DRAFT INTERNATIONAL STANDARD
ISO/DIS 17987-5

ISO/TC **22**/SC **3** Secretariat: **DIN**

Voting begins on: **2014-01-10** Voting terminates on: **2014-04-10**

Road vehicles — Local Interconnect Network (LIN) —

Part 5:
Application Programmers Interface (API)

Véhicules routiers — Réseau Internet local (LIN) —

Partie 5: Interface du programmeur d'application (API)

ICS: 43.020

Reference number
ISO/DIS 17987-5:2013(E)

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 17987-5 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17987 consists of the following parts, under the general title *Road vehicles — Local interconnect network (LIN)*:

— *Part 1: General information and use case definition*

— *Part 2: Transport protocol and network layer services*

— *Part 3: Protocol specification*

— *Part 4: Electrical Physical Layer (EPL) specification 12 V/24 V*

— *Part 5: Application Programmers Interface (API)*

— *Part 6: Protocol conformance test specification*

— *Part 7: Electrical Physical Layer (EPL) conformance test specification*

Introduction

This document set specifies the use cases, the communication protocol and physical layer requirements of an in-vehicle communication network called "Local Interconnect Network (LIN)".

The Local interconnect network (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 transportation.

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:

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

— Bitrate support from 1 kbps to 20 kbps;

— Deterministic schedule table based frame communication;

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

— Status management that provides error handling and error signalling;

— Transport layer that allows large amount of data to be transported (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;

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the protocol and physical layer requirements specified by ISO 17987 are structured according to Table°1.

Table°1°—°LIN specifications applicable to the OSI layers

Applicability	OSI seven layer	LIN	Vehicle manufacturer enhanced diagnostics
Seven layer according to ISO°7498-1 and ISO/IEC 10731	Application (layer 7)	ISO 17987-1, ISO 17987-5	ISO°14229-1, ISO°14229-7
	Presentation (layer 6)	ISO 17987-5	vehicle manufacturer specific
	Session (layer 5)	ISO 17987-3	ISO°14229-2
	Transport (layer 4)	ISO 17987-2	
	Network (layer 3)		
	Data link (layer 2)	ISO 17987-3, ISO 17987-6	
	Physical (layer 1)	ISO 17987-4, ISO 17987-7	

Table°1 shows ISO 17987 Parts 2, 3, 4, 6 and 7 being the common standards for the OSI layers 1 through 3 for LIN and the vehicle manufacturer enhanced diagnostics.

The vehicle manufacturer enhanced diagnostics column shows application layer services covered by ISO 14229-7 which have been defined in compliance with diagnostic services established in ISO 14229-1, but are not limited to use only with them. ISO 14229-7 is also compatible with most diagnostic services defined in national standards or vehicle manufacturer's specifications. The presentation layer is defined vehicle manufacturer specific. The session layer services are covered by ISO 14229-2. The transport protocol and network layer services are specified in ISO 17987-2.

Road vehicles — Local interconnect network (LIN) — Part 5: Application Programmers Interface (API)

1 Scope

This part of ISO 17987 has been established in order to define the LIN Application Programmers Interface (API). The LIN API is a network software layer that hides the details of a LIN network configuration (e.g. how signals are mapped into certain frames) for a user making an application program for an arbitrary ECU. The user will be provided an API, which is focused on the signals transported on the LIN network. A tool takes care of the step from network configuration to program code. This will provide the user with configuration flexibility.

This document defines a mandatory interface to a software LIN device driver implemented in the 'C' programming language. Thus, hardware implementations are not standardized nor are implementations in other programming languages.

2 Normative references

The following referenced documents are indispensable for the application 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 17987-1, *Road vehicles – Local interconnect network (LIN) – Part 1: General information and use case definition*

ISO 17987-2, *Road vehicles – Local interconnect network (LIN) – Part 2: Transport protocol and network layer services*

ISO 17987-3, *Road vehicles – Local interconnect network (LIN) – Part 3: Protocol specification*

ISO 17987-4, *Road vehicles – Local interconnect network (LIN) – Part 4: Electrical Physical Layer (EPL) specification 12 V/24 V*

ISO 17987-6, *Road vehicles – Local interconnect network (LIN) – Part 6: Protocol conformance test specification*

ISO 17987-7, *Road vehicles – Local interconnect network (LIN) – Part 7: Electrical Physical Layer (EPL) 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 in ISO 17987-2, ISO 17987-3 and ISO 17987-4 apply.

3.2 Symbols

||	logical OR binary operation

3.3 Abbreviated terms

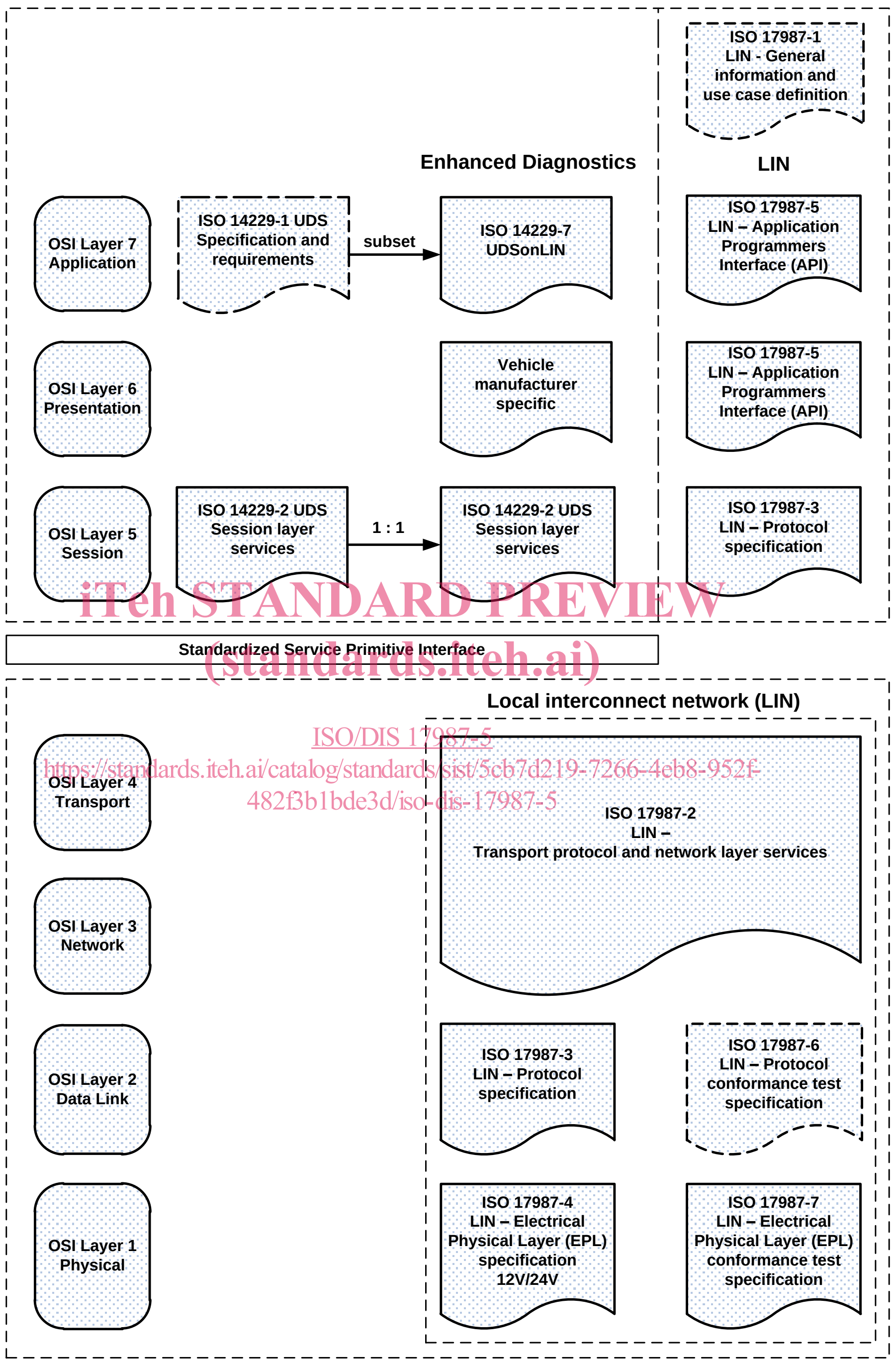
AC	alternate current
API	application programmers interface
ms	millisecond
OSI	open systems interconnection
PDU	protocol data unit
RX	Rx pin of the transceiver
UART	universal asynchronous receiver transmitter

4 Conventions

ISO 17987 and ISO 14229-7 [5] are based on the conventions specified in the OSI Service Conventions (ISO/IEC°10731) [2] as they apply for physical layer, protocol, network & transport protocol and diagnostic services.

5 Document overview

Figure°1 illustrates the document reference according to OSI model.



ISO 17987-1 LIN - General information and use case definition
Enhanced Diagnostics
LIN
OSI Layer 7 Application
ISO 14229-1 UDS Specification and requirements
subset
ISO 14229-7 UDSonLIN
ISO 17987-5 LIN – Application Programmers Interface (API)
OSI Layer 6 Presentation
Vehicle manufacturer specific
ISO 17987-5 LIN – Application Programmers Interface (API)
OSI Layer 5 Session
ISO 14229-2 UDS Session layer services
1 : 1
ISO 14229-2 UDS Session layer services
ISO 17987-3 LIN – Protocol specification
Standardized Service Primitive Interface
Local interconnect network (LIN)
OSI Layer 4 Transport
ISO 17987-2 LIN – Transport protocol and network layer services
OSI Layer 3 Network
OSI Layer 2 Data Link
ISO 17987-3 LIN – Protocol specification
ISO 17987-6 LIN – Protocol conformance test specification
OSI Layer 1 Physical
ISO 17987-4 LIN – Electrical Physical Layer (EPL) specification 12V/24V
ISO 17987-7 LIN – Electrical Physical Layer (EPL) conformance test specification

Figure°1°— LIN document reference according to OSI model

6 Specification release version information

The specification release version of this part of ISO 17987 is: 2.2.

7 Requirements

7.1 LIN cluster generation

The LIN Description file (LDF) (see ISO 17987-2) is parsed by a tool and generates a configuration for the LIN device driver. The Node Capability Language specification (NCF) is normally not used in this process since its intention is to describe a hardware slave node, and therefore, does not need the API. See ISO 17987-2 for a description of the workflow and the roles of the LDF and NCF.

7.2 Concept of operations

7.2.1 General

The API is split in three areas:

— LIN core API,

— LIN node configuration and identification API,

— LIN transport layer API (optional).

7.2.2 LIN core API

The LIN core API handles initialization, processing and a signal based interaction between the application and the LIN core. This implies that the application does not have to bother with frames and transmission of frames. Notification exists to detect transfer of a specific frame if this is necessary, see 7.3.5. API calls to control the LIN core also exist.

Two versions exist of most of the API calls

— Static calls embed the name of the signal or interface in the name of the call, and

— Dynamic calls provide the signal or interface as a parameter.

NOTE The named objects (signals, schedules) are defined in the LDF may extend their names with the channel postfix name (see channel postfix name definition in ISO 17987-2).

7.2.3 LIN node configuration and identification API

The LIN node configuration and identification API is service-based (request/response), i.e., the application in the master node calls an API routine that transmits a request to the specified slave node and awaits a response. The slave node device driver automatically handles the service.

The behaviour of the LIN node configuration and identification API is covered in the node configuration and identification (see ISO 17987-3).

7.2.4 LIN transport layer API

The LIN transport layer is message based. Its intended use is to work as a transport layer for messages to a diagnostic message parser outside of the LIN device driver. Two exclusively alternative APIs exist one raw that allows the application to control the contents of every frame sent and one messaged-based that performs the full transport layer function.

The behaviour of the LIN transport layer API is defined in ISO 17987-2.

7.3 API conventions

7.3.1 General

The LIN core API has a set of functions all based on the idea to give the API a separate name space, in order to minimize the risk of conflicts with existing software. All functions and types will have the prefix "l_" (lowercase "L" followed by an "underscore").

Table°2°—°API functions overview

Function	Description
DRIVER AND CLUSTER MANAGEMENT	
l_sys_init	l_sys_init performs the initialization of the LIN core.
SIGNAL INTERACTION	
scalar signal read	Reads and returns the current value of the signal.
scalar signal write	Reads and returns the current value of the signal.
byte array read	Reads and returns the current values of the selected bytes in the signal.
byte array write	Sets the current value of the selected bytes in the signal specified by the name sss to the value specified.
NOTIFICATION	
l_flg_tst	Returns a C boolean indicating the current state of the flag specified by the name of the static API call, i.e. returns zero if the flag is cleared, non-zero otherwise.
l_flg_clr	Sets the current value of the flag specified by the name of the static API call to zero.
SCHEDULE MANAGEMENT	
l_sch_tick	The l_sch_tick function follows a schedule.
l_sch_set	Sets up the next schedule to be followed by the l_sch_tick function for a certain interface.
INTERFACE MANAGEMENT	
l_ifc_init	l_ifc_init initializes the controller specified by the name, i.e. sets up internal functions such as the baud rate.
l_ifc_goto_sleep	This call requests slave nodes on the cluster connected to the interface to enter bus sleep mode by issuing one go to sleep command.
l_ifc_wake_up	The function will transmit one wake up signal.
l_ifc_ioctl	This function controls functionality that is not covered by the other API calls.
l_ifc_rx	The application program is responsible for binding the interrupt and for setting the correct interface handle (if interrupt is used).
l_ifc_tx	The application program is responsible for binding the interrupt and for setting the correct interface handle (if interrupt is used).
l_ifc_aux	This function may be used in the slave nodes to synchronize to the break/sync field sequence transmitted by the master node on the interface specified by iii.
l_ifc_read_status	This function will return the status of the previous communication.

Table°2°—°*(continued)*

Function	Description
USER PROVIDED CALL-OUTS	
l_sys_irq_disable	The user implementation of this function shall achieve a state in which no interrupts from the LIN communication can occur.
l_sys_irq_restore	The user implementation of this function shall restore the interrupt level identified by the provided l_irqmask previous.
NODE CONFIGURATION	
ld_is_ready	This call returns the status of the last requested configuration service.
ld_check_response	This call returns the result of the last node configuration service.
ld_assign_frame_id_range	This call assigns the protected identifier of up to four frames in the slave node with the addressed NAD.
ld_assign_NAD	This call assigns the NAD (node diagnostic address) of all slave nodes that matches the initial_NAD, the supplier ID and the function ID.
ld_save_configuration	This call will make a save configuration request to a specific slave node with the given NAD, or to all slave nodes if NAD is set to broadcast.
ld_read_configuration	This call will serialize the current configuration and copy it to the area (data pointer) provided by the application.
ld_set_configuration	The function will configure the NAD and the PRIDs according to the configuration given by data.
ld_conditional_change_NAD	This call changes the NAD if the node properties fulfil the test specified by id, byte, mask and invert.
IDENTIFICATION	
ld_read_by_id	The call requests the slave node selected with the NAD to return the property associated with the id parameter.
ld_read_by_id_callout	This callout is used when the master node transmits a read by identifier request with an identifier in the user defined area.
INITIALIZATION	
ld_init	This call will (re)initialize the raw and the messaged-based layers on the interface.
RAW API	
ld_put_raw	The call queues the transmission of 8 bytes of data in one frame. The data is sent in the next suitable frame.
ld_get_raw	The call copies the oldest received diagnostic frame data to the memory specified by data.
ld_raw_tx_status	The call returns the status of the raw frame transmission function.
ld_raw_rx_status	The call returns the status of the raw frame receive function.
MESSAGE-BASED API	
ld_send_message	The call packs the information specified by data and DataLength into one or multiple diagnostic frames.
ld_receive_message	The call prepares the LIN diagnostic module to receive one message and store it in the buffer pointed to by data.
ld_tx_status	The call returns the status of the last made call to ld_send_message.
ld_rx_status	The call returns the status of the last made call to ld_receive_message.

7.3.2 Data types

The LIN core shall define the following types:

—	l_bool	0 is false, and non-zero (> 0) is true,
—	l_ioctl_op	implementation dependent,
—	l_irqmask	implementation dependent,
—	l_u8	unsigned 8 bit integer,
—	l_u16	unsigned 16 bit integer.

In order to gain efficiency, the majority of the functions will be static functions (no parameters are needed, since one function exist per signal, per interface, etc.).

7.3.3 Driver and cluster management

7.3.3.1 l_sys_init

Table 3 defines the l_sys_init.

Table 3 — l_sys_init

Prototype	l_bool l_sys_init (void)
Applicability	Master and slave nodes.
Description	l_sys_init performs the initialization of the LIN core. The scope of the initialization is the physical node i.e. the complete node (see node composition definition in ISO 17987-2). The call to the l_sys_init is the first call a user shall use in the LIN core before using any other API functions.
Return value	Zero if the initialization succeeded. Non-zero if the initialization failed.

7.3.4 Signal interaction

In all signal API calls below the sss is the name of the signal, e.g. l_u8_rd_engine speed ().

7.3.4.1 Signal types

The signals will be of three different types:

— l_bool for one bit signals; zero if false, non-zero otherwise,

— l_u8 for signals of the size 2 – 8 bits,

— l_u16 for signals of the size 9 – 16 bits.