanced with timings;

— a node operating on more than one cluster has been clarified;

— packing a signal in more than one frame has been clarified;

— NAD $7E_{16}$ (functional NAD) is reserved as functional address for diagnostics;

— node capability language specification is extended with new parameters;

— the configuration language specification is updated to reflect the changes made; node attributes, node composition, event-triggered frames and configuration are added.

B.3.3 Changes between LIN 2.1 and 2.2A

Spelling corrections and clarifications have been implemented resulting in minor slave node incompatibilities possible.

Wake up frame definition changed (rising edge after dominant pulse triggers the wake up).

B.3.4 Changes between LIN 2.2A and ISO 17987

ISO 17987 is the ISO version of LIN 2.2A. The document has been modified to meet the ISO drafting rules and guidelines.

— The node configuration and identification service ReadByIdentifier have been extended with following identifiers:

  — ID = 2_{10} to receive a negative response that is used for bit timing measurement in LIN conformance testing;

  - ID = 3_{10} to obtain the new introduced LDF and NCF revision version.
- SID $B8_{16}$ has been defined for LIN auto addressing service replacing legacy SID $B5_{16}$.
- Optional big-endian signal encoding variant specified for a LIN network.

Annex C
(informative)

LIN auto addressing methods

C.1 Auto addressing method and method IDs

In order to assign unique node addresses to slave nodes, an auto addressing concept is introduced using the request Assign NAD via auto addressing as shown in Table C.1.

Table C.1 — Assign NAD via Auto Addressing Request

NAD	PCI	SID	D1	D2	D3	D4	D5
Initial NAD	n/a	n/a	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	New NAD
$7F_{16}$	06_{16}	$B8_{16}$ ($B5_{16}$ legacy only. The entire LIN cluster uses either $B5_{16}$ or $B8_{16}$)	FF_{16}	$7F_{16}$	Auto Addressing Sub Function ID	Auto Addressing Method ID	New NAD

D3 and D4 are used for special auto addressing needs shown in Table C.2.

Table C.2 — Example of auto addressing sub function

D3	function	D4	D5
01_{16}	Initialization	Auto Addressing Method ID	FF_{16}
02_{16}	Assign NAD	Auto Addressing Method ID	new NAD
03_{16}	Store NAD in Slave	Auto Addressing Method ID	FF_{16}
04_{16}	Assign NAD Finished	Auto Addressing Method ID	FF_{16}

C.2 Auto addressing method ID

Table C.3 identifies existing auto addressing methods along with the supplier and contact information for each method. It is updated on an ongoing basis by the ISO 17987 committee or group assigned by the ISO 17987 committee.

The methods and test specifications are not within the scope of this document.

Table C.3 shows the definition of auto addressing method IDs.

Table C.3 — Definition of auto addressing method IDs

D4 method ID	Auto addressing method name	Contact information
01_{16}	Extra Wire Daisy Chain	NXP
02_{16}	Bus Shunt Method 1	ELMOS
03_{16}	LIN Switch Method	NXP
$F1_{16}$	Bus Shunt Method 2	ELMOS
$F2_{16}$	Reserved for bus direction method (not complete)	Infineon

Bibliography

[1] ISO 14229-1, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

[2] ISO 14229-2, *Road vehicles — Unified diagnostic services (UDS) — Part 2: Session layer services*

[3] ISO 14229-7, *Road vehicles — Unified diagnostic services (UDS) — Part 7: UDS on LIN implementation (UDSonLIN)*

[4] ISO 17987-7, *Road vehicles —Local Interconnect network (LIN) — Part 7: Electrical Physical Layer (EPL) conformance test specification*

[5] ISO/IEC 7498-1, *Information processing systems — Open systems interconnection — Basic reference model*

[6] ISO/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

[7] ISO/TR 17987-5, *Road vehicles —Local Interconnect network (LIN) — Part 5: Application Programmers Interface (API)*

[8] SAE J2602/1:2012-11, *LIN Network for Vehicle Applications*

ICS 01.040.43; 43.020
Price based on 44 pages