INTERNATIONAL STANDARD

ISO 20794-2

First edition
2020-02

Road vehicles — Clock extension peripheral interface (CXPI) —

Part 2: Application layer

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

Partie 2: Couche Application

Reference number
ISO 20794-2:2020(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 this document, 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) as they apply for all layers and the diagnostic services.

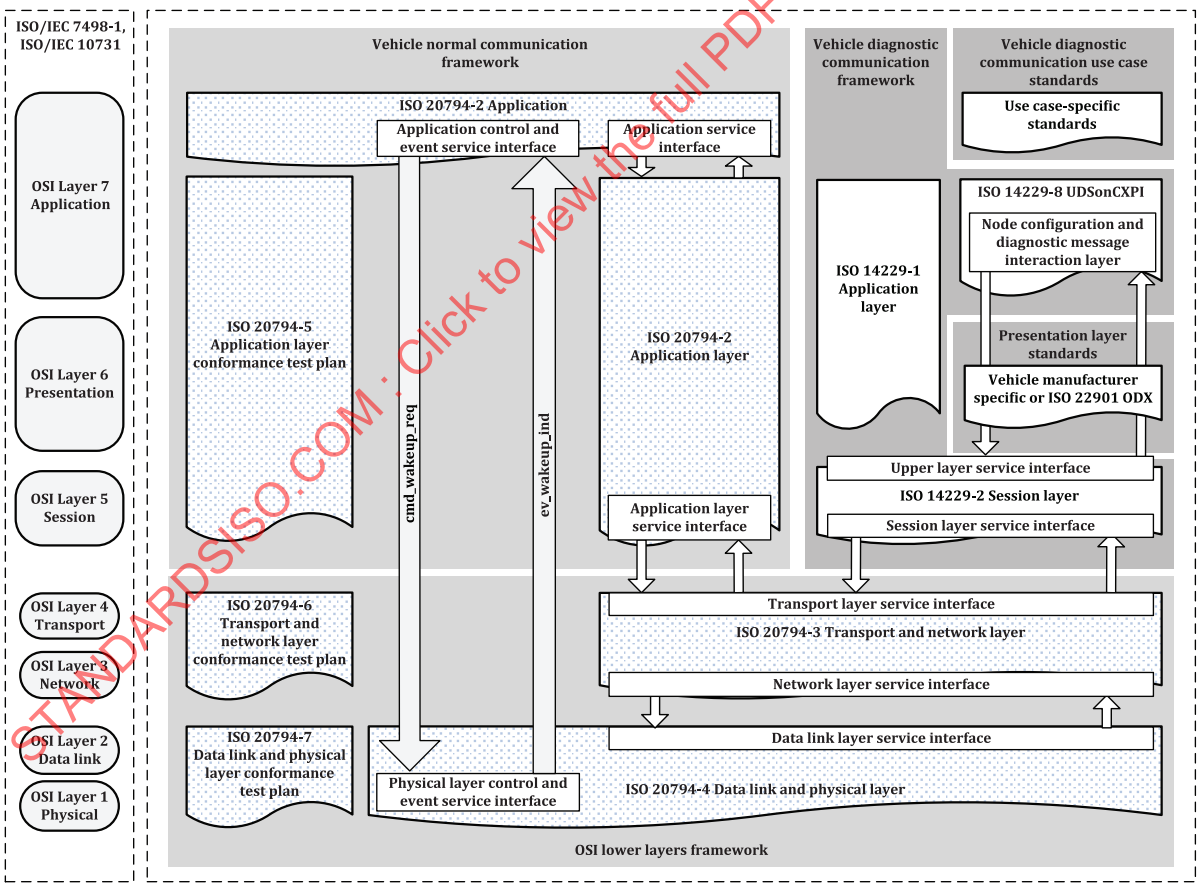


Figure 1 — ISO 20794 documents reference according to OSI model

Road vehicles — Clock extension peripheral interface (CXPI) —

Part 2: Application layer

1 Scope

This document describes the application layer protocol including the application measurement and control data management, message transfer and fault management.

The application and application layer contain the following descriptions:

— message structure;

— communication method;

— network management (optional);

— measurement and control data; and

— error handling.

This document also specifies:

— the service interface; and

— the service interface parameters.

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

ISO 20794-4, *Road vehicles — Clock extension peripheral interface (CXPI) — Part 4: Data link layer and physical layer*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20794-3, ISO 20794-4, 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
clock master
node that transmits *clock* (3.4) to the *lower OSI layers* (3.2)

3.2
lower OSI layer
below the application layer

3.3
master node
node that provides the *schedule* (3.10) master management (including polling method), the *primary clock* (3.6), and optionally the sleep message transmission management

3.4
clock
pulse that synchronises all nodes

3.5
normal state
state which enables transmission and reception of messages

3.6
primary clock
clock (3.4) that is provided by the *master node* (3.3)

3.7
publisher
node providing a message response containing application measurement and/or control data

3.8
request identifier
ReqId
parameter that requests dedicated measurement and/or control data

3.9
request type identifier
ReqTypeId
parameter that enables the polling method for dedicated measurement data and/or control data

3.10
schedule
origin of periodic message transmission

3.11
secondary clock
clock (3.4) that is provided by a dedicated *slave node* (3.13)

3.12
sequence
transmission and reception procedure of messages

3.13
slave node
node other than *master node* (3.3)

3.14
subscriber
master or *slave node* (3.13) that receives the data within a message

3.15
wake-up pulse
stimulus initiated by a node used for wake-up of other nodes

4 Symbols and abbreviated terms

4.1 Symbols

$t_{clock\_start\_m}$	time that the master node requests the clock to the lower OSI layers at the latest
$t_{clock\_stop\_m}$	time that the master node stops to request the clock after master node receives the sleep message notification
$t_{cxpi\_network\_error}$	judgment time of the CXPI network error
t_{sleep_s}	time that each slave node transits to sleep state after the node receives the sleep message notification
t_{wakeup_m}	minimum time that master node starts the request of any request field first for the wake-up sequence
$t_{wakeup\_recovery\_s}$	time that slave node starts the request of the second wake-up pulse after request of the first wake-up pulse
t_{wakeup_s}	maximum time until the slave node wakes up by the wake-up sequence
$t_{wakeup\_schedule\_m}$	maximum time until master node starts the request of any request field (ReqId) or request field (ReqTypeId) first for the wake-up sequence

4.2 Abbreviated terms

AL	application layer
APP	application
CRC	cyclic redundancy check
DLC	data length code
ECU	electronic control unit
LSB	least significant bit
MSB	most significant bit
Mtype	message type
NMInfo	network management information
NormalCom	normal communication
OSI	open systems interconnection
param	parameter
PDU	protocol data unit
ReqId	request identifier

ReqTypeId	request type identifier
SCT	sequence count
SI	service interface
SIP	service interface parameter

5 Conventions

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

6 Introduction to application and application layer

6.1 Application properties

The application has the following properties:

— communication methods;

— message types;

— network management (optional wake-up, sleep);

— state management (state machine);

— measurement and/or control data; and

— error handling.

6.2 Application layer properties

The application layer has the following properties:

— message exchange;

— message structure; and

— service interface and parameters.

6.3 Message transmission

A message consists of a request field and a response field. The node that corresponds to a request identifier composed in the request field can transmit the response field.

6.4 Communication methods

Two communication methods are supported:

— Event-triggered method

Each node can request a request protected identifier field and response field based on an internal event occurrence.

— Polling method

The master node requests a request protected type identifier field (event request in polling method) to the lower OSI layers and then each node can transmit a request protected identifier field. The node corresponding to the request protected identifier field transmits the response field.

6.5 Message types

The application layer supports the following message types:

— request message field;

— response message field; and

— request sleep message field.

6.6 Error handling

The error handling is based on the following measures:

— CXPI network errors; and

— sequence count error (optional).

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 application and application layer services.

7.2 SIP — Data type definitions

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

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 — `Unsigned Word` = 16-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 — Mtype, message type

This requirement specifies the message type parameter values of the CXPI service interface.

REQ	0.2 SIP — Mtype, message type
The `Mtype` parameter shall be of data type `Enum` and shall be used to identify the message 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 values 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}] (fixed value)	

7.6 SIP — PDU, protocol data unit

This requirement specifies the protocol data unit parameter values 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 packet data (PDU) content of the request or response packet to be transmitted/received. Range: [00_{16} to FF_{16}]	

7.7 SIP — Length, length of PDU

This requirement specifies the length of PDU 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 requested transmission/notified reception. If Mtype = NormalCom then 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 values 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 event wake-up indication information. Table 1 describes the network management 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 interface parameter value indicates the reception of the wake-up pulse event from the lower OSI layers. This parameter is optional if `cmd_wakeup_pulse_on` in Table 2 is (optional).
`ev_dominant_pulse_detect`	This service interface parameter value indicates the reception of the dominant pulse event from the lower OSI layers. This parameter is optional if `cmd_wakeup_pulse_on` in Table 2 is (optional).
`ev_clk_detect`	This service interface parameter value indicates the reception of the clock detection event from the lower OSI layers.
`ev_clk_loss`	This service interface parameter value indicates the reception of the clock loss event from the lower OSI layers.

7.9 SIP — cmd_wakeup_req, command wake-up request

This requirement specifies the command wake-up request parameter values 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 specifies 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 interface parameter value commands the clock generator to turn on the clock transmission to the lower OSI layers.
`cmd_clk_generator_off`	This service interface parameter value commands the clock generator to turn off the clock transmission to the lower OSI layers.
`cmd_wakeup_pulse_on (optional)`	This service interface parameter value commands the transmission of the wake-up pulse to the lower OSI layers.

7.10 SIP — NMInfo, network management information

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

REQ	0.9 SIP — NMInfo, network management information
The `NMInfo` parameter shall be of data type `2-bit Bit String` and shall contain the NetMngt information in the response field. 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 values of the CXPI service interface.

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

7.12 SIP — Result, result

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

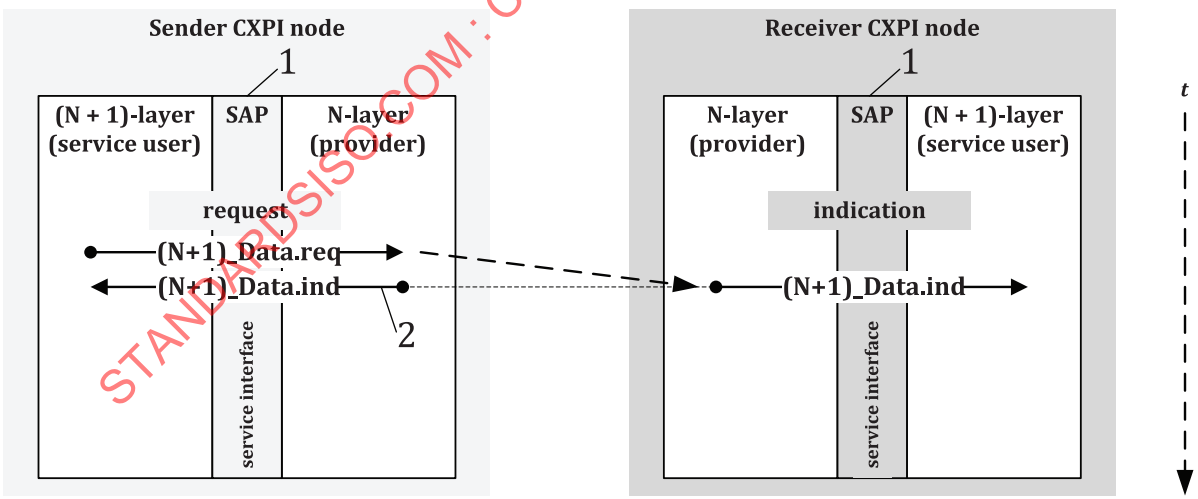
REQ	0.11 SIP — Result, result
The `Result` parameter shall be of data type `16-bit Bit String` and shall contain the status relating to the outcome of a service execution. If two or more errors are discovered at the same time, then the transport or network layer entity shall set the appropriate error bit in the `Result` parameter. Range: [`OK, DLL_Arb_Lost, Err_DLL_Byte, Err_DLL_CRC, Err_DLL_DLC, Err_DLL_DLCext, Err_DLL_Parity, Err_DLL_Framing, Err_NL_TIMEOUT_A, Err_TL_Ptype, Err_TL_PCI_DL_Value, Err_APP_SCT`] Range: [0_2 to 1_2] (1-bit per result)	

8 SI — Service interface (SI) definition to application and lower OSI layers

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

The service interface defines the service and parameter mapping to the application and the lower OSI layers.

Figure 2 shows the application A_Data.req and A_Data.ind service interface.



Key

1 service access point

2 read back from CXPI network provided by lower OSI layer

Figure 2 — A_Data.req and A_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 (A_Data.req).

All nodes receive the request protected identifier field of type NormalCom (A_Data.ind). The node, which has corresponding PDU data transmits the response PDU (A_Data.req) and all nodes receive the response PDU (A_Data.ind).

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

This requirement specifies the application service interface parameter mapping between application and application layer and to the lower OSI layers.

REQ	0.12 SI — A_Data.req and A_Data.ind service interface parameter mapping between application and application layer and to the lower OSI layers
The A_Data.req and A_Data.ind service interface parameter mapping is specified in Table 3.	

The normal communication (NormalCom) message consists of an A_ReqId (request field) or an A_PDU (response field). The request field has no A_PDU and therefore the A_Length = NULL. The response field has an A_PDU and therefore the A_Length > 0.

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

Application (service user)	Application layer (service provider)	A_Data.req and A_Data.ind parameter validity NormalCom with A_Length			
		= NULL		≥'0'	
		.req	.ind	.req	.ind
Mtype	A_Mtype	X[a]	X[a]	X[a]	X[a]
Length	A_Length	—[b]	—[b]	X[a]	X[a]
ReqId	A_ReqId	X[a]	X[a]	X[a]	X[a]
ReqTypeId[b]	A_ReqTypeId[b]	X[a]	X[a]	—[b]	—[b]
PDU	A_PDU	—[b]	—[b]	X[a]	X[a]
NMInfo	A_NMInfo	—[b]	—[b]	X[a]	X[a]
SCT	A_SCT	—[b]	—[b]	X[a]	X[a]
Result	A_Result	—[b]	X[a]	—[b]	X[a]

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

[a] Supported "X".

[b] Not supported "—".

9 Application (APP)

9.1 APP — Message exchange

A message consists of a request protected type identifier field or a request protected identifier field and a response field (PDU). The request protected identifier field is transmitted by a node. The node that corresponds to the request protected identifier field value can request the response field (PDU) to the lower OSI layers.

9.2 APP — Communication methods

9.2.1 APP — General

The CXPI communication protocol supports two communication methods.

9.2.2 APP — Event-triggered method

Each node can transmit a request protected identifier field (message request) in case it wants to transmit a corresponding response field (response message).

REQ	8.1 APP — Communication methods — Event-triggered method
If all nodes' application software supports the event-triggered method, it shall immediately request a request protected identifier field to the lower OSI layers. The node corresponding to the request protected identifier field shall request the response field (PDU) including application measurement and/or control data.	

Figure 3 shows a sequence example of the event-triggered method. This method is asynchronous with the master node. The node '1' requests the request PID 'B', then the node '2' requests the response message PDU 'B', which corresponds to the request PID 'B'. Then an event occurs in the node '3'. The node '3' requests request PID 'C' and then node '3' requests the response PDU 'C', which corresponds to request PID 'C'.

Then an event occurs in the node '2' application. Node '2' requests the request (PID) 'B'. The node '2' requests request PID 'B' and then node '2' requests the response PDU 'B', which corresponds to request PID 'B'.

The node '1' requests the request PID 'B' periodically, then the node '2' requests the response PDU 'B', which corresponds to request PID 'B'. The slave nodes may request PIDs periodically too.

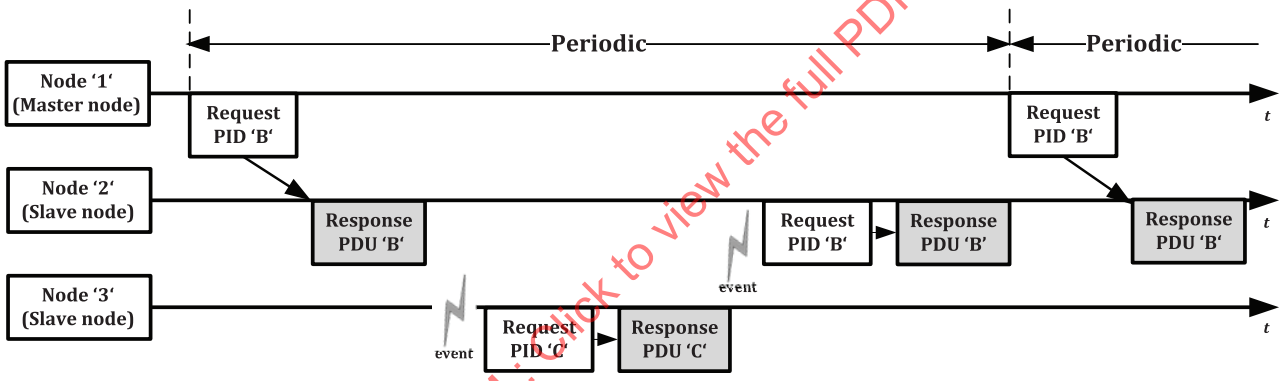


Figure 3 — Message sequence of event-triggered method

9.2.3 APP — Polling method

There are two kinds of requests that the master node requests about the request protected identifier field and the request protected type identifier field. When the master node requests the request protected type identifier field (event request in polling method) to the lower OSI layers, each node can request the request protected identifier field. If multiple nodes request the request protected identifier field at the same time and the request protected identifier fields collide on CXPI network, a non-destructive arbitration takes place in the data link layer and the higher priority request protected identifier field is transmitted to the CXPI network. The node corresponding to the request protected identifier field requests the response field (PDU). The slave node only requests PDUs when it receives the request protection identifier field or request protected type identifier field from the master node. The slave node cannot request PDUs voluntarily.

Although the polling method lowers the responsivity of event compared with the event-triggered method, the polling method can be selected to guarantee communicative periodicity.

REQ	8.2 APP — Communication methods — Polling method
If the master node application is in polling method, then the master node shall periodically request a request protected type identifier field (event request in polling method) or a request protected identifier field, or both, to the lower OSI layers. The node corresponding to the request protected identifier field shall request the response field (A_PDU) including application measurement and/or control data.	

Figure 4 shows an example of the polling method sequence. When the master node '1' requests the periodic request protected identifier field PID 'B' to the lower OSI layers, each node can request the response field. In this example, node '2' requests the response field PDU 'B'.

Then an event occurs in node '3'. The master node '1' requests a periodic request protected type identifier field (PTYPE) to the lower OSI layers, which allows all nodes to request a request protected identifier field. In this example, node '3' requests a request protected identifier field PID 'C' followed by a response field PDU 'C'.

The master node '1' requests the periodic request protected identifier field PID 'C' to the lower OSI layers and node '3' requests the response field PDU 'C'.

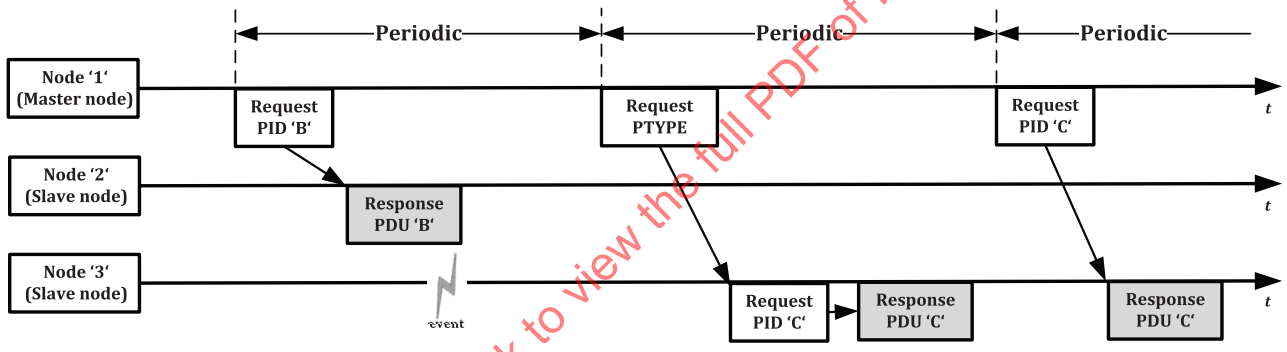


Figure 4 — Message sequence of polling method

9.3 APP — Network management (NM)

9.3.1 APP — General

The network management describes the wake-up/sleep function, which allows a node to transit to a sleep state. The wake-up/sleep function is an optional feature. Implementation of the wake-up/sleep function is selected for each node.

The master node checks the response field (PDU) with NMInfo = sleep permission of all slave nodes. If all slave nodes confirm sleep permission, the master node requests a normal communication (NormalCom) message with a Sleep PDU.

Master node trigger:

— The master node with an internal event, which requires data to be requested, transmits the synchronisation clock pulse to the lower OSI layers.

Slave node trigger:

— A slave node with an internal event, which requires data to be requested, transmits a wake-up pulse to the lower OSI layers.

— A slave node receives the synchronisation clock from the clock master, which enables the application to transmit the data to the lower OSI layers.

9.3.2 APP — Normal, standby, and sleep states

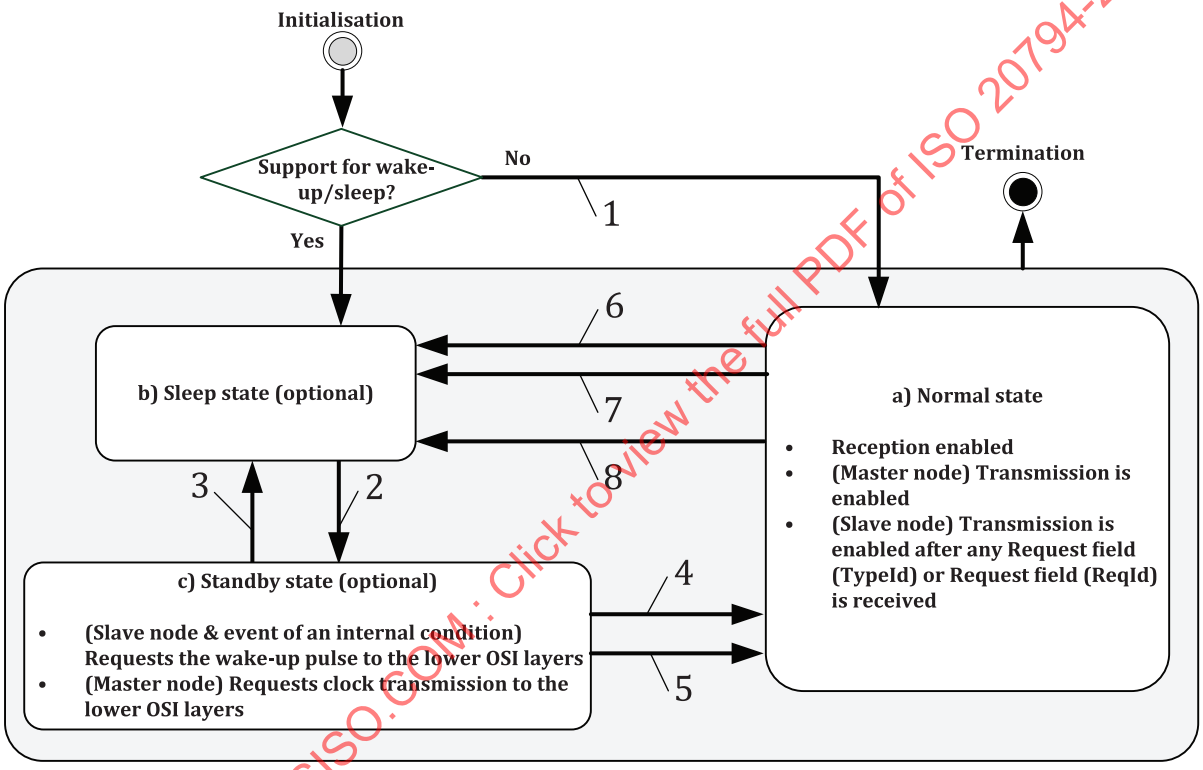
This state manages the state of wake-up/sleep function for each node as shown in Figure 5. The state machine supports the following three states:

a) Normal state (mandatory);

b) Standby state (optional); and

c) Sleep state (optional).

REQ	8.3 APP — NM — Normal, standby, and sleep states
The state machine shall comply to the states and state transits as specified in Figure 5.	



Key

1 master node: requests clock transmission
2 wake-up condition occurred (optional)
3 CXPI network error and sleep permission (optional)
4 master node requests clock transmission (optional)
5 slave node notification of the clock (optional)
6 CXPI network error and sleep permission (optional)
7 master node: when sleep condition is satisfied (optional)
8 slave node: when preset time elapses after sleep message is received (optional)

Figure 5 — NM state machine

9.3.3 APP — Normal state

9.3.3.1 APP — General

In normal state, the clock is generated by the master node and received by the slave nodes. The normal state enables transmission and reception of messages.

9.3.3.2 APP — Event-triggered method in normal state

This request specifies the normal state of an event-triggered method.

REQ	8.4 APP — NM — Event-triggered method in normal state — Wake-up/sleep not supported
After power ON the node shall transit into normal state if the wake-up/sleep and standby state features are not supported.	

REQ	8.5 APP — NM — Event-triggered method in normal state — Master node starts message transmission
The master node shall begin message transmission after it enters the normal state.	

REQ	8.6 APP — NM — Event-triggered method in normal state — Slave node starts message transmission
The slave node shall transmit a response field after it receives a supported request field (ReqId). Table 4 specifies the condition of a slave node to enable message transmission.	

Table 4 — Condition that transmission of message of slave nodes is possible

Condition	Executable processing
When it receives any request field (ReqId) from the lower OSI layers, and there is no error.	Transmission of response field that corresponds to request field (ReqId).
When it wants to transmit request field (ReqId).	Transmission of request field (ReqId).

9.3.3.3 APP — Polling method in normal state

This request specifies the polling method in normal state.

REQ	8.7 APP — NM — Polling method in normal state
The application shall transit to normal state, if power-on and wake-up/sleep and standby are not supported and shall begin the message transmission.	

REQ	8.8 APP — NM — Polling method in normal state — Master node starts message transmission
The master node shall transmit either any request identifier field (ReqId) or the request type identifier field (ReqTypeId) at least once after transit to the normal state.	

REQ	8.9 APP — NM — Polling method in normal state –Slave node starts message transmission
The slave node shall not begin message transmission as long as it has not received either any identifier field (ReqId) or the request type identifier that the master node transmitted after transit to the normal state.	

Table 5 specifies the condition, which enables the transmission of a message of a node.

Table 5 — Condition when the transmission of message of slave nodes becomes possible

Condition	Executable processing
When it receives any request field (ReqId) from the lower OSI layers, and there is no error.	Allow the transmission of response field that corresponds to request field (ReqId).

Table 5 *(continued)*

Condition	Executable processing
When it receives any request field (ReqTypeId) from the lower OSI layers, and there is no error.	Allow the transmission of request field (ReqId) after the request field (ReqTypeId) of the master node has been transmitted.

9.3.4 APP — Sleep state (optional)

The sleep state denotes the state, in which a node stops transmitting and receiving of messages.

REQ	8.10 APP — NM — Sleep state (optional)
A node, that is supporting wake-up/sleep function shall transit to sleep state upon initialisation as specified in Figure 5. In addition, it shall also transit to sleep state after the sleep processing is executed from the normal state, and transit from standby state or normal state due to the CXPI network error. During the sleep state, when each node receives the wake-up condition, it shall transit to the standby state.	

REQ	8.11 APP — NM — Wake-up/sleep supported
After power ON the node shall transit into sleep state if the wake-up/sleep feature is supported.	

9.3.5 APP — Standby state (optional)

The standby state denotes the state, where the node which transmits the wake-up pulse also transmits the clock. The standby state is not allowed message transmission and reception.

REQ	8.12 APP — NM — Standby state (optional)
The standby state shall transit only from the sleep state. The master node shall transit to the normal state if the clock transmission to the lower OSI layers becomes active. If a slave node is in standby state and generates the event of an internal factor, it requests a wake-up pulse transmission to the lower OSI layer, and after the clock transmission from a master node is notified, it transits to normal mode. If the clock is not notified, the slave node shall request the wake-up pulse transmission again.	

If a slave node is in standby mode and does not generate the event of an internal factor, it transits to normal mode, after the clock transmission from a master node is notified.

REQ	8.13 APP — NM — Wake-up/sleep supported
When supporting wake-up/sleep, the normal state shall transit only from standby state. After the node transits to the normal state, it shall begin the message transmission. The slave node shall transmit a message after the slave node received a request field (ReqId) that the master node transmits.	

9.3.6 APP — Wake-up/sleep function (optional)

9.3.6.1 APP — General

The wake-up condition is determined per CXPI network and consists of an internal condition (e.g. detecting that the ignition is turned on) and an external condition, which is notified by the wake-up pulse reception from the lower OSI layers.

If the wake-up/sleep function is implemented, it has the following five operations:

a) wake-up function of master node trigger;

b) wake-up function of slave node trigger;

c) re-request of wake-up pulse transmission;

d) master node sleep function request; and

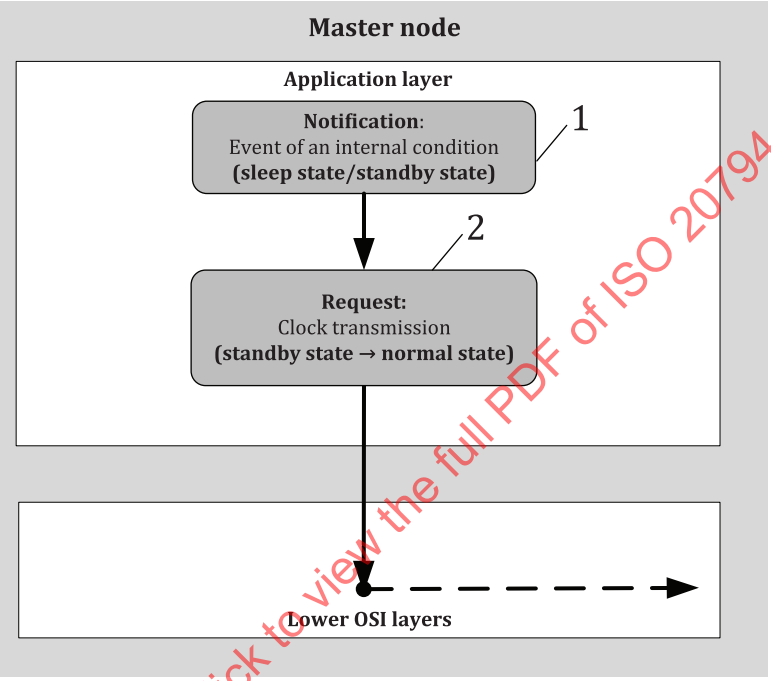
e) multi clock master sequence processing.

REQ	8.14 APP — NM — Wake-up/sleep function(optional)
After power ON the node shall transit into sleep state if the wake-up/sleep feature is supported.	

9.3.6.2 APP — Master node wake-up function

9.3.6.2.1 APP — Master node wake-up request/notification

The master node transmits a clock request onto the CXPI network based on an internal event to wake-up sleeping CXPI cluster.



Key

1 master node transits from sleep state into standby state due to an internal event

2 master node transmits a clock to the lower OSI layers and transits from standby state into normal state

Figure 6 — Master node generated wake-up request sequence

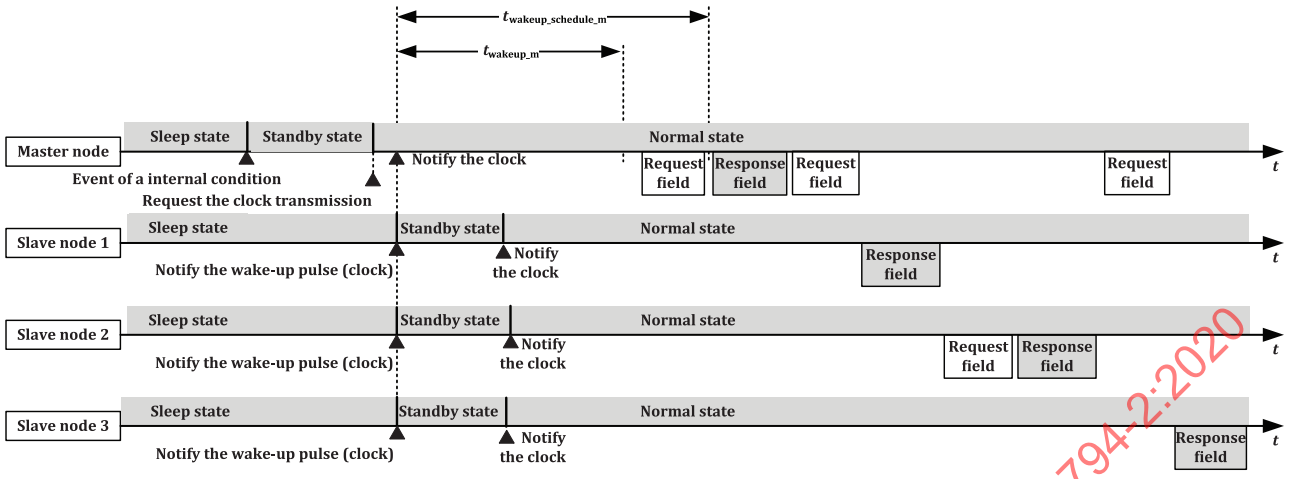
REQ	8.15 APP — NM — Master node wake-up request/notification — Internal event notification
If an internal event occurs in the master node (see Figure 6), the master node shall transit from sleep state into standby state.	

REQ	8.16 APP — NM — Master node wake-up request/notification — Transit from standby state into normal state
After the master node requests the clock transmission to the lower OSI layers to wake-up other nodes, the master node transits from standby state to normal state.	

9.3.6.2.2 APP — Master node wake-up sequence

This subclause describes the processing of wake-up according to an internal condition of the master node (see Figure 7).

REQ	8.17 APP — NM — Master node wake-up request/notification — Master node wake-up sequence
The master node shall start transmitting the request field (ReqId) or (ReqTypeId) within $t_{wakeup\_schedule\_m}$ time after t_{wakeup_m} time.	



Key

sending request field

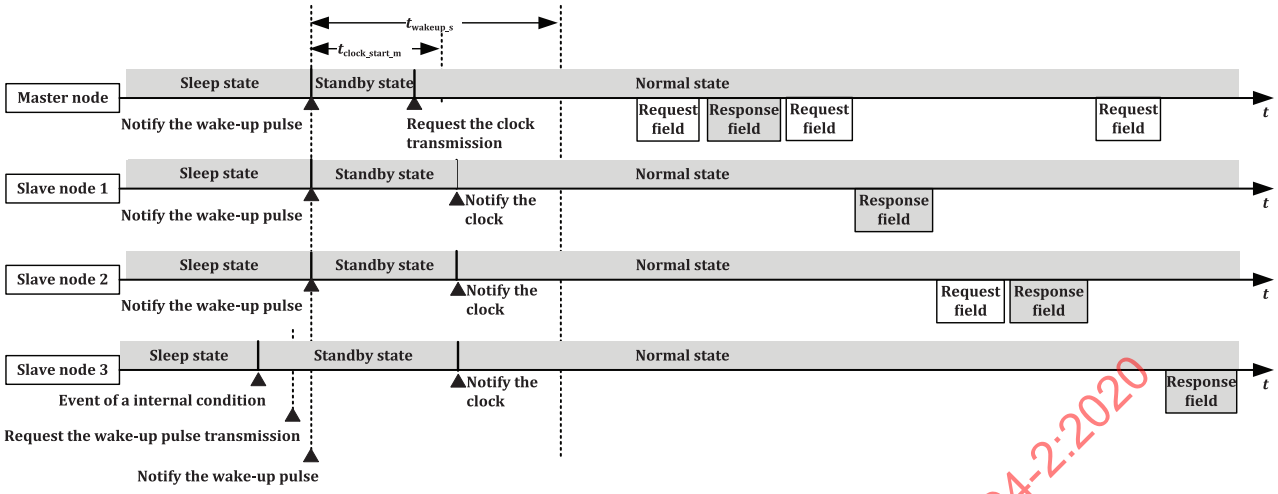
response by node corresponding to request field

state of node

Figure 7 — Wake-up sequence of master node trigger

This requirement describes the processing of wake-up according to an internal condition of the slave node (see Figure 8).

REQ	**8.18 APP — NM — Master node wake-up request/notification — Master node wake-up sequence — $t_{clock\_start\_m}$ time**
If the master node is in sleep state and receives a wake-up pulse notification from the lower OSI layers, it shall transit from sleep state into standby state and requests to transmit the synchronisation clock to the lower OSI layers within $t_{clock\_start\_m}$ time (see Figure 8).	



Key

- sending request field
- response by node corresponding to request field
- state of node

Figure 8 — Master node clock transmission timing

9.3.6.2.3 Master node sleep request/notification

REQ	8.19 APP — NM — Master node wake-up request/notification — Master node sleep request/notification
The sleep message payload shall have a length of 8 byte. The sleep message is specified in Table 6.	

At the time of the normal mode, the sleep message is used, in order that the master node makes each slave node prepare the sleep state. Table 6 shows the message configuration. The sleep message has a fixed length and a fixed value for ReqId and a fixed A_PDU.

Table 6 — Sleep message

Sleep ReqId	Sleep A_PDU
1 byte	8 byte
$1F_{16}$ (fixed)	00_{16} FF_{16} FF_{16} FF_{16} FF_{16} FF_{16} FF_{16} FF_{16} (fixed)

This requirement specifies the sleep conditions.

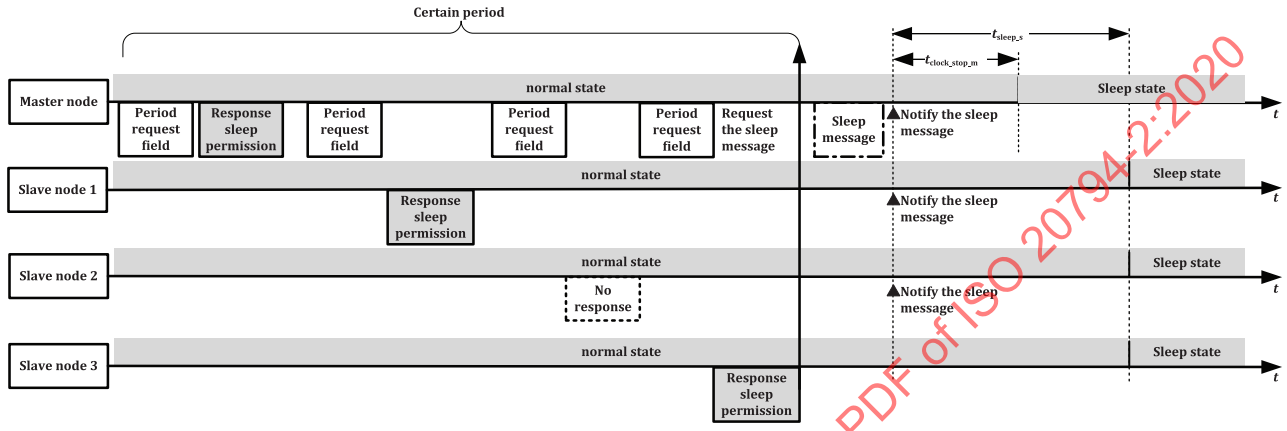
REQ	8.20 APP — NM — Wake-up/sleep function (optional) — Master node sleep function — Sleep conditions
If the sleep condition specified in Table 7 is fulfilled, the master node shall recognise that the condition is met for sleep message transmission.	

A slave node transits to sleep state after reception of the sleep message (see Figure 9).

REQ	8.21 APP — NM — Wake-up/sleep function (optional) — Condition of receive sleep message reception
The receiver shall ignore the data bytes after the 2nd data byte in the data field.	

Table 7 — Condition of transmission of sleep message by master node

Condition	Description
Transmission of sleep message	When the master node receives a response field notification from each slave node with sleep permission, which is indicated by setting the service interface parameter "NMInfo = sleep_ind permission" and the master node does not transmit a sleep prohibition (sleep_ind = 0) request and does not receive the message of sleep prohibition (sleep_ind = 0) notification from any node for a certain period.



Key

periodic sending of request field

response by node corresponding to Request field, including sleep permission signal in frame

sleep message sent by master

state of node

Figure 9 — Sleep sequence

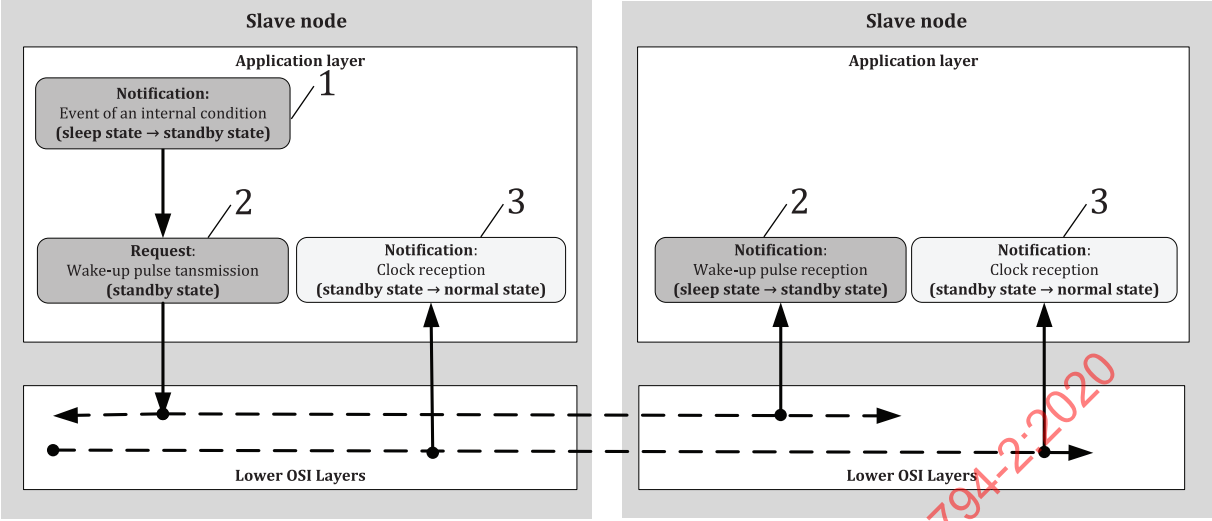
REQ	**8.22 APP — NM — Wake-up/sleep function (optional) — Master node sleep function — $t_{clock\_stop\_m}$ time**
After receiving the sleep message (read back from CXPI network) notification, the master node shall stop the clock request and enter sleep state within $t_{clock\_stop\_m}$ time.	

REQ	**8.23 APP — NM — Wake-up/sleep function (optional) — Master node sleep function — $t_{clock\_stop\_m}$ time**
The master node shall stop the clock within $t_{clock\_stop\_m}$ time after detecting an error.	

9.3.6.3 APP — Slave node wake-up function

9.3.6.3.1 APP — Slave node wake-up request/notification

Slave node receiving the event of an internal condition in a sleeping CXPI cluster may request a wake-up.



Key

1 slave node 1 transits from sleep state into standby state due to an internal event

2 slave node 1 transmits wake-up pulse request to the lower OSI layers
slave node 2 receives wake-up pulse notification and transits from sleep state into standby state

3 all slave nodes receive the synchronisation clock notification and transit from standby state into normal state

Figure 10 — Slave node generated wake-up request sequence

REQ	8.24 APP — NM — Slave node wake-up request/notification — Internal event notification
If an internal event occurs in the slave node (see Figure 10), the slave node transmits a wake-up pulse to the lower OSI layers in order to wake-up other nodes.	

REQ	8.25 APP — NM — Slave node wake-up request/notification — Wake-up sequence by dominant pulse and clock notification — Transit from standby state into normal state
If a slave node recognises a clock, it shall transit from standby state into normal state.	

9.3.6.3.2 APP — Slave node wake-up sequence

9.3.6.3.2.1 APP — General

This subclause describes the processing of wake-up according to an internal condition of the slave node. An internal event triggers the request for wake-up pulse in the slave node (see 9.3.6.3.1). In case the slave node is in sleep state and a dominant pulse is notified from the lower OSI layers caused by noise and no clock received, the wake-up sequence as specified in 9.3.6.2.3 and 9.3.6.4.2.4 is also permitted to simplify the electrical circuit.

9.3.6.3.2.2 APP — Wake-up sequence by wake-up pulse notification

This requirement specifies the wake-up sequence of a wake-up pulse notification.

REQ	8.26 APP — NM — Slave node wake-up sequence — Wake-up sequence by wake-up pulse notification — t_{wakeup_s} time
If a slave node receives the wake-up pulse notification, it shall transit into standby state within t_{wakeup_s} time (see Figure 11).	

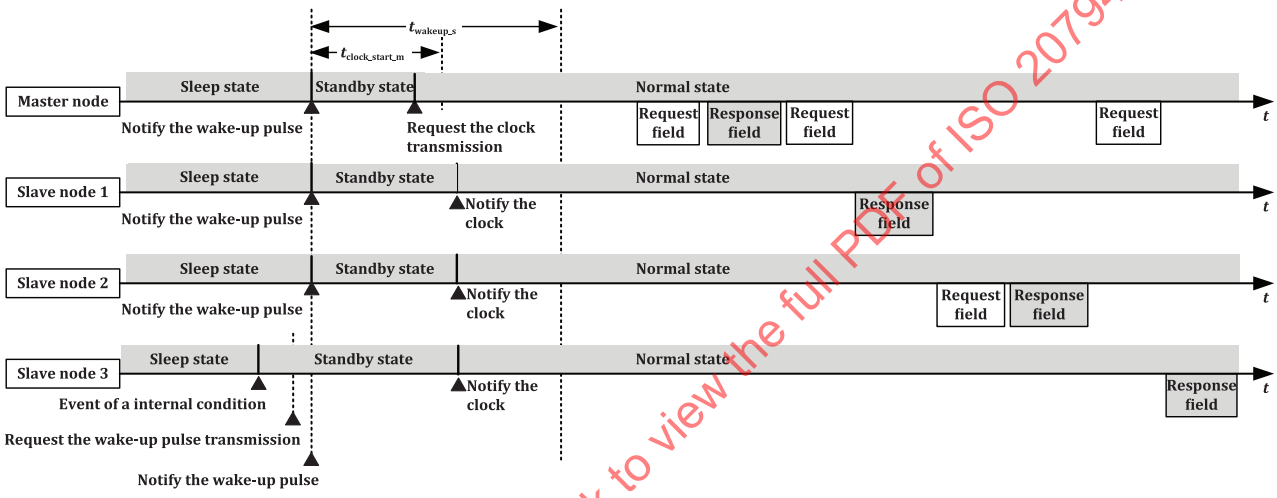
The wake-up pulse includes the clock requested by the master node (see ISO 20794-4).

REQ	8.27 APP — NM — Slave node wake-up sequence — Wake-up sequence by wake-up pulse notification — Transit to normal state
If a slave node receives the clock notification, it shall transit to normal state (see Figure 11).	

The master node starts transmitting any request field (ReqId) or request field (ReqTypeId) within $t_{wakeup\_schedule\_m}$ time after t_{wakeup_m} time.

REQ	8.28 APP — NM — Slave node wake-up sequence — Wake-up sequence by wake-up pulse notification — Response field (PDU) transmission
A slave node shall transmit a corresponding response field (PDU) request to the lower OSI layers after reception of a corresponding request field (ReqTypeId or ReqId) notification (see Figure 11).	

Figure 11 shows the wake-up sequence of slave node trigger.



Key

- sending request field
- response by node corresponding to request field
- state of node

Figure 11 — Wake-up sequence of slave node trigger

9.3.6.3.2.3 APP — Wake-up sequence by dominant pulse and clock notification

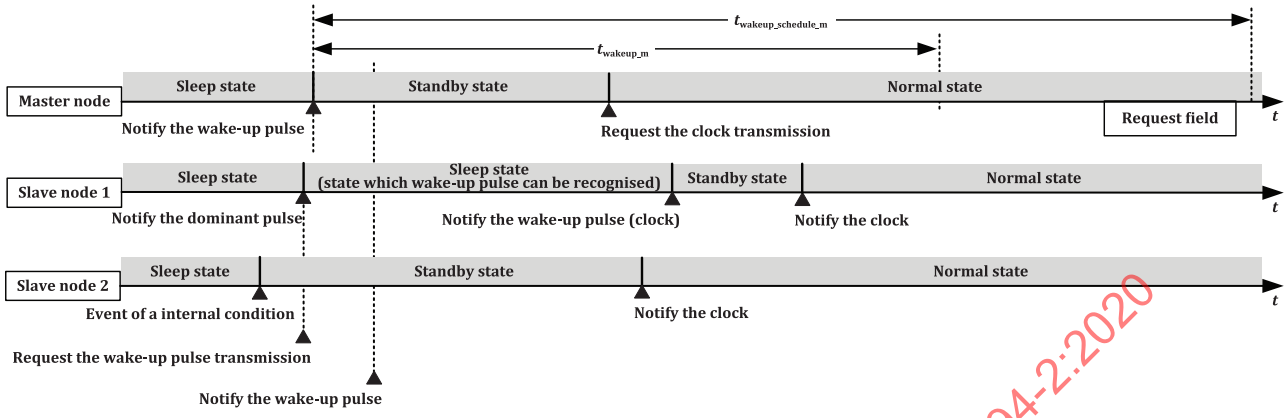
The slave node transmits a wake-up pulse request, but the other slave nodes recognize it as a dominant pulse. This requirement specifies the wake-up sequence of dominant pulse notification.

REQ	8.29 APP — NM — Slave node wake-up sequence — Wake-up sequence by dominant pulse and clock notification — Sleep state
After receiving the dominant pulse notification, the slave node shall be capable of recognising the wake-up pulse. Figure 12 specifies the wake-up sequence.	

After the master node transmits a clock request, the slave nodes, that do not transmit the wake-up pulse notification, receive the wake-up pulse notification. The slave nodes transit from sleep state into standby state. If the slave nodes recognise a clock, it shall transit from standby state into normal state.

The master node starts transmitting any request field (ReqId) or request field (ReqTypeId) within $t_{wakeup\_schedule\_m}$ time after t_{wakeup_m} time. A slave node transmits a corresponding response field

(PDU) to the lower OSI layers after reception of a corresponding request field (ReqId) (see REQ 8.24 and Figure 11).



Key

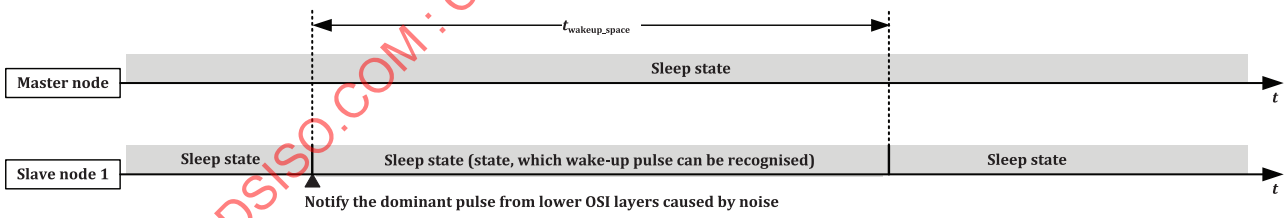
sending request field

state of node

Figure 12 — Wake-up sequence by dominant pulse and clock notification

9.3.6.3.2.4 APP — Sleep sequence of no clock after dominant pulse received

REQ	8.30 APP — NM — Slave node wake-up sequence — Sleep sequence of no clock after dominant pulse received
After the slave node receives the dominant pulse notification and $t_{wakeup\_space\_s}$ time exceeds, the slave node shall transit to sleep state. Figure 13 specifies the sleep sequence.	



Key

state of node

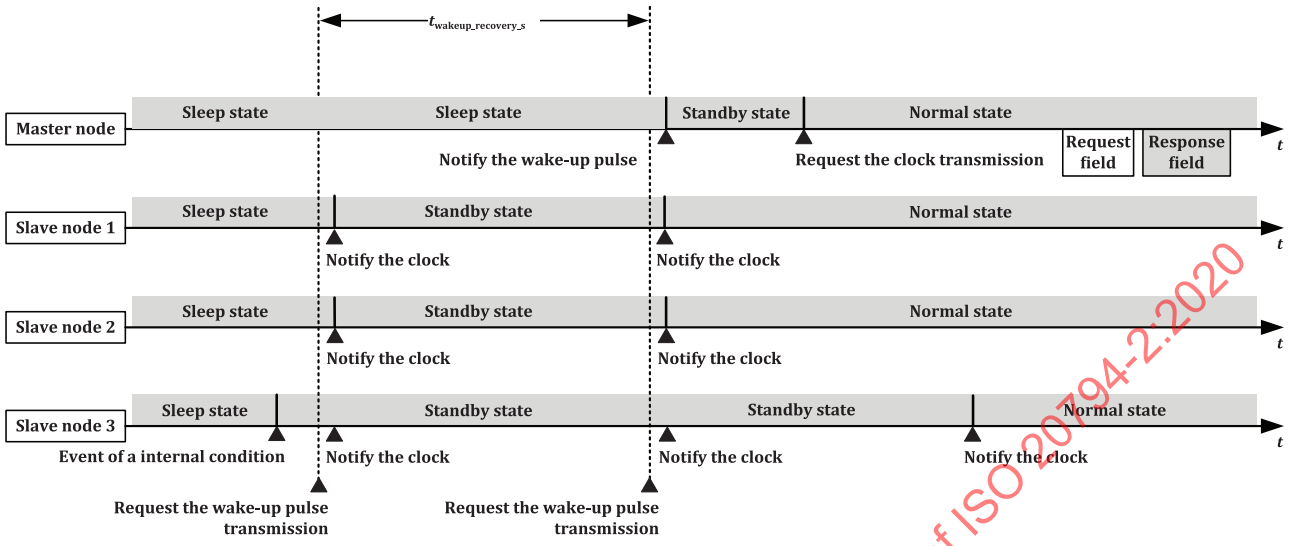
Figure 13 — Sleep sequence of no clock after dominant pulse received

9.3.6.4 APP — Slave node re-request of wake-up pulse

If a slave node is in sleep state and an internal event requires measurement and/or control data transmission, the slave node requests a wake-up pulse to the lower OSI layers.

REQ	8.31 APP — NM — Slave node re-request of wake-up pulse
If a slave node transmits the wake-up pulse to the lower OSI layers and does not receive a clock from the lower OSI layers after the $t_{wakeup\_recovery\_s}$ time, it shall transmit a 2nd wake-up pulse to the lower OSI layers (see Figure 14).	

When the clock request is not transmitted by the master node (2nd wake-up pulse already retransmitted), the wake-up sequence can be executed again after sleep state by the CXPI network error processing.



Key

sending request field

response by node corresponding to request field

state of node

Figure 14 — Re-request sequence of wake-up pulse transmission

9.3.6.5 APP — Slave node sleep function

This requirement specifies the slave node sleep message. Figure 15 shows the message reception of a sleep notification to the application in the slave node.

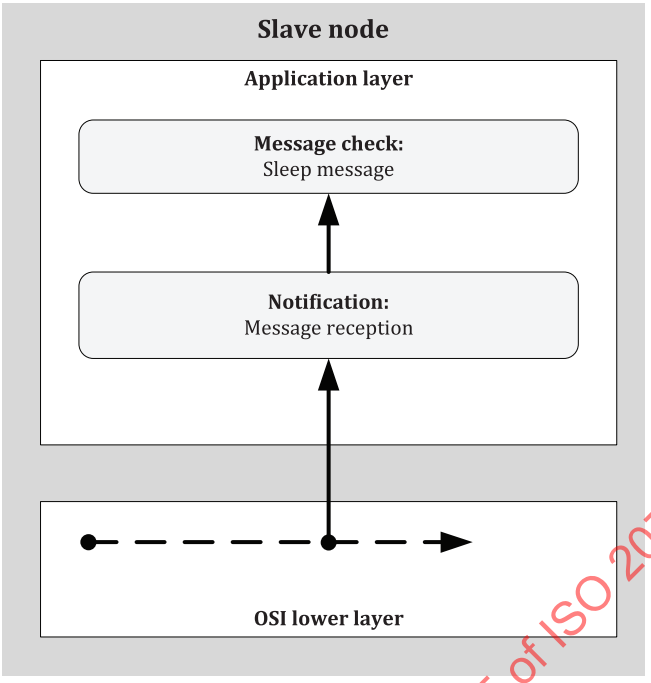
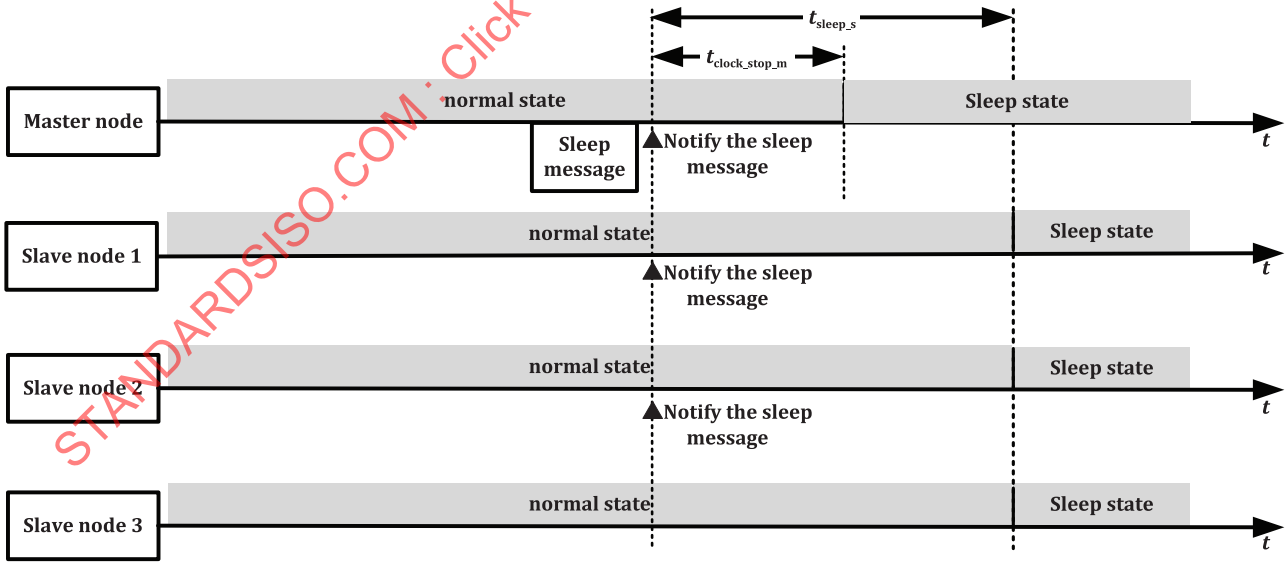


Figure 15 — Slave node sleep message

REQ	8.32 APP — NM — Slave node sleep function — Reception of sleep message
If the slave node receives the sleep message notification from the lower OSI layers it shall transit into sleep state within t_{sleep_s} time (see Figure 16 and Table 8). The sleep message is specified in Table 6.	



Key

sleep message sent by master

state of node

Figure 16 — Slave node receives sleep message

9.3.7 APP — Wake-up/sleep sequence parameter

This requirement specifies the wake-up/sleep sequence parameter.

REQ	8.33 APP — NM — Wake-up/sleep sequence parameter
Table 8 specifies the timing parameters of the wake-up and sleep sequence (see 9.3.6.2, 9.3.6.3, 9.3.6.4, 9.3.6.5, and 9.4).	

Table 8 — Wake-up/sleep sequence parameter

Param no.	Signal name	Starting point	Ending point		unit
			minimum	maximum	
1	t_{wakeup_s}[b]	Wake-up pulse notification[a]	—	50	ms
2	t_{wakeup_m}	Wake-up pulse notification[a]	70	—	ms
3	$t_{wakeup\_schedule\_m}$	Wake-up pulse notification[a]	—	100	ms
4	$t_{clock\_start\_m}$[b]	Wake-up pulse notification[a]	—	50	ms
5	$t_{wakeup\_space\_s}$[c]	Dominant pulse notification	70	170	ms
6	$t_{wakeup\_recovery\_s}$[c]	Wake-up pulse requestment	60	250	ms
7	$t_{clock\_stop\_m}$	Sleep message notification[d]	0	30	t_{bit}
8	t_{sleep_s}	Sleep message notification	25	50	ms
9	t_{check_clock}[e]	Transit time from sleep state (system supports wake-up/sleep) or normal state time (system does not support wake-up/sleep)	2 × $t_{wakeup\_recovery\_s}$	$t_{cxpi\_network\_error}$	—
10	$t_{cxpi\_network\_error}$	Refer to 9.6.2	—	—	

NOTE The signal name "$t_{..\_m}$" is a parameter only for the master node, and "$t_{..\_s}$" is a parameter only for the slave node.

[a] Clock is also one of the wake-up pulses.

[b] This time is shorter than the time, in which the master node starts the transmission of the first request protected type identifier or request protected identifier field ($<t_{wakeup\_m}$).

[c] It is longer than the time which master node starts the transmission of clock ($>t_{clock\_start\_m}$).

[d] If an error occurs in a sleep message, then the time of detecting an error is the starting point.

[e] It is after the time which primary clock master shall transmit the clock and before the time which secondary clock master transits to sleep state due to the CXPI network error.

9.4 APP — Multi clock master sequence processing

This subclause describes the optional processing of the secondary clock master in case the primary clock master fails. The secondary clock master operates in case it does not receive clock transmissions from the lower OSI layers when the system transits to normal state. This optional processing may also take over the master schedule.

This function prevents temporary loss of clock transmissions in case the primary clock master stops operating because of a failure. Implementation of this function is optional and at network design time.

If the primary clock master does not transmit the clock to the lower OSI layers due to a failure, the secondary clock master re-requests the wake-up pulse transmission. If the primary clock master does not request the clock transmission to the lower OSI layer, each node does not detect the clock and either requests the secondary clock master or detects a CXPI network error.

After the primary clock master recovers the clock transmission, the clock of the secondary clock master and the clock of the primary master collide in the lower OSI layers.

REQ	8.34 APP — Multi clock master sequence processing — Clock condition of secondary clock master
The secondary clock master shall start the clock transmission and shall start the master schedule if the conditions specified in Table 9 are met. The secondary clock master shall stop the clock transmission if the conditions specified in Table 9 are met. The clock transmission sequence shall be implemented according to Figure 17.	

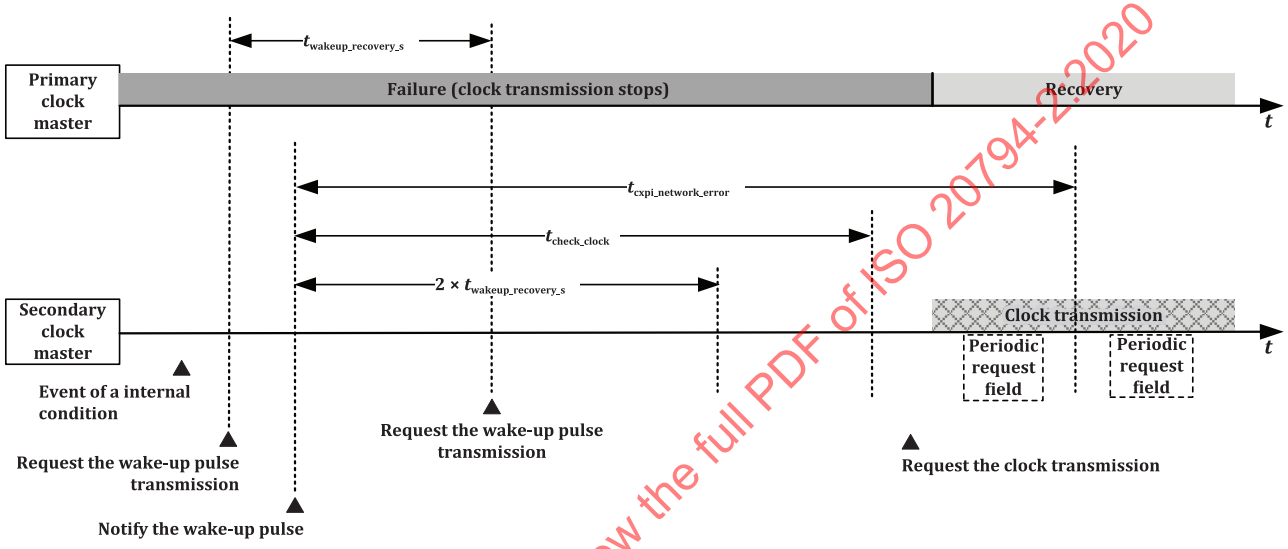


Figure 17 — Sequence of clock master switch

Table 9 — Clock start and stop condition of the secondary clock master

Condition	Description
Clock start condition of secondary clock master	When all of the following conditions are met: — The secondary clock master detects no clock after t_{check_clock} from initialisation (no support for wake-up/sleep); — The secondary clock master detects no clock after t_{check_clock} from when the wake-up-pulse is received (support for wake-up/sleep).
Clock stop condition of secondary clock master	When all of the following conditions are met: — The secondary clock master outputs the clock; — The receiving error[a] continues 1 s or more or the receiving error[a] of 5 messages or more occurs in 10 messages.
[a] The receiving error turns into the CRC error, the parity error, the data length error, sequence count error, and the framing error (see ISO 20794-4).	

9.5 APP — Measurement and/or control data

9.5.1 APP — Publisher and subscriber data

The purpose of publish application measurement and/or control data is to exchange measurement data and/or control data to all nodes. The publisher mechanism is initiated by a master node schedule or event-triggered request field transmission. The publisher of the application measurement and/or control data provides the corresponding response field to the request field.

The subscriber mechanism is followed by those nodes, which has a response field matching the received request field parameter value. All subscribers of the application measurement and/or control data receive the application PDU (assuming no errors were detected).

9.5.2 APP — Measurement and/or control data management

This requirement specifies the measurement and control data management.

REQ	8.35 APP — Measurement and/or control data management — Response field (PDU)
Application measurement and/or control data shall be transmitted in the response field (PDU).	

REQ	8.36 APP — Measurement and/or control data management — Publisher
The publisher of the application measurement and/or control data (PDU) shall always provide the response to the request protected identifier field.	

REQ	8.37 APP — Measurement and/or control data management — Subscriber
All subscribers of the application measurement and/or control data (PDU) shall receive the application PDU (assuming no errors were detected).	

9.5.3 APP — Measurement and/or control data types

Measurement and control data are either of type scalar values or byte array.

REQ	8.38 APP — Measurement and/or control data types — Data types
A scalar application measurement and control data shall have a length of 1 bit to 16 bit. A byte array has a length of 0 byte to 255 byte.	

Scalar application measurement and/or control data are treated as unsigned integers. A 1-bit scalar application measurement and/or control data are of the logic'0' and '1'.

REQ	8.39 APP — Measurement and/or control data types — Single publisher
Each application measurement and/or control data shall have one publisher, i.e. it shall be always sent by the same node in the CXPI cluster.	

Zero, one or multiple nodes may subscribe to the application measurement and/or control data.

REQ	8.40 APP — Measurement and/or control data types — Initial values
All application measurement and/or control data in the subscribing node(s) shall use the initial values as defined by the application.	

REQ	8.41 APP — Measurement and/or control data types — Initial value duration
The initial value for a published application measurement and/or control data shall be valid until the node writes a new value to this application measurement and/or control data. The initial value for a subscribed application measurement and/or control data shall be valid until a new updated value is received by another node.	

9.5.4 APP — Measurement and/or control data consistency

This requirement specifies the measurement and/or control data consistency.

REQ	8.42 APP — Measurement and/or control data consistency
Scalar application measurement and/or control data writing or reading shall be atomic operations, i.e. it shall not be possible for an application to receive an application measurement and/or control data value that is partly updated. This shall also apply to byte arrays.	

NOTE No consistency is guaranteed by this document between any application measurement and/or control data.

9.5.5 APP — Assignment of ReqId

This requirement specifies the assignment of a ReqId.

REQ	8.43 APP — Measurement and control data — Assignment of ReqId
The request protected identifier field shall include a value within the range of 01_{16} to $7F_{16}$. The request protected identifier field values specified in Table 10 shall not be used for proprietary purposes.	

The request protected identifier field values are designed freely for each system, excluding the assigned request protected identifier field values as specified in Table 10. Although, the request protected identifier field uses 01_{16} to $7F_{16}$, 00_{16} is used by the request protected type identifier field.

Table 10 — Assigned request protected identifier field list

ReqId	Name	Usage
$2F_{16}$	Request identifier in inspection (for ECU supplier)	Inspection
$6F_{16}$	Response identifier in inspection (for ECU supplier)	Inspection
$1F_{16}$	Request identifier message for NormalCom sleep PDU or DiagNodeCfg PDU	Request sleep PDU or diagnostic request (see ISO 14229-8)
$5F_{16}$	Request identifier for DiagNodeCfg response message	Diagnostic response (see ISO 14229-8)
$3F_{16}$	Request identifier in inspection (for semiconductor vender)	Inspection, may be used for conformance test purpose to request error message including all Result parameters as specified in 7.12.
$7F_{16}$	Reserved for future extension	—

9.5.6 APP — Priority of ReqId

The priority of a message is decided by the priority of the ReqId. A message with a higher priority ReqId always wins CXPI network access compared to a message with a lower priority ReqId.

REQ	8.44 APP — Measurement and control data — Priority of ReqId
The priority of the message shall be determined by the priority of the ReqId as specified in Table 11.	

The “P” column in Table 11 specifies the priority of a ReqId (see ISO 20794-4:2020, 8.8 DLL — Byte arbitration). The priority of a ReqId is ranked so that the bit of the value '0' gives priority to the bit of the value '1' as compared with the order from LSB to MSB of each ReqId.

Table 11 — Priority of ReqId (excluding the assigned ReqId)

P[a]	ReqId	Binary #	P[a]	ReqId	Binary #	P[a]	ReqId	Binary #	P[a]	ReqId	Binary #
1	40_{16}	1000000	33	42_{16}	1000010	65	41_{16}	1000001	97	43_{16}	1000011
2	20_{16}	0100000	34	22_{16}	0100010	66	21_{16}	0100001	98	23_{16}	0100011
3	60_{16}	1100000	35	62_{16}	1100010	67	61_{16}	1100001	99	63_{16}	1100011
4	10_{16}	0010000	36	12_{16}	0010010	68	11_{16}	0010001	100	13_{16}	0010011
5	50_{16}	1010000	37	52_{16}	1010010	69	51_{16}	1010001	101	53_{16}	1010011
6	30_{16}	0110000	38	32_{16}	0110010	70	31_{16}	0110001	102	33_{16}	0110011
7	70_{16}	1110000	39	72_{16}	1110010	71	71_{16}	1110001	103	73_{16}	1110011
8	08_{16}	0001000	40	$0A_{16}$	0001010	72	09_{16}	0001001	104	$0B_{16}$	0001011
9	48_{16}	1001000	41	$4A_{16}$	1001010	73	49_{16}	1001001	105	$4B_{16}$	1001011
10	28_{16}	0101000	42	$2A_{16}$	0101010	74	29_{16}	0101001	106	$2B_{16}$	0101011
11	68_{16}	1101000	43	$6A_{16}$	1101010	75	69_{16}	1101001	107	$6B_{16}$	1101011

[a] P = Priority.

Table 11 *(continued)*

P[a]	ReqId	Binary #	P[a]	ReqId	Binary #	P[a]	ReqId	Binary #	P[a]	ReqId	Binary #
12	18_{16}	0011000	44	$1A_{16}$	0011010	76	19_{16}	0011001	108	$1B_{16}$	0011011
13	58_{16}	1011000	45	$5A_{16}$	1011010	77	59_{16}	1011001	109	$5B_{16}$	1011011
14	38_{16}	0111000	46	$3A_{16}$	0111010	78	39_{16}	0111001	110	$3B_{16}$	0111011
15	78_{16}	1111000	47	$7A_{16}$	1111010	79	79_{16}	1111001	111	$7B_{16}$	1111011
16	04_{16}	0000100	48	06_{16}	0000110	80	05_{16}	0000101	112	07_{16}	0000111
17	44_{16}	1000100	49	46_{16}	1000110	81	45_{16}	1000101	113	47_{16}	1000111
18	24_{16}	0100100	50	26_{16}	0100110	82	25_{16}	0100101	114	27_{16}	0100111
19	64_{16}	1100100	51	66_{16}	1100110	83	65_{16}	1100101	115	67_{16}	1100111
20	14_{16}	0010100	52	16_{16}	0010110	84	15_{16}	0010101	116	17_{16}	0010111
21	54_{16}	1010100	53	56_{16}	1010110	85	55_{16}	1010101	117	57_{16}	1010111
22	34_{16}	0110100	54	36_{16}	0110110	86	35_{16}	0110101	118	37_{16}	0110111
23	74_{16}	1110100	55	76_{16}	1110110	87	75_{16}	1110101	119	77_{16}	1110111
24	$0C_{16}$	0001100	56	$0E_{16}$	0001110	88	$0D_{16}$	0001101	120	$0F_{16}$	0001111
25	$4C_{16}$	1001100	57	$4E_{16}$	1001110	89	$4D_{16}$	1001101	121	$4F_{16}$	1001111
26	$2C_{16}$	0101100	58	$2E_{16}$	0101110	90	$2D_{16}$	0101101	—	—	—
27	$6C_{16}$	1101100	59	$6E_{16}$	1101110	91	$6D_{16}$	1101101	—	—	—
28	$1C_{16}$	0011100	60	$1E_{16}$	0011110	92	$1D_{16}$	0011101	—	—	—
29	$5C_{16}$	1011100	61	$5E_{16}$	1011110	93	$5D_{16}$	1011101	—	—	—
30	$3C_{16}$	0111100	62	$3E_{16}$	0111110	94	$3D_{16}$	0111101	—	—	—
31	$7C_{16}$	1111100	63	$7E_{16}$	1111110	95	$7D_{16}$	1111101	—	—	—
32	02_{16}	0000010	64	01_{16}	0000001	96	03_{16}	0000011	—	—	—

[a] P = Priority.

9.6 APP — Error handling

9.6.1 APP — General

The application has the function to detect the following errors:

a) CXPI network error;

b) sequence count (SCT) error (optional).

9.6.2 APP — CXPI network error

9.6.2.1 APP — Support for wake-up/sleep (optional)

This requirement specifies the CXPI network error for support for wake-up/sleep.

REQ	8.45 — APP — CXPI network error — Support for wake-up/sleep (optional)
In standby state or normal state, each node shall detect a CXPI network error when it does not receive a message until the judgment time for a CXPI network error. Table 12 specifies the error detection node, the state of error detection, and the processing after error detection Table 13 specifies the CXPI network error condition.	