INTERNATIONAL STANDARD

ISO 20794-4

First edition
2020-02

Road vehicles — Clock extension peripheral interface (CXPI) —

Part 4: Data link layer and physical layer

Véhicules routiers — Interface du périphérique d'extension d'horloge (CXPI) —

Partie 4: Couches de liaison de données et physique

Reference number
ISO 20794-4:2020(E)

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

Published in Switzerland

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 of the voluntary nature of standards, 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 www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

A list of all parts in the ISO 20794 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

ISO 20794 (all parts) specifies the application (partly), application layer, transport layer, network layer, data link layer, and physical layer requirements of an in-vehicle network called "clock extension peripheral interface (CXPI)".

CXPI is an automotive low-speed single-wire network. It is an enabler for reducing vehicle weight and fuel consumption by reducing wire counts to simple devices like switches and sensors.

CXPI serves as and is designed for automotive control applications, for example door control group, light switch, and HVAC (Heating Ventilation and Air Conditioning) systems.

The CXPI services, protocols, and their key characteristics are specified in different parts according to the OSI layers.

— Application and application layer
  — application measurement and control data communication to exchange information between applications in different nodes based on message communication;
  — wake-up and sleep functionality;
  — two kinds of communication methods can be selected at system design by each node:
    i) the event-triggered method, which supports application measurement- and control-based (event-driven) slave node communication, and
    ii) the polling method, which supports slave node communication based on a periodic master schedule;
  — performs error detection and reports the result to the application;
  — application error management.
— Transport layer and network layer
  — transforms a message into a single packet;
  — adds protocol control information for diagnostic and node configuration into each packet;
  — adds packet identifier for diagnostic and node configuration into each packet;
  — performs error detection and reports the result to higher OSI layers.
— Data link layer and physical layer
  — provides long and short data frames;
  — adds a frame identifier into the frame;
  — adds frame information into the frame;
  — adds a cyclic redundancy check into the frame;
  — performs byte-wise arbitration and reports the arbitration result to higher OSI layers;
  — performs frame type detection in reception function;
  — performs error detection and reports the result to higher OSI layers.
  — performs Carrier Sense Multiple Access (CSMA);
  — performs Collision Resolution (CR);

— generates a clock, which is transmitted with each bit to synchronise the connected nodes on the CXPI network;

— supports bit rates up to 20 kbit/s.

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[1], which structures communication systems into seven layers.

Figure 1 illustrates an overview of communication frameworks beyond the scope of this document including related standards:

— vehicle normal communication framework, which is composed of ISO 20794-2, and ISO 20794-5;

— vehicle diagnostic communication framework, which is composed of ISO 14229-1, ISO 14229-2[3], and ISO 14229-8[4];

— presentation layer standards, e.g. vehicle manufacturer specific or ISO 22901-1 ODX[6];

— lower OSI layers framework, which is composed of ISO 20794-3, ISO 20794-4, ISO 20794-6, and ISO 20794-7 conformance testing.

ISO 20794 (all parts) and ISO 14229-8[4] are based on the conventions specified in the OSI Service Conventions (ISO/IEC 10731)[1] as they apply for all layers and the diagnostic services.

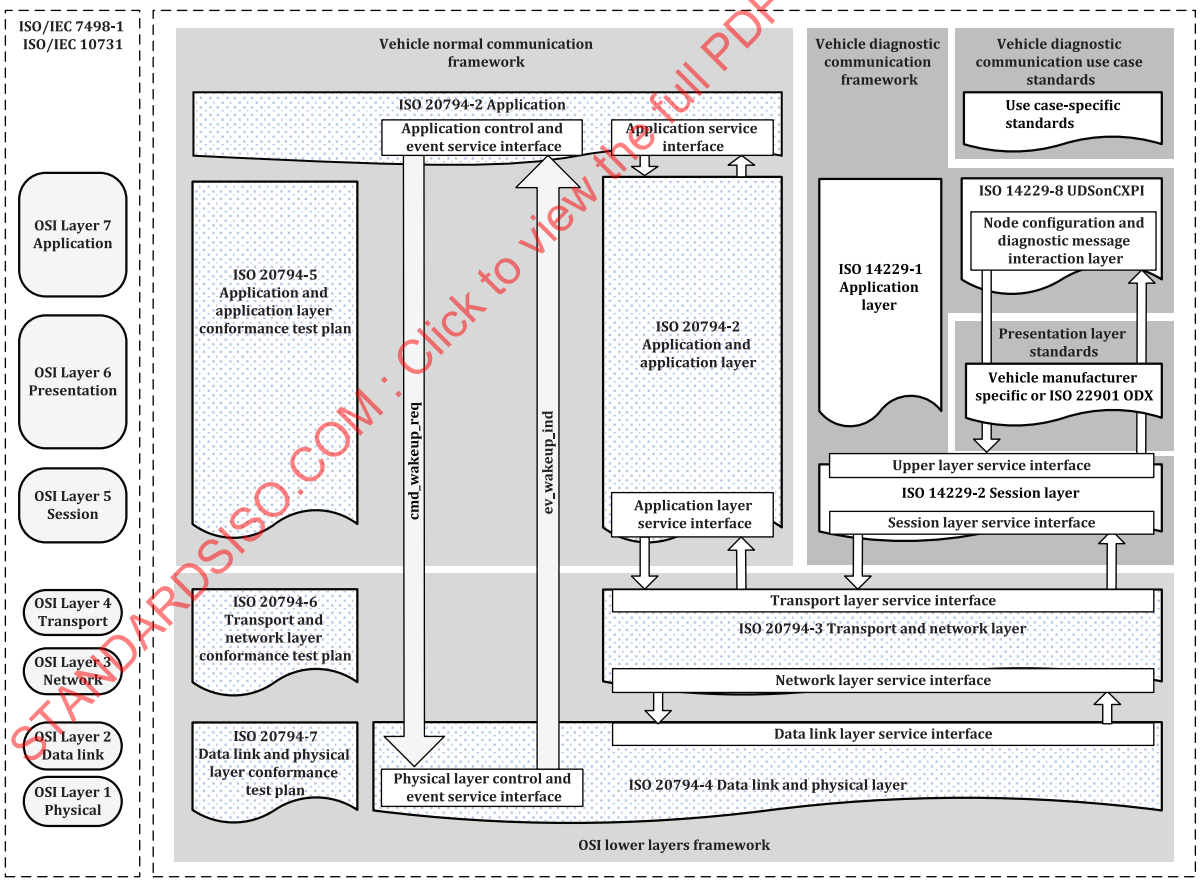


Figure 1 — ISO 20794 documents reference according to OSI model

Road vehicles — Clock extension peripheral interface (CXPI) —

Part 4: Data link layer and physical layer

1 Scope

This document specifies the CXPI data link layer and the CXPI physical layer.

The DLL is based on:

— priority-based CXPI network access;

— non-destructive content-based arbitration;

— broadcast frame transfer and acceptance filtering; and

— node related error detection and error signalling.

The CXPI physical layer (PHY) requirements comprise of:

— physical signalling (PS) sub-layer, which specifies the requirements of the clock generation function, the encoding and decoding of CXPI frames, and bit-wise collision resolution logic;

— physical media attachment (PMA) sub-layer, which specifies the requirements of the signal shaping waveform logic;

— physical media dependent (PMD) sub-layer, which specifies the requirements of the CXPI network termination, electrostatic discharge protection, etc., and device connector requirements; and

— physical media (PM), which specifies the requirements of the CXPI network cable/wiring harness.

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/IEC 7498-1, *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*

ISO 20794-2, *Road vehicles — Clock extension peripheral interface (CXPI) — Part 2: Application layer*

ISO 20794-3, *Road vehicles — Clock extension peripheral interface (CXPI) — Part 3: Transport and network layer*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20794-2, ISO 20794-3, and ISO/IEC 7498-1 and the following apply.

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

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

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

3.1
bit rate
transmission speed (bit/s) of the protocol within the specification range of transmit frames

3.3
higher OSI layer
OSI layer 3 or more than higher number of layers

3.4
idle state
state of the CXPI network when no frames are exchanged (idle state) and only the clock exists to synchronise the nodes

3.5
inter frame space
IFS
time interval between frames on the CXPI network

4 Symbols and abbreviated terms

4.1 Symbols

C_{BUS}	total bus capacitance
C_{LINE}	capacity of CXPI network
C_{MASTER}	capacity of master node
C_{SLAVE}	capacity of slave node
kbit/s	kilobit per second
LEN_{BUS}	length of CXPI bus wire
R_{BUS}	total bus resistor including all slave nodes resistors and master node resistor
R_{MASTER}	master node resistor
R_{SLAVE}	slave node resistor
t_{bit}	bit time
t_{bit_ref}	bit time of reference bit rate
$t_{rx\_dif\_cont}$	difference of the dominant time between logical value '1' and logical value '0'
$t_{rx\_wakeup\_clk}$	time that the receiving clock master detects the width of dominant level as the wake-up pulse.
t_{rx_wakeup}	time that the receiving node detects each width of dominant level in the wake-up pulse.
$t_{rx\_wakeup\_space}$	limitation time of acceptance second dominant pulse in the wake-up pulse from first dominant pulse.

t_{tx_wakeup}	time that the transceiver node transmits the dominant voltage of the wake-up pulse
$t_{tx\_wakeup\_space}$	interval time between two of dominant level of transmitting wake-up pulse
$t_{tx\_0\_lo}$	dominant time of logical value '0'
$t_{tx\_0\_lo\_dom}$	dominant time of logical value '0' (TH_{tx_dom} = 30 % of V_{SUP})
$t_{tx\_0\_pd}$	at the time of logical value '0' outputs, time from the LO level detection of the CXPI network until falling the voltage TH_{tx_dom} = 30 % of V_{SUP}
$t_{tx\_1\_lo}$	dominant time of logical value '1'
$t_{tx\_1\_lo\_dom}$	dominant time of logical value '1' (TH_{tx_dom} = 30 % of V_{SUP})
$t_{tx\_1\_lo\_rec}$	dominant time of logical value '1' (TH_{tx_rec} = 70 % of V_{SUP})
TH_{tx_dom}	dominant threshold voltage of the driver node
TH_{tx_rec}	recessive threshold voltage of the driver node
TH_{rx_dom}	dominant threshold voltage of the received node
TH_{rx_rec}	recessive threshold voltage of the received node
V_{BUS}	voltage of CXPI network
V_{BUS_CNT}	centre recessive threshold voltage of the received node
V_{BUSdom}	dominant voltage of the received node
V_{BUSrec}	recessive voltage of the received node
V_{HYS}	hysteresis voltage between the recessive threshold voltage and the dominant threshold voltage of the received node
V_{rec_master}	maximum recessive level of logical value “1”
V_{th_dom}	measured value of the dominant threshold voltage of the received node
V_{th_rec}	measured value of the recessive threshold voltage of the received node

4.2 Abbreviated terms

AC	alternating current
CRC	cyclic redundancy check
CT	count
DLC	data length code
DLL	data link layer
ECU	electronic control unit
FI	frame information
Ftype	frame type
HI	high

IBS	inter byte space
ID	identifier
IFS	inter frame space
ind	indicator
L_	link
LO	low
LSB	least significant byte
MSB	most significant byte
NM	network management
NRZ	non-return to zero
OSI	open systems interconnection
PDU	protocol data unit
PID	protected identifier
PHY	physical layer
PM	physical media
PMA	physical media attachment
PMD	physical media dependent
PS	physical signalling
PTYPE	protected type identifier
PWM	pulse width modulation
ReqId	request identifier
ReqTypeId	request type identifier
RX_{PWM}	PMA receiver interface signal
RXD_{NRZ}	PS receiver interface signal
SIP	service interface parameter
TH	threshold
TX_{PWM}	PMA transmit interface signal
TXD_{NRZ}	PS transmitter interface signal

5 Conventions

This document is based on the conventions discussed in the OSI Service Conventions as specified in ISO/IEC 10731.

6 Introduction to data link layer and physical layer

6.1 Frames

Information on the CXPI network is sent in fixed format frames of different limited lengths. When the CXPI network is in idle state, any connected node is allowed to start the transmission of a Protected Identifier (PID). The CXPI network is in idle state when no frames are transmitted.

6.2 Frame collision avoidance

If two or more nodes start to transmit a PID value at the same time, the bus access conflict is resolved by content-based arbitration using the PID value. The transmitter with the PID value of the highest priority gains the bus access.

6.3 Error detection and indication

For detecting errors, the following measures are specified:

— byte value error (`Err_DLL_Byte`);

— cyclic redundancy check value error (`Err_DLL_CRC`);

— data length code value error (`Err_DLL_DLC`);

— data length code extension value error (`Err_DLL_DLCext`);

— parity value error (`Err_DLL_Parity`); and

— framing value error (`Err_DLL_Framing`).

To confirm an error occurrence during data transmission and reception, an error indication to the higher OSI layers is provided.

6.4 Clock transmission and detection

The node (master node by default), that generates the clock, continuously transmits the clock to the CXPI network. Clock existence is detected by the falling edge of the clock on the CXPI network. A node performs communication synchronised with the node, that generates the clock.

7 Service interface parameters (SIP)

7.1 SIP — General

The following subclauses specify the service interface parameters and data types, which are used by the CXPI data link and physical layer services.

7.2 SIP — Data type definitions

This requirement specifies the data type definitions of the CXPI service interface.

REQ	0.1 SIP — Data type definitions
The data types shall be in accordance to: — `Enum` = 8-bit enumeration — `Unsigned Byte` = 8-bit unsigned numeric value — `Byte Array` = sequence of 8-bit aligned data — `2-bit Bit String` = 2-bit binary coded — `8-bit Bit String` = 8-bit binary coded — `16-bit Bit String` = 16-bit binary coded	

7.3 SIP — Ftype, frame type

This requirement specifies the frame type parameter value of the CXPI service interface.

REQ	0.2 SIP — Ftype, frame type
The `Ftype` parameter shall be of data type Enum and shall be used to identify the frame type and range of address information included in a service call. Range: [`NormalCom, DiagNodeCfg`]	

7.4 SIP — ReqId, request identifier

This requirement specifies the request identifier parameter value of the CXPI service interface.

REQ	0.3 SIP — ReqId, request identifier
The `ReqId` parameter shall be of data type `Unsigned Byte` and shall contain the request identifier. Range: [01_{16} to $7F_{16}$]	

7.5 SIP — ReqTypeId, request type identifier

This requirement specifies the request type identifier parameter value of the CXPI service interface.

REQ	0.4 SIP — ReqTypeId, request type identifier
The `ReqTypeId` parameter shall be of data type `Unsigned Byte` and shall contain the request type identifier. Range: [00_{16}] in DLL NOTE `ReqTypeId` is used by the application to enable the polling method. It has a fixed value.	

7.6 SIP — PDU, protocol data unit

This requirement specifies the protocol data unit parameter value of the CXPI service interface.

REQ	0.5 SIP — PDU, protocol data unit
The `PDU` parameter shall be of data type `Byte Array` and shall contain the frame data (PDU) content of the request or response frame to be transmitted/received. Range: [00_{16} to FF_{16}]	

7.7 SIP — Length, length of PDU

This requirement specifies the length parameter value of the CXPI service interface.

REQ	0.6 SIP — Length, length of PDU
The `Length` parameter shall be of data type `Unsigned Byte` and shall contain the length of the PDU to be transmitted/received. Range: [00_{16} to FF_{16}]	

7.8 SIP — ev_wakeup_ind, event wake-up indication (optional)

This requirement specifies the event wake-up indication parameter value of the CXPI service interface.

REQ	0.7 SIP — ev_wakeup_ind, event wake-up indication (optional)
The `ev_wakeup_ind` parameter shall be of data type `Enum` and shall include the network management information in the response, which is composed of wake-up indication information of a CXPI node. Table 1 describes the network management information values. Range: [`ev_wakeup_pulse_detect, ev_dominant_pulse_detect, ev_clk_detect, ev_clk_loss`]	

Table 1 — ev_wakeup_ind, event wake-up indication (optional)

Enum values	Description
`ev_wakeup_pulse_detect`	This service primitive parameter value indicates the reception of the wake-up pulse event from the DLL. This parameter is optional if `cmd_wakeup_pulse_on` in Table 2 is (optional).
`ev_dominant_pulse_detect`	This service primitive parameter value indicates the reception of the dominant pulse event from the DLL. This parameter is optional if `cmd_wakeup_pulse_on` in Table 2 is (optional).
`ev_clk_detect`	This service primitive parameter value indicates the reception of the clock detection event from the DLL.
`ev_clk_loss`	This service primitive parameter value indicates the reception of the clock loss event from the DLL.

7.9 SIP — cmd_wakeup_req, command wake-up request

This requirement specifies the command wake-up request parameter value of the CXPI service interface.

REQ	0.8 SIP — cmd_wakeup_req, command wake-up request
The `cmd_wakeup_req` parameter shall be of data type `Enum` and shall include the wake-up request command information to wake-up a CXPI node. Table 2 describes the `cmd_wakeup_req` values. Range: [`cmd_clk_generator_on, cmd_clk_generator_off, cmd_wakeup_pulse_on`]	

Table 2 — cmd_wakeup_req values

Enum values	Description
`cmd_clk_generator_on`	This service primitive parameter value commands the clock generator to turn on the clock transmission to the DLL.
`cmd_clk_generator_off`	This service primitive parameter value commands the clock generator to turn off the clock transmission to the DLL.
`cmd_wakeup_pulse_on (optional)`	This service primitive parameter value commands the transmission of the wake-up pulse to the DLL.

7.10 SIP — NMInfo, network management information

This requirement specifies the network management information parameter value of the CXPI service interface.

REQ	0.9 SIP — NMInfo, network management information
The `NMInfo` parameter shall be of data type `Enum` and shall contain the NM information in the response field. Type: `Enum` Range: 00_2 = [no request for `wakeup_ind, sleep_ind` prohibition] 01_2 = [no request for `wakeup_ind, sleep_ind` permission] 10_2 = [request for `wakeup_ind, sleep_ind` prohibition] 11_2 = [request for `wakeup_ind, sleep_ind` permission]	

7.11 SIP — SCT, sequence count

This requirement specifies the sequence count parameter value of the CXPI service interface.

REQ	0.10 SIP — SCT, sequence count
The `SCT` parameter shall be of data type `Unsigned Byte` and shall contain the sequence count information in the response field to be transmitted/received. Type: `2-bit Unsigned Byte` Range: [00_2 `to` 11_2]	

7.12 SIP — Result, result

This requirement specifies the result parameter value of the CXPI service interface.

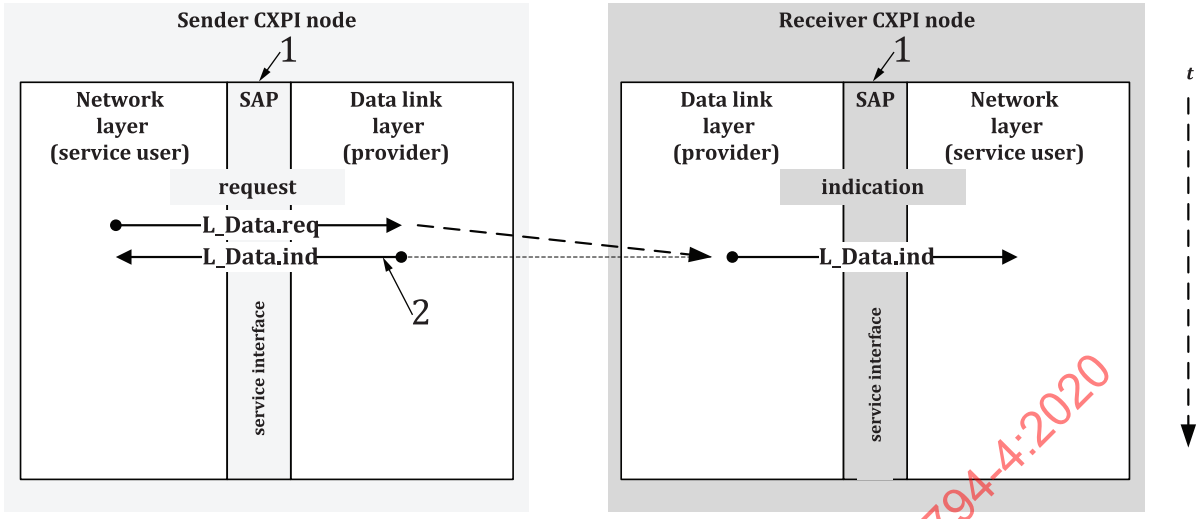
REQ	0.11 SIP — Result, result
The `Result` parameter shall be of data type `Bit String` and shall contain the status relating to the outcome of a ReqField or RespField execution. Range: [`OK, DLL_Arb_Lost, Err_DLL_Byte, Err_DLL_CRC, Err_DLL_DLC, Err_DLL_DLCext, Err_DLL_Parity, Err_DLL_Framing`] Range: [0_2 `to` 1_2]	

8 Data link layer (DLL)

8.1 SI — L_Data.req and L_Data.ind service interface

The service interface defines the service primitives and parameter mapping to the DLL.

Figure 2 shows the DLL `L_Data.req` and `L_Data.ind` service interface.



Key

1 service access point

2 read back from CXPI network provided by lower OSI layer

t time

Figure 2 — DLL L_Data.req and L_Data.ind service interface

For normal communication (NormalCom) the sender node transmits, if master node, either a request protected type identifier field (polling method), or a request protected identifier field (`L_Data.req`). All nodes receive the request protected identifier field of type NormalCom (`L_Data.ind`). The node, which has corresponding PDU data transmits the response PDU (`L_Data.req`) and all nodes receive the response PDU (`L_Data.ind`).

For diagnostic and node configuration (DiagNodeCfg) the sender node transmits a request message including a protected identifier field and the N_PDU (`L_Data.req`) onto the CXPI network. All nodes receive the DiagNodeCfg request message (`L_Data.ind`). The node(s), which is (are) targeted by the request message, transmit a DiagNodeCfg response message (`L_Data.req`) onto the CXPI network.

8.2 SI — L_Data.req and L_Data.ind service interface parameter mapping

This requirement specifies the data link layer service interface parameter mapping to lower OSI layers.

REQ	2.1 SI — L_Data.req and L_Data.ind service interface parameter mapping
The `L_Data.req` and `L_Data.ind` service interface parameter mapping is specified in Table 3.	

The normal communication frame (NormalCom) either consists of a `L_ReqId (L_PID)` only (`L_Length = NULL`) or `L_ReqId (L_PID)` and a response `L_PDU (L_Length` $\geq 00_{16}$).

The diagnostic and node configuration frame (DiagNodeCfg) always consist of a `L_ReqId (L_PID)` and a response `L_PDU`. Therefore, the `L_Length` $> 00_{16}$.

Table 3 — L_Data.req and L_Data.ind service interface parameter mapping

Higher OSI layers (service user)	Data link layer (service provider)	L_Data.req and L_Data.ind parameter validity					
		NormalCom with L_Length = NULL		NormalCom with L_Length ≥00_{16}		DiagNodeCfg with L_Length >00_{16}	
		.req	.ind	.req	.ind	.req	.ind
N_Ptype	L_Ftype	X[a]	X[a]	X[a]	X[a]	X[a]	X[a]
N_Length	L_Length	—[c]	—[c]	X[a]	X[a]	X[a]	X[a]
N_ReqId	L_ReqId	X[a]	X[a]	X[a]	X[a]	X[a]	X[a]
N_ReqTypeId[b]	L_ReqTypeId[b]	X[a]	X[a]	—[c]	—[c]	—[c]	—[c]
N_PDU	L_DATA	—[c]	—[c]	X[a]	X[a]	X[a]	X[a]
N_NMInfo	L_NMInfo	—[c]	—[c]	X[a]	X[a]	X[a]	X[a]
N_SCT	L_SCT	—[c]	—[c]	X[a]	X[a]	X[a]	X[a]
N_Result	L_Result	—[c]	X[a]	—[c]	X[a]	—[c]	X[a]

NOTE A service interface call either includes a ReqId or a ReqTypeId parameter.

[a] Supported "X".

[b] Not used if DiagNodeCfg.

[c] Not supported "—".

8.3 DLL — Service interface with L_Ftype parameter mapping

8.3.1 DLL — L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length = NULL)

This requirement specifies the data link layer frame type NormalCom with L_Length = NULL via the L_Data.req and L_Data.ind interface.

REQ	2.2 DLL — L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length = NULL)

The L_Data.req and L_Data.ind service interface with L_Ptype = NormalCom with L_Length = NULL shall use the service interface parameters specified in Table 3 and Figure 3.

If the DLL provider is the sender then the following mapping shall be implemented for a L_Data.req:

— L_Ftype = N_Ptype(NormalCom);

— L_ReqTypeId = N_ReqTypeId;

— L_ReqTypeId[Bit 6 to Bit 0] = N_ReqTypeId;

— L_ReqTypeId[Bit 7] = ODD_Parity(N_ReqTypeId);

— L_ReqId[Bit 6 to Bit 0] = N_ReqId;

— L_ReqId[Bit 7] = ODD_Parity(N_ReqId);

— L_Length = NULL.

If the DLL provider is the receiver then the following mapping shall be implemented for a L_Data.ind:

— N_Ptype = L_Ftype(NormalCom);

— N_ReqTypeId = L_ReqTypeId;

— N_ReqId = L_ReqId;

— N_Length = NULL;

— N_Result = L_Result.

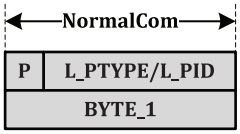


Figure 3 — DLL L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length = NULL)

8.3.2 DLL — L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length ≥ 0016)

This requirement specifies the data link layer frame type NormalCom with L_Length ≥ 00_{16} via the L_Data.req and L_Data.ind interface.

REQ	2.3 DLL — L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length ≥ 00_{16})

The L_Data.req and L_Data.ind service interface with L_Ftype = NormalCom with L_Length ≥ NULL shall use the service interface parameters specified in Table 3 and Figure 4.

If the DLL provider is the sender then the following mapping shall be implemented for a L_Data.req:

— L_Ftype = N_Ptype(NormalCom);

— L_ReqId[Bit 6 to Bit 0] = N_ReqId;

— L_ReqId[Bit 7] = ODD_Parity(N_ReqId);

— If (0 <= N_Length ≤ 12) then (L_FI_DLC = N_Length) else
(L_FI_DLC = 1111_2 and L_FI_DLCext = N_Length);

— L_PDU = [F_FI, N_PDU, L_CRC];

— L_MNInfo = N_MNInfo;

— L_SCT = N_SCT.

If the DLL provider is the receiver then the following mapping shall be implemented for a L_Data.ind:

— N_Ptype = L_Ftype(NormalCom);

— N_ReqId = L_ReqId[Bit 6 to Bit 0];

— N_Length = L_Length;

— If (L_FI_DLC ≤ 12) then (N_Length = L_FI_DLC) else
(N_Length = L_FI_DLCext);

— N_PDU = [L_PDU, w/o L_FI, w/o L_CRC];

— N_MNInfo = L_MNInfo;

— N_SCT = L_SCT;

— N_Result = L_Result.

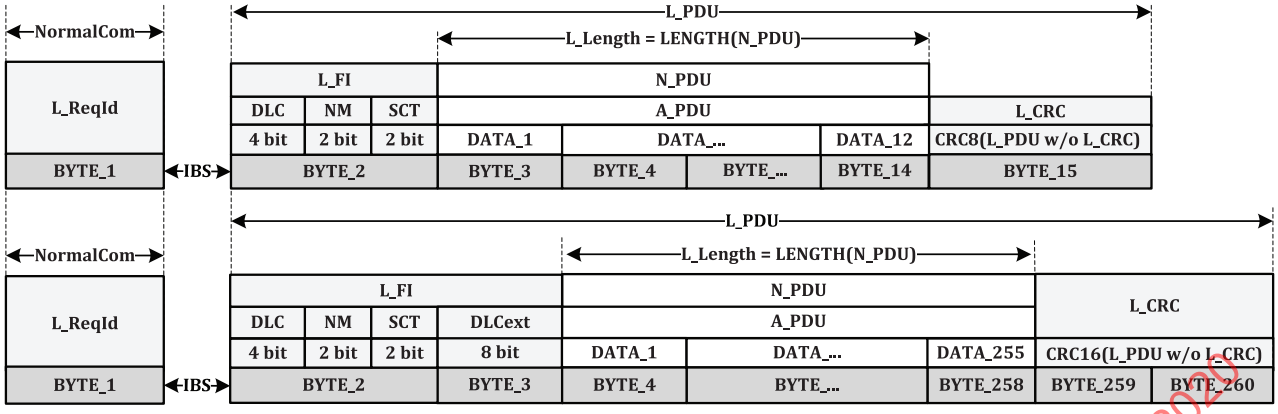


Figure 4 — DLL L_Data.req and L_Data.ind with L_Ftype = NormalCom (L_Length ≥ 0016)

8.3.3 DLL — L_Data.req and L_Data.ind interface with L_Ftype = DiagNodeCfg

This requirement specifies the data link layer frame type DiagNodeCfg via the L_Data.req and L_Data.ind interface.

REQ	2.4 DLL — L_Data.req and L_Data.ind interface with L_Ftype = DiagNodeCfg

The L_Data.req and L_Data.ind service interface with L_Ftype = DiagNodeCfg shall use the service interface parameters specified in Table 3 and in Figure 5.

If the DLL provider is the sender then the following mapping shall be implemented for a L_Data.req:

— L_Ftype = N_Ptype(DiagNodeCfg);

— L_Length = LENGTH(N_PDU);

— L_ReqId[Bit 6 to Bit 0] = N_ReqId;

— L_ReqId[Bit 7] = ODD_Parity(N_ReqId);

— If (0 < N_Length ≤ 12) then (L_FI_DLC = N_Length) else
(L_FI_DLC = 1111_2 and L_FI_DLCext = N_Length);

— L_PDU = [F_FI, N_PDU, L_CRC];

— L_MNInfo = N_MNInfo;

— L_SCT = N_SCT.

If the DLL provider is the receiver then the following mapping shall be implemented for a L_Data.ind:

— N_Ptype = L_Ftype(DiagNodeCfg);

— N_ReqId = L_ReqId[Bit 6 to Bit 0];

— If (L_FI_DLC ≤ 12) then (N_Length = L_FI_DLC) else
(N_Length = L_FI_DLCext);

— N_PDU = [L_PDU, w/o L_FI, w/o L_CRC];

— N_MNInfo = L_MNInfo;

— N_SCT = L_SCT;

— N_Result = L_Result.

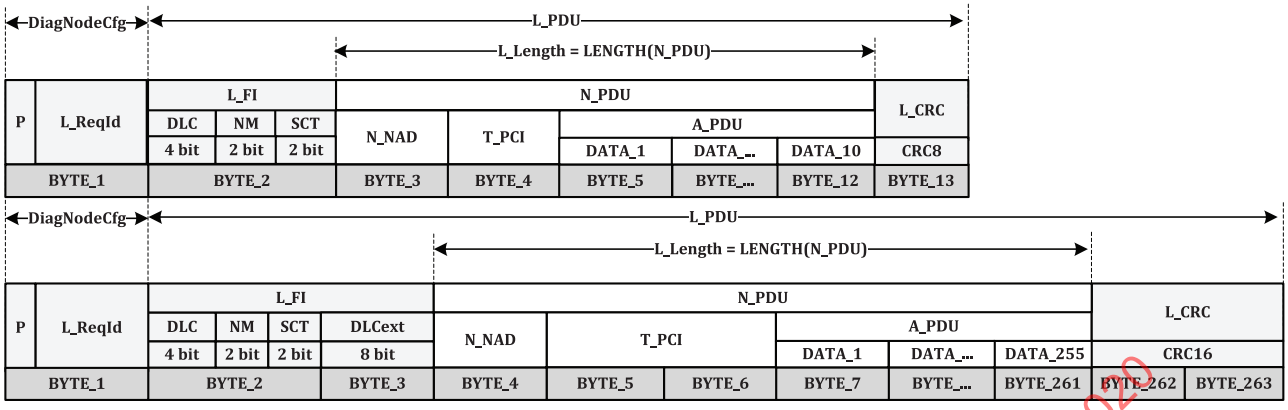


Figure 5 — DLL L_Data.req and L_Data.ind with L_Ftype = DiagNodeCfg

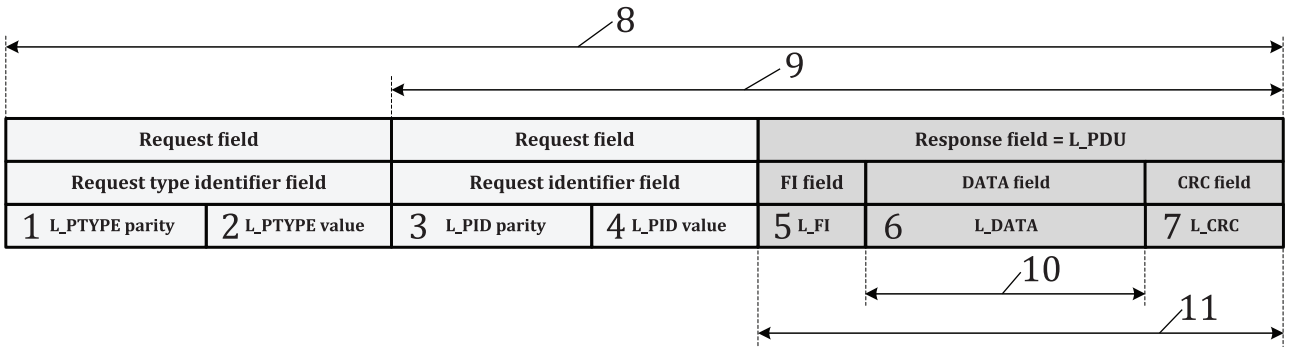
8.4 DLL — Frame fields

8.4.1 DLL — Frame field definition

This requirement specifies the CXPI frame fields.

REQ	2.5 DLL — Frame field definition
A frame shall consist of a request protected type identifier field (polling method only), a request protected identifier field, and a response PDU field (see Figure 6). The `L_PID` or `L_PTYPE` value shall be used for arbitration on the CXPI network in order to gain access to the CXPI network. The protected type identifier has highest priority. The response field shall consist of the `L_FI` field, the `N_PDU` field, and the `L_CRC` field.	

A node transmits the request field of the frame on the CXPI network and if arbitration is successful, either the same node or a responding node transmits the response field of the frame.



Key

1 protected type identifier parity
2 protected type identifier
3 protected identifier parity
4 protected identifier
5 frame information field
6 data field
7 cyclic redundancy check field
8 request type identifier field, request identifier field, response field
9 request identifier field, response field
10 L_Length = LENGTH(N_PDU)
11 L_PDU

Figure 6 — DLL frame field definition

8.4.2 DLL — Request type identifier and request identifier fields

8.4.2.1 DLL — Request type identifier field

This requirement specifies the `L_PTYPE` field.

REQ	2.6 DLL — Request type identifier field
The request type identifier field [`L_PTYPE_Parity, L_PTYPE`] shall contain the parity bit in one bit, and the `L_PTYPE` value in seven bits (see Table 4).	

The master node transmits a request type identifier field, if the polling method is activated. A node can then transmit a request identifier field.

Table 4 — Configuration of request type identifier field

L_PTYPE parity (bit 7)	L_PTYPE value (bit 6 to 0)
Odd parity bit (fixed at 1_2)	Fixed at 00_{16}

8.4.2.2 DLL — Request identifier field

This requirement specifies the L_PID parameter.

REQ	2.7 DLL — Request identifier field
The request identifier field [`L_PID_Parity, L_PID`] shall contain the parity bit in one bit, and the `L_PID` value in seven bits (see Table 5).	

All nodes can transmit a request identifier field.

Table 5 — Configuration of request identifier field

L_PID parity (bit 7)	L_PID value (bit 6 to 0)
Odd parity bit	01_{16} to $7F_{16}$

8.4.2.3 DLL — L_PTYPE and L_PID parity determination

This requirement specifies the L_PID parity determination.

REQ	2.8 DLL — L_PTYPE and L_PID parity determination
For ID0 to ID6 bits, if the count of bits with a value of '1' is even, the parity bit value shall be set to '1' making the total count of '1s' in the whole set [including the parity bit (P)] an odd number. If the count of bits with a value of '1' is odd, the count is already odd so the parity bit's (P) value shall be '0'. The odd parity shall be calculated according to the Formula (1).	

Definition of formula:

$$P = \neg(ID0 \oplus ID1 \oplus ID2 \oplus ID3 \oplus ID4 \oplus ID5 \oplus ID6) \quad (1)$$

8.4.3 DLL — L_FI (frame information) field

8.4.3.1 DLL — L_FI configuration

This requirement specifies the L_FI of the response field.

REQ	2.9 DLL — L_FI configuration
The frame information (L_FI) field shall contain the L_FI_DLC (data link data length code), the L_FI_NM (data link network management parameter), the L_FI_SCT (data link frame sequence count), and the L_FI_DLCext (data length code extension).	

8.4.3.2 DLL — L_FI_DLC (data length code)

This requirement specifies the L_FI_DLC of the response field.

REQ	2.10 DLL — L_FI_DLC (data length code)
The L_FI_DLC defines the data length of the number of data bytes contained in a response field of a frame. The length of the response field of the frame is changed according to the value of the L_FI_DLC (see 7.10 and Table 6). If the data length is set to 12, the transmitting node sets the L_FI_DLC to '1100_2'. The data length shall be set to 12, even if the L_FI_DLC value is greater than 12 at the receiving node. If the L_FI_DLC value is set to '1111_2' the operation shall be in accordance with the definition in 8.4.3.3.	

Table 6 — Configuration of L_FI (L_FI_DLC ≠'1111_2')

L_FI_DLC (bit 7 to 4)	L_FI_NM (bit 3 to 2)		L_FI_SCT (bit 1 to 0)
Data length 0 byte to 12 byte	wakeup_ind (see 7.10)	sleep_ind (see 7.10)	Frame sequence count

8.4.3.3 DLL — L_FI_DLCext (data length code extension)

This requirement specifies the L_FI_NM of the response field.

REQ	2.11 DLL — L_FI_DLCext (data length code extension)
The L_FI_DLCext (extension) is an optional feature of the DLL. If the L_FI_DLCext is supported by the DLL, it shall be used for long frame transmissions. The value of the L_FI_DLC shall be set to '1111_2' and the value of the L_FI_DLCext (1 byte) shall be set to the data length by the transmitting node. The data up to 255 byte shall be set following the L_FI_DLCext value (see Table 7).	

Table 7 — Configuration of L_FI (L_FI_DLC ='1111_2')

L_FI_DLC (bit 7 to 4)	L_FI_NM (bit 3 to 2)		L_FI_SCT (bit 1 to 0)	L_FI_DLCext (bit 7 to 0)
Fixed at 1111_2	wakeup_ind (see 7.10)	sleep_ind (see 7.10)	Frame sequence count	Data length $0 \leq$ L_FI_DLCext ≤ 255

8.4.3.4 DLL — L_FI_NM (network management)

This requirement specifies the L_FI_NM of the response field.

REQ	2.12 DLL — DLL — L_FI_NM (network management)
The L_FI_NM (ev_wakeup_ind, sleep_ind) parameter shall be filled with the N_NMInfo (see 7.10) parameter as specified by the higher OSI layer service interface.	

8.4.3.5 DLL — L_FI_SCT (sequence count)

This requirement specifies the L_FI_SCT of the response field.

REQ	2.13 DLL — L_FI_SCT (sequence count)
The L_FI_SCT bits shall be filled with the (numeric) value as specified by the service primitive interface to the higher OSI layers.	

8.4.4 DLL — L_DATA (data field)

This requirement specifies the L_DATA (data field) of the response field.

REQ	2.14 DLL — L_DATA (data field)
The data field shall contain the L_DATA, which is the payload data.	

This requirement specifies the L_DATA (big endian).

REQ	2.15 DLL — L_DATA (data field) — Big endian
The byte order shall be big endian (order in L_DATA: MSB, LSB), when the data of frame includes 2 or more bytes of data. If the L_FI_DLC value is 2 or more data byte, MSB of the data shall be set to first byte in the order of transmitting to the CXPI network, LSB of the data shall be set to last byte in the order of transmitting to the CXPI network.	

The transmitting node can freely store data in the data of frame. The data length is the number of bytes described with L_FI_DLC of frame information, and it stores the data of 0 byte to 12 byte. Figure 7 shows the L_DATA bytes of the frame and named L_DATA_1, L_DATA_2, to L_DATA_12 in the order of transmitting to the CXPI network.

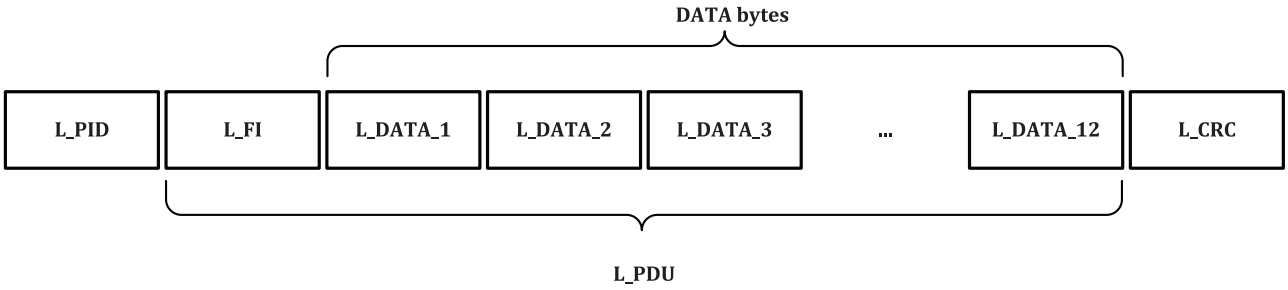


Figure 7 — Data bytes within a frame

8.4.5 DLL — L_CRC field

8.4.5.1 DLL — L_CRC field determination

This requirement specifies the L_CRC field in the response field.

REQ	2.16 DLL — L_CRC field determination
The L_CRC (data link cyclic redundancy check) parameter field shall be determined according to the conditions: — L_CRC8: see 8.4.5.2 if L_Length ≤ 12 (L_FI_DLC ≠ '1111_2'). — L_CRC16: see 8.4.5.3 if L_Length ≤ 255 (L_FI_DLC = '1111_2').	

8.4.5.2 DLL — CRC8 calculation

This requirement specifies the DLL CRC8 calculation.

REQ	2.17 DLL — CRC8 calculation
The L_CRC8 calculation shall be performed if the L_FI_DLC ≠ '1111_2' for both, the transmitting and the receiving nodes. The L_CRC8 operation shall be performed in bit order as it is received from the CXPI network, and it shall exclude the start bit and a stop bit of each byte. An inversion of logical value '0_2' and '1_2' before and after the calculation shall not be done. The L_CRC8 operation of transmission node shall be performed according to the range defined in Table 8 in the data which transmits according to Formula (2), and the result shall be stored to L_CRC8 as it transmits from LSB of crc_rg in order on the CXPI network. The L_PTYPE field shall be excluded from L_CRC8 operand. The L_CRC8 operation of the receiving node shall perform the logic as specified in Table 8 according to Formula (2) and it shall support error detection. The initial value of crc_rg shall be '00_{16}'.

Definition of formula:

$$CRC8 = X8 + X4 + X1 + 1 \quad (2)$$

Table 8 — CRC8 operand processing

Operand	L_PID (1 byte) L_FI (1 byte) L_DATA (0 byte to 12 byte)

```
#define MSB_CRC8_R     (0xC8)       // x8 + x4 + x1 + x0
static unsigned char GetCRC8_R(const void *buff, size_t size)
{
    unsigned char *p = (unsigned char *)buff;
    unsigned char crc_rg = 0x00;
    int i;
    for(crc_rg = 0x00 ; size != 0 ; size--){
        crc_rg ^= *p++;
        for (i = 0 ; i < 8 ; i++){
            if (crc_rg & 0x01){
                crc_rg >>= 1;
                crc_rg ^= MSB_CRC8_R;
            }
            else{
                crc_rg >>= 1;
            }
        }
    }
    return crc_rg;
}
```


8.4.5.3 DLL — CRC16 calculation

This requirement specifies the DLL L_CRC16 calculation.

REQ	2.18 DLL — CRC16 calculation
The L_CRC16 calculation shall be performed if the L_FI_DLC = '1111_2' for both, transmitting and the receiving nodes. The L_CRC16 operation shall be performed in bit order which is received from CXPI network, and it shall exclude the start bit and a stop bit. An inversion of logical '0_2' and '1_2' before and after the calculation shall not be done. The L_CRC16 operation of transmission node shall be performed according to the range defined in Table 9 in the data which transmits according to Formula (3), and the result shall store to L_CRC16 as it transmits from LSB of crc_rg in order on the CXPI network. The L_PTYPE field shall be excluded from L_CRC16 operand. The L_CRC16 operation of receiving node shall perform the logic as specified in Table 9 according to Formula (3) and it shall support error detection. The initial value of crc_rg shall be '0000_{16}'.	

Definition of formula:

$$CRC16=X16+X12+X5+1 \quad (3)$$

Table 9 — CRC16 operand processing

Range	L_PID (1 byte) L_FI (2 byte) L_DATA (0 byte to 255 byte)

```
#define MSB_CRC16_R     (0x8408)       // x16 + x12 + x5 + x0
static unsigned short GetCRC16_R(const void *buff, size_t size)
{
    unsigned char *p = (unsigned char *)buff;
    unsigned short crc_rg = 0x0000;
    int i;
        for (crc_rg = 0x0000; size != 0; size--){
        crc_rg ^= *p++;
        for (i = 0; i < 8; i++){
            if (crc_rg & 0x0001){
                crc_rg >>= 1;
                crc_rg ^= MSB_CRC16_R;
            }
            else{
                crc_rg >>= 1;
            }
        }
    }
    return crc_rg;
}
```

8.5 DLL — Internal operation

The DLL exchanges frames with the physical layer.

REQ	2.19 DLL — Internal operation — Unsegmented frame
The internal operation of the CXPI DLL shall specify unsegmented frame transmission and reception.	

This requirement specifies the frame serialisation and bit order.

REQ	2.20 DLL — Internal operation — Frame serialisation and bit order
Figure 8 specifies frame serialisation and bit order.	

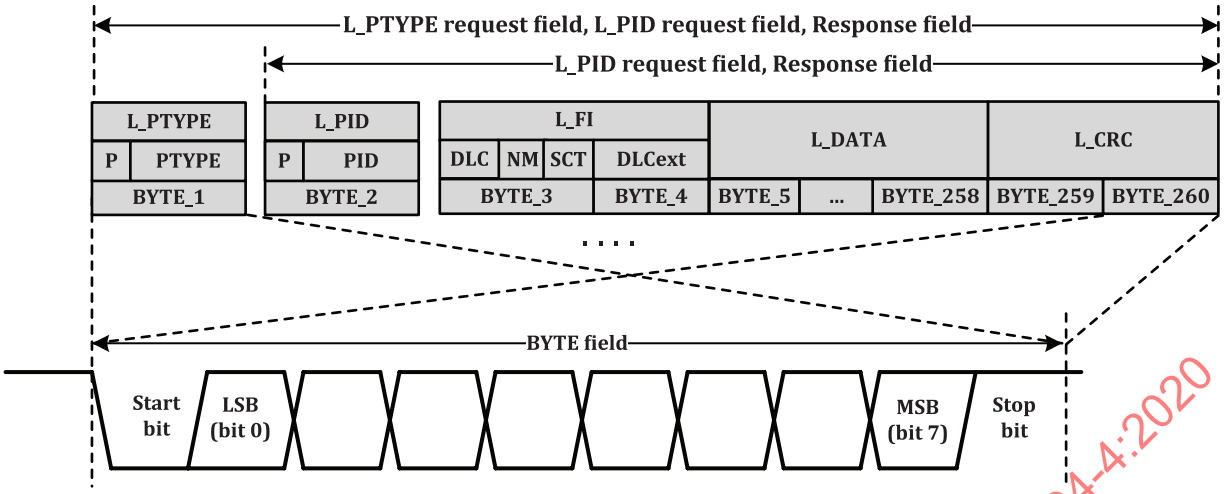


Figure 8 — DLL frame serialisation and bit order

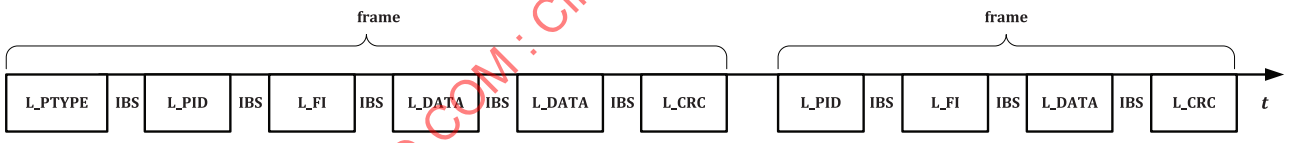
8.6 DLL — Timing parameters

8.6.1 DLL — IBS timing handling

When byte data is continuously transmitted at the time of transmission of response, the interval of each byte data is called IBS (inter-byte-space) in a frame.

REQ	2.21 DLL — IBS timing handling
The transmitting node shall use an IBS of less than 9-bit length or more than 1-bit length with a logical value of '1'.	

Figure 9 shows the relation between byte data and IBS that composes the frame when byte data is continuously transmitted.



Key

t time

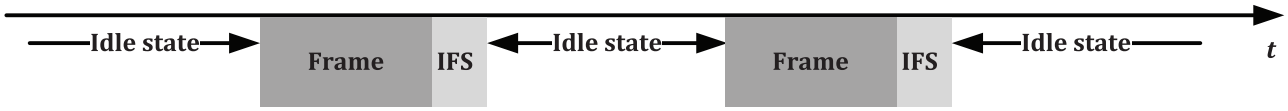
Figure 9 — DLL relation between byte data and IBS that compose a frame

8.6.2 DLL — IFS timing handling

The bit field with the consecutive logical value '1', called IFS (inter-frame-space) exists just behind the frame, and it is used for the separation to the next frame. All nodes generate an IFS after the reception of the CRC in the response field.

REQ	2.22 DLL — IFS timing handling
The IFS shall be inserted after the `L_CRC` of a transmitted frame. The IFS shall consist of logical value '1' of 20-bit length or more.	

Figure 10 shows the relation of frame, IFS and idle state.



Key

t time

Figure 10 — DLL relation of frame, IFS and idle state

8.6.3 DLL — Beginning condition of frame

This requirement specifies the beginning condition of frame.

REQ	2.23 DLL — Beginning condition of frame
The reception beginning condition of frame for the receiving node shall be when either of the definition columns is established. — After logical value '1' of 10-bit length or more from a stop bit receiving completion or transmission completion. It shall include at the time of framing error occurring to receiving and transmission completion of a stop bit. — After IFS. The transmission beginning condition of frame for the receiving node shall be when either of the definition column is established. — After IFS. It may perform from a stop bit receiving completion or transmitting completion. In that case, it shall include at the time of framing error occurring to receiving and transmission completion of a stop bit. NOTE Immediately after wake-up, the reception beginning condition of frame is only “after IFS” for synchronisation with frame.	

Figure 11 shows the timing chart of the wake-up condition for the DLL. While Node X and Node A communicate, Node X and Node A change the state after a fixed time elapse with a stop bit as the starting point.

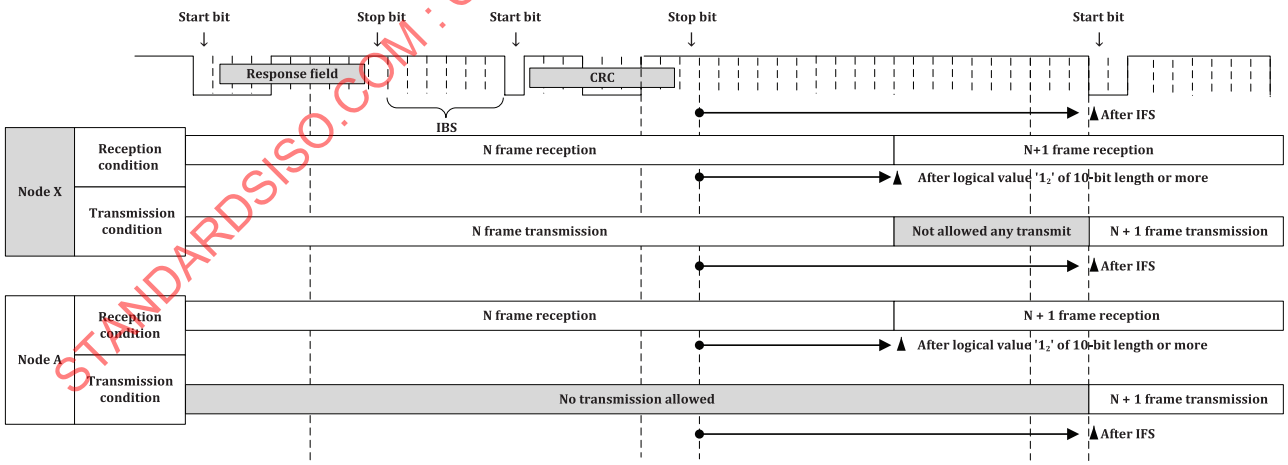


Figure 11 — Wake-up condition for DLL

Figure 12 shows the timing chart of the beginning condition of a frame after wake-up. Since the position of a stop bit is unknown, Node B changes the state after the IFS.

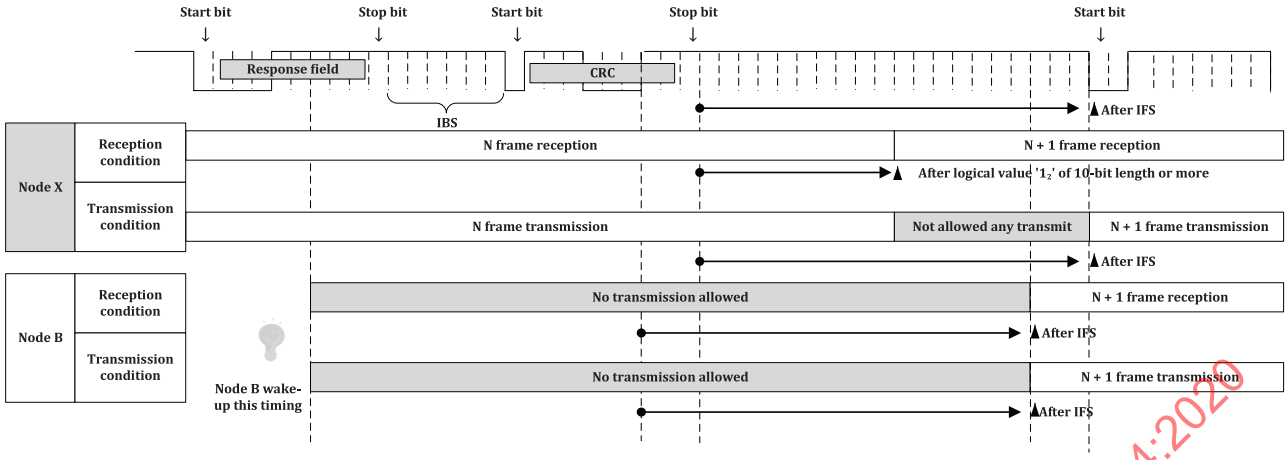
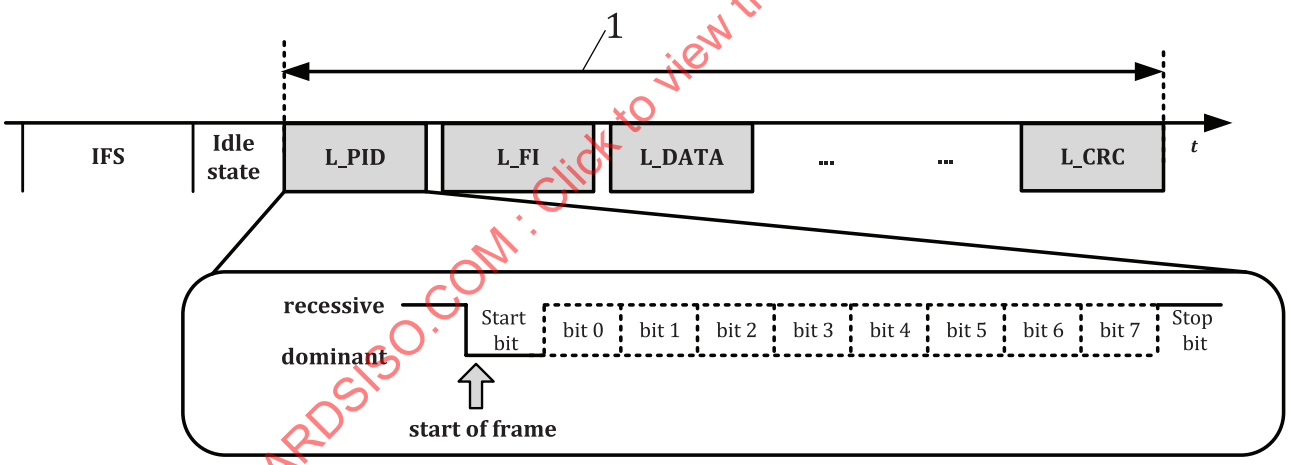


Figure 12 — Wake-up condition for DLL (after wake-up)

8.6.4 DLL — Start of frame

This requirement specifies the start of frame.

REQ	2.24 DLL — Start of frame
The start bit received first after IFS is detected shall be the start bit of a frame (see Figure 13).	



Key

1 frame

t time

Figure 13 — DLL start of frame

8.6.5 DLL — Frame transmission time

This requirement specifies the frame transmission time. The calculation excludes the IFS.

REQ	2.25 DLL — Frame transmission time
The total frame transmission time shall be comprised of the frame length, which is specified from the start bit of the start of frame to the stop bit of the frame end including the inter byte spaces (IBS). Table 10 specifies the maximum frame length.	

Table 10 — Maximum frame transmission time calculation

Frame type (byte)	Transmission mode	Maximum frame length (t_{bit})
L_FI_DLC ≠'1111_2'	response transmission time after L_PID	$t_{pid\_response}$ = [30 + (10 × data length)] × (1 + 0,9)[a]
	response transmission time after L_PTYPE	When there is L_PTYPE field: 10 + 9 + $t_{pid\_response}$[b]
L_FI_DLC = '1111_2'	response transmission time after L_PID	$t_{pid\_response\_long}$ = [50 + (10 × data length)] × (1 + 0,9)[a]
	response transmission time after L_PTYPE	When there is L_PTYPE field: 10 + 9 + $t_{pid\_response\_long}$[b]
Sleep frame (data length = 8)	response transmission time after L_PID	$t_{pid\_response}$ = [30 + (10 × data length)] × (1 + 0,9)[c]

NOTE It is recommended to use an IBS = 6 for implementations.

[a] 0,9 in the formula indicates the total of IBS, which is composed of L_PID field and response field.

[b] If there is a L_PTYPE field, the maximum frame length shall be long that L_PTYPE field 10-bit and following maximum 9 bit of IBS are added maximum length of L_PID field and response field.

[c] For the maximum frame length of response transmission after L_PID, data length shall be 8 byte.

8.7 DLL — Completion of frame

8.7.1 DLL — General

This subclause describes the completing condition of frame, L_PTYPE field, and L_PID field respectively.

8.7.2 DLL — Completing condition of L_PTYPE field

This requirement specifies the completing condition of L_PTYPE field.

REQ	2.26 DLL — Completing condition of L_PTYPE field
Table 11 specifies the completing condition of L_PTYPE field.	

Table 11 — Completing condition of L_PTYPE field

Item	Description
Completing condition	— Transmitting node Complete transmission of all bits of L_PTYPE field to CXPI network. — Receiving node Reception of all bits of the L_PTYPE field and confirmation that there is no error.

8.7.3 DLL — Completing condition of L_PID field

This requirement specifies the completing condition of L_PID field.

REQ	2.27 DLL — Completing condition of L_PID field
Table 12 specifies the completing condition of L_PID field.	

Table 12 — Completing condition of L_PID field

Item	Description
Completing condition	— Transmitting node Complete transmission of all bits of `L_PID` field to CXPI network. — Receiving node Complete reception of all bits of the `L_PID` field and confirmation that there is no error.

8.7.4 DLL — Completing condition of frame

This requirement specifies the completing condition of frame.

REQ	2.28 DLL — Completing condition of frame
Table 13 specifies the completing condition of frame.	

Table 13 — Completing condition of frame

Item	Description
Completing condition	— Transmitting node The frame is completed when there is no error until the transmission of the last bit of `L_CRC` and detecting the logical value '1' continuously during the 10-bit after `L_CRC`. — Receiving node The frame is completed when there is no error until receiving the last bit of `L_CRC` and detecting the logical value '1' continuously during the 10-bit after `L_CRC`.

8.8 DLL — Byte arbitration

Each CXPI node performs byte comparison between a transmitted and the received byte from the CXPI network. If there is no equal match of the byte values, the transmission is aborted.

REQ	2.29 DLL — Byte arbitration
The DLL reception function shall perform byte arbitration by comparing the transmitted byte to the physical layer with the received byte from the physical layer. — If there is an equal match between both bytes, then the remaining bytes of the frame shall be transmitted. — If there is no equal match between both bytes, then the transmission of remaining bytes shall be stopped and shall set an arbitration lost (`DLL_Arb_Lost`) in the L_Result parameter. — If the compared bytes are not equal, the transmitting node shall stop transmission and shall transit into reception mode.	

NOTE At the time of the start of a frame transmission, the existence of data reception is checked, and frame transmission is started when no data are received.

8.9 DLL — Function models

8.9.1 DLL — Transmission logic

The DLL transmission function model specifies the logic and the mapping of service interface parameters to the DLL-specific frame parameters. The DLL transmission logic distinguishes between frame type L_Ftype = NormalCom and L_Ftype = DiagNodeCfg transmissions.

If the `L_Ftype = NormalCom` then the length of the N_PDU determines, whether a `NormalCom[L_ReqId]` or a `NormalCom[L_ReqId, L_PDU]` is transmitted to the lower OSI layer. If the `L_Ftype = DiagNodeCfg` then a `DiagNodeCfg[L_ReqId, L_PDU]` is transmitted to the lower OSI layer.

REQ	2.30 DLL — Function models — DLL — Transmission logic
The DLL transmission function model shall be in accordance with the logic specified in Figure 14.	

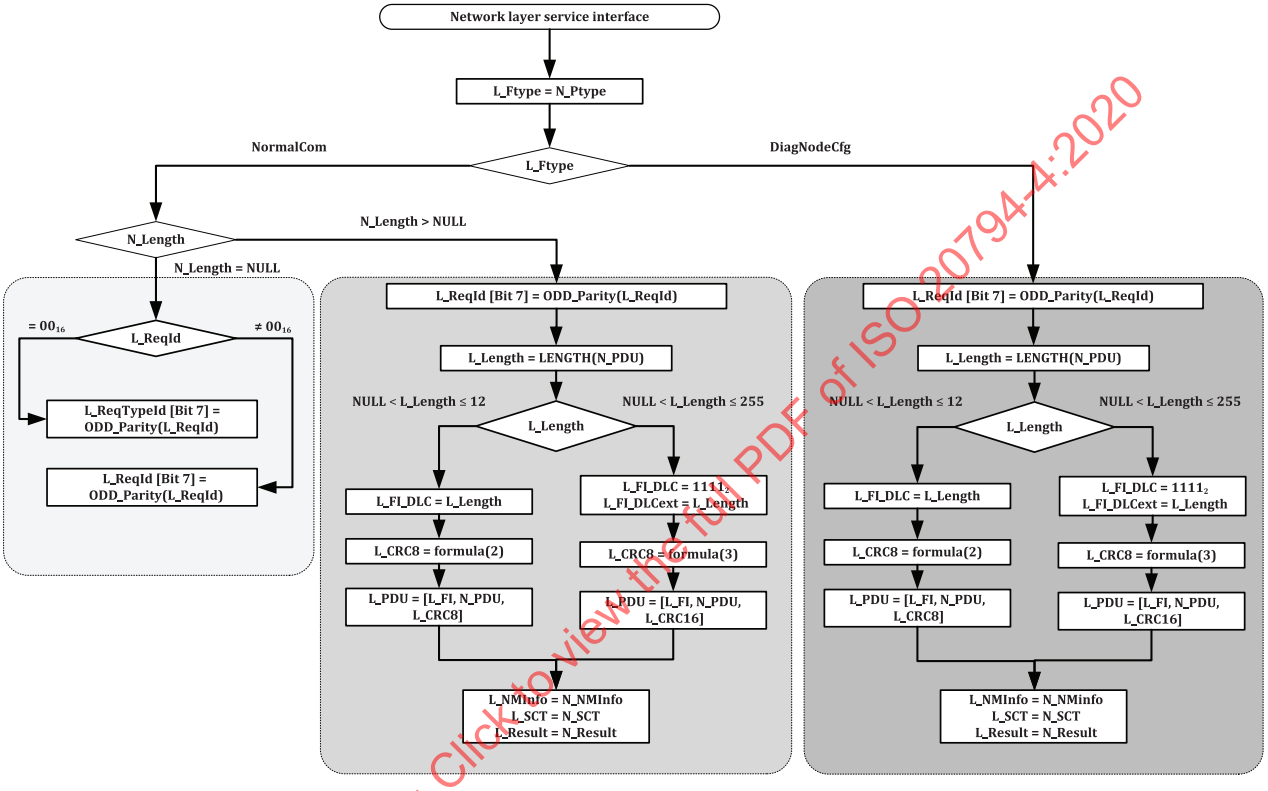


Figure 14 — DLL transmission logic

8.9.2 DLL — Reception logic

The DLL reception function model specifies the logic and the mapping of service interface parameters to the NL-specific packet parameters. The DLL reception logic distinguishes between frame type `L_Ftype = NormalCom` and `L_Ftype = DiagNodeCfg` receptions.

REQ	2.31 DLL — Function models — DLL — Reception logic
The DLL reception function model shall be in accordance with the logic specified in Figure 15.	

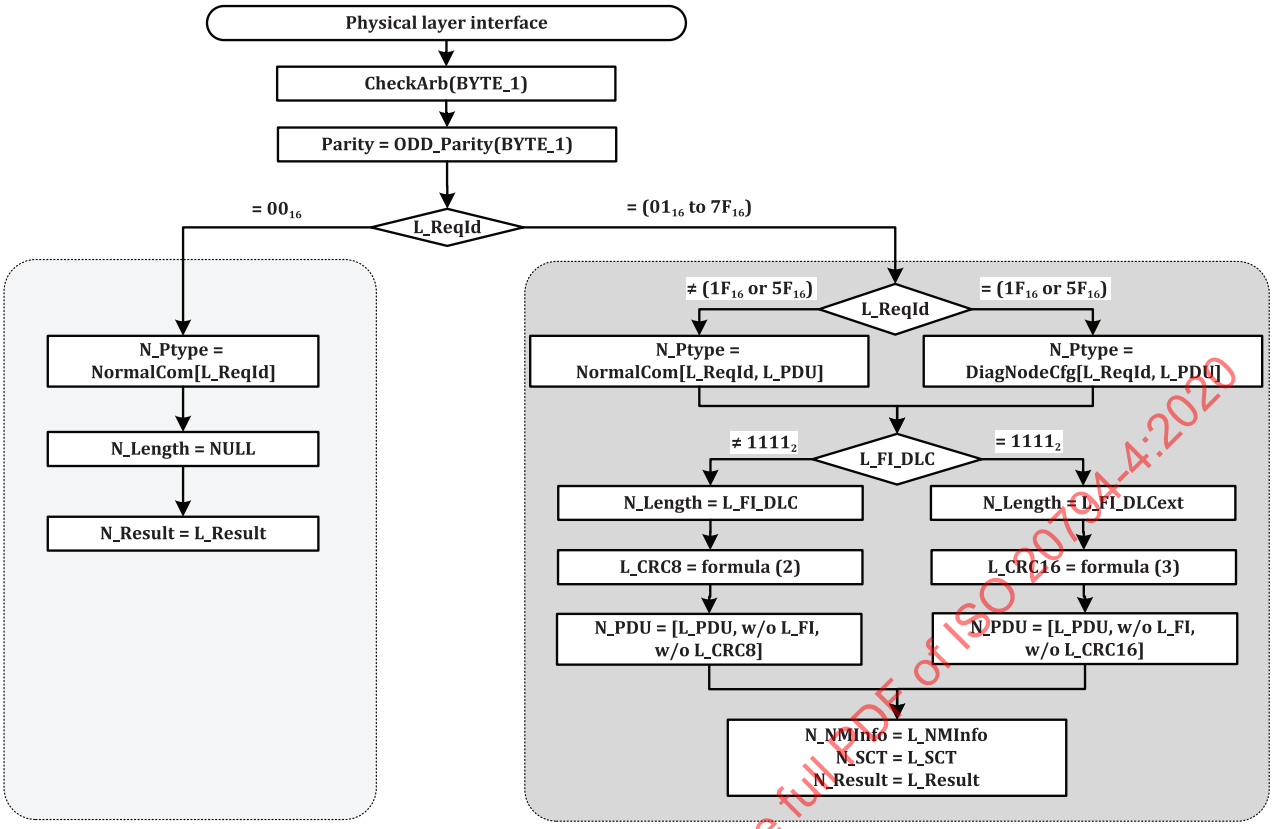


Figure 15 — DLL reception function model

8.10 DLL — Error detection

8.10.1 DLL — General

The DLL has the function to detect the following errors:

a) Byte error (`Err_DLL_Byte`);

b) CRC error (`Err_DLL_CRC`);

c) Parity error (`Err_DLL_Parity`);

d) Data length code error (`Err_DLL_DLC`);

e) Data length code extension error (`Err_DLL_DLCext`);

f) Framing error (`Err_DLL_Framing`);

8.10.2 DLL — Byte error (Err_DLL_Byte)

This requirement specifies the byte error.

REQ	2.32 DLL — Byte error (Err_DLL_Byte)
The transmitting node shall compare the byte value (including start bit of request field, stop bit of request field and response field) of the transmitted byte and the received byte and shall indicate the byte error when there is no match. The node shall stop the transmission. If the transmitting node detects the byte error it shall set the byte error (`Err_DLL_Byte`) in the `L_Result` parameter.	

8.10.3 DLL — CRC error (Err_DLL_CRC)

This requirement specifies the L_CRC error.

REQ	2.33 DLL — CRC error (Err_DLL_CRC)
If the CRC operation result (including L_PID, L_FI, and L_DATA) differs from the received L_CRC, the receiving node shall discard the frame and shall set the CRC error (Err_DLL_CRC) in the L_Result parameter.	

8.10.4 DLL — Parity error (Err_DLL_Parity)

This requirement specifies the parity error.

REQ	2.34 DLL — Parity error (Err_DLL_Parity)
If the receiving node detects an even number of bits of the received request field with a logical value of '1', the receiving node shall discard the received request filed and shall set the parity error (Err_DLL_Parity) in the L_Result parameter.	

8.10.5 DLL — Data length code error (Err_DLL_DLC)

This requirement specifies the data length code error.

REQ	2.35 DLL — Data length code error (Err_DLL_DLC)
If the transmitting node receives more or less bytes as indicated in the data length value, the transmitting node shall stop the transmission and shall set a data length code error (Err_DLL_DLC) in the L_Result parameter. If the receiving node receives more or less bytes as indicated in the L_FI_DLC or L_FI_DLCext value, the receiving node shall discard the frame and shall set a data length code error (Err_DLL_DLC) in the L_Result parameter.	

REQ	2.36 DLL — Data length code error (Err_DLL_DLC) — L_FI_DLC = [D_{16} to E_{16}]
If Ftype = DiagNodeCfg and if the receiving node receives a frame with L_Length of L_FI_DLC equal to D_{16} to E_{16}, the frame shall be ignored.	

REQ	2.37 DLL — Data length code error (Err_DLL_DLC) — L_FI_DLC = [D_{16} to E_{16}] L_Result = Err_DLL_DLC
If Ftype = DiagNodeCfg and if the receiving node receives a frame with L_Length of L_FI_DLC equal to D_{16} to E_{16}, it shall set the Err_DLL_DLC in the L_Result parameter.	

8.10.6 DLL — Data length code extension error (Err_DLL_DLCext)

This requirement specifies the data length code extension error.

REQ	2.38 DLL — Data length code error (Err_DLL_DLC) — L_FI_DLCext = [0_{16} to C_{16}]
If Ftype = DiagNodeCfg and if the receiving node receives a frame with L_Length of L_FI_DLCext equal to 0_{16} to C_{16}, the frame shall be ignored.	

REQ	2.39 DLL — Data length code error (Err_DLL_DLC) — L_FI_DLCext = [0_{16} to C_{16}] — L_Result = Err_DLL_DLCext
If Ftype = DiagNodeCfg and if the receiving node receives a frame with L_Length of L_FI_DLCext equal to 0_{16} to C_{16}, it shall set the Err_DLL_DLCext in the L_Result parameter.	

8.10.7 DLL — Framing error (Err_DLL_Framing)

This requirement specifies the framing error.

REQ	2.40 DLL — Framing error (Err_DLL_Framing)
If the stop bit of the received byte is a logical value '0', the transmitting node shall stop the transmission and shall set the `Err_DLL_Framing` in the `L_Result` parameter. If the stop bit of the received byte is a logical value '0', the receiving node shall stop the reception, shall discard the frame, and shall set the `Err_DLL_Framing` in the `L_Result` parameter.	

8.10.8 DLL — Exception handling for L_FI_DLC = '1111_2' detection

This requirement specifies the exception handling for L_Length > 12 detection.

REQ	2.41 DLL — Exception handling for L_Length = '1111_2' detection
If a node does not support frames with `L_Length > 12`, the node shall disregard the `L_FI_DLC` value without setting `L_Result` indicating an error.	

9 Physical layer (PHY)

9.1 PHY — Overview

The CXPI physical layer (PHY) comprises the physical signalling (PS) sub-layer, which is implemented in the CXPI protocol or the CXPI transceiver or the CXPI System Base Chip (SBC). It also comprises the physical media attachment (PMA) sub-layer, which is implemented normally in the CXPI transceiver or the CXPI SBC, which also may implement additional protection circuitry, not in scope of this document. The CXPI PHY also comprises the physical media dependent (PMD) sub-layer, which includes cable, and the termination resistors.

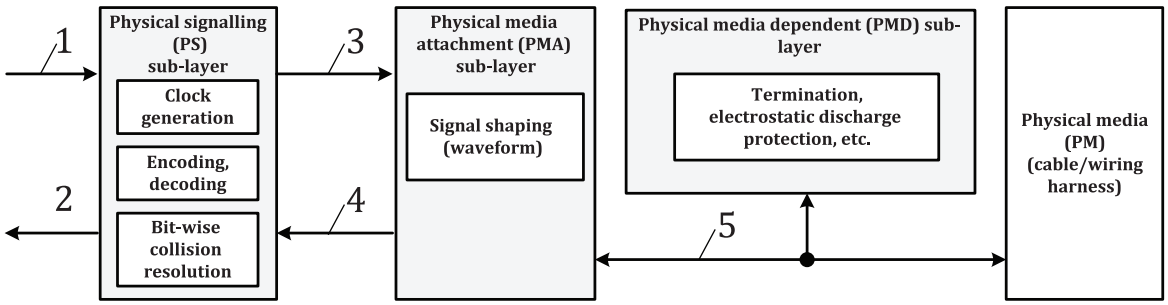
Figure 16 specifies an overview of the PHY sub-layers. The PHY sub-layers are comprises the:

— Physical signalling (PS) sub-layer, which specifies the requirements of the clock generation function, the encoding and decoding of CXPI frames, and the bit-wise collision resolution logic.

— Physical media attachment (PMA) sub-layer, which specifies the requirements of the signal shaping waveform logic.

— Physical media dependent (PMD) sub-layer, which specifies the requirements of the CXPI network termination, electrostatic discharge protection, etc., and device connector requirements.

— Physical media (PM), which specifies the requirements of the CXPI network cable/wiring harness.



Key

1 TXD_{NRZ}
2 RXD_{NRZ}
3 TX_{PWM}
4 RX_{PWM}
5 CXPI network

Figure 16 — PHY sub-layers overview

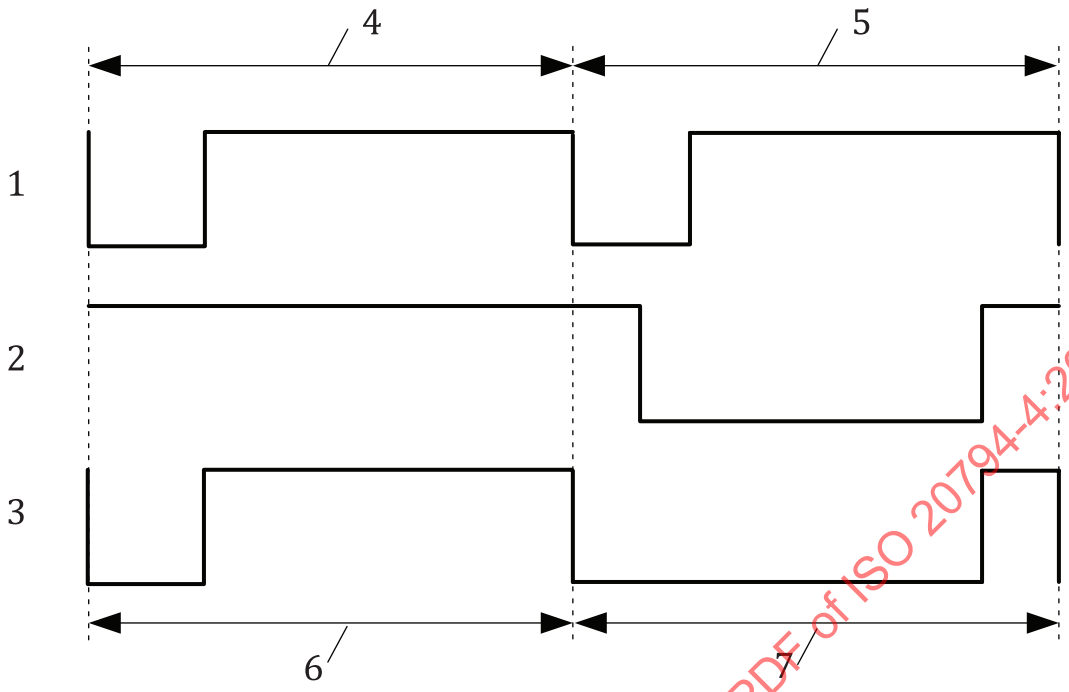
9.2 PHY — Concept of waveform generation

Figure 17 shows the concept of waveform generation of the logical values '1' and '0':

a) TX_{PWM} for logical values '1' and '0' for clock master,

b) TX_{PWM} for logical values '1' and '0' for other nodes than clock master,

c) CXPI network integrated output of a) and b).



Key

1 TX_{PWM} of clock master
2 TX_{PWM} of another node
3 CXPI network integrated output of a) and b)
4 logical value '1'
5 logical value '0'
6 1st bit
7 2nd bit

Figure 17 — PHY concept of waveform generation of logical values '1' and '0'

9.3 PHY — Physical signalling (PS) requirements

9.3.1 PHY — PS general

The PS has the following properties:

— encoding function;

— decoding function;

— clock generation;

— clock synchronisation;

— detection of clock existence;

— bit synchronisation;

— bit-wise collision resolution.

9.3.2 PHY — PS physical interface configuration

Figure 18 shows an example of the physical interface configuration.

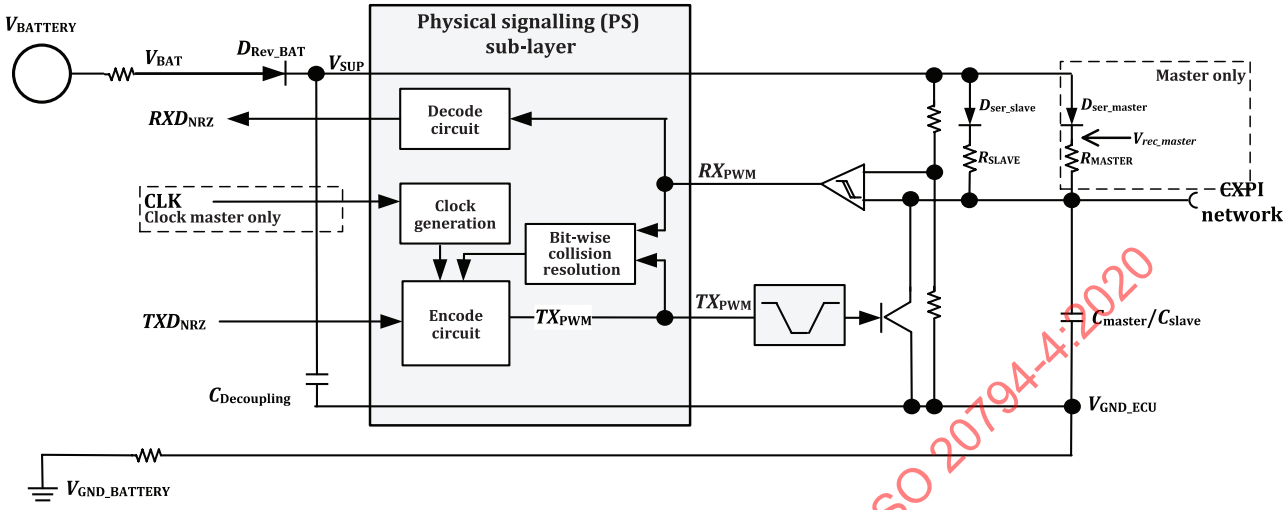


Figure 18 — Example of PHY PS physical interface configuration

9.3.3 PHY — PS bit rate

This requirement specifies the PS system bit rate.

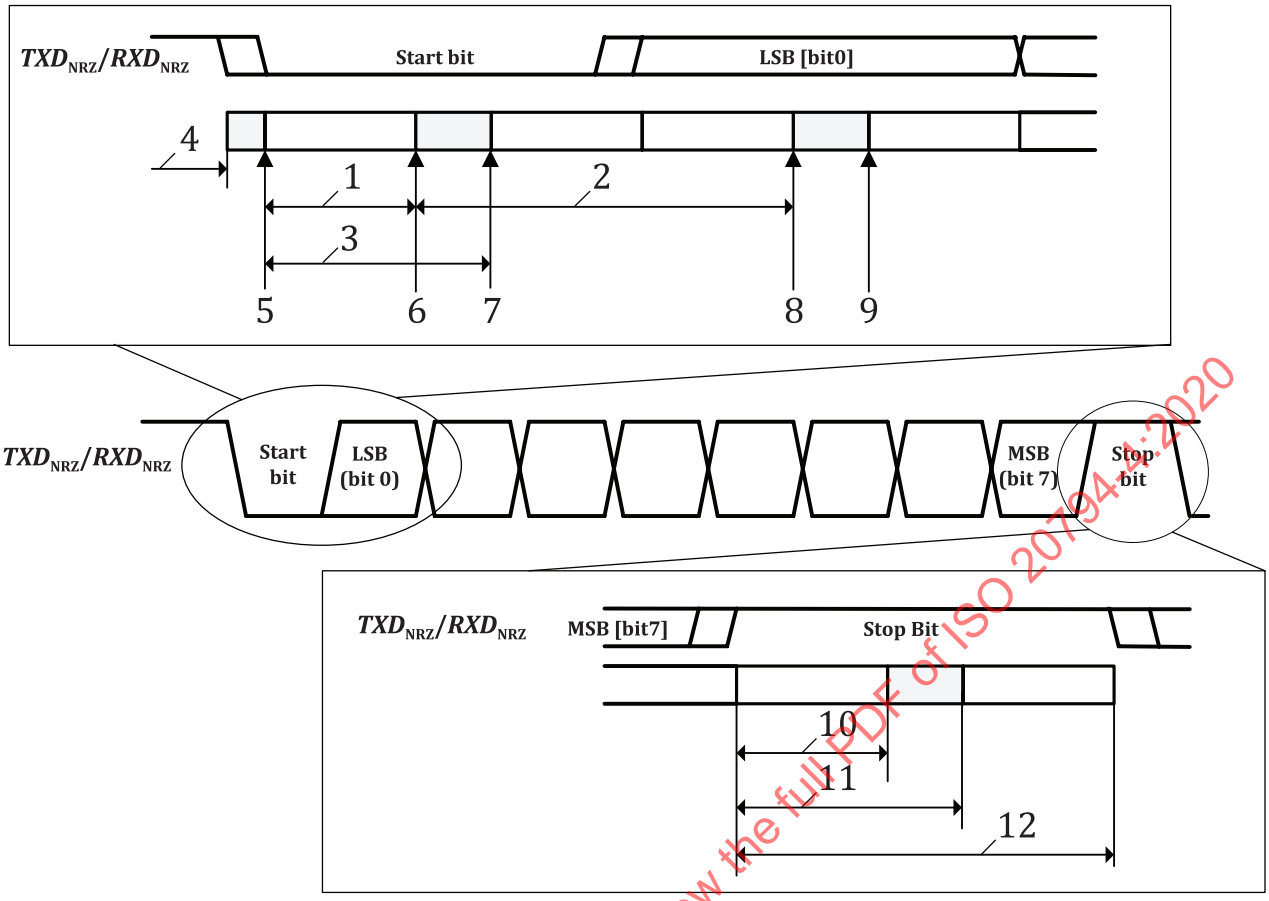
REQ	1.1 PHY — PS bit rate
The PS sub-layer shall transmit symbols (logical value '1_2' and logical value '0_2') with a bit rate of 20 kbit/s at maximum.	

The recommended maximum bit rate is 19,2 kbit/s.

9.3.4 PHY — PS bit sample timing

Signal edges from HI level to LO level can be used for synchronisation.

REQ	1.2 PHY — PS bit sample timing
To ensure proper operation under worst case conditions, the parameters specified below and in Figure 19 shall be fulfilled. — Value of accuracy of the byte field detection (t_{bfs}) is 1/16 t_{bit} as typical and 2/16 t_{bit} as maximum. — Earliest bit sample time (t_{ebs}) is 7/16 t_{bit} as minimum. — Latest bit sample time (t_{lbs}) is 10/16 t_{bit} – t_{bfs}.	



Key

1	$t_{ebs} = 7/16\ t_{bit}$	7	LBS[0]
2	$t_{ebs}[1] - t_{ebs}[0] = t_{bit}$	8	EBS[1]
3	$t_{lbs} = 10/16\ t_{bit} - t_{bfs}$	9	LBS[1]
4	t_{bfs}	10	$t_{ebs} = 7/16\ t_{bit}$
5	falling edge of start bit detect	11	$t_{lbs} = 10/16\ t_{bit} - t_{bfs}$
6	EBS[0]	12	t_{bit}

Figure 19 — PHY PS network system bit sample timing

9.3.5 PHY — PS encoding and decoding logic

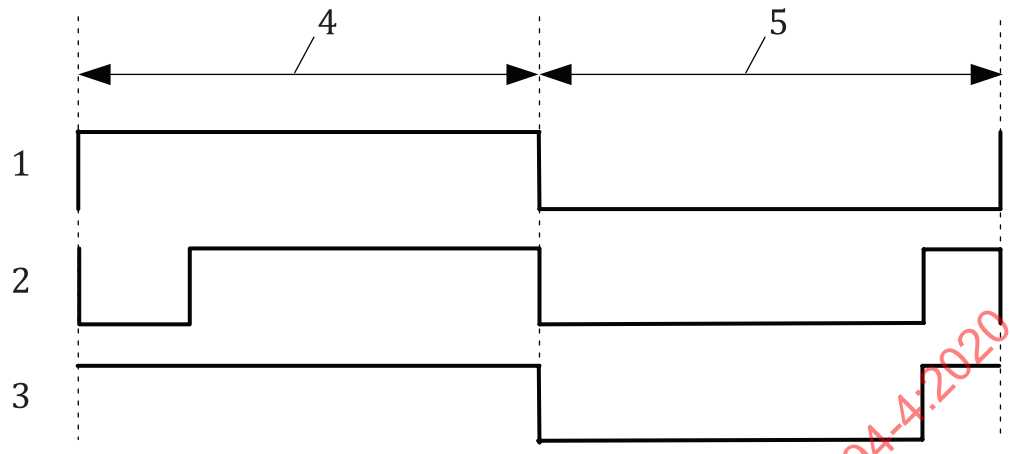
9.3.5.1 PHY — PS general

In the normal state, the PWM encoding and PWM decoding are implemented. This subclause describes each processing.

9.3.5.2 PHY — PS encoding function

This requirement specifies the function.

REQ	1.3 PHY — PS encoding function
The encoding function of the clock master shall convert the NRZ signal to a PWM signal mixing with the clock. The encoding function of the node shall convert the NRZ signal to a PWM signal. Figure 20 specifies the waveform of TXD_{NRZ} and TX_{PWM}.	



Key

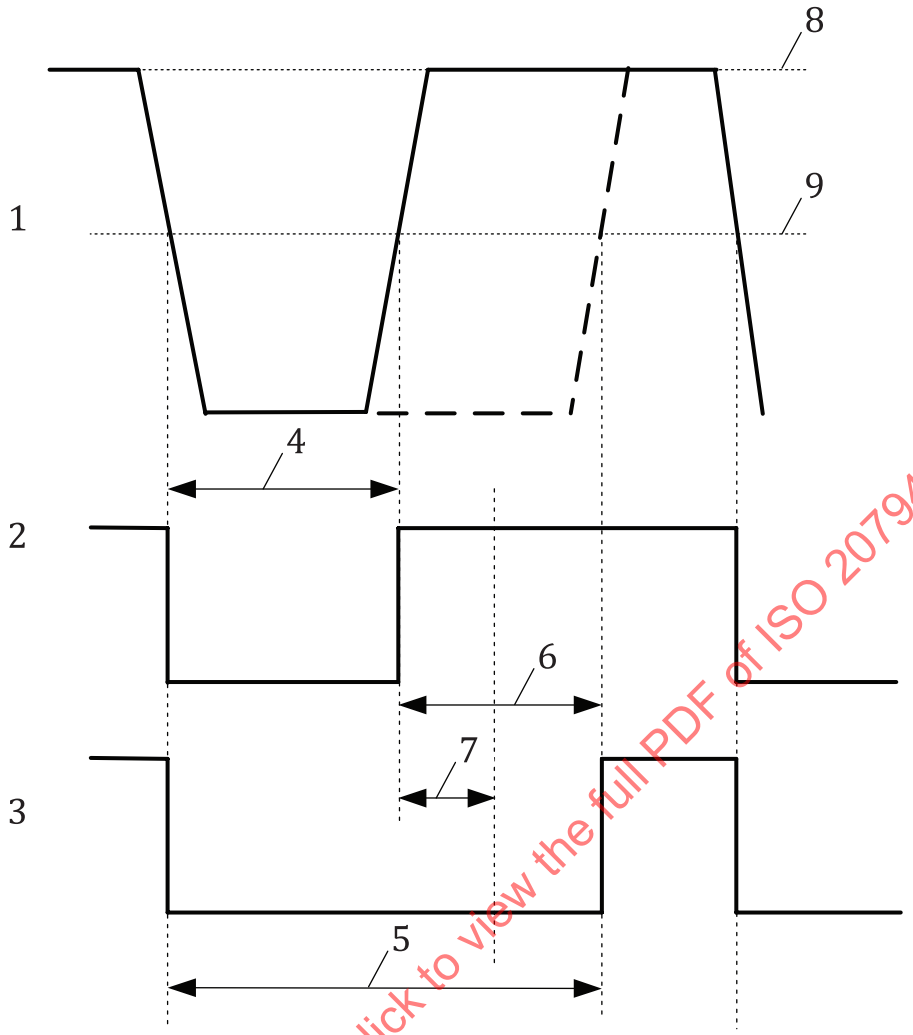
1 TXD_{NRZ}
2 TX_{PWM} (clock master)
3 TX_{PWM} (node)
4 logical value '1'
5 logical value '0'

Figure 20 — PHY PS conversion of the waveforms of TXD_{NRZ} and TX_{PWM}

9.3.5.3 PHY — PS decoding function

This requirement specifies the decoding function.

REQ	1.4 PHY — PS decoding function
The decoding function shall convert the PWM signal to an NRZ signal. Figure 21 specifies the difference of width of LO level at the constant threshold that receiving node discriminates logical value '1' and logical value '0'. Table 14 specifies the difference of width of LO level parameter. If rescessive voltage of CXPI network does not reach V_{rec_master}, the fall edge of next bit will become early. As a consequence, extends the LO width of the logical value '1', and therefore here it prescribes the min value of $t_{sample\ cont}$.	



Key

1 CXPI network
2 logical value '1' of RX_{PWM}
3 logical value '0' of RX_{PWM}
4 $t_{rx\_1\_lo\_cont}$
5 $t_{rx\_0\_lo\_cont}$
6 $t_{rx\_dif\_cont}$
7 t_{sample_cont}
8 V_{rec_master}
9 V_{BUS_CNT}

Figure 21 — PHY PS difference of width of LO level

Table 14 — Difference of width of LO level parameter

Param no.	Symbol	min.	typ.	max.	unit	Description
1	$t_{rx\_dif\_cont}$	2,5	—	—	µs	$t_{rx\_dif\_cont} = (t_{rx\_0\_lo\_cont} - t_{rx\_1\_lo\_cont})$; t_{bit} = 50 µs; $t_{rx\_dif\_cont} = 0{,}05 \times t_{bit}$.
2	t_{sample_cont}	0,8	—	—	µs	The value in which this parameter is added to the width of LO level of last logical value '1' is to be a threshold used for the decoding processing. t_{bit} = 50 µs; $t_{sample\_cont} = 0{,}016 \times t_{bit}$.

9.3.6 PHY — PS clock generation

The clock is a signal (logical value '1_2' or logical value '0_2') including the periodical following edge on the PMA sub-layer as shown in Figure 22.

REQ	1.5 PHY — PS clock generation
Clock transmission during the period which is not transmitting the data shall transmit consecutively the waveform equal to the logical value '1_2'.	

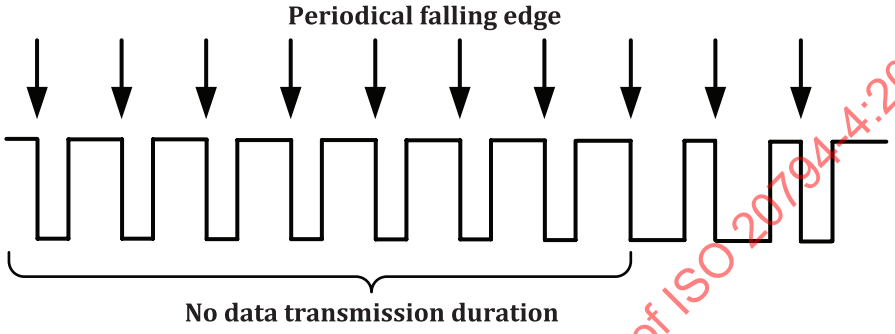
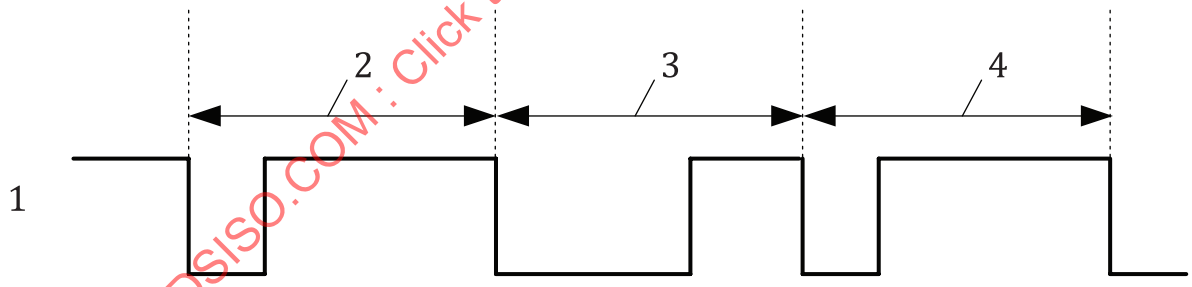


Figure 22 — PHY PS example of clock on the PMA sub-layer

REQ	1.6 PHY — PS clock generation — Transmitter jitter
Figure 23 specifies the transmitter jitter of the falling TX_{PWM} edge. Table 15 specifies the transmitter jitter parameter.	



Key

1 TX_{PWM}
2 t_{bit_ref} (bit width of reference bit rate)
3 $t_{bit\_ref} \times (1 + \Delta t_{bit\_cont})$
4 $t_{bit\_ref} \times (1 + \Delta t_{bit\_cont})$

Figure 23 — PHY PS transmitter jitter of the falling TX_{PWM} edge

Table 15 — PS transmitter jitter parameter

Param no.	Symbol	min.	typ.	max.	unit	Description
3	Δt_{bit_cont}	−0,005	—	+0,005	—	Transmitter jitter of falling TX_{PWM} edge. The jitter tolerance is related to bit width t_{bit_ref}.

The clock is stopping as shown in Figure 24.

REQ	1.7 PHY — PS clock generation — Clock stop
During the clock stopping, the potential of the TX_{PWM} shall be fixed to HI level.	

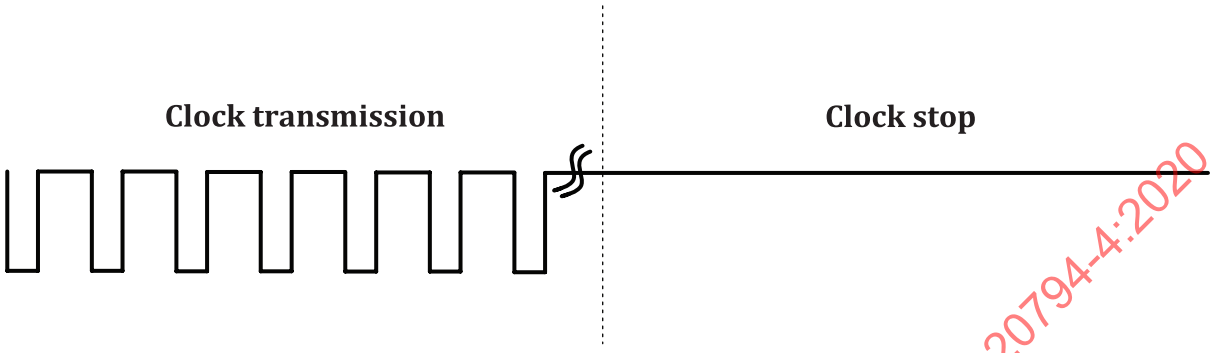
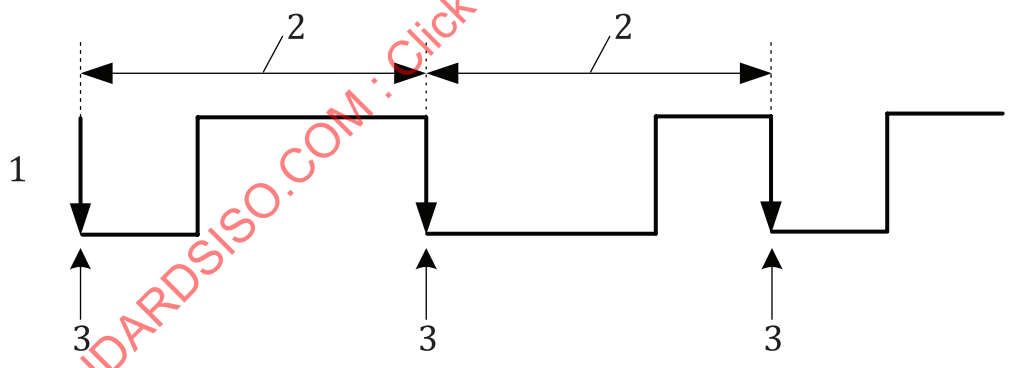


Figure 24 — PHY PS clock stop

9.3.7 PHY — PS node clock synchronization and bit synchronization

This requirement specifies the PS node clock synchronisation and bit synchronization.

REQ	1.8 PHY — PS node clock synchronization and bit synchronization
The node shall perform communication processing by synchronizing with the clock which determines the start of bit by the timing of the waveform of the PMA sub-layer changing from HI level to LO level (see Figure 25).	



Key

1 RX_{PWM}
2 waveform for 1 bit
3 start of bit

Figure 25 — PHY PS start of bit

9.3.8 PHY — PS detection of clock existence

This requirement specifies the PS detection of clock existence.

REQ	1.9 PHY — PS detection of clock existence
Table 16 specifies the requirement of clock existence detection.	

Table 16 — Clock existence condition

Condition	Description
with clock	Detecting the edge of clock on the RX_{PWM} except in case of the wake-up pulse sent by own node. Set the `ev_wakeup` to '`ev_clk_detect`'.
without clock	There is no edge except for 5 ms or more, on the RX_{PWM}. Set the `ev_wakeup` to '`ev_clk_loss`'.

9.3.9 PHY — PS bit-wise collision resolution

In the normal state, each node compares any received bit from the RX_{PWM} with the related bit transmitted to the TX_{PWM}.

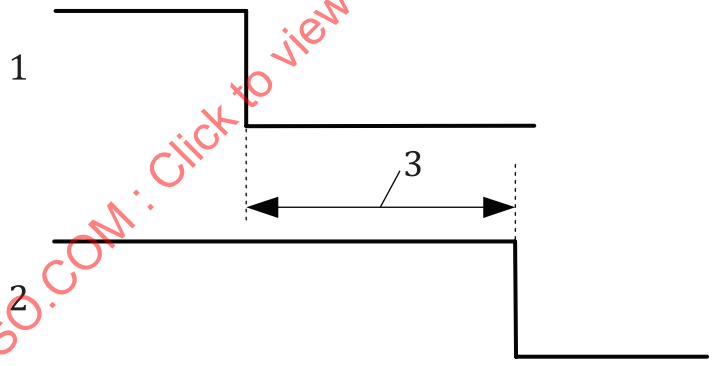
REQ	1.10 PHY — PS bit-wise collision resolution
The transmitting node shall discontinue the transmission of the next bit, when the read-back value of a bit in the PMA sub-layer is different compared to the transmitted bit.	

When the value of the bit is corresponding, the node may continuously transmit to the PMA sub-layer.

When two or more nodes begin transmitting simultaneously, the node that transmits the highest priority frame completes the transmission.

9.3.10 PHY — PS AC parameters

Figure 26 shows propagation delay of node transmitter logical value '0' timing parameters.

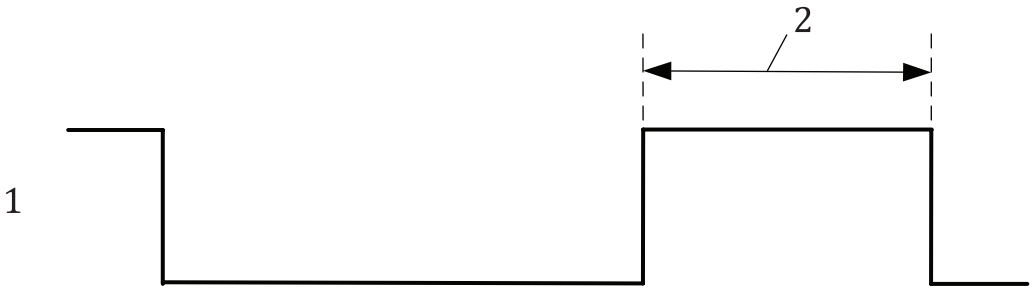


Key

1 RX_{PWM}
2 TX_{PWM}
3 $t_{tx\_0\_pd\_cont}$

Figure 26 — PHY PS propagation delay of node transmitter logical value '0'

Figure 27 shows receiving node detects HI level.



Key

1 RX_{PWM}

2 $t_{rx\_0\_hi\_cont}$

Figure 27 — PHY PS receiving node detects HI level

REQ	1.11 PHY — PS AC parameters
Table 17 specifies the AC parameter of the PS entity.	

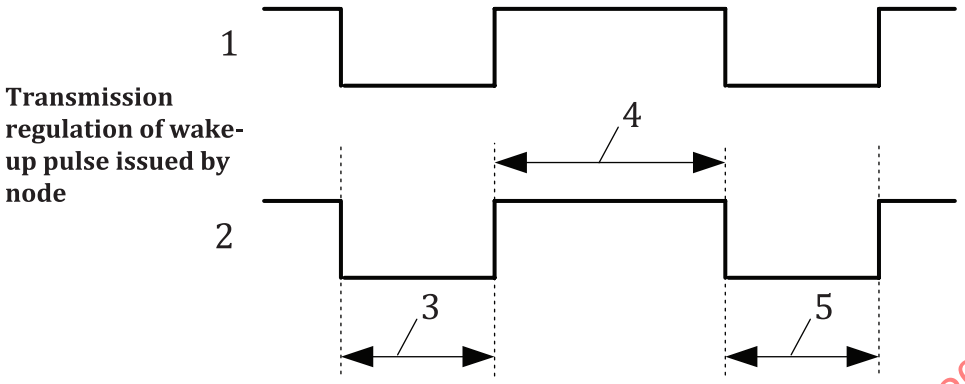
Table 17 — PS entity AC parameter

Param no.	Symbol	min.	typ.	max.	unit	Description
4	$t_{tx\_0\_pd\_cont}$	—	—	0,5	µs	At the time of logical value '0' outputs, time from the LO level detection of the RX_{PWM} until falling of the TX_{PWM} (see Figure 26). t_{bit} = 50 µs; $t_{tx\_0\_pd\_cont} = 0{,}01 \times t_{bit}$.
5	$t_{rx\_0\_hi\_cont}$	1	—	—	µs	The time that should be detected the receiving node is HI level (see Figure 27). t_{bit} = 50 µs; $t_{rx\_0\_hi\_cont} = (0{,}06 \times t_{bit}) - t_{rx\_pwm\_pd\_sym}$.

9.3.11 PHY — PS node transmission of wake-up pulse

Nodes can issue wake-up pulses.

REQ	1.12 PHY — PS node transmission of wake-up pulse
The higher OSI layers shall command the PS layer using the interface in 9.7 to send a wake-up pulse. Figure 28 specifies the waveform of the wake-up pulse and Table 18 specifies the parameters.	



Key

1 TXD_{NRZ}
2 TX_{PWM}
3 t_{tx_wakeup}
4 $t_{tx\_wakeup\_space}$
5 t_{tx_wakeup}

Figure 28 — PHY PS wake-up pulse waveform

The node may stop transmitting second dominant voltage of wake-up pulse, when the node detects the wake-up pulse on the CXPI network in the act of transmitting the wake-up pulse by the node.

Table 18 — PS network system parameter of wake-up pulse

Param no.	Symbol	Node	min.	typ.	max.	unit	Description
6	t_{tx_wakeup}	node	400	—	2 500	µs	Specifications of dominant level of the wake-up pulse that transmitting node transmits to PMA sub-layer. min: t_{bit} = 50 µs; t_{tx_wakeup} = 8 × t_{bit}.
7	$t_{tx\_wakeup\_space}$	node	5	—	10	ms	Interval time between two of dominant level of transmitting wake-up pulse.

9.4 PHY — Physical medium attachment (PMA) requirements

9.4.1 PHY — PMA general

Waveform shaping is a responsibility of the PMA. Figure 29 shows an example of the physical interface configuration.

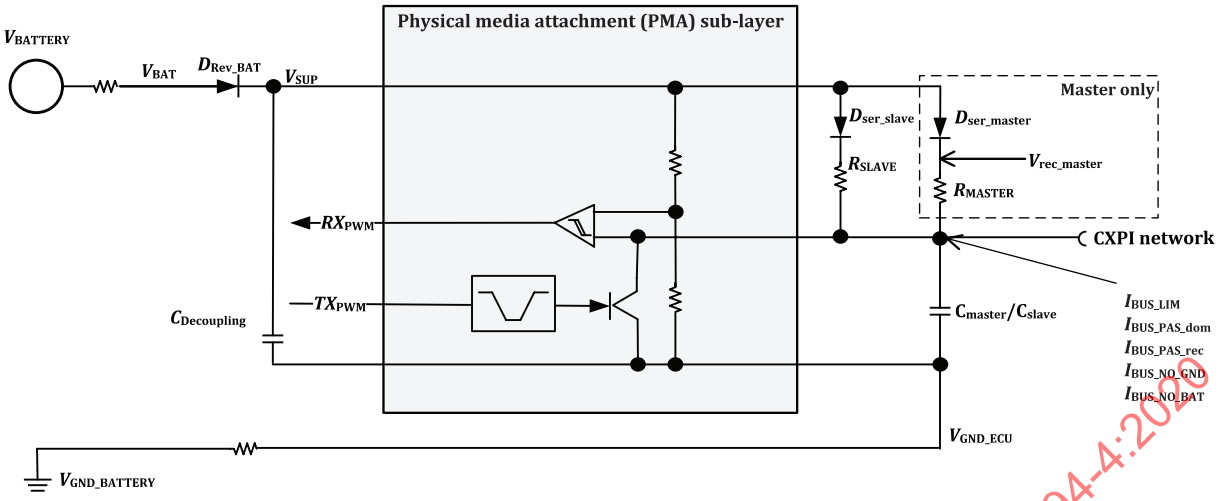


Figure 29 — Example of PHY PMA physical interface configuration

9.4.2 PHY — PMA electrical parameters

This requirement specifies the PMA electrical parameters.

REQ	1.13 PHY — PMA electrical parameters
Table 19 specifies the PMA electrical parameters.	

Table 19 — PMA electrical parameters

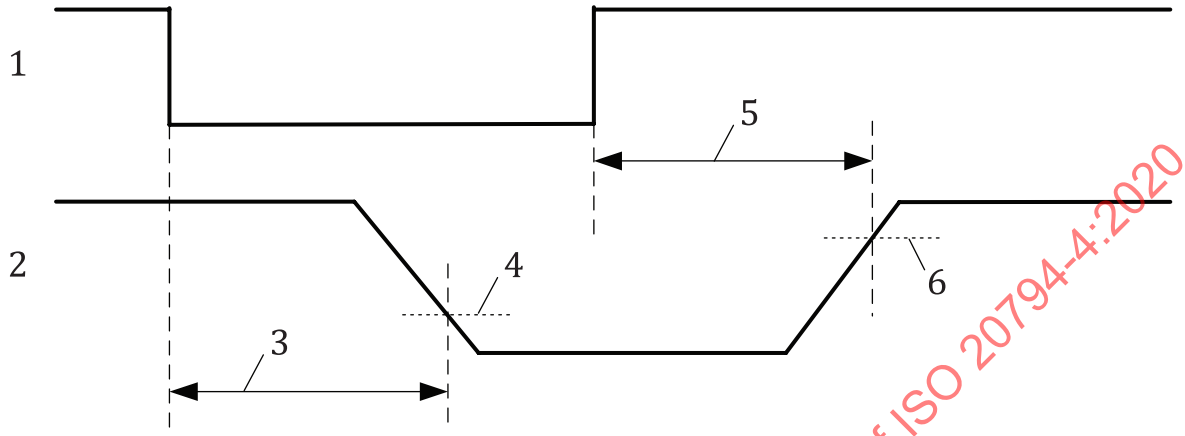
Param no.	Symbol	min.	typ.	max.	unit	Description
8	V_{BAT}	8	—	18	V	Operating voltage range
9	V_{SUP}	7,0	—	18	V	Supply voltage range
10	I_{BUS_LIM}	40	—	200	mA	Current limitation for driver dominant state driver on $V_{BUS} = V_{BAT\_max}$
11	$I_{BUS\_PAS\_dom}$	−1	—	—	mA	Input leakage current at the receiver including pull-up resistor as specified in Table 23 Param 40 driver off $V_{BUS} = 0$ V $V_{BAT} = 12$ V
12	$I_{BUS\_PAS\_rec}$	—	—	20	µA	Driver off 8 V < V_{BAT} < 18 V 8 V < V_{BUS} < 18 V $V_{BUS} \geq V_{BAT}$
13	$I_{BUS\_NO\_GND}$	−1	—	1	mA	Control unit disconnected from ground $V_{GND} = V_{SUP}$ 0 V < V_{BUS} < 18 V $V_{BAT} = 12$ V Loss of local ground shall not affect communication in the residual network.

Table 19 *(continued)*

Param no.	Symbol	min.	typ.	max.	unit	Description
14	$I_{BUS\_NO\_BAT}$	—	—	100	µA	V_{BAT} disconnected $V_{SUP} = V_{GND}$ 0 V < V_{BUS} < 18 V Node shall sustain the current that can flow under this condition. Bus shall remain operational under this condition.
15	V_{Shift_BAT}	0	—	0,1	V_{BAT}	Battery shift $V_{Shift\_BAT} = V_{BATTERY} - V_{Shift\_GND} - V_{BAT}$ $V_{BATTERY}$ = Voltage of vehicle battery
16	V_{Shift_GND}	0	—	0,1	V_{BAT}	GND shift $V_{Shift\_GND} = V_{GND} - V_{GND\_BATTERY}$ V_{GND} = Voltage of GND $V_{GND\_BATTERY}$ = Voltage of vehicle battery GND
17	$V_{Shift\_Difference}$	0	—	0,08	V_{BAT}	Difference between battery shift and GND shift $V_{Shift\_Difference} = \lvert V_{Shift\_BAT} - V_{Shift\_GND} \rvert$
18	V_{BUSdom}	—	—	0,423	V_{SUP}	Receiver dominant state
19	V_{BUSrec}	0,556	—	—	V_{SUP}	Receiver recessive state
20	V_{BUS_CNT}	0,475	0,5	0,525	V_{SUP}	$V_{BUS\_CNT} = (V_{th\_dom} + V_{th\_rec}) / 2$
21	V_{HYS}	—	—	0,133	V_{SUP}	$V_{HYS} = V_{th\_rec} - V_{th\_dom}$

9.4.3 PHY — PMA AC parameters

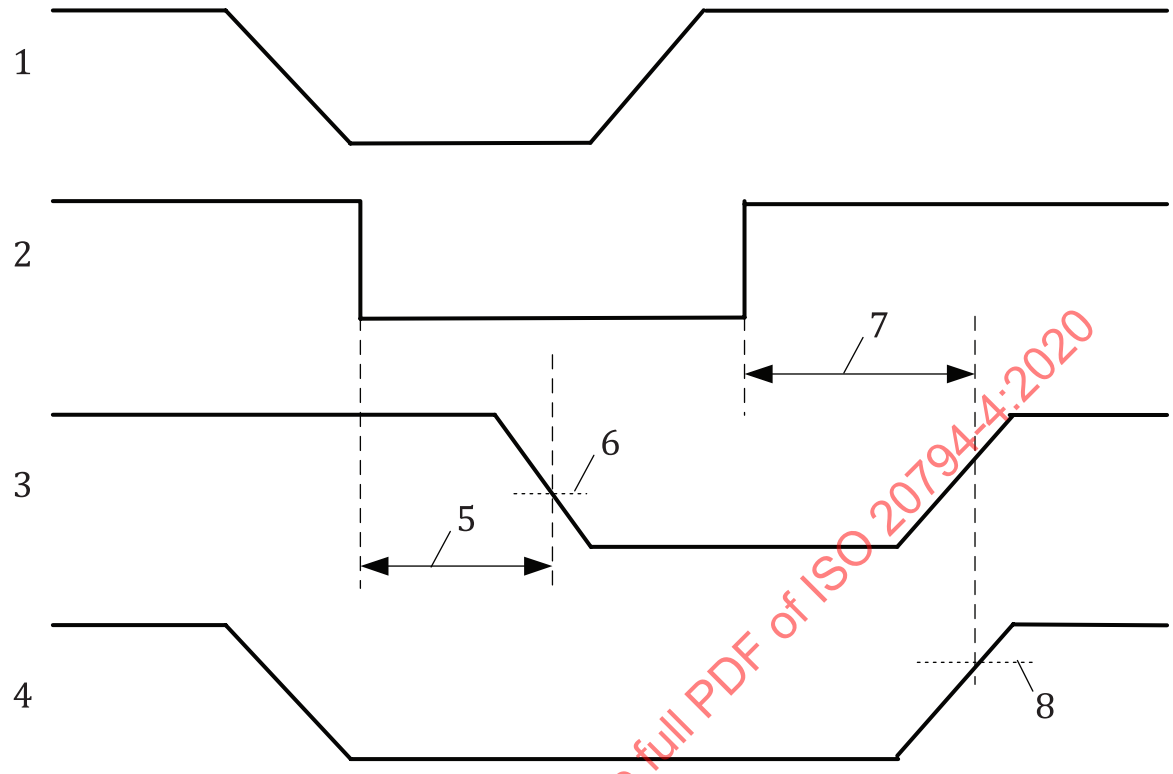
Figure 30 and Figure 31 show the PHY PMA propagation delay of transmitter timing parameters that impact the behaviour of the CXPI network.



Key

1 TX_{PWM} (clock master)
2 CXPI network
3 $t_{tx\_pwm\_pdf\_clk}$
4 TH_{tx_dom}
5 $t_{tx\_pwm\_pdr}$
6 TH_{tx_rec}

Figure 30 — PHY PMA propagation delay of transmitter (clock master)

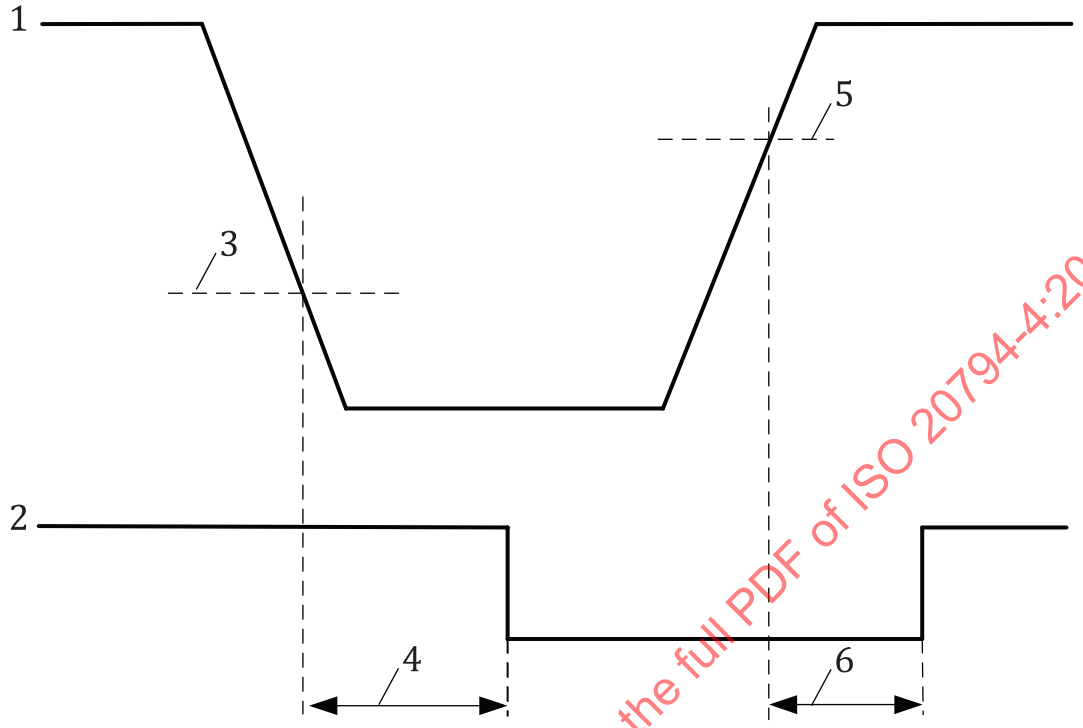


Key

1	CXPI network (output to the CXPI network of clock master)	5	$t_{tx\_pwm\_pdf}$
2	TX_{PWM} (node)	6	TH_{tx_dom}
3	CXPI network (output to the CXPI network of node)	7	$t_{tx\_pwm\_pdr}$
4	CXPI network	8	TH_{tx_rec}

Figure 31 — PHY PMA propagation delay of transmitter (node)

Figure 32 shows the PHY PMA propagation delay of receiver timing parameters that impact the behaviour of the CXPI network.



Key

1 CXPI network
2 RX_{PWM}
3 V_{th_dom}
4 $t_{rx\_pwm\_pdf}$
5 V_{th_rec}
6 $t_{rx\_pwm\_pdr}$

Figure 32 — PHY PMA propagation delay of receiver