IEC 61158-4-3

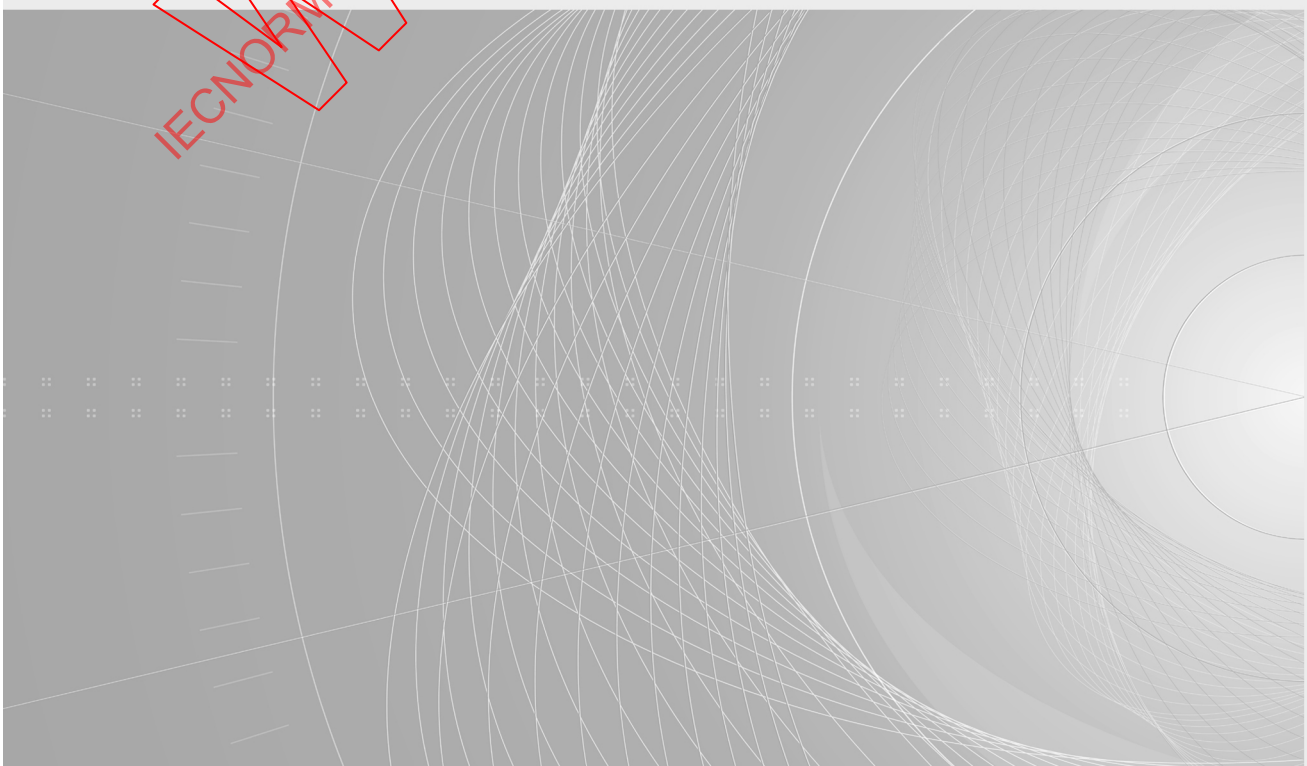
Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 4-3: Data-link layer protocol specification – Type 3 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-3: Spécification du protocole de la couche liaison de données – Éléments de Type 3



IEC 61158-4-3:2007-12(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 61158-4-3

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 4-3: Data-link layer protocol specification – Type 3 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-3: Spécification du protocole de la couche liaison de données – Éléments de Type 3

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XH
CODE PRIX

ICS 25.040.40; 35.100.20

ISBN 978-2-8322-1776-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-3: Data-link layer protocol specification – Type 3 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the IEC 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

IEC draws attention to the fact that it is claimed that compliance with this standard may involve the use of patents as follows, where the [xx] notation indicates the holder of the patent right:

Type 3 and possibly other types:

DE 36 43 979 C2	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung
DE 36 43 979 A1	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung

IEC takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights are registered with IEC. Information may be obtained from:

[SI]: SIEMENS AG
Ludwig Winkel
Siemensallee 84
D-76181 Karlsruhe
Germany

Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61158-4-3 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This bilingual version (2014-08) corresponds to the English version, published in 2007-12.

This first edition and its companion parts of the IEC 61158-4 subseries cancel and replace IEC 61158-4:2003. This edition of this part constitutes an editorial revision.

This edition of IEC 61158-4 includes the following significant changes from the previous edition:

a) deletion of the former Type 6 fieldbus, and the placeholder for a Type 5 fieldbus data link layer, for lack of market relevance;

b) addition of new types of fieldbuses;

c) division of this part into multiple parts numbered -4-1, -4-2, …, -4-19.

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/474/FDIS	65C/485/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

The French version of this standard has not been voted upon.

This publication has been drafted in accordance with ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under http://webstore.iec.ch in the data related to the specific publication. At this date, the publication will be:

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

NOTE The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

The list of all the parts of the IEC 61158 series, under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the "three-layer" fieldbus reference model described in IEC/TR 61158-1.

The data-link protocol provides the data-link service by making use of the services available from the physical layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer data-link entities (DLEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

a) as a guide for implementors and designers;

b) for use in the testing and procurement of equipment;

c) as part of an agreement for the admittance of systems into the open systems environment;

d) as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-3: Data-link layer protocol specification – Type 3 elements

1 Scope

1.1 General

The data-link layer provides basic time-critical messaging communications between devices in an automation environment.

This protocol provides communication opportunities to a pre-selected "master" subset of data-link entities in a cyclic asynchronous manner, sequentially to each of those data-link entities. Other data-link entities communicate only as permitted and delegated by those master data-link entities.

For a given master, its communications with other data-link entities can be cyclic, or acyclic with prioritized access, or a combination of the two.

This protocol provides a means of sharing the available communication resources in a fair manner. There are provisions for time synchronization and for isochronous operation.

1.2 Specifications

This standard specifies

a) procedures for the timely transfer of data and control information from one data-link user entity to a peer user entity, and among the data-link entities forming the distributed data-link service provider;

b) the structure of the fieldbus DLPDUs used for the transfer of data and control information by the protocol of this standard, and their representation as physical interface data units.

1.3 Procedures

The procedures are defined in terms of

a) the interactions between peer DL-entities (DLEs) through the exchange of fieldbus DLPDUs;

b) the interactions between a DL-service (DLS) provider and a DLS-user in the same system through the exchange of DLS primitives;

c) the interactions between a DLS-provider and a Ph-service provider in the same system through the exchange of Ph-service primitives.

1.4 Applicability

These procedures are applicable to instances of communication between systems which support time-critical communications services within the data-link layer of the OSI or fieldbus reference models, and which require the ability to interconnect in an open systems interconnection environment.

Profiles provide a simple multi-attribute means of summarizing an implementation's capabilities, and thus its applicability to various time-critical communications needs.

1.5 Conformance

This standard also specifies conformance requirements for systems implementing these procedures. This standard does not contain tests to demonstrate compliance with such requirements.

2 Normative references

The following referenced documents are indispensable for the application of this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61158-2 (Ed.4.0), *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 2: Physical layer specification and service definition*

IEC 61158-3-3, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 3-3: Data link service definition – Type 3 elements*

ISO/IEC 2022, *Information technology – Character code structure and extension techniques*

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

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

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

ISO 1177, *Information processing – Character structure for start/stop and synchronous character oriented transmission*

3 Terms, definitions, symbols and abbreviations

For the purposes of this standard, the following terms, definitions, symbols and abbreviations apply.

3.1 Reference model terms and definitions

This standard is based in part on the concepts developed in ISO/IEC 7498-1 and ISO/IEC 7498-3, and makes use of the following terms defined therein.

3.1.1 called-DL-address [7498-3]

3.1.2 calling-DL-address [7498-3]

3.1.3 centralized multi-end-point-connection [7498-1]

3.1.4 correspondent (N)-entities [7498-1]
correspondent DL-entities (N=2)
correspondent Ph-entities (N=1)

3.1.5 demultiplexing [7498-1]

3.1.6 DL-address [7498-3]

3.1.7 DL-address-mapping [7498-1]

3.1.8 DL-connection [7498-1]

3.1.9 DL-connection-end-point [7498-1]

3.1.10 DL-connection-end-point-identifier [7498-1]

3.1.11 DL-connection-mode transmission [7498-1]

3.1.12 DL-connectionless-mode transmission [7498-1]

3.1.13 DL-data-sink [7498-1]

3.1.14 DL-data-source [7498-1]

3.1.15 DL-duplex-transmission [7498-1]

3.1.16 DL-facility [7498-1]

3.1.17 DL-local-view [7498-3]

3.1.18 DL-name [7498-3]

3.1.19 DL-protocol [7498-1]

3.1.20 DL-protocol-connection-identifier [7498-1]

3.1.21 DL-protocol-control-information [7498-1]

3.1.22 DL-protocol-data-unit [7498-1]

3.1.23 DL-protocol-version-identifier [7498-1]

3.1.24 DL-relay [7498-1]

3.1.25 DL-service-connection-identifier [7498-1]

3.1.26 DL-service-data-unit [7498-1]

3.1.27 DL-simplex-transmission [7498-1]

3.1.28 DL-subsystem [7498-1]

3.1.29 DL-user-data [7498-1]

3.1.30 flow control [7498-1]

3.1.31 layer-management [7498-1]

3.1.32 multiplexing [7498-3]

3.1.33 naming-(addressing)-authority [7498-3]

3.1.34 naming-(addressing)-domain [7498-3]

3.1.35 naming-(addressing)-subdomain [7498-3]

3.1.36 (N)-entity [7498-1]
DL-entity
Ph-entity

3.1.37 (N)-interface-data-unit [7498-1]
DL-service-data-unit (N=2)
Ph-interface-data-unit (N=1)

3.1.38 (N)-layer [7498-1]
DL-layer (N=2)
Ph-layer (N=1)

3.1.39 (N)-service [7498-1]
DL-service (N=2)
Ph-service (N=1)

3.1.40 (N)-service-access-point [7498-1]
DL-service-access-point (N=2)
Ph-service-access-point (N=1)

3.1.41 (N)-service-access-point-address [7498-1]
DL-service-access-point-address (N=2)
Ph-service-access-point-address (N=1)

3.1.42 peer-entities [7498-1]

3.1.43 Ph-interface-control-information [7498-1]

3.1.44 Ph-interface-data [7498-1]

3.1.45 primitive name [7498-3]

3.1.46 reassembling [7498-1]

3.1.47 recombining [7498-1]

3.1.48 reset [7498-1]

3.1.49 responding-DL-address [7498-3]

3.1.50 routing [7498-1]

3.1.51 segmenting [7498-1]

3.1.52 sequencing [7498-1]

3.1.53 splitting [7498-1]

3.1.54 synonymous name [7498-3]

3.1.55 systems-management [7498-1]

3.2 Service convention terms and definitions

This standard also makes use of the following terms defined in ISO/IEC 10731 as they apply to the data-link layer:

3.2.1 acceptor

3.2.2 asymmetrical service

3.2.3 confirm (primitive);
requestor.deliver (primitive)

3.2.4 deliver (primitive)

3.2.5 DL-confirmed-facility

3.2.6 DL-facility

3.2.7 DL-local-view

3.2.8 DL-mandatory-facility

3.2.9 DL-non-confirmed-facility

3.2.10 DL-provider-initiated-facility

3.2.11 DL-provider-optional-facility

3.2.12 DL-service-primitive; primitive

3.2.13 DL-service-provider

3.2.14 DL-service-user

3.2.15 DL-user-optional-facility

3.2.16 indication (primitive) acceptor.deliver (primitive)

3.2.17 multi-peer

3.2.18 request (primitive); requestor.submit (primitive)

3.2.19 requestor

3.2.20 response (primitive); acceptor.submit (primitive)

3.2.21 submit (primitive)

3.2.22 symmetrical service

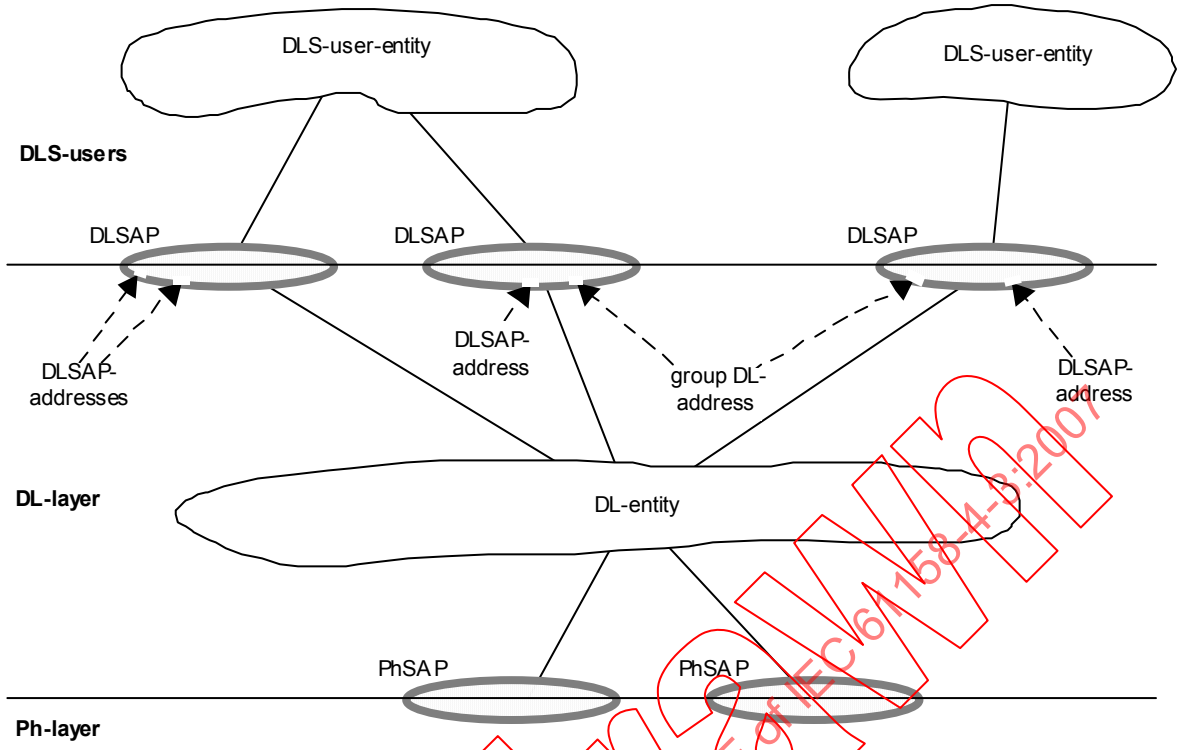
3.3 Common terms and definitions

NOTE Many definitions are common to more than one protocol Type; they are not necessarily used by all protocol Types.

3.3.1
DL-segment, link, local link
single DL-subnetwork in which any of the connected DLEs may communicate directly, without any intervening DL-relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication

3.3.2
DLSAP
distinctive point at which DL-services are provided by a single DL-entity to a single higher-layer entity

NOTE This definition, derived from ISO/IEC 7498-1, is repeated here to facilitate understanding of the critical distinction between DLSAPs and their DL-addresses. (See Figure 1.)



NOTE 1 DLSAPs and PhSAPs are depicted as ovals spanning the boundary between two adjacent layers.

NOTE 2 DL-addresses are depicted as designating small gaps (points of access) in the DLL portion of a DLSAP.

NOTE 3 A single DL-entity may have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

Figure 1 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses

3.3.3
DL(SAP)-address
either an individual DLSAP-address, designating a single DLSAP of a single DLS-user, or a group DL-address potentially designating multiple DLSAPs, each of a single DLS-user.

NOTE This terminology is chosen because ISO/IEC 7498-3 does not permit the use of the term DLSAP-address to designate more than a single DLSAP at a single DLS-user.

3.3.4
(individual) DLSAP-address
DL-address that designates only one DLSAP within the extended link

NOTE A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

3.3.5
extended link
DL-subnetwork, consisting of the maximal set of links interconnected by DL-relays, sharing a single DL-name (DL-address) space, in which any of the connected DL-entities may communicate, one with another, either directly or with the assistance of one or more of those intervening DL-relay entities

NOTE An extended link may be composed of just a single link.

3.3.6
frame
denigrated synonym for DLPDU

3.3.7
group DL-address
DL-address that potentially designates more than one DLSAP within the extended link. A single DL-entity may have multiple group DL-addresses associated with a single DLSAP. A single DL-entity also may have a single group DL-address associated with more than one DLSAP

3.3.8
node
single DL-entity as it appears on one local link

3.3.9
receiving DLS-user
DL-service user that acts as a recipient of DL-user-data

NOTE A DL-service user can be concurrently both a sending and receiving DLS-user.

3.3.10
sending DLS-user
DL-service user that acts as a source of DL-user-data

3.4 Additional Type 3 definitions

3.4.1
acknowledge DLPDU
reply DLPDU that contains no DLSDU

3.4.2
address extension
DLSAP address or region/segment address

3.4.3
bit time
time to transmit one bit

3.4.4
confirmed message exchange
complete data transfer with request and acknowledgement or response DLPDU

3.4.5
controller_type
hardware class of the communications entity

3.4.6
current master
token holder

3.4.7
data DLPDU
DLPDU that carries a DLSDU from a local DLS-user to a remote DLS-user

3.4.8
DL_status
status that specifies the result of the execution of the associated request

3.4.9
GAP
range of station (DLE) DL-addresses from this station (TS) to its successor (NS) in the logical token ring, excluding stations above HSA

3.4.10
GAP maintenance
registration of new Master and slave stations

3.4.11
isochronous mode
special operational mode that implies both a constant (isochronous) cycle with a fixed schedule of high and low priority messages, and the synchronization of the DLS-users with this constant (isochronous) cycle

3.4.12
local DLS-user
DLS-user that initiates the current service

3.4.13
message exchange
complete confirmed or unconfirmed data transfer

3.4.14
region/segment address
address extension that identifies a particular fieldbus subnetwork

NOTE This supports DL-routing between fieldbusses.

3.4.15
request data
DLSDU provided by the remote DLS-user to the local DLS-user

3.4.16
remote DLE
addressed DLE of a service request (that is, the intended receiving DLE of any resulting send/request DLPDU)

3.4.17
remote DLS-user
addressed DLS-user of a service request (that is, the intended receiver of any resulting indication primitive)

3.4.18
reply DLPDU
DLPDU transmitted from a remote DLE to the initiating (local) DLE, and possibly other DLEs

NOTE When the remote DLE is a Publisher, the reply DLPDU also can be sent to several remote DLEs.

3.4.19
response DLPDU
reply DLPDU that carries a DLSDU from a remote DLS-user to local DLS-user

3.4.20
send data
DLSDU provided by a local DLS-user to a remote DLS-user

3.4.21
send/request DLPDU
DLPDU that carries either a request for data or a DLSDU or both from a local DLS-user to a remote DLS-user

3.4.22
time master
device which is able to send clock synchronization DLPDUs

NOTE Link devices have time master functionality.

3.4.23
time receiver
device which is able to be time synchronized by a time Master

3.4.24
token holder
Master station that controls bus access

3.4.25
token passing
medium access method, in which the right to transmit is passed from master station to master station in a logical ring

3.5 Common symbols and abbreviations

3.5.1 Data units

3.5.1.1	**DLPDU**	DL-protocol data unit
3.5.1.2	**DLSDU**	DL-service data unit
3.5.1.3	**PhIDU**	Ph-interface data unit
3.5.1.4	**PhPDU**	Ph-protocol data unit

3.5.2 Miscellaneous

3.5.2.1	**DL-**	data link layer (as a prefix)
3.5.2.2	**DLCEP**	DL-connection endpoint
3.5.2.3	**DLE**	DL-entity (the local active instance of the Data Link layer)
3.5.2.4	**DLL**	DL-layer
3.5.2.5	**DLM-**	DL-management (as a prefix)
3.5.2.6	**DLMS**	DL-management-service
3.5.2.7	**DLS**	DL-service
3.5.2.8	**DLSAP**	DL-service access point
3.5.2.9	**FIFO**	first-in first-out (queuing method)
3.5.2.10	**LLC**	logical link control
3.5.2.11	**MAC**	medium access control
3.5.2.12	**OSI**	open systems interconnection
3.5.2.13	**Ph-**	physical layer (as a prefix)
3.5.2.14	**PhE**	Ph-entity (the local active instance of the Physical layer)
3.5.2.15	**PhL**	Ph-layer

3.5.2.16 PhS Ph-service

3.5.2.17 PhSAP Ph-service access point

3.5.2.18 QoS quality of service

3.6 Type 3 symbols and abbreviations

3.6.1 ACK acknowledge(ment) DLPDU

3.6.2 ASM active spare time message

3.6.3 ASP active spare time period

3.6.4 Bus ID bus identification, an address extension (region/DL-segment address) that identifies a particular bus as supporting routing between DL-segments

3.6.5 CRX character receive execution

3.6.6 CS clock synchronization

3.6.7 CTX character transmit execution

3.6.8 DA destination address of a DLPDU

3.6.9 DAE destination address extension(s) of a DLPDU, which convey D_SAP_index and/or destination bus ID

3.6.10 D_SAP destination service access point, the DLSAP associated with the remote DLS-user

3.6.11 D_SAP_index destination service access point index – that component of a DLSAP address which designates a DLSAP and remote DLS-user within the remote DLE

3.6.12 DXM data exchange multicast

3.6.13 ED end delimiter of a DLPDU

3.6.14 EOA END-OF-ACTIVITY

3.6.15 EOD END-OF-DATA

3.6.16 EODA END-OF-DATA-AND-ACTIVITY

3.6.17 EXT address extension bit of a DLPDU

3.6.18 FC frame control (frame type) field of a DLPDU

3.6.19 FCB frame count bit of a DLPDU (FC field) used to eliminate lost or duplicated DLPDUs

3.6.20 FCV frame count bit valid bit of a DLPDU, indicates whether the FCB is to be evaluated

3.6.21 FCS frame check sequence (synchronous) or frame checksum (asynchronous)

3.6.22 FLC fieldbus link control

3.6.23 G GAP update factor, the number of token rotations between GAP maintenance (update) cycles

3.6.24 GAPL GAP list containing the status of all stations in this station's GAP

3.6.25 IsoM isochronous mode

3.6.26 Hd Hamming distance, a measure of DLPDU integrity, the minimum number of bit errors that can cause acceptance of a spurious DLPDU

3.6.27 HSA highest station address installed (configured) on this fieldbus

3.6.28 L length of the information field, the part of a DLPDU that is checked by the FCS

3.6.29 LE field giving the length of a DLPDU beyond the fixed part

3.6.30 LEr field that repeats the length to increase DLPDU integrity

3.6.31 LMS list of master stations

3.6.32 LR local resource not available or not sufficient (DL/DLM_status of the service primitive)

3.6.33 LS local service not activated at DL-service access point or local DLSAP not activated (DL/DLM_status of the service primitive)

3.6.34 lsb least significant bit of a field or octet

3.6.35 max the arithmetic maximum function

3.6.36 MCT multicast

3.6.37 MP message transfer message retry transfer periods

3.6.38 mr number of retries

3.6.39 msb most significant bit of a field or octet

3.6.40 MSRD DLS: Send and Request Data with Multicast reply

3.6.41 mt number of retries per token rotation

3.6.42 n number of stations

3.6.43 NA no acknowledgement/response (DL/DLM_status of the service primitive)

3.6.44 na number of active stations

3.6.45 NIL locally determined value

3.6.46 NO not ok (DL/DLM_status of the service primitive)

3.6.47 np number of passive stations

3.6.48 NR no response DL-data acknowledgement negative and send data ok (DL_status of the service primitive)

3.6.49 NRZ non-return-to-zero (PhL), an encoding technique where transitions occur only when successive data bits have different values

3.6.50 NS next station, the station to which this master will pass the token

3.6.51 OK service finished according to the rules (DL/DLM_status of the service primitive)

3.6.52 PhICI PhL Interface Control Information [ISO/IEC 7498-1]

3.6.53 PhPCI PhL Protocol Control Information [ISO/IEC 7498-1]

3.6.54 PON power on transition occurs at a station

3.6.55 PS previous station, the station which passes the token to this master station

3.6.56 PSP passive spare time period

3.6.57 RDH response DL-data high and no resource for send data (DL/DLM_status of the service primitive)

3.6.58 RDL response DL/DLM-data low and no resource for send data (DL/DLM_status of the service primitive)

3.6.59 Res reserved

3.6.60 RET retry

3.6.61 RR no resource for send data and no response DL-data available (negative acknowledgement) (DL/DLM_status of the service primitive)

3.6.62 RS no service, or no service activated at remote DLSAP (negative acknowledgement) (DL/DLM_status of the service primitive)

3.6.63 RSYS system message rate (in message transfer periods per second) at which confirmed DL-message exchanges are performed

3.6.64 SA source address of a DLPDU

3.6.65 SAE source address extension(s) of a DLPDU, which convey S_SAP_index and/or source bus ID

3.6.66 SC single character acknowledge DLPDU

3.6.67 SD1 to SD4 start delimiters of asynchronous DLPDU transmission

3.6.68 SDA Send Data with Acknowledge (DL-service)

3.6.69 SDA_H/L Send Data with Acknowledge high/low (DLPDU Function)

3.6.70 SDX start delimiter of asynchronous or synchronous DLPDU transmission

3.6.71 SDL1 to SDL5 start delimiters of synchronous DLPDU transmission

3.6.72 SDN Send Data with No Acknowledge (DL-service)

3.6.73 SDN_H/L Send Data with No Acknowledge high/low (DLPDU Function)

3.6.74 SM state machine

3.6.75 SOA START-OF-ACTIVITY

3.6.76 SRC send receive control

3.6.77 SRD Send and Request Data with Reply (DL-service)

3.6.78 SRD_H/L Send and Request Data with Reply high/low (DLPDU Function)

3.6.79 SRU send receive unit

3.6.80 S_SAP source service access point, the DLSAP associated with the initiating local DLS-user

3.6.81 S_SAP_index source service access point index – a component of a DLSAP-address which designates that DLSAP within the DLE at which the transaction is being initiated

3.6.82 Stn station, a device implementing a DLE with a fieldbus DL-address

3.6.83 SYN synchronizing bits of a DLPDU (period of idle), which guarantees the specified DLPDU integrity and allows for receiver synchronization

3.6.84 TA/R time to transmit an acknowledgement/response DLPDU

3.6.85 TASM ASM message time

3.6.86 t_{BIT} bit time

3.6.87 TCSI clock synchronization interval time

3.6.88 T_{CT} isochronous cycle time

3.6.89 TGUD GAP update time

3.6.90 T_{ID} idle time

3.6.91 TIM timer state machine

3.6.92 T_{MP} message transfer period

3.6.93 TP token transfer period

3.6.94 TPSP passive spare time

3.6.95 TPTG post transmission gap time

3.6.96 TQUI quiet time

3.6.97 TRCT real isochronous cycle time

3.6.98 T_{RD} receive delay time

3.6.99 TRDY ready time

3.6.100 TRES spare time

3.6.101 T_{RR} real rotation time

3.6.102 TS this station

3.6.103 TS/R send/request DLPDU time

3.6.104 T_{SD} send delay time

3.6.105 TSDI station delay of initiator

3.6.106 TSDR station delay of responder

3.6.107 TSET setup time

3.6.108 T_{SH} time shift

3.6.109 T_{SL} slot time

3.6.110 T_{SM} safety margin time

3.6.111 T_{SR} system reaction time

3.6.112 TSYN synchronization time

3.6.113 TSYNI synchronization interval time

3.6.114 T_{TC} token cycle time

3.6.115 T_{TD} transmission delay time

3.6.116 T_{TF} token DLPDU time

3.6.117 T_{TH} token holding time

3.6.118 T_{TO} timeout time

3.6.119 T_{TP} token transfer period

3.6.120 T_{TR} target rotation time

3.6.121 UART universal asynchronous receiver/transmitter

3.6.122 UC UART character

3.6.123 UE negative acknowledgement, remote user interface error (DL/DLM_status of the service primitive)

4 Common DL-protocol elements

4.1 Frame check sequence

4.1.1 General

Any reference to bit *K* of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^K.

NOTE 1 This is sometimes referred to as “little endian” bit numbering.

As in other International Standards (see Note 2), DLPDU-level error detection is provided by calculating and appending a multi-bit frame check sequence (FCS) to the other DLPDU fields during transmission to form a "systematic code word"[1)] of length ***n*** consisting of ***k*** DLPDU message bits followed by ***n - k*** redundant bits, and by calculating during reception that the message and concatenated FCS form a legal (*n*,*k*) code word. The mechanism for this checking is as follows:

NOTE 2 For example, ISO/IEC 3309, ISO/IEC 8802 and ISO/IEC 9314-2.

The generic form of the generator polynomial for this FCS construction is specified in equation (4) and the polynomial for the receiver’s expected residue is specified in equation

1) W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes* (2nd edition), MIT Press, Cambridge, 1972.

(9). The specific polynomials for DL-protocol type 3 are specified in Table 1. An exemplary implementation is shown in Annex B.

Table 1 – FCS length, polynomials and constants by Type 3 synchronous

Item	Value
n-k	16
G(x)	$X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$ (see Notes 1, 2, 3)
R(x)	$X^{15} + X^{14} + X^{13} + X^9 + X^8 + X^7 + X^4 + X^2$ (see Note 4)

NOTE 1 Code words D(X) constructed from this G(X) polynomial have Hamming distance 4 for lengths ≤ 344 octets and Hamming distance 5 for lengths ≤ 15 octets.

NOTE 2 This G(X) polynomial is relatively prime to all, and is thus not compromised by any, of the polynomials commonly used in DCEs (modems): the differential encoding polynomial $1 + X^{-1}$ and all primitive scrambling polynomials of the form $1 + X^{-j} + X^{-k}$.

NOTE 3 This G(X) polynomial is the optimal 16-bit polynomial for burst error detection over DLPDUs of 300 octets or less when the statistics of the error burst have a Poisson distribution (as is the usual case).

NOTE 4 The remainder R(x) should be 1110 0011 1001 0100 (X^{15} to X^0, respectively) in the absence of errors.

4.1.2 At the sending DLE

The original message (that is, the DLPDU without an FCS), the FCS, and the composite message code word (the concatenated DLPDU and FCS) shall be regarded as vectors M(X), F(X), and D(X), of dimension *k*, *n* - *k*, and *n*, respectively, in an extension field over Base Galois Field(2). If the message bits are $m_1 \ldots m_k$ and the FCS bits are $f_{n-k-1} \ldots f_0$, where

$m_1 \ldots m_8$ form the first octet sent,

$m_{8N-7} \ldots m_{8N}$ form the *N*th octet sent,

$f_7 \ldots f_0$ form the last octet sent, and

m_1 is sent by the first PhL symbol(s) of the message and f_0 is sent by the last PhL symbol(s) of the message (not counting PhL framing information),

NOTE 1 This "as transmitted" ordering is critical to the error detection properties of the FCS.

then the message vector M(X) shall be regarded to be

$$M(X) = m_1X^{k-1} + m_2X^{k-2} + \ldots + m_{k-1}X^1 + m_k \quad (1)$$

and the FCS vector F(X) shall be regarded to be

$$F(X) = f_{n-k-1}X^{n-k-1} + \ldots + f_0 \quad (2)$$
$$= f_{15}X^{15} + \ldots + f_0$$

The composite vector D(X), for the complete DLPDU, shall be constructed as the concatenation of the message and FCS vectors

$$D(X) = M(X)\,X^{n-k} + F(X) \quad (3)$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \ldots + f_0$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{16} + f_{15}X^{15} + \ldots + f_0$$

The DLPDU presented to the PhL shall consist of an octet sequence in the specified order.

The redundant check bits $f_{n-k-1} \ldots f_0$ of the FCS shall be the coefficients of the remainder F(X), after division by G(X), of $L(X)\,(X^k + 1) + M(X)\,X^{n-k}$

where G(X) is the degree *n-k* generator polynomial for the code words

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \ldots + 1 \quad (4)$$
$$= X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{6} + X^{3} + X^{2} + X + 1$$

and L(X) is the maximal weight (all ones) polynomial of degree *n-k-1*

$$L(X) = \frac{X^{n-k} + 1}{X + 1} = X^{n-k-1} + X^{n-k-2} + \ldots + X + 1 \quad (5)$$
$$= X^{15} + X^{14} + X^{13} + X^{12} + \ldots + X^{2} + X + 1$$

That is,

$$F(X) = L(X)\,(X^{k} + 1) + M(X)\,X^{n-k} \ (\text{modulo}\ G(X)) \quad (6)$$

NOTE 2 The L(X) terms are included in the computation to detect initial or terminal message truncation or extension by adding a length-dependent factor to the FCS.

NOTE 3 As a typical implementation when n-k = 16, the initial remainder of the division is preset to all ones. The transmitted message bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (4). The ones complement of the resulting remainder is transmitted as the (n-k)-bit FCS, with the coefficient of X^{n-k-1} transmitted first.

4.1.3 At the receiving DLE

The octet sequence indicated by the PhE shall be concatenated into the received DLPDU and FCS, and regarded as a vector V(X) of dimension ***u***

$$V(X) = v_1X^{u-1} + v_2X^{u-2} + \ldots + v_{u-1}X + v_u \quad (7)$$

NOTE 1 Because of errors **u** can be different than **n**, the dimension of the transmitted code vector.

A remainder R(X) shall be computed for V(X), the received DLPDU and FCS, by a method similar to that used by the sending DLE (see 4.1.2) in computing F(X)

$$R(X) = L(X)\,X^{u} + V(X)\,X^{n-k} \ (\text{modulo}\ G(X)) \quad (8)$$
$$= r_{n-k-1}X^{n-k-1} + \ldots + r_0$$

Define E(X) to be the error code vector of the additive (modulo-2) differences between the transmitted code vector D(X) and the received vector V(X) resulting from errors encountered (in the PhS provider and in bridges) between sending and receiving DLEs.

$$E(X) = D(X) + V(X) \quad (9)$$

If no error has occurred, so that E(X) = 0, then R(X) will equal a non-zero constant remainder polynomial

$$R_{ok}(X) = L(X)\,X^{n-k} \ (\text{modulo}\ G(X)) \quad (10)$$

whose value is independent of D(X). Unfortunately R(X) will also equal $R_{ok}(X)$ in those cases where E(X) is an exact non-zero multiple of G(X), in which case there are “undetectable” errors. In all other cases, R(X) will not equal $R_{ok}(X)$; such DLPDUs are erroneous and shall be discarded without further analysis.

NOTE 2 As a typical implementation, the initial remainder of the division is preset to all ones. The received bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (4).

5 Overview of the DL-protocol

NOTE Annex A specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. The specification of Annex A is complementary to the textual specification in this and related clauses in the body of this standard. In case of conflict the requirements of Annex A take precedence.

5.1 General

From the requirements of the various application fields, for example, process control, factory automation, power distribution, building automation, primary process industry etc., the following characteristic features of the fieldbus data-link protocol are resulting (see Table 2).

Table 2 – Characteristic features of the fieldbus data-link protocol

Feature	Description
Station types	Masters (active stations, with bus access control); Slaves (passive stations, without bus access control); preferably at most 32 masters, optionally up to 127, if the applications are not time critical
Station addressing	0 to 127 (127 = global addresses for broad-cast and multicast messages), address extension for region/DL-segment address and service access address (DLSAP), 6 bit each
Bus access	Hybrid: decentralized and central; "token-passing" between master stations and "Master-Slave" between Master and slave stations
Data transfer services	Send Data with/without Acknowledge Send and Request Data with Reply Send and Request Data with Multicast Reply Clock Synchronization
DLPDU length	max. 255 octets per DLPDU, 0 to 246 octets for each data unit without address extension
Transmission speed	Depending on network topology and cable lengths, for example, step-wise from 9,6 to 12000 kBit/s
Transmission characteristic	Half duplex, asynchronous and synchronous transmission
Data integrity asynchronous transmission	Messages with Hamming distance (Hd) = 4, sync slip detection, special sequence to avoid loss or duplication of data
Data integrity synchronous transmission	Hamming distance (Hd) = 4 for DLPDU lengths shorter than 255 octets and Hd = 5 for lengths shorter than 15 octets, special sequence to avoid loss or duplication of data

NOTE 1 The DLPDU format for asynchronous transmission is the format FT 1.2 (asynchronous transmission with start-stop synchronization) for Telecontrol Equipment and Systems, which is specified in IEC 60870-5-1.

NOTE 2 The DLPDU format for synchronous transmission is based on octet characters.

5.2 Overview of the medium access control and transmission protocol

The fieldbus data-link layer uses controlled medium access accomplished by a hybrid medium access method: a decentralized method according to the principle of **token-passing** is underlain by a central method according to the **master-slave** principle. Medium access control may be exercised by each master station (active station). The token-passing is characterized by the following features:

a) Token-passing allows fair media access for all token holders.

EXAMPLE: When four token holders produce the same amount of similar priority data they will share the media so that on average each of them can use 25% of the available message transfer time. With the token-passing procedure, rules exist to share the message transfer time between the token holders without discrimination of any of them. In case of usage of the whole of the available token holding time by one token holder in one token rotation this token holder is limited to one high priority message in the next token rotation after which it has to transfer the token immediately.

b) Token-passing guarantees short reaction time.

EXAMPLE: For urgent messages, a maximum delay is specified for delivery of the information. Thus, a configurable time (T_{TR}) specifies the target rotation time. Independent of the number of token holders and the amount of messages that have to be sent, at least one urgent message can be sent by each token holder. Thus, a short reaction time is guaranteed for one urgent message from each token holding station in each token holding period.

c) Token-passing allows flexible (re-)configuration.

EXAMPLE: When a token holder is switched on or off, it will be automatically included or excluded in the logical ring. In the next token rotation after detection and inclusion in the logical token ring, the new token holder has the same rights to send messages as the other token holders. The sharing of the bus message transfer time is organized automatically without changing of parameters of the other token holders.

Medium access control may be exercised by each master station (active station) if the station has the token. The token is passed from master station to master station in a logical ring and

thus determines the instant, when a master station may access the medium. Controlled token passing is managed by each station knowing its predecessor (previous station, PS), the station from which it receives the token. Furthermore, each station knows its successor (next station, NS), that is, the station to which the token is transmitted, and its own address (This station, TS). Each master station determines the PS and NS addresses after the initialization of the operating parameters for the first time and then later dynamically according to the algorithm described in 5.3.2.4.

The Master-Slave principle is characterized by the following features:

Communication is always initiated by a master station which has the permission for medium access, the token. slave stations (passive stations) act neutrally in respect to medium access, that is, they do not transmit independently but only on request. If the logical ring consists of only one Master and several slave stations then it is a pure Master-Slave system.

The following error conditions, exceptions and operational states in the system are dealt with:

— multiple tokens,
— lost token,
— error in token passing,
— duplicate station addresses,
— stations with faulty transmitter or receiver,
— adding and removing stations during operation,
— any combinations of master and slave stations.

5.3 Transmission modes and DL-entity

5.3.1 Overview

The exchange of messages takes place confirmed or unconfirmed. A confirmed message exchange consists of a master station's send/request DLPDU and the associated acknowledgement or response DLPDU of a Master or a slave station. User data may be transmitted in the send/request DLPDU as well as in the response DLPDU. The acknowledgement DLPDU does not contain any user data (for DLPDU formats see clause 7).

An unconfirmed message exchange takes place only in case of token transmission and in case of the transmission of data without acknowledgement (for example, necessary for broadcast messages). In both modes of operation, there is no acknowledgement. In broadcast messages a master station (initiator) addresses all other stations at the same time by means of a global address (highest station address, all address bits are binary "1").

All stations except the respective token holder (initiator) shall in general monitor all requests. The stations acknowledge or respond only when they are addressed. The acknowledgement or response shall arrive within a predefined time, the slot time, otherwise the initiator repeats, depending on the predefined retry limit, the request if it is not a "first request" (see 6.4, FCB). A new request or a token shall not be issued by the initiator before the expiration of a waiting period after receiving an acknowledge or response, the idle time (see 5.5).

If the responder does not acknowledge or respond after a predefined number of retries (see Table 5), it is marked as "non-operational". If a responder is "non-operational", a later unsuccessful request will not be repeated.

The modes of transmission operation define the sequence of the message transfer periods. Three modes are distinguished

1) Token handling.

2) Send operation.

3) Send and request operation.

5.3.2 Token procedures

5.3.2.1 Token circulation

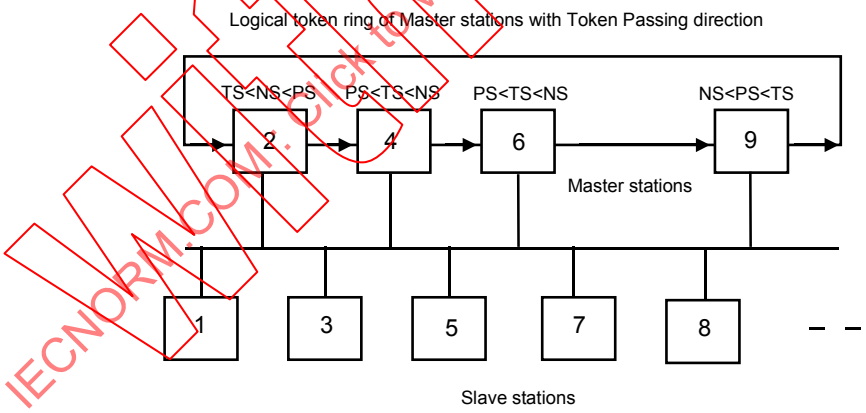
The token is passed from master station to master station in ascending numerical order of station addresses by means of the token DLPDU (see 7.4). To close the logical token ring, the station with the highest address passes the token to the station with the lowest address, see Figure 2.

5.3.2.2 Token reception

If a master station (TS) receives a token DLPDU addressed to itself from a station which is registered as Previous station (PS) in its list of master stations (LMS), it **owns the token** and may execute message transfer periods. The LMS is generated in a master station in the "Listen_Token" state (see 8.2.4) after power on and is updated and corrected, if necessary, later on upon each receipt of a token DLPDU.

If the token transmitter is not the registered PS, the addressee assumes an error and shall not accept the token. Only a subsequent retry of the same PS is accepted and results in the token receipt, because the token receiver shall assume now that the logical ring has changed. It replaces the originally recorded PS in its LMS by the new one.

Figure 2 shows the logical token-passing ring.



where

TS is this station (address);

PS is previous station (address);

NS is next station (address);

master stations are 2, 4, 6 and 9 (addresses as an example);

slave stations are 1, 3, 5, 7 and 8 (addresses as an example).

Figure 2 – Logical token-passing ring

5.3.2.3 Token transmission

After the master station has finished its message transfer periods - contingent maintenance of the GAP station list (GAPL, see 5.3.2.4) included - it passes the token to its successor (NS) by transmitting the token DLPDU. The functionality of its transceiver is checked by simultaneous monitoring (see 8.2.11, "Pass_Token" state).

If, after transmitting the token DLPDU and after expiration of the synchronization time within the slot time (see 5.5), the token transmitter receives a valid DLPDU, that is, a DLPDU header without any errors, it assumes that its NS owns the token and executes message transfer periods. If the token transmitter receives an invalid DLPDU, it assumes that another master station is transmitting. In both cases it ceases monitoring the token passing and retires, that is, it enters the "Active_Idle" state (see 8.2.5).

If the token transmitter does not recognize any bus activity within the slot time, it repeats the token DLPDU and waits another slot time. It retires thereafter, if it recognizes bus activity within the second slot time. Otherwise, it repeats the token DLPDU to its NS for a last time. If, after this second retry, it recognizes bus activity within the slot time, it retires.

If, after the second retry, there is no bus activity, the token transmitter tries to pass the token to the next but one master station (NS of NS). It continues repeating this procedure until it has found a successor from its LMS. If it does not succeed, the token transmitter assumes that it is the only one left in the logical token ring and transmits the token to itself. If it finds a NS again in a later station registration, it tries again to pass the token.

5.3.2.4 Addition and removal of stations

Master and slave stations may be connected to or disconnected from the transmission medium at any moment. Each master station in the logical token ring is responsible for the addition of new stations in the range from the own station address (TS) to the next station (NS), excluding TS and NS. This address range is called GAP and is represented in the GAP List (GAPL), except the address range between Highest Station Address (HSA: 2 to 126) and 127, which does not belong to the GAP.

Each master station in the logical token ring examines its address range (all GAP addresses) after expiration of the GAP update timer (see 5.5.3.11) for changes concerning Master and slave stations. This is accomplished by examining **one** address per token receipt, using the "Request DL Status with Reply" request DLPDU (see Table 3, b7=1, Code-No 9: Format 7.1.1 a) and 7.1.2 a)).

Upon receiving the token, GAP maintenance starts immediately after all queued message transfer periods have been executed, if there is still transmission time available (see 5.3.2.6). Otherwise, GAP maintenance starts upon the next or the consecutive token receipts after the high priority message transfer periods and other low priority services invoked before the previous GAP maintenance have been performed (see 5.3.2.7). In realizations, care is necessary to ensure that GAP maintenance and low priority message transfer periods do not block each other.

GAP addresses are examined in ascending order, except the GAP which surpasses the HSA. For example, if the HSA and address 0 are not used by a master station, the master station with the highest address examines address 0 after checking the HSA.

If a station acknowledges positively with the DL status "Master station not ready to enter logical token ring" or "Slave station" (see Table 3, b7=0, Code-No 0, no SC, and Figure 20), it is accordingly marked in the GAPL and the next address is checked. If a station answers with the state "Master station ready to enter logical token ring", the token holder changes its GAPL and LMS and passes the token to the new NS. This station, which has newly been admitted to the logical token ring, has already built up its LMS, when it was in the "Listen_Token" state, so that it is able to determine its GAP range and its NS.

If a station answers with the DL status "Master station in logical token ring", then the token holder does not change its GAP and passes the token to the NS given in the LMS. Thus the skipped master station may retire from the bus in case of permanent not receiving the token. In this case the master station shall enter the "Listen_Token" state. In this state it generates a new LMS and remains in this state until it is addressed once more by a "Request DL Status with Reply" transmitted by its predecessor (PS).

Stations which were registered in the GAPL and which do not respond to a "Request DL Status with Reply" are removed from the GAPL and are recorded as unused station addresses.

Due to performance requirements repeated requests of "Request DL Status with Reply" are not desired.

5.3.2.5 (Re)initialization of the logical token ring

Initialization is primarily a special case of updating the LMS and GAPL. If after power on (PON) of a master station a time-out is encountered in the "Listen_Token" state (no bus activity within Time-out Time T_{TO} (see 5.5)), then the master station shall claim the token ("Claim_Token" state) and start initialization.

When the entire fieldbus system is started, the master station with the lowest station address starts initialization. By transmitting two token DLPDUs addressed to itself (DA = SA = TS) it informs any other master stations (entering a NS into their LMS) that it is now the only station in the logical token ring. Then it transmits a "Request DL Status with Reply" DLPDU to each station in an incrementing address sequence, in order to register other stations. If a station responds with "Master station not ready to enter logical token ring" or with "Slave station" it is entered in the GAPL. The first master station, which answers with "Master station ready to enter logical token ring", is registered as NS in the LMS and thus closes the GAP range of the token holder. Then the token holder passes the token to its NS.

Reinitialization becomes necessary after loss of the token, which causes a time-out (no bus activity). In this case, an entire bus initialization sequence is not required, because LMS and GAPL already exist in the master stations. The time-out expires first in the master station with the lowest address. It takes the token and starts executing regular message transfer periods or passes the token to its NS.

5.3.2.6 Token rotation time

After receiving a token, the DL-entity may carry out high priority and low priority message transfer periods according to the token rotation time. The token rotation time is measured by using the token-rotation-timer. At the beginning the token-rotation-timer is loaded with the target rotation time T_{TR} and decremented with each bit time. The token-rotation-timer stops if the value zero is reached.

The token rotation time shall be measured according to the following rules:

— immediately after token reception the master station shall read the current value of the token-rotation-timer (token holding time T_{TH}) to calculate the real rotation time T_{RR} (difference between the target rotation time and the current value of token-rotation-timer) and shall start the token-rotation-timer with the value target rotation time (T_{TR})

— the T_{RR} represents always the token rotation time of the last token rotation (from the viewpoint of this station)

A system's minimum target rotation time depends on the number of master stations and thus on the token transfer period (T_{TP}) and the duration of high priority message transfer periods (high T_{MP}). The predefined target rotation time T_{TR} shall also contain sufficient time for low priority message transfer periods and a safety margin for potential retries.

In order to keep within the system reaction time required by the field of application, the target rotation time T_{TR} of the token in the logical ring shall be specified.

The system reaction time is defined as the maximum time interval (worst case) between two consecutive high priority message transfer periods of a master station, measured at the DLS-user interface at maximum bus load.

In order to achieve a target rotation time as short as possible, it is recommended for the DLS-user (see IEC 61158-5-3), to declare only important data as high priority message transfer periods and to restrict their length (for example, ≤ 32 octets for the DLSDU, see 6.6).

If the transfer periods defined in 5.6 (equations (52) and (59) as well as (53) and (60)) are included and possible retries are taken into consideration, the operating parameter "target rotation time T_{TR}" (see 5.6 and Table 5), which is necessary for initialization (min T_{TR}), is calculated for the DLS-user as follows:

$$\min T_{TR} = na \times T_{TP} + (na + 1) \times \text{high } T_{MP} + k \times \text{low } T_{MP} + mt \times T_{RMP} \quad (11)$$

where

- na is the number of master stations;
- k is the estimated number of low priority message transfer periods per token rotation;
- T_{TP} is the token transfer period (see 5.6.1.1 and 5.6.2.1);
- T_{MP} is the message transfer period, depending on DLPDU length (see 5.6.1.2 and 5.6.2.2);
- mt is the numbers of message retry transfer periods per token rotation;
- T_{RMP} is the message retry transfer period.

The first term contains one token transfer period per master station. The second term contains **one** high priority message transfer period for *N+1* master stations. The third term contains the estimated number of low priority message transfer periods per token rotation. The fourth term serves as a safety margin for potential retries. The maximum reaction time for high priority message transfer periods is guaranteed for all bus loads.

5.3.2.7 Message priorities

In the parameter service_class of the DL services the DLS-user (application layer) can choose between two priorities: low and high. The priority is passed to the DLE with the service request.

After token reception, each master station may always execute **one** high priority message transfer period including retries in the case of an error independently of the token holding time T_{TH}.

Further high or low priority message transfer periods may be executed according to the following rules if T_{TH} is still available (see 5.5.5).

- High or low priority messages may be carried out if the calculated real rotation time (T_{RR}) is less than the current value of the token-rotation-timer before the instant of execution of message sending.
- Once a high or low priority message transfer period is started, it is always completed, including any required retry (retries), even if the token-rotation-timer is less than or equal to the value of T_{RR} during the execution.

— If there is no T_{TH} available (the T_{RR} is greater than the current value of the token-rotation-timer before the instant of execution of message sending) then the token shall be passed to NS immediately.

NOTE 1 The prolongation of the token holding time T_{TH} automatically results in a shortening of transmission time for message transfer periods at the next token receipt.

NOTE 2 For further explanation refer to Annex A and Annex C.

5.3.3 Send or send/request mode

In the send or send/request mode, single message transfer periods are conducted sporadically. The master station's DL-entity initiates this mode due to a local user's request upon receipt of the token. If there are several requests, this mode of operation may be continued until the maximum allowed token holding time expires.

5.4 Service assumed from the PhL

5.4.1 Asynchronous transmission

5.4.1.1 PhS transmission and reception services

The PhL data service includes two service primitives. A request primitive is used to request a service by the DL-entity; an indication primitive is used to indicate a reception to the DL-entity. The names of the respective primitives are as follows:

Ph-ASYN-DATA request

Ph-ASYN-DATA indication

The temporal relationship of the primitives is shown in Figure 3.

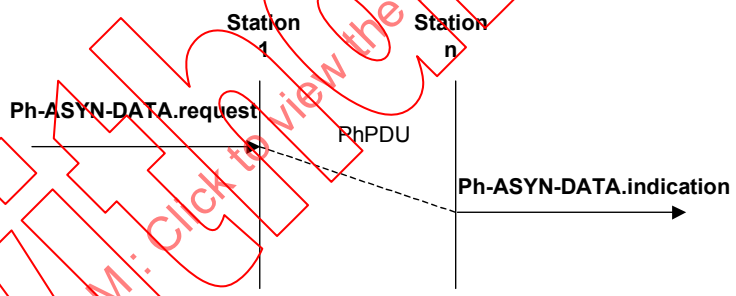


Figure 3 – PhL data service for asynchronous transmission

5.4.1.2 Detailed specification of the service and message exchange

This subclause describes in detail the service primitives and the related parameters in an abstract way. The parameters contain information needed by the PhL-entity.

Parameters of the primitives:

Ph-ASYN-DATA request (DL_symbol)

The parameter DL_symbol shall have one of the following values:

a) ZERO corresponds to a binary "0",

b) ONE corresponds to a binary "1",

c) SILENCE disables the transmitter when no valid DL symbol is to be transmitted.

The Ph-ASYN-DATA request primitive is passed from the DL-entity to the PhL-entity to request that the given symbol shall be sent to the fieldbus medium.

The reception of this primitive shall cause the PhL-entity to attempt encoding and transmission of the DL-symbol.

The Ph-ASYN-DATA request is a primitive, which shall only be generated once per DL-symbol period (t_{BIT}). The PhL-entity may confirm this primitive with a locally defined confirmation primitive.

Ph-ASYN-DATA indication (DL_symbol)

The parameter DL_symbol shall have one of the following values:

— ZERO corresponds to a binary "0",

— ONE corresponds to a binary "1".

The Ph-ASYN-DATA indication primitive is passed from the PhL-entity to the DL-entity to indicate that a DL-symbol was received from the fieldbus medium.

The Ph-ASYN-DATA indication is a primitive, which shall only be generated once per received DL-symbol period (t_{BIT}).

5.4.2 Synchronous transmission

5.4.2.1 General

This subclause defines the assumed physical service (PhS) primitives and their constraints on use by the DLE.

NOTE Proper layering requires that an (N+1)-layer entity not be concerned with, and that an (N)-service interface not overly constrain, the means by which an (N)-layer provides its (N)-services. Thus the Ph-service interface does not require DLEs to be aware of internal details of the PhE (for example, preamble, postamble and DLPDU delimiter signal patterns, number of bits per baud), and should not prevent the PhE from using appropriate evolving technologies.

5.4.2.2 Assumed primitives of the PhS

The granularity of transmission in the fieldbus protocol is one octet. This is the granularity of PhS-user data and timing information exchanged at the PhL – DLL interface.

5.4.2.2.1 PhS characteristics reporting service

The PhS is assumed to provide the following service primitive to report essential PhS characteristics used in DLL transmission, reception, and scheduling activities:

Ph-CHARACTERISTICS indication (minimum-data-rate, framing-overhead)

where

minimum-data-rate – specifies the effective minimum rate of data conveyance in bits per second, including any timing tolerances, of any PhE on the local link.

NOTE 1 A PhE with a nominal data rate of 1 Mbit/s ± 0,01 % would specify a minimum data rate of 0,9999 Mbit/s.

framing-overhead – specifies the maximum number of bit periods, where
period = 1/data rate,
used in any transmission for PhPDUs which do not directly convey data (for example, PhPDUs conveying preamble, DLPDU delimiters, postamble, inter-DLPDU “silence”, and so on).

NOTE 2 If the framing overhead is F and two DL message lengths are L_1 and L_2, then the time to send two immediately consecutive messages of lengths L_1 and L_2 will be at least as great as the time required to send one message of length $L_1 + F + L_2$.

If this framing-overhead is more than the DLEs configured per-DLPDU-PhL-overhead, V(PhLO), then the DLE shall report this discrepancy to DL-management and shall not issue Ph-DATA requests while the discrepancy exists.

NOTE 3 This restriction prohibits DLE transmission while this discrepancy exists. The DLEs local station management may remedy this discrepancy by reconfiguring V(PhLO) to a greater value.

5.4.2.2.2 PhS transmission and reception services

The PhS is assumed to provide the following service primitives for transmission and reception:

Ph-DATA request (class, data);

Ph-DATA indication (class, data);

Ph-DATA confirm (status)

where

class – specifies the Ph-interface-control-information (PhICI) component of the Ph-interface-data-unit (PhIDU). For a Ph-DATA request, its possible values are

START-OF-ACTIVITY – transmission of the PhPDUs which precede Ph-user data should commence;

DATA – the single-octet value of the associated data parameter should be transmitted as part of a continuous correctly-formed transmission;

END-OF-DATA-AND-ACTIVITY – the PhPDUs which terminate Ph-user data should be transmitted after the last preceding octet of Ph-user data, culminating in the cessation of active transmission;

For a Ph-DATA indication, its possible values are:

START-OF-ACTIVITY – reception of an apparent transmission from one or more PhEs has commenced;

DATA – the associated data parameter was received as part of a continuous correctly-formed reception;

END-OF-DATA – the ongoing continuous correctly formed reception of Ph-user data has concluded with correct reception of PhPDUs implying END-OF-DATA;

END-OF-ACTIVITY – the ongoing reception (of an apparent transmission from one or more PhEs) has concluded, with no further evidence of PhE transmission;

END-OF-DATA-AND-ACTIVITY – simultaneous occurrence of END-OF-DATA and END-OF-ACTIVITY;

data – specifies the Ph-interface-data (PhID) component of the PhIDU. It consists of one octet of Ph-user data to be transmitted (Ph-DATA request) or which was received successfully (Ph-DATA indication).

status – specifies either success or the locally detected reason for inferring failure.

The Ph-DATA confirm primitive provides the critical physical timing feedback necessary to inhibit the DLE from starting a second transmission before the first is complete. The final Ph-DATA confirm of a transmission shall not be issued until the PhE has completed the current transmission.

5.4.2.3 Notification of PhS characteristics

The PhE has the responsibility for notifying the DLE of those characteristics of the PhS which are relevant to DLE operation. This notification is accomplished by the PhE by issuing a single Ph-CHARACTERISTICS indication primitive at each of the PhEs PhSAPs at PhE startup.

5.4.2.4 Transmission of Ph-user data

The PhE determines the timing of all transmissions. When a DLE has a DLPDU to transmit, and the DL-protocol gives that DLE the right to transmit, then the DLE shall send the DLPDU, including a concatenated FCS, by making a well-formed sequence of Ph-DATA requests, consisting of a single request specifying START-OF-ACTIVITY; followed by three to 300 consecutive requests, inclusive, specifying DATA; and concluded by a single request specifying END-OF-DATA-AND-ACTIVITY.

The PhE signals its completion of each Ph-DATA request, and its readiness to accept a new Ph-DATA request, with a Ph-DATA confirm primitive; the status parameter of the Ph-DATA confirm primitive conveys the success or failure of the associated Ph-DATA request. A second Ph-DATA request should not be issued until after the Ph-DATA confirm corresponding to the first request has been received from the PhE.

5.4.2.5 Reception of Ph-user data

The PhE reports a received transmission with a well-formed sequence of Ph-DATA indications, which shall consist of either

a) a single indication specifying START-OF-ACTIVITY; followed by consecutive indications specifying DATA; followed by a single indication specifying END-OF-DATA; and concluded by a single indication specifying END-OF-ACTIVITY, or

b) a single indication specifying START-OF-ACTIVITY; followed by consecutive indications specifying DATA; followed by a single indication specifying END-OF-DATA-AND-ACTIVITY, or

c) a single indication specifying START-OF-ACTIVITY; optionally followed by one or more consecutive indications specifying DATA; and concluded by a single indication specifying END-OF-ACTIVITY.

NOTE This last sequence is indicative of an incomplete or incorrect reception. Detection of an error in the sequence of received PhPDUs, or in the PhEs reception process, disables further Ph-DATA indications with a class parameter specifying DATA, END-OF-DATA, or END-OF-DATA-AND-ACTIVITY until after both the end of the current period of activity and the start of a subsequent period of activity have been reported by Ph-DATA indications specifying END-OF-ACTIVITY and START-OF-ACTIVITY, respectively.

In the first two cases, the DLE concatenates the received data and then attempts to parse it into a DLPDU followed by a concatenated FCS. In the last case the DLE discards all reported data and reports the event to DL-management.

5.5 Operational elements

5.5.1 Overview

The following times T are measured in bits. A time t in seconds (s) shall therefore be divided by the bit time t_{BIT}.

5.5.2 Bit time t_{BIT}

The bit time t_{BIT} is the time, which elapses during the transmission of one bit. It is equivalent to the reciprocal value of the data rate:

$$t_{BIT} = \frac{1}{\text{data rate}} \ [\text{s bit}^{-1}] \quad (12)$$

5.5.3 Asynchronous transmission

5.5.3.1 Synchronization time (T_{SYN})

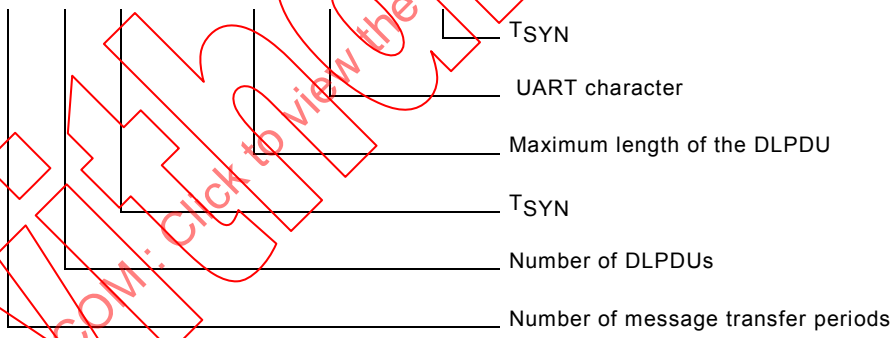
The synchronization time T_{SYN} is the minimum time interval during which each station shall receive idle state (idle = binary "1") from the transmission medium before it may accept the beginning of a send/request DLPDU or token DLPDU. The value of the synchronization time shall be set with:

$$T_{SYN} = 33 \text{ bit} \quad (13)$$

5.5.3.2 Synchronization interval time (T_{SYNI})

The synchronization interval time T_{SYNI} is the maximum allowed time interval between two consecutive synchronization times (T_{SYN}), in order to detect “permanent transmitters”. The value of the synchronization interval time shall be set with:

$$T_{SYNI} = 2 \times (2 \times (33 \text{ bit} + 255 \times 11 \text{ bit})) + 33 \text{ bit} = 11\,385 \text{ bit} \quad (14)$$



This value regards two complete message transfer periods, each of which consists of two DLPDUs of maximum length and the related synchronization times. A transmission disturbance is permitted in one of those synchronization times.

5.5.3.3 Station delay time (T_{SDx})

The station delay time T_{SDx} is the period of time which may elapse between the transmission or receipt of a DLPDUs last bit until the transmission or receipt of a following DLPDUs first bit (with respect to the transmission medium, that is, including line receiver and transmitter). The following three station delays are defined:

a) Station Delay of Initiator (station transmitting request or token DLPDU):

   TSDI

b) Minimum Station Delay of Responders (stations which acknowledge or respond):

   min T_{SDR}

c) Maximum Station Delay of Responders:

   max T_{SDR}

5.5.3.4 Quiet time (T_{QUI})

When transposing the NRZ signals into a different signal coding, the transmitter fall time after switching off the transmitter (at the initiator) shall be taken into account if it is greater than T_{SDR}.

During this quiet time T_{QUI}, transmission and receipt of DLPDUs shall be disabled. This shall also be taken into account when using self-controlled repeaters, whose switching time shall be taken into consideration. The implementation shall ensure, that the following condition is fulfilled:

$$T_{QUI} < \min T_{SDR} \quad (15)$$

In order to fulfil this condition, it may be necessary to prolong min T_{SDR}.

5.5.3.5 Ready time (T_{RDY})

The ready time T_{RDY} is the time within which a master station shall be ready to receive an acknowledgement or response after transmitting a request. The implementation shall ensure, that the following condition is fulfilled:

$$T_{RDY} < \min T_{SDR} \quad (16)$$

In order to fulfil this condition it may be necessary to prolong min T_{SDR}.

When transposing NRZ signals into a different signal coding, the quiet time shall also be taken into consideration when switching off the transmitter. The receiver shall not be enabled before this time:

$$T_{QUI} < T_{RDY} \quad (17)$$

In order to fulfil this condition, it may be necessary to prolong T_{RDY} and thus min T_{SDR} accordingly.

5.5.3.6 Safety margin (T_{SM})

The following time interval is specified as safety margin T_{SM}:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} + T_{QUI} \quad (18)$$

T_{SET} is the set-up time, which expires from the occurrence of an event (for example, an interrupt on the last bit of a sent DLPDU or when synchronization time expires) until the necessary reaction is performed (for example, to start synchronization time or to enable the receiver).

5.5.3.7 Idle time (T_{IDx})

The idle time T_{IDx} is the time which expires at the initiator either after receipt of a DLPDUs last bit (measured at the line receiver) as idle = binary "1" **on the transmission medium**, until a new DLPDUs first bit is transmitted on the medium (including line transmitter) or between transmitting the last bit of a DLPDU which is not to be acknowledged and transmitting the first bit of the next DLPDU. The idle time shall be at least the synchronization time plus the safety margin T_{SM} (see Figure 4, Figure 5 and Figure 6 case a)).

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (19)$$

At high data rates (see Figure 4, case b) and c) and Figure 5 case b)) the synchronization time is very short, hence the station delays become significant and shall be taken into consideration.

Two idle times are distinguished. After an acknowledgement, response or token DLPDU the idle time shall be calculated as follows:

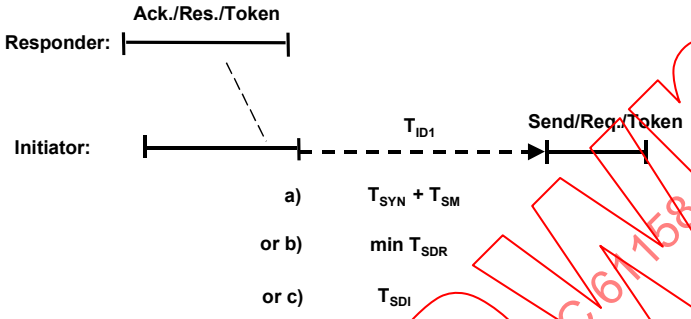


Figure 4 – Idle time T_{ID1}

$$T_{ID1} = \max((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (20)$$

Care shall be taken that the time to update the LMS is not greater than T_{ID1}. This can be accomplished by prolonging T_{SET} or min T_{SDR}. If the necessary prolongation cannot be reached with the range of value of T_{SET}, than the min T_{SDR} shall be made longer.

After a send DLPDU which is **not** to be acknowledged (SDN or CS), the idle time shall be calculated as shown in Figure 5 and equation 34.

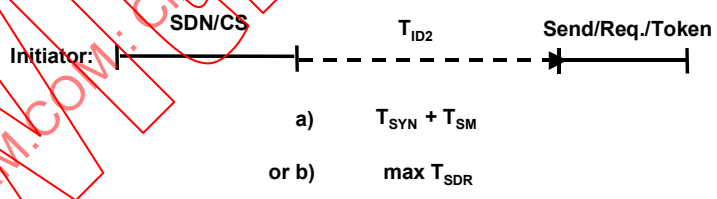


Figure 5 – Idle time T_{ID2} (SDN, CS)

$$T_{ID2} = \max((T_{SYN} + T_{SM}), (\max T_{SDR})) \quad (21)$$

Also, after a response DLPDU (MSRD) the idle time shall be calculated as shown in Figure 6 and equation 34.

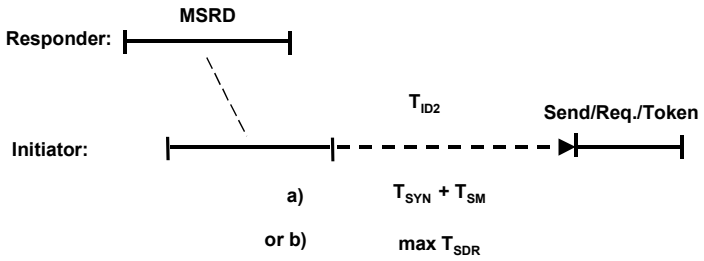


Figure 6 – Idle time T_{ID2} (MSRD)

5.5.3.8 Transmission delay time (T_{TD})

The transmission delay time T_{TD} is the maximum time, which elapses on the transmission medium between transmitter and receiver when a DLPDU is transmitted. When computing it, delay times of repeaters shall be considered, if necessary.

EXAMPLE for computing the transmission delay time: given a line length of 200 m without repeaters, t_{TD} is approximately 1 µs and thus at 500 kbit/s

T_{TD} = (1µs × 500 kbit/s) = 0,5 bit.

5.5.3.9 Slot time (T_{SL})

The slot time T_{SL} is the maximum time the initiator shall wait for the complete receipt of the first character (see 6.1.1, UART character, 1 UC = 11 bits) of the immediate acknowledgement or response DLPDU, after transmitting the last bit of a send/request DLPDU (including the line transmitter) (see Figure 7). Furthermore, T_{SL} is the maximum time the initiator waits for the token receiver's first UART character (1.UC) after transmitting a token DLPDU (see Figure 8). Theoretically, two slot times are distinguished. After a send/request DLPDU the slot time shall be calculated as follows:

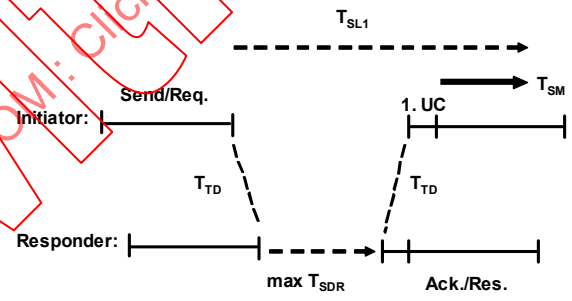


Figure 7 – Slot time T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + 11 \text{ bit} + T_{SM} \quad (22)$$

After a token DLPDU the slot time shall be calculated as follows:

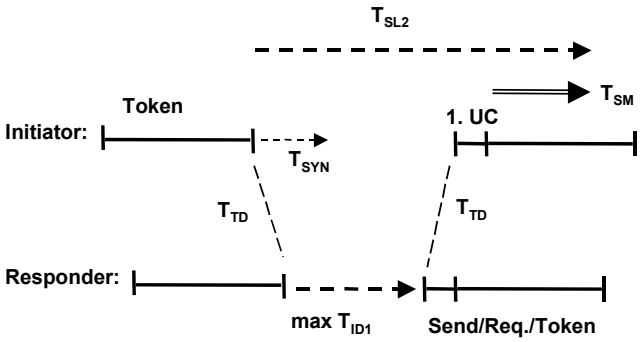


Figure 8 – Slot time T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + 11 \text{ bit} + T_{SM} \quad (23)$$

In order to simplify the realization, only **one** slot time, the longer one, may be used in the system. This does not influence the system reaction time negatively, as the slot time is a pure monitoring time.

$$T_{SL} = \max (T_{SL1}, T_{SL2}) \quad (24)$$

5.5.3.10 Time-out time (T_{TO})

The time-out time T_{TO} serves to monitor the Master and slave stations' bus activity and idle time. Monitoring shall start either immediately after PON, in the "Listen_Token" or "Passive_Idle" state, or later after receiving the last bit of a DLPDU. It shall end after receiving the first bit of the following DLPDU. If the phase of no bus activity times out, the bus shall be regarded as inactive. (This is an error case, for example, due to a lost token DLPDU.) The time-out interval shall be set to:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (25)$$

For master stations: n = station address (0 to 126)

For slave stations: n = 130, independent of the station's address

The first term ensures that there is sufficient difference to the maximum permissible idle time between two DLPDUs. The second term ensures that not all master stations claim the token at the same moment after an error has occurred.

5.5.3.11 GAP update time (T_{GUD})

The GAP update time T_{GUD} serves for initializing GAP maintenance by the master station. After the first generation of the GAPL, update of the GAP image is cyclically initialized after every interval T_{GUD}. This initialization takes place at the next possible token receipt, if there is still token holding time available after the regular DLPDU transfer periods, or during later token holding phases. The GAP update time is a multiple of the target rotation time T_{TR} and shall be set to:

$$T_{GUD} = G \times T_{TR} \quad (\text{where } 1 \leq G \leq 100) \quad (26)$$

T_{TR} is the target rotation time (see 5.3.2.6).

5.5.3.12 Isochronous mode

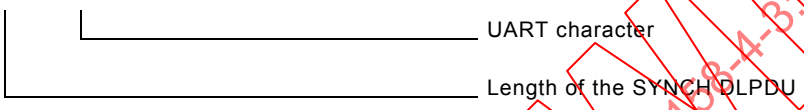
5.5.3.12.1 Isochronous cycle time (T_{CT})

The isochronous cycle time T_{CT} serves to supervise the isochronous cycle. The first cycle starts by sending the SYNCH DLPDU after reception of the token DLPDU.

5.5.3.12.2 IsoM synchronization DLPDU time (T_{SYNCH})

The isochronous mode synchronization DLPDU time describes the time, which is needed to send the SYNCH DLPDU at the start of a new IsoM cycle. The value of the IsoM synchronization message time shall be set with:

$$T_{SYNCH} = 13 \times 11 \text{ bit} = 143 \text{ bit} \quad (27)$$



5.5.3.12.3 Active spare time DLPDU time (T_{ASM})

The active spare time DLPDU time describes the duration that is needed to send an active spare time (ASP) DLPDU.

$$T_{ASM} = T_{ID1} + (6 \times 11 \text{ bit}) \quad (28)$$

5.5.3.12.4 Real isochronous cycle time (T_{RCT})

The real isochronous cycle time is the read value of the isochronous-cycle-timer.

5.5.3.12.5 Spare time (T_{RES})

Within the isochronous cycle the token is passed by the IsoM Master after the transmission of all high priority messages and the number of configured low priority messages. After receiving the next token addressed to the IsoM Master, the station read the isochronous cycle timer (T_{RCT}) in order to calculate the remaining time before the next SYNCH DLPDU is to be sent.

$$T_{RES} = T_{CT} - T_{RCT} \quad (29)$$

Within the spare time, at least one active spare time DLPDU shall be sent. For further active spare time, DLPDUs shall be valid.

$$T_{RES} > T_{ASM} + T_{ID1} \quad (30)$$

5.5.3.12.6 Passive spare time (T_{PSP})

The passive spare time denotes the part of isochronous cycle where the IsoM Master shows no bus activities. It shall be less than the minimum time-out time T_{TO} under consideration of possible delays in the stations and during transmission. The passive spare time is defined as follow:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (31)$$

$$T_{PSP} > T_{ID1} + T_{ASM} + \max T_{SDR} \quad (32)$$

5.5.3.12.7 Time shift (T_{SH})

The time shift is the difference between the measured cycle time and the calculated cycle time when the passive-spare-timer expires.

$$T_{SH} = T_{RCT} - T_{CT} \quad (33)$$

5.5.3.13 Send delay time (T_{SD})

The send delay time is the time that elapses between the receiving of a DL-CS-TIME-EVENT.request primitive at the DLE of the time master and the DLPDUs last bit of a resulting TE DLPDU (see 7.7) transmitted by the time master.

5.5.3.14 Receive delay time (T_{RD})

The receive delay time is the time that elapses at a time receiver between the receiving of the last bit of a TE DLPDU (see 7.7) and the receiving of a DLPDUs last bit of a CV DLPDU (see 7.8).

5.5.3.15 Clock synchronization interval time (T_{CSI})

The clock synchronization interval time is used to monitor the synchronization sequences within the DLE.

5.5.4 Synchronous transmission

5.5.4.1 Synchronization time (T_{SYN})

The synchronization time is the minimum time interval during which each station shall receive no activity from the transmission medium before it may accept the beginning of a send/request DLPDU or token DLPDU.

The synchronization time shall correspond to the post-transmission gap time (T_{PTG}) that is defined in 9.2.8 of IEC 61158-2. Its value shall be set to at least 4 bit and may be increased by the DLMS-user up to 32 bit.

$$T_{SYN} = T_{PTG} = T_{QUI} \quad (34)$$

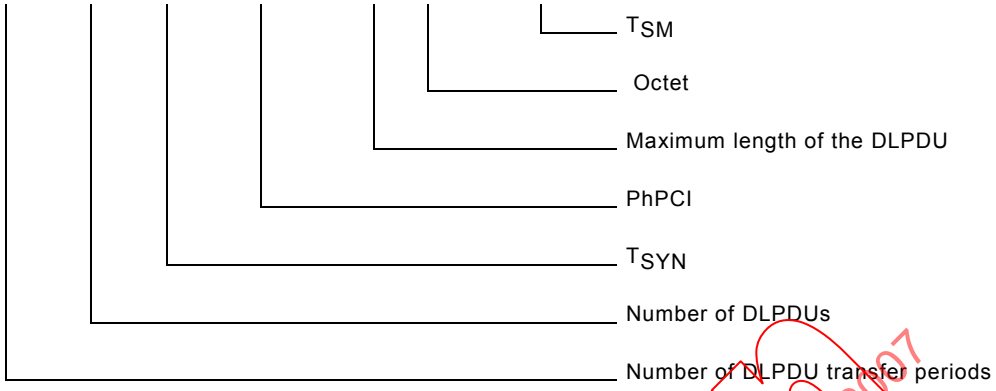
$$T_{SYN} = 4 \text{ to } 32 \text{ bit} \quad (35)$$

5.5.4.2 Synchronization interval time (T_{SYNI})

The synchronization interval time T_{SYNI} is the maximum allowed time interval between two consecutive synchronization times (T_{SYN}), in order to detect "permanent transmitters".

The value of T_{SYNI} shall be set as follows:

$$T_{SYNI} = 2 \times (2 \times (32 \text{ bit} + 80 \text{ bit} + 255 \times 8 \text{ bit})) + 64 \text{ bit} = 8\,672 \text{ bit} \quad (36)$$



This value regards two complete message sequences, each of which consists of two DLPDUs of maximum length and the associated maximum PhL Protocol Control Information (PhPCI: Preamble, Start Delimiter, End Delimiter) and the maximum synchronization time (post-transmission gap). A transmission disturbance is permitted in one of those synchronization times.

5.5.4.3 Station delay time (T_{SDx})

The station delay time T_{SDx} is the period of time, which may elapse between the transmission or receipt of a DLPDUs last octet until the transmission or receipt of a following DLPDUs first octet (with respect to the transmission medium, that is, including line receiver and transmitter). The following three station delays are defined:

a) Station Delay of Initiator (station transmitting request or token DLPDU)

T_{SDI}

b) Minimum Station Delay of Responders (station that acknowledges or responds)

min T_{SDR}

c) Maximum Station Delay of Responders

max T_{SDR}

5.5.4.4 Quiet time (T_{QUI})

The transmitter fall time or repeater switch time corresponds to the post-transmission gap time (T_{SYN}). The following shall apply:

$$T_{QUI} = T_{PTG} = T_{SYN} \quad (37)$$

5.5.4.5 Ready time (T_{RDY})

The ready time T_{RDY} is the time within which a Master station shall be ready to receive an acknowledgement or response after transmitting a request. The implementation shall ensure, that the following condition is fulfilled:

$$T_{RDY} < \min T_{SDR} \quad (38)$$

In order to fulfil this condition it may be necessary to prolong min T_{SDR}.

During the quiet time T_{QUI}, transmission and receipt of DLPDUs shall be disabled. The implementation shall ensure, that the following condition is fulfilled:

$$T_{QUI} = T_{PTG} = T_{SYN} < T_{RDY} \quad (39)$$

In order to fulfil this condition the min T_{SDR} shall be increased according to equation (38) if necessary.

5.5.4.6 Safety margin (T_{SM})

The safety margin T_{SM} is defined as the time interval:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} \quad (40)$$

T_{SET} is the set-up time, which expires from the occurrence of an event (for example, an interrupt on the last octet of a sent DLPDU or on synchronization time expiration) until the execution of the necessary reaction.

5.5.4.7 Idle time (T_{IDx})

The idle time T_{IDx} is the time which expires at the initiator after a Ph-DATA indication primitive until sending of a new DLPDU with Ph-DATA request primitive or after passing a Ph-DATA request primitive with a Ph-DATA confirm primitive to transmit a DLPDU which is not to be acknowledged until passing a new Ph-DATA request primitive for transmitting the next DLPDU. The idle time shall be at least the synchronization time plus the safety margin T_{SM}.

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (41)$$

Two idle times are distinguished (see description of idle time in 5.5.3.7, Figure 4, Figure 5, and Figure 6). After an acknowledgement, response or token DLPDU the idle time shall be calculated as follows:

$$T_{ID1} = \max ((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (42)$$

Care shall be taken that the time to update the LMS is not greater than T_{ID1}. This can be accomplished by prolonging T_{SET} or min T_{SDR}. If the necessary prolongation cannot be reached with the range of value of T_{SET}, then the min T_{SDR} shall be made longer.

After a send DLPDU, which is not to be acknowledged (SDN, CS), the idle time shall be calculated as follows:

$$T_{ID2} = \max ((T_{SYN} + T_{SM}), (\max T_{SDR})) \quad (43)$$

5.5.4.8 Transmission delay time (T_{TD})

The transmission delay time T_{TD} is the maximum time, which elapses on the transmission medium between transmitter and receiver when a DLPDU is transmitted. When computing it, delay times of repeaters shall be considered if necessary. It shall also conform to any restrictions placed on it in IEC 61158-2, where it is called “propagation delay”.

5.5.4.9 Slot time (T_{SL})

The slot time T_{SL} is the maximum time the initiator shall wait after passing a Ph-DATA request primitive for transmitting a request DLPDU from the Ph-DATA confirm primitive until receiving

the first Ph-DATA indication primitive as an indication of receiving the immediate acknowledgement or response (see Figure 9). Furthermore, T_{SL} is the maximum time the initiator shall wait for a Ph-DATA indication primitive after the token DLPDU as reaction to receiving the first DLPDU octet from the token receiver. Theoretically, two slot times are distinguished (see Figure 10). After a send/request DLPDU the following slot time shall be calculated:

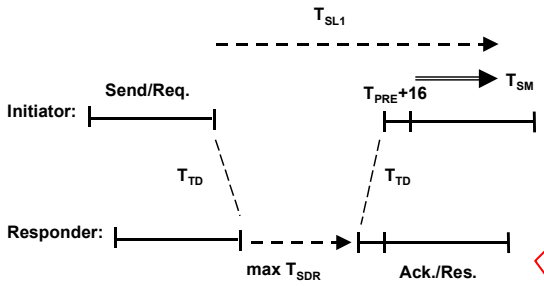


Figure 9 – Slot time T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (44)$$

After a token DLPDU the following slot time shall be calculated:

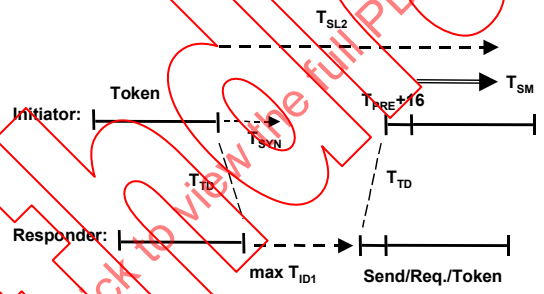


Figure 10 – Slot time T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (45)$$

where T_{PRE} is Preamble period (see IEC 61158-2)

In order to simplify the realization, only the longer slot time may be used in the system. This does not influence the system reaction time negatively, as the slot time is merely a monitoring time.

$$T_{SL} = \max (T_{SL1}, T_{SL2}) \quad (46)$$

5.5.4.10 Time-out time (T_{TO})

The time-out time T_{TO} serves to monitor the Master and Slave stations' bus activity and idle time. Monitoring shall start either immediately after PON, in the "Listen_Token" or "Passive_Idle" state, or later after the reception of a Ph-DATA indication primitive. It shall end upon receipt of a Ph-DATA indication primitive for reception of the first octet of a following DLPDU. If the phase of no bus activity times out, the bus shall be regarded as inactive. (This is an error case, for example, due to a lost token DLPDU.). The time-out time shall be set to the following value:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (47)$$

For Master stations: n = station address (0 to 126)

For Slave stations: n = 130, independent of their station address

5.5.4.11 GAP update time (T_{GUD})

5.5.3.11 shall apply.

5.5.4.12 Isochronous Mode

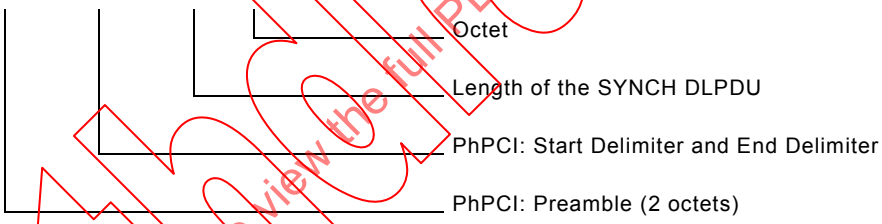
5.5.4.12.1 Isochronous cycle time (T_{CT})

5.5.3.12.1 shall apply.

5.5.4.12.2 IsoM synchronization DLPDU time (T_{SYNCH})

The isochronous mode synchronization DLPDU time describes the time, which is needed in order to send the SYNCH DLPDU at the start of a new IsoM cycle. The value of the IsoM synchronization message time shall be set with:

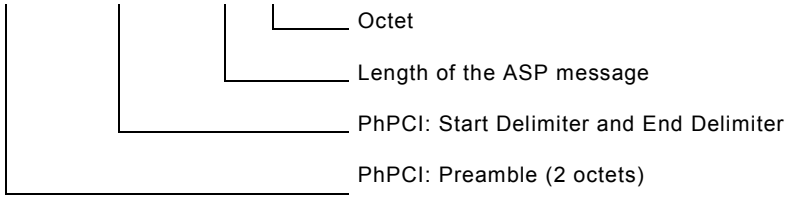
$$T_{SYNCH} = 16 \text{ bit} + 16 \text{ bit} + 13 \times 8 \text{ bit} = 136 \text{ bit} \quad (48)$$



5.5.4.12.3 Active spare time DLPDU time (T_{ASM})

The active spare time DLPDU time describes the duration that is needed to send an active spare time (ASP) DLPDU.

$$T_{ASM} = T_{PTG} + 16 \text{ bit} + 16 \text{ bit} + (6 \times 8 \text{ bit}) \quad (49)$$



5.5.4.12.4 Real isochronous cycle time (T_{RCT})

5.5.3.12.4 shall apply.

5.5.4.12.5 Spare time (T_{RES})

5.5.3.12.5 shall apply.

5.5.4.12.6 Passive spare time (T_{PSP})

The passive spare time denotes the part of isochronous cycle where the IsoM Master shows no bus activities. It shall be less than the minimum time-out time T_{TO} under consideration of possible delays in the stations and during transmission. The passive spare time is defined as follow:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (50)$$

$$T_{PSP} > T_{ID1} + T_{ASM} + \max T_{SDR} \quad (51)$$

5.5.4.12.7 Time shift (T_{SH})

5.5.3.12.7 shall apply.

5.5.4.13 Send delay time (T_{SD})

The send delay time is the time that elapses between a passed DL-CS-TIME-EVENT request primitive to the DLE of the time master and the DLPDUs last octet of a resulting TE DLPDU (see 7.7) transmitted by the time master.

5.5.4.14 Receive delay time (T_{RD})

The receive delay time is the time that elapses by a time receiver between the receiving of a DLPDUs last octet of a TE DLPDU (see 7.7) and the last octet of a CV DLPDU (see 7.8).

5.5.4.15 Clock synchronization interval time (T_{CSI})

Subclause 5.5.3.15 shall apply.

5.5.5 Timers and counters

5.5.5.1 Asynchronous transmission

5.5.5.1.1 Timers

In order to measure the token rotation time and to realize the supervisory times the following timers shall be implemented:

token-rotation-timer, idle-timer, slot-timer, time-out-timer, syn-interval-timer, GAP-update-timer, isochronous-cycle-timer, passive-spare-timer, send-delay-timer and receive-delay-timer.

token-rotation-timer: When a Master station receives the token, this timer is loaded with the target rotation time T_{TR} and decremented each bit time. When the station again receives the token, the timer value, the remaining time or token holding time T_{TH}, is read and the timer reloaded with T_{TR}. The real rotation time T_{RR} results from the difference T_{TR} - T_{TH}.

The token-rotation-timer can be read out at every moment, representing always the value of the actual token rotation. Low priority message transfer periods may be processed if at the instant of processing the real token rotation time is less than the value of the actual token rotation.

idle-timer: This timer monitors the idle state (binary "1"), the synchronization time, immediately on the bus line. The synchronization time preceding each request is necessary for unambiguous receiver synchronization. The idle-timer of Slave stations and Master stations "without token" is loaded with T_{SYN} after the transmission or receipt of a DLPDUs

last bit and then decremented each bit time. The receiver shall be enabled immediately after the timer has expired. The timer of a Master station "with token" is loaded according to the data transmission service with T_{ID1} or T_{ID2} (see 5.5.3). A new request or token DLPDU may only be transmitted after expiration of the timer. When the signalling level is binary "0", the timer is always reloaded.

slot-timer: This timer in a Master station monitors after a request or token pass whether the receiving station responds or becomes active within the predefined time T_{SL}, the slot time. After transmission of a DLPDUs last bit this timer is loaded with T_{SL} and decremented each bit time as soon as the receiver is enabled. If the timer expires before a DLPDUs first bit is received, an error has occurred. Then a retry or a new message transfer period shall be initiated.

time-out-timer: This timer monitors bus activity in Master and Slave stations. After the transmission or receipt of a DLPDUs last bit the timer is loaded with a multiple of the slot time (see 5.5.3) and decremented each bit time as long as no new DLPDU is received. If the timer expires, a fatal error has occurred, which for the Master station causes a (re)initialization. The DLMS-user of the Slave and Master station receives a time-out notification.

syn-interval-timer: Master and Slave stations use this timer to monitor the transmission medium whether a receiver synchronizing (T_{SYN}, idle state, idle = binary "1") occurs within T_{SYNI}. Each time the receiver is synchronized, the timer is loaded with T_{SYNI} (see 5.5.3). From the beginning of a DLPDU (first start bit) the timer is decremented each bit time as long as no new T_{SYN} is detected. If the timer expires, an error has occurred on the transmission medium, for example, stuck at "0" or permanent "0" / "1" edges. The DLMS-user is notified accordingly.

GAP-update-timer: Only Master stations need this timer. Its expiration indicates the moment for GAP maintenance. After a complete GAP check, which may last several token rotations (segmented; see 5.3.2.4), the timer is loaded with a multiple (G) of the target rotation time T_{TR} (see 5.3.2.6).

NOTE For ease of implementation, this timer can be implemented as a counter, decremented at each token receipt.

isochronous-cycle-timer: The timer supervises the isochronous cycle. It is loaded with value equal to 0 at the starting point of the isochronous mode (after receiving the first token) and increments every bit time. The timer is readable at any time. The timer is started either by loading with the value equal to 0 at the first beginning of the isochronous cycle or after expiration of the passive-spare-timer. In cases the timer is started too late, it is loaded with the value of the time difference between the real isochronous cycle time of the last cycle and the isochronous cycle time.

passive-spare- timer: This timer monitors the idle time before sending a SYNCH message. It shall be set according to the time difference between T_{CT} and the current value of the isochronous-cycle-timer T_{RCT} at the end of the last ASP message if the difference is greater than 0. The transmission of the last bit of the last ASP message in an isochronous cycle starts this timer. In the isochronous mode after expiration of the passive-spare-timer, a SYNCH message shall be sent and a synch notification event shall be sent to the DLMS-user.

send-delay-timer: When the time master receives a DL-CS-TIME-EVENT request primitive from the local DLS-user, this timer is started with the value = $2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after confirmation of the transmission of the last portion of a TE DLPDU (see 7.7). The value of send delay time is calculated as the difference between the start value ($2 \times T_{CSI}$) and the read send-delay-timer value. The calculated send delay time is passed to the local DLS-user. In case the timer has expired, the DLS-user is notified of a clock synchronization sequence violation.

receive-delay-timer: When the time receiver receives the DLPDUs last bit of a TE DLPDU (see 7.7), this timer is started with the value = 2 × T_{CSI} and decremented each bit time. The timer is stopped after the receiving the last bit of a CV DLPDU (see 7.8). The value of the receive delay time is calculated as difference between the start value (2 × T_{CSI}) and the read receive-delay-timer value. The calculated receive delay time is passed to the local DLS-user. In case the timer expired a clock synchronization sequence violation is notified to the DLS-user.

When a Master station enters the "Listen_Token" state, the idle-timer is loaded with T_{SYN}, the time-out-timer with T_{TO}, the syn-interval-timer with T_{SYNI} and the other timers are cleared. When a Slave station enters the "Passive_Idle" state, the time-out-timer is loaded with T_{TO} and the syn-interval-timer with T_{SYNI}.

5.5.5.1.2 Counters

For installation and maintenance the following pairs of counters (DL-variables) may be used:

For Master stations:

— counter for transmitted DLPDUs (DLPDU_sent_count), except for the SDN and Request DL Status with Reply services

— counter for transmitted DLPDU retries (Retry_count)

Additional counters may be provided for each individual remote station:

— counter for transmitted DLPDUs (DLPDU_sent_count_sr), except for the SDN and Request DL Status with Reply services and for token DLPDUs

— counter for transmitted DLPDUs with no response or erroneous response (Error_count), except for the SDN and Request DL Status with Reply services and for token DLPDUs

For Slave and Master stations:

— counter for received valid start delimiters (SD_count)

— counter for received invalid start delimiters (SD_error_count)

When a station enters the "Listen_Token" or "Passive_Idle" state, the counters are cleared and enabled. If a counter reaches its maximum, counting of this counter as well as of the related comparative counter is stopped. When clearing a counter the related comparative counter is also cleared and they are enabled again. The DLMS-user may access these counters using the Set/Read Value services.

5.5.5.2 Synchronous transmission

5.5.5.2.1 Timers

As explained in 5.5.5.1.1, the following timers shall be implemented to measure the token rotation time and to realize the monitor timers:

Token-rotation-timer, idle-timer, slot-timer, time-out-timer, syn-interval-timer, GAP-update-timer, isochronous-cycle-timer, passive-spare-timer, send-delay-timer and receive-delay-timer.

token-rotation-timer: The functionality of this timer is as defined in 5.5.5.1.1.

idle-timer: This timer monitors the idle state T_{PTG} = T_{SYN} = T_{QUI} on the bus line. The idle-timer in the Master station with the token is loaded with T_{ID1} or T_{ID2} depending on the data transmission service (see 5.5.4). The timer is decremented every bit time, when after a Ph-DATA request primitive (PhICI specifying END-OF-DATA-AND-ACTIVITY) either the Ph-DATA

confirm primitive at the transmitter or the Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) at the receiver is transferred. A new request or a new token DLPDU may be transmitted only after expiration of the timer.

slot-timer: After a request from or a token transfer by a Master station, this timer of the Master station monitors whether the receiving station responds or becomes active within the defined slot time T_{SL}. The timer is initialized with T_{SL} and is decremented every bit time after each transmission of a DLPDU. This DLPDU transmission is indicated by a Ph-DATA confirm primitive after transfer of a Ph-DATA request primitive (PhICI specifying END-OF-DATA-AND-ACTIVITY). If the timer expires before a DLPDU has been received, as indicated by a Ph-DATA indication primitive (PhICI specifying START-OF-ACTIVITY), an error has occurred. As a result of that, a retry or a new message transfer period is initiated.

time-out-timer: This timer monitors bus activity in Master and Slave stations. After transfer of the last Ph-DATA request primitive with PhICI specifying END-OF-DATA-AND-ACTIVITY and return of the corresponding Ph-DATA confirm primitive, or after receiving a Ph-DATA indication primitive with PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY, the timer is loaded with a multiple of the slot time (see 5.5.4) and is decremented every bit time as long as no Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY has been received. If the timer expires, a fatal error has occurred, which for the Master station causes a (re)initialization. The DLMS-user of the Slave or Master station respectively receives a time-out notification.

syn-interval-timer: Master and Slave stations use this timer to monitor the transmission medium for "permanent transmitters". After every Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, the timer is loaded with the value T_{SYNI} (see 5.5.4) and decremented every bit time, as long as no Ph-DATA indication primitive with PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY has been received. If the timer expires, an error of the transmission medium has occurred. The DLMS-user receives a corresponding notification.

GAP-update-timer: This timer operates in the same way as described in 5.5.5.1.1.

isochronous-cycle-timer: The functionality of this timer is as defined in 5.5.5.1.1.

passive-spare- timer: This timer monitors the idle time before sending a SYNCH message. It shall be set according to the time difference between T_{CT} and the current value of the isochronous-cycle-timer T_{RCT} at the end of the last ASP message if the difference is greater than 0. The transmission of the last bit of the last ASP message in an isochronous cycle starts this timer. In the isochronous mode after expiration of the passive-spare-timer, a SYNCH message shall be sent and a synch notification event shall be sent to the DLMS-user.

send-delay-timer: When the time master receives a DL-CS-TIME-EVENT request primitive from the local DLS-user, this timer is started with the value = $2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after confirmation of the transmission of the last portion of a TE DLPDU (see 7.7). The value of send delay time is calculated as the difference between the start value ($2 \times T_{CSI}$) and the read send-delay-timer value. The calculated send delay time is passed to the local DLS-user. In case the timer has expired, the DLS-user is notified of a clock synchronization sequence violation.

receive-delay-timer: When the time receiver receives the DLPDUs last bit of a TE DLPDU (see 7.7) indicated by a Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) has been received, this timer is started with the value = $2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after the receiving of the DLPDUs last bit of a CV DLPDU (see 7.8) indicated by a Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) has been received. The value of the receive delay time is calculated as difference between the start value ($2 \times T_{CSI}$) and the read receive-delay-timer value. The calculated receive delay time is passed to the

local DLS-user. In case the timer expired a clock synchronization sequence violation is notified to the DLS-user.

5.5.5.2.2 Counters

The specifications given in 5.5.5.1 shall apply for the optional counters.

5.6 Cycle and system reaction times

5.6.1 Asynchronous transmission

5.6.1.1 Token transfer period

The base load in a system with several Master stations, that is, the busload caused by medium access control (token DLPDUs) and not by regular message transfer periods, is determined by the token period TP. The total base load per token rotation results from *na* (number of Master stations) token cycles. The transfer period T_{TP} is composed of the token DLPDU time T_{TF}, the transmission delay time T_{TD} and the idle time T_{ID1}. T_{ID1} results from the station delay time T_{SDI} or the synchronization time T_{SYN} respectively. T_{TP} is measured in bits (see Figure 11).

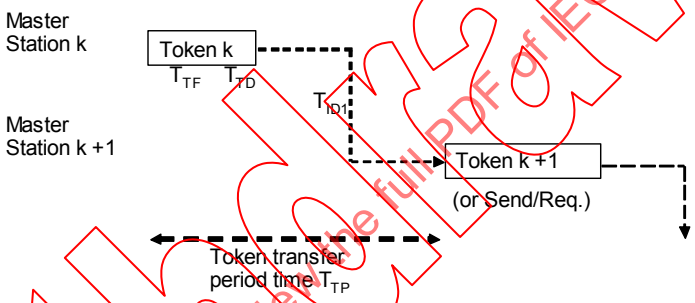


Figure 11 – Token transfer period

$$T_{TP} = T_{TF} + T_{TD} + T_{ID1} \quad (52)$$

The token DLPDU time T_{TF} is determined by the number of UART characters, UC, in the token DLPDU. A UART character always consists of 11 bits (see 6.1.1) and hence the token DLPDU comprises altogether 33 bits. The transmission delay time T_{TD} depends on the line length (about 5 ns/m without repeater) and is mostly substantially less than the other times. The idle time T_{ID1}, which elapses between the token DLPDUs, contains the station delay time T_{SDI} of the token receiver on the one hand, on the other hand the synchronization time T_{SYN} + T_{SM} shall be used, if this sum is larger than T_{SDI} (see 5.5.3). This mainly occurs at low data rates (< 100 kbit/s).

5.6.1.2 Message transfer period

A message transfer period MP consists of the send/request DLPDU and the acknowledgement or response DLPDU. The transfer period is composed of the DLPDU transmission times, the transmission delay times and the station delay times.

The station delay time T_{SDR} elapses between request and acknowledgement or response. This time is needed for decoding the request and assembling the acknowledgement or response DLPDU. It depends on the protocol implementation in the station and is mostly substantially greater than the transmission delay time T_{TD}. The idle time T_{ID}, which elapses between acknowledgement or response and new request, contains also the station delay time (see 5.5.3). However, T_{SYN} + T_{SM} shall be used, if this sum is greater than T_{SDI}.

Figure 12 shows the Message transfer periods.

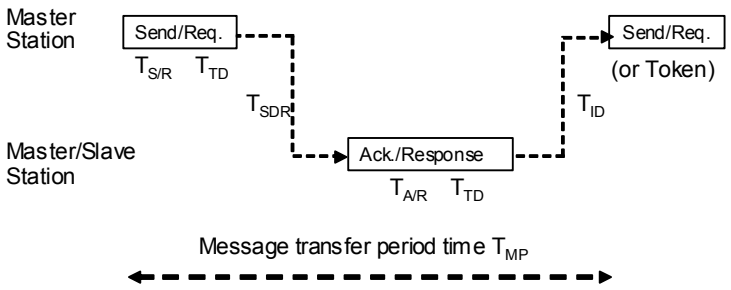


Figure 12 – Message transfer period

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{ID} + 2 \times T_{TD} \quad (53)$$

If the initiator does not receive a valid response DLPDU, caused by disturbances at a station or at the medium, the initiator will repeat the send/request DLPDU depending on its retry limit and the state of the addressed station. For this retry message transfer period (RMP) another transfer period T_{RMP} applies. T_{RMP} is composed of the send/request DLPDU transmission time and the slot time.

$$T_{RMP} = T_{S/R} + T_{SL} \quad (54)$$

At data rates < 100 kbit/s decoding and evaluation of DLPDUs may partially keep pace with their reception. Station delay times become considerably shorter then.

The DLPDU transmission times ($T_{S/R}$, $T_{A/R}$) are determined by the number of UART characters (UC). Thus, they are calculated as follows:

$T_{S/R}$ = a × 11 bit, where "a" is the number of UC in a send/request DLPDU

$T_{A/R}$ = b × 11 bit, where "b" is the number of UC in an ack/response DLPDU

EXAMPLE a = 6, for the request DLPDU: $T_{S/R}$ = 66 bit; b = 59, for the response DLPDU:$T_{A/R}$ = 649 bit (50 octet DATA_UNIT)

5.6.1.3 System reaction times

The message rate R_{SYS} in the system equals the possible number of message transfer periods per second:

$$R_{SYS} = 1 / t_{MP};\ t_{MP} = T_{MP} \times t_{BIT} \quad (55)$$

The maximum system reaction time T_{SR} in a system with **one** Master station and n Slave stations (Master-Slave system) is calculated from the message transfer period and the number of Slave stations. If message retries are allowed, T_{SR} is calculated as follows:

$$T_{SR} = np \times T_{MP} + mp \times T_{RMP} \quad (56)$$

where

np is the number of Slave stations

mp is the number of message retry transfer periods.

T_{RMP} is the message retry transfer period

The maximum system reaction time in a system with **several** Master stations and Slave stations equals the target rotation time:

$$T_{SR} = T_{TR} \quad \text{(see 5.3.2.6)} \tag{57}$$

5.6.1.4 Isochronous cycle time

The isochronous cycle time starts with the transmission of the SYNCH DLPDU by the IsoM Master. The IsoM Master transmits all high priority messages and the configured number of low priority messages. To permit a multi-master environment the token is passed to the next Master station thus leaving other Master stations the necessary time to perform their actions.

$$T_{CT} = T_{SYNCH} + N \times T_{TP} + \sum_{i=1}^{N} P_i + T_{RES} \tag{58}$$

where

N is the number of master stations within token ring

P_i = DLPDU time by master station I (includes isochronous master's cyclic DLPDUs).

The positive difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) represents the spare time T_{RES}. Depending on the amount of time available, a number of active spare time messages (ASM) are sent. At least one ASM message shall be sent. After each sending of the ASM message, the spare time shall be calculated. The transmission of the last ASM is succeeded by the start of the passive-spare-timer if the difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) is greater than zero. The sending of a new SYNCH DLPDU starts after expiration of the passive-spare-timer. If the difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) is less than or equal to zero then the sending of a new SYNCH DLPDU starts immediately. The transmission of a new SYNCH DLPDU marks the beginning of the next cycle. The start of the next cycle is notified to the DLMS-user with a synch notification.

In parallel, the value of isochronous-cycle-timer is read and compared with the calculated cycle time. If the result is zero or within the allowed time shift ($maxT_{SH}$) no message is sent to the DLMS-user. If the result is not in the range of the allowed time shift a Synch_Delay event with the value of the difference between real and calculated cycle time is sent to the DLMS-user.

5.6.2 Synchronous transmission

5.6.2.1 Token transfer period

Similar to asynchronous transmission, the following shall apply for the token transfer period T_{TP}:

$$T_{TP} = T_{TF} + T_{TD} \tag{59}$$

Due to the DLPDU character (see 6.1.2) and the changed DLPDU format (see 7.4.2), the token DLPDU time T_{TF} is 80 bit with a preamble of 16 bit and a post-transmission gap time of 8 bit.

NOTE Only the transmission speed of 31,25 kbit/s may be selected.

5.6.2.2 Message transfer period

The following shall apply for the message transfer period T_{MP} :

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{PTG} + 2 \times T_{TD} \quad (60)$$

The specifications for the message transfer period stipulated in 5.6.1.2 shall apply except for the following cases:

The PDU transmission times ($T_{S/R}$, $T_{A/R}$) are determined by the number of PhL octets. Thus, they are calculated as follows:

$T_{S/R}$ = a, where “a” is the number of octets in a request DLPDU (≤ 255)

$T_{A/R}$ = b, where “b” is the number of octets in an ack/response DLPDU (≤ 255).

EXAMPLE a = 10, for the request DLPDU: $T_{S/R}$ = 80 bit (additionally to 5.6.1.2: 2 octet Preamble and 2 octet PhL start/end delimiter); b = 63, for the response DLPDU: $T_{A/R}$ = 504 bit (additionally to 5.6.1.2: 2 octet Preamble and 2 octet PhL start/end delimiter)

5.6.2.3 System reaction times

See 5.6.1.3.

5.6.2.4 Isochronous cycle time

See 5.6.1.4.

6 General structure and encoding of DLPDUs, and related elements of procedure

6.1 DLPDU granularity

6.1.1 Asynchronous transmission – UART character

6.1.1.1 General

Each DLPDU shall consist of a number of UART characters (UC). Each UART character is a start-stop character for asynchronous transmission, structured as shown in Figure 13.

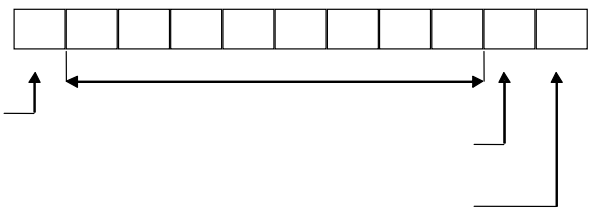


Figure 13 – UART character

The presentation of UART characters is based on ISO/IEC 1177 and ISO/IEC 2022.

6.1.1.2 Transmission rule

Each UART character shall consist of 11 bits: a start bit (ST) which shall be always binary "0", 8 information bits (I) which may be binary "0" or binary "1", an even parity bit (P) which may be binary "0" or binary "1" and a stop bit (SP) which shall be always binary "1".

6.1.1.3 Bit synchronizing

The receiver's bit synchronizing shall always start with the falling edge of the start bit, that is, at the transition from binary "1" to binary "0". The start bit and all consecutive bits shall be scanned in the middle of the bit time. The start bit shall be binary "0" in the middle of the bit, otherwise the synchronizing shall be regarded as failed and shall be stopped. The synchronizing of the UART character ends with the stop bit being binary "1". If a binary "0" bit is encountered instead of the stop bit, a synchronizing error or UART character error shall be assumed and reported and the next leading edge of a start bit shall be waited for.

A maximum deviation of ± 0,3 % of the nominal data rate (bit-period) for transmission and receipt shall not be exceeded for data rates of less than 1 500 kbit/s. For data rates of 1500 kbit/s and higher a maximum deviation of ± 0,03 % of the nominal data rate (bit-period) shall not be exceeded.

6.1.2 Synchronous transmission

Each octet of the DLPDU shall be structured as shown in Figure 14 for synchronous transmission.

Figure 14 – Octet structure

6.2 Length octet (LE, LEr)

The two length octets of identical value in the DLPDU header of the format for variable data length shall contain the number of information octets in the DLPDU body. These comprise: DA, SA, FC and the DATA_UNIT. The value shall cover the range from 4 to 249, so that a maximum of 246 octets may be transmitted in a DLPDUs DATA_UNIT (see 6.6). A value < 4 is not permitted, as a DLPDU contains at least DA, SA, FC and **one** DATA octet. The longest DLPDU may contain a total of 255 octets. Figure 15 illustrates the Length octet coding.

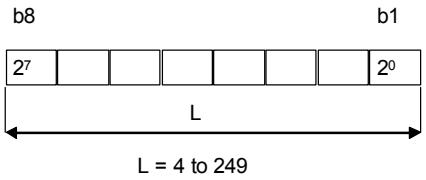


Figure 15 – Length octet coding

6.3 Address octet

6.3.1 Destination and source station address (DA and SA)

The two address octets in the DLPDU header (request, acknowledgement and response DLPDUs) shall contain the destination (DA) and source (SA) station address. The short acknowledgement DLPDU does not contain these two address octets. Figure 16 illustrates the Address octet coding.

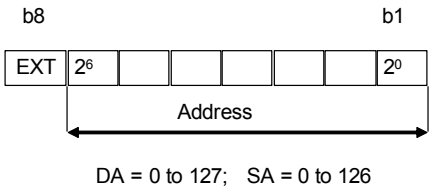


Figure 16 – Address octet coding

Address 127 (b1 to b7 = 1) is reserved as global address for broadcast and multicast messages (DLPDU to all stations or a group of stations selected by means of a DL-service-access-point; it may only occur in SDN and CS, and the response DLPDU of MSRD).

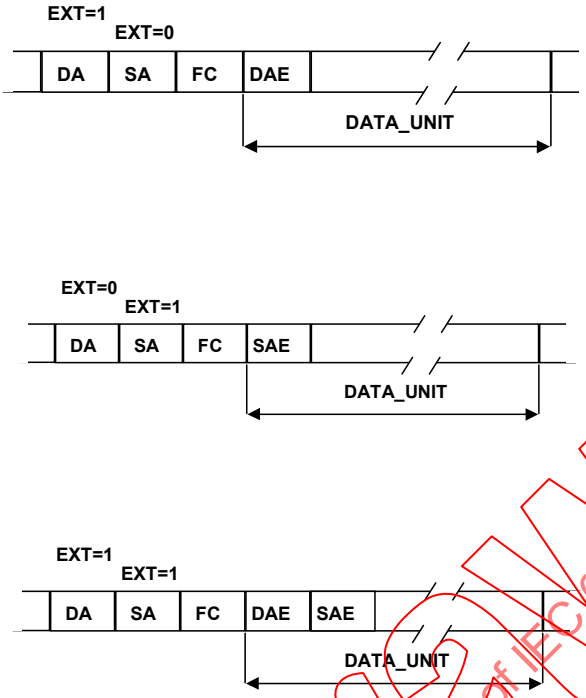
Thus, 127 station addresses (0 to 126) are available for Slave and Master stations, of which preferably no more than 32 may be occupied by Master stations. For non time-critical applications, there may be up to 127 Master stations. As at least one Master station is required, a maximum of 126 addresses for Slave stations is allowed.

Except for MSRD, the send/request DLPDUs address octets shall be sent back mirrored in the acknowledgement or response DLPDU. That is, SA of the acknowledgement or response DLPDU shall contain the destination station address and DA shall contain the source station address of the send/request DLPDU. For MSRD the DA of the response DLPDU shall contain the address 127 and SA of the response DLPDU shall contain the destination station address of the send/request DLPDU.

6.3.2 Address extension (EXT)

Address extensions apply only to DLPDUs with DATA_UNIT. The EXT bit (extension) shall indicate a destination and/or source address extension (DAE, SAE) (see Figure 17), which shall immediately follows the FC octet in the DATA_UNIT. It may be distinguished between access address (DL-service-access-point, DLSAP, see 6.3.4) and region/DL-segment address. Both address types may also occur simultaneously, as each address extension contains an EXT bit again (see bit b8 in Figure 18).

The address extensions of the send/request DLPDU shall be sent back mirrored in the response DLPDU.

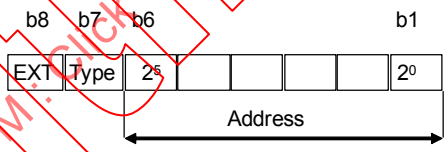


where

EXT = 0: No address extension in the DATA_UNIT

EXT = 1: Address extension follows in the DATA_UNIT

Figure 17 – DAE/SAE octet in the DLPDU



where

b8 (EXT) denotes an additional address extension:

0: No additional address extension octet

1: One additional address extension octet follows immediately. The following order shall be followed:

First octet: region/DL-segment address with b7=1, b8=1

Second octet: DLSAP with b7=0, b8=0.

b7 denotes the type:

0: 6 bit DL-service-access-point (DLSAP): DAE = 0 to 63; SAE = 0 to 62

1: 6 bit region/DL-segment address to realize hierarchical systems, values are not specified in this standard.

Figure 18 – Address extension octet

6.3.3 Address check

A receiver shall check the destination address information in a DLPDU addressed to itself against TS according to the following rules.

a) If there is no region/DL-segment address in the DLPDU existent (DA[b8=0] or DAE[b7=0]), it shall only check the DA on equality with TS.

b) If there is a region/DL-segment address in the DLPDU existent (DAE[b7=1]), it shall check the DAE and the DA on equality with region/DL-segment address and TS of the addressed station.

c) If the receiver of the station, which is addressed with a region/DL-segment address, does not contain a region/DL-segment address then the DLPDU is not addressed to this station.

6.3.4 DL-service-access-point (DLSAP)

At the DLS interface, (see IEC 61158-3-3) a data transmission service (or several thereof) is processed via a DL-service-access-point (DLSAP). Several DLSAPs may exist at the same time in Master and Slave stations. The related DLSAP-address shall be transmitted together with the message, except for the DLSAP-addresses NIL and CS.

The address extensions DAE and SAE shall be used for the transmission of DLSAPs. The source service access point index (S_SAP_index), which represents the access address of the local user to the DL, shall be transmitted in the SAE octet. The destination service access point index (D_SAP_index), which represents one or all access addresses of the remote user to the DL, shall be transmitted in the DAE octet. S_SAP_index values from 0 to 62 and D_SAP_index values from 0 to 63 may be chosen. The D_SAP_index value 63 denotes the global access address. This D_SAP_index may only be used for the Send Data with No Acknowledge service.

NOTE For DLPDUs with a Function = SDA_H/L or SRD_H/L (see Table 3) a D_SAP_index of 63 in the send/request DLPDU results in a negative acknowledge (RS).

If the transmission of DLSAP addresses is omitted for reasons of DLPDU efficiency, the data transmission services shall be processed via the **default DLSAP** at the initiator or the responder or both. In these cases, all DLPDUs shall be transmitted without the related address extension (SAE or DAE or both). At the DLS interface the default DLSAP is mandatory and is addressed with the value NIL.

The DLSAP-address CS used for the time synchronization services Clock Value and Time Event is omitted in the related DLPDUs for the same reason.

6.4 Control octet (FC)

6.4.1 General

The control octet in the DLPDU header shall indicate the DLPDU type, such as send/request DLPDU and acknowledgement or response DLPDU. In addition, the control octet shall contain the function Code No and the control information, which prevents loss and multiplication of messages, or the station type with the DL status. Figure 19 and Figure 20 illustrate the FC octet coding.

b8	b7	b6	b5	b4	b3	b2	b1
0/1	1	FCB	FCV	2^3	Function code No		2^0

where

b8 = 0/1, b7 = 1: send/request DLPDU

FCB: is Frame count bit: 0 or 1, alternating

FCV: is Frame count bit valid:

0: alternating function of FCB is invalid

1: alternating function of FCB is valid

Function code No: shall be as described in Table 3.

Figure 19 – FC octet coding for send/request DLPDUs

b8	b7	b6	b5	b4	b3	b2	b1
Res	0	Station type and DL status		2^3	Function code No		2^0

where

DLPDU type b7 = 0: is acknowledgement or response DLPDU

b6 and b5 are station type and DL status combined

b6 b5

0 0 Slave station

0 1 Master station not ready to enter logical token ring

1 0 Master station ready to enter logical token ring

1 1 Master station in logical token ring

Res: means reserved (the sender shall set to binary "0", the receiver does not have to interpret this bit)

Function code No: shall be as described in Table 3.

Figure 20 – FC octet coding for acknowledgement or response DLPDUs

Table 3 shows the function code No of the control octet for both send/request DLPDUs and acknowledgement or response DLPDUs.

Table 3 – Transmission function code

Code	Function	Format in subclause described	Possible for the following stations	
	DLPDU type b8 = 1 & b7 = 1		Initiator	Receiver/ Responder
0	Clock Value	7.3.1 , 7.3.2	M	M and S
1…15	Reserved			
	DLPDU type b8 = 0 & b7 = 1			
0	Time Event	7.1.1 , 7.1.2	M	M and S
1,2	Reserved			
3	Send data with scknowledge low	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M	M and S
4	Send data with no acknowledge low		M	M and S
5	Send data with acknowledge high		M	M and S
6	Send data with no acknowledge high		M	M and S
7	Send and request data multicast	7.2.1 , 7.3.1 , 7.2.2 , 7.3.2	M	M and S
8	Reserved			
9	Request DL-status with reply	7.1.1 , 7.1.2	M	M and S
10, 11	Reserved			
12	Send and request data low	7.1.1 , 7.2.1 , 7.3.1	M	M and S
13	Send and request data high	7.1.2 , 7.2.2 , 7.3.2	M	M and S
14	Request ident with reply	7.1.1 , 7.1.2	M	M and S
15	Reserved			
	DLPDU type b7 = 0			
0	Acknowledgement positive (OK)	7.1.1 , 7.1.2 [a]	M and S	M
1	ACK negative DL/DLM User Error (UE)	7.1.1 , 7.1.2	M and S	M
2	ACK negative no resource for send data (& no response DL data) (RR)		M and S	M
3	ACK negative no service activated (RS)		M and S	M
4 to 7	Reserved		M and S	M
8	Response DL/DLM data low (& send data ok) (DL)	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M and S	M
9	ACK negative no response DL/DLM data, (& send data ok) (NR)	7.1.1 , 7.1.2 [a]	M and S	M
10	Response DL data high, (& send data ok) (DH)	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M and S	M
11	Reserved		M and S	M
12	Response DL data low, no resource for send data (RDL)	7.2.1 , 7.3.1	M and S	M
13	Response DL data high, no resource for send data (RDH)	7.2.2 , 7.3.2	M and S	M
14, 15	Reserved			

where M: Master, S: Slave

	b4	b1
Function code No 0:	0	0 0 0
Function code No 15:	1	1 1 1

( ) Value of the L/M_status parameter of the service primitives (see IEC 61158-3-3).

[a] Also possible Short Acknowledgement SC = E5H; exception: if request DL-status with reply, no SC is permitted.

6.4.2 Frame count bit

The frame count bit FCB (b6) prevents the duplication of messages at the responder and the loss at the initiator. However, "Send Data with No Acknowledge" (SDN), "Request DL Status with Reply", "Request Ident with Reply", “Time Event” and “Clock Value” are excluded from this.

In order to manage the security sequence, the initiator shall carry a FCB for each responder. When an send/request DLPDU is transmitted to a responder for the first time or to a responder currently marked as "non operational", the associated FCB shall be set unambiguously. The initiator shall achieve this by an send/request DLPDU with FCV=0 and FCB=1. The responder shall classify such a DLPDU as first message transfer period and store FCB=1 together with the initiator's address (SA and optional SAE[b7=1]) (see Table 4). This message transfer period is not repeated by the initiator.

If a responder supports the region/DL-segment addressing and the send/request DLPDU contains a source region/DL-segment address (SAE[b7=1]), which is unequal to the own region/DL-segment address (DAE[b7=1]), then the responder shall store the SAE[b7=1] together with the SA.

In the following send/request DLPDUs to the same responder the initiator shall set FCV=1 and toggle FCB with each new send/request DLPDU. The responder shall evaluate FCB when receiving an send/request DLPDU addressed to itself with FCV=1. A FCB changed in comparison with the same initiator's (same SA and optional same SAE[b7=1]) preceding send/request DLPDU shall be considered as confirmation of the preceding message transfer period's correct completion. If the send/request DLPDU originates from a different initiator (different SA or optional different SAE[b7=1]), there shall be no evaluation of the FCB. In both cases, the responder shall store the FCB with the source address (SA and optional SAE[b7=1]) until it receives a new DLPDU addressed to itself.

If an acknowledgement or response DLPDU is missing or corrupted, the FCB shall **not** be changed by the initiator in the retry; this indicates the faulty preceding message transfer period. If a responder receives an send/request DLPDU with FCV=1 and the same FCB as in the same initiator's (same SA and optional same SAE[b7=1]) immediately preceding send/request DLPDU, a retry shall be carried out. As a result, the responder shall again transmit the acknowledgement or response DLPDU kept in readiness.

The responder shall keep ready the preceding acknowledgement or response DLPDU for a potential retry, until it detects a confirmation of the preceding message transfer period's correct completion as described above, or until it receives a token DLPDU or a DLPDU with a changed address (SA or DA or optional SAE[b7=1] or DAE[b7=1]).

For "Send Data with No Acknowledge", "Request DL Status with Reply", "Request Ident with Reply", “Time Event” and “Clock Value”, FCV and FCB are both zero for the following DLPDU types; the responder does not need to analyse FCB:

Table 4 – FCB, FCV in responder

b6	b5	←- bit position		
FCB	FCV	Condition	Meaning	Action
0	0	DA = TS/127	Request with no ack Request DL-status with reply Request ident with reply Time event Clock event	Last ack or reply may be deleted
0/1	0/1	DA ≠ TS	Request to other responder	Last ack or reply may be deleted
1	0	DA = TS	First request	FCBM := 1 SAM := SA last Ack or Reply may be deleted
0/1	1	DA = TS SA = SAM FCB ≠ FCBM	New request	last Ack or delete Reply FCBM := FCB have Ack or Reply ready for Retry
0/1	1	DA = TS SA = SAM FCB = FCBM	Request retry	FCBM := FCB repeat Ack or Reply and keep it ready
0/1	1	DA = TS SA ≠ SAM	New initiator	FCBM := FCB SAM := SA have Ack or Reply ready for Retry
—	—	Token-DLPDU	—	last Ack or Reply may be deleted
NOTE FCBM is the stored FCB and SAM is the stored SA.				

6.5 DLPDU content error detection

6.5.1 Asynchronous transmission – frame checksum (FCS)

The 8-bit (1 octet) frame checksum that is required for Hamming distance 4 in a DLPDU shall always immediately precede the end delimiter. The checksum shall be structured as shown in Figure 21.

Figure 21 – FCS octet coding

In DLPDUs of fixed length with no data field (see 7.1.1) the checksum shall be calculated from the two's-complement arithmetic sum of DA, SA and FC (without start and end delimiters).

In DLPDUs of fixed length with data field (see 7.2.1) and in DLPDUs with variable data field length (see 7.3.1) the checksum shall additionally include the DATA_UNIT.

6.5.2 Synchronous transmission –frame check sequence (FCS)

A 16-bit (2 octet) frame check sequence is required. It shall be calculated and appended to the DLPDU as specified in Clause 5, where its Hamming distance properties are also described.

6.6 DATA_UNIT

6.6.1 General

The DATA_UNIT shall consist of the User Data (DLSDU) of the DLS/DLMS-user and optionally of one or more address extension octets (see Figure 22). Up to 4 address extension octets are permissible (see 6.3.1). The User Data comprises a maximum of 242 up to 246 octets depending on the number of used address extension octets.

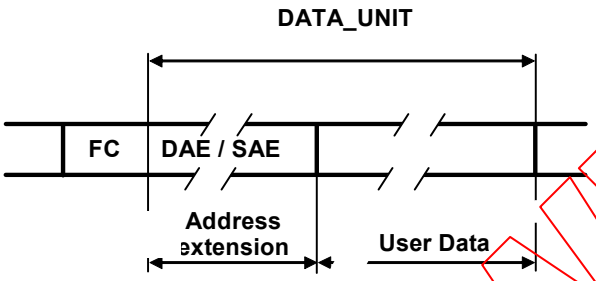
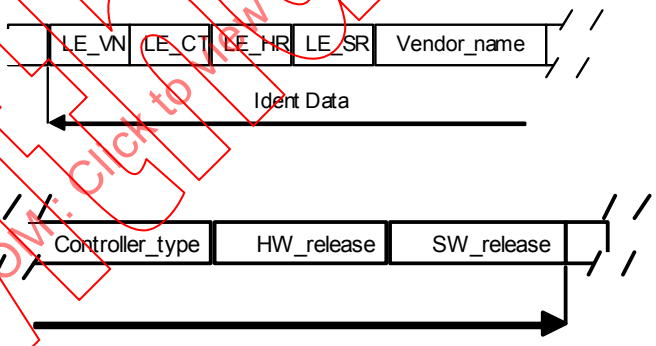


Figure 22 – Data field

The following user data are defined for the Ident DLPDU:

6.6.2 Ident user data

As shown in Figure 23 the Ident user data shall contain the station's Ident_List with vendor name (Vendor_name), fieldbusfieldbus controller type (controller_type) and hardware and software release (HW/SW_release). It comprises a maximum of 200 octets:



where

LE_VN, LE_CT, LE_HR, LE_SR in each case the length of the corresponding data field in Octets (1 octet each, dual coding, significance as in Figure 21).

Vendor_name (VN) is the name of manufacturer as an ASCII string (ISO 7 Bit Code, b8=0).

Controller_type (CT) is the hardware controller type as an ASCII string (ISO 7 Bit Code, b8=0).

HW_release (HR) is the hardware release of controller as an ASCII string (ISO 7 Bit Code, b8=0).

SW_release (SR) is the software release of controller as an ASCII string (ISO 7 Bit Code, b8=0).

Figure 23 – Ident user data

6.7 Error control procedures

6.7.1 Asynchronous transmission

Line protocol errors, for example, character framing errors, overrun errors and parity errors, and transmission protocol errors, for example faulty start delimiters, frame check octets and end delimiters, invalid DLPDU length, response times, etc. shall result in the following station reactions:

A send/request DLPDU or token DLPDU that has been received incorrectly by a station shall not be processed, acknowledged or answered. The initiator shall retry the request after expiration of the slot time. The request shall also be retried if the acknowledgement or response was corrupted. The initiator shall complete a request only after having received a valid response or if the retry (retries) was not (were not) successful (see Table 5). This means that a "Send/Request" shall be kept until the responder confirms its correct receipt by an acknowledgement or response or the retry (retries) was not (were not) successful. In the same way, a responder shall terminate a "Request" or "Send/Request" only if a new request with altered frame count bit is received or another station is addressed (see 6.4, FCB).

If a station does not acknowledge or respond after retry (retries), it shall be marked as "non operational". When processing the following requests, the initiator shall transmit the request to this station without retry, until the station acknowledges or responds correctly again. After positive acknowledgement, the initiator shall mark again the addressed station as "operational". When processing the next request, the initiator shall continue the original mode of operation with this station.

6.7.2 Synchronous transmission

Errors in the line protocol (see Clause 9 of IEC 61158-2) and in the medium access protocol, such as erroneous start octets and FCS octets, DLPDU length, response times etc. shall result in the station reactions specified in 6.7.1.

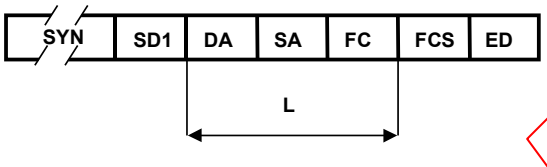
7 DLPDU-specific structure, encoding and elements of procedure

7.1 DLPDUs of fixed length with no data field

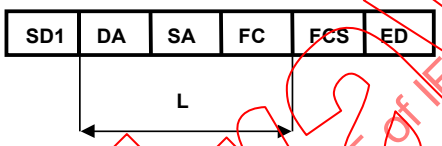
7.1.1 Asynchronous transmission

The formats of DLPDUs of fixed length with no data field shall be as shown in Figure 24.

a) Format of the request DLPDU:



b) Format of the acknowledgement DLPDU:



c) Format of the short acknowledgement DLPDU:

where

SYN is the synchronization period, a minimum of 33 line idle bits
SD1 is the start delimiter, value: 10H
DA is the destination address
SA is the source address
FC is the frame control
FCS is the frame checksum
ED is the end delimiter, value: 16H
L is the information field length, fixed number of octets: L = 3
SC is the single character, value: E5H.

Figure 24 – DLPDUs of fixed length with no data field

Transmission Rules

1) Line idle state shall correspond to signalling level binary "1".
2) Each request DLPDU shall be preceded by at least 33 line idle bits (synchronization time).
3) No idle states are allowed between a DLPDUs UART characters.
4) The receiver shall check:
   — for each UART character: start bit, stop bit and parity bit (even),
   — for each DLPDU: start delimiter, DA, SA, FCS and end delimiter,
   — and the synchronization time in case of a request DLPDU.

   If the check fails, the entire DLPDU shall be discarded.

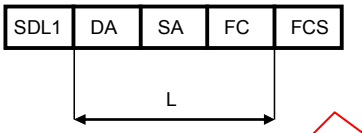
SC and SD1 (as well as SD2 and SD3, see 7.2.1 and 7.3.1) have Hamming distance Hd=4 and are safe against being shifted (see for example IEC 60870-5-1), that is, the single character SC appears to be a DLPDU with Hd=4.

For requests to be acknowledged (Send Data with Acknowledge), only SC is a permissible positive acknowledgement. For requests to be answered (Send and Request Data with Reply), SC is permissible if no data is available (see Table 3, b7=0, Code-No 9).

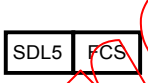
7.1.2 Synchronous transmission

The formats of DLPDUs of fixed length with no data field shall be as shown in Figure 25.

a) Format of the request DLPDU and format of the acknowledge DLPDU:



b) Format of the short acknowledgement DLPDU:



where

SDL1 is the start octet 1 (start delimiter 1 data link), code: 10H
SDL5 is the start octet 5 (start delimiter 5 data link), code: E5H
DA is the destination address
SA is the source address
FC is the frame control
FCS is the frame check sequence, 2 octets
L is the information field length, fixed number of octets: L = 3.

Figure 25 – DLPDUs of fixed length with no data field

Transmission Rules

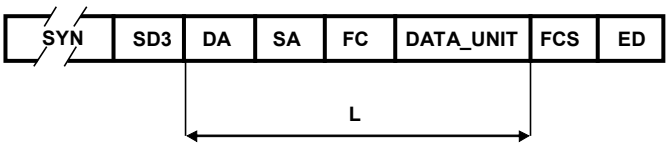
In addition to the transmission rules of the Ph-layer in Clause 9 of IEC 61158-2, the receiver shall check the SDL, DA/SA and FCS octets for each DLPDU. If the check fails, the whole DLPDU shall be discarded.

7.2 DLPDUs of fixed length with data field

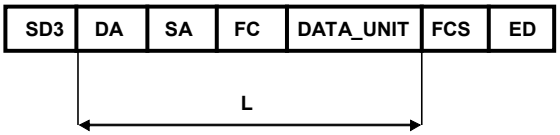
7.2.1 Asynchronous transmission

The format of DLPDUs of fixed length with data field shall be as shown in Figure 26.

a) Format of the send/request DLPDU:



b) Format of the response DLPDU:



where

SYN	is the synchronization period, a minimum of 33 line idle bits
SD3	is the start delimiter, value: A2H
DA	is the destination address
SA	is the source address
FC	is the frame control
DATA_UNIT	is the data field, fixed length (L-3) = 8 octets
FCS	is the frame checksum
ED	is the end delimiter, value: 16H
L	is the information field length, fixed number of octets: L = 11.

Figure 26 – DLPDUs of fixed length with data field

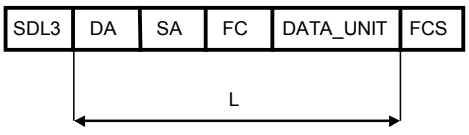
Transmission Rules

The same transmission rules shall apply as for DLPDUs of fixed length with no data field (see 7.1.1).

7.2.2 Synchronous transmission

The formats of DLPDUs of fixed length with data field shall be as shown in Figure 27.

a) Format of the send/request and response DLPDUs:



where

SDL3	is the start delimiter 3 data link, code: A2H
DA	is the destination address
SA	is the source address
FC	is the frame control
DATA_UNIT	is the data field, fixed length (L-3) = 8 octets
FCS	is the frame check sequence, 2 octets
L	is the information field length, fixed number of octets: L = 11.

Figure 27 – DLPDUs of fixed length with data field

Transmission Rules

The same transmission rules shall apply as for DLPDUs of fixed length with no data field (see 7.1.2).

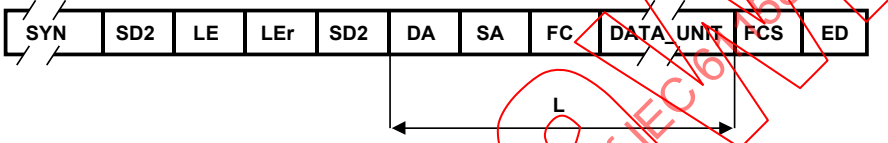
7.3 DLPDUs with variable data field length

7.3.1 Asynchronous transmission

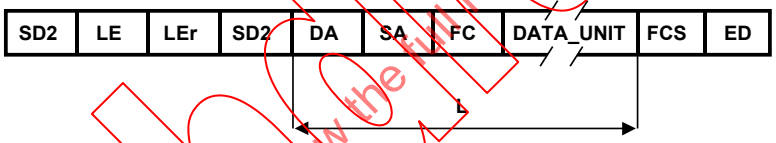
For a variable number of data octets the length information shall also be transmitted in the DLPDU. This length information shall be contained twice in a fixed DLPDU header at the beginning of the DLPDU. Thus it is protected with Hd = 4 and safe against slip.

The format of DLPDUs with variable data field length shall be as shown in Figure 28.

a) Format of the send/request DLPDU:



b) Format of the response DLPDU:



where

SYN	is the synchronization period, a minimum of 33 line idle bits
SD2	is the start delimiter, value: 68H
LE	is the octet length, allowed values: 4 to 249
LEr	is the octet length repeated
DA	is the destination address
SA	is the source address
FC	is the frame control
DATA_UNIT	is the data field, variable length (L-3), max. 246 octets
FCS	is the frame checksum
ED	is the end delimiter, value: 16H
L	is the information field length, variable number of octets: L = 4 to 249.

Figure 28 – DLPDUs with variable data field length

Transmission Rules

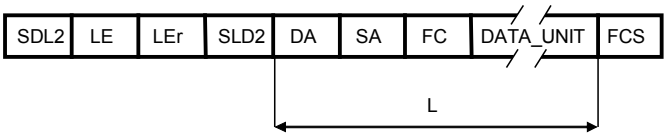
The same transmission rules as for DLPDUs of fixed length with no data field shall be applied (see 7.1.1).

In addition to transmission rule 4, LE shall be identical to LEr, and the information octets shall be counted from the destination address (DA) up to the frame checksum (FCS) and the result shall be compared with LE.

7.3.2 Synchronous transmission

The formats of DLPDUs with variable data field length shall be as shown in Figure 29.

a) Format of the send/request and response DLPDUs:



where

SDL2	is the start delimiter 2 data link, code: 68H
LE	is the length, value: 4 to 249
LEr	is the length (repeated)
DA	is the destination address
SA	is the source address
FC	is the frame control
DATA_UNIT	is the data field, variable length (L-3), max. 246 octets
FCS	is the frame check sequence, 2 octets
L	is the information field length, variable number of octets: L = 4 to 249.

Figure 29 – DLPDUs with variable data field length

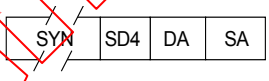
Transmission Rules

The same transmission rules as for DLPDUs of fixed length with no data field shall apply (see 7.1.2). In addition, the receiver shall check if LE and LEr coincide. The information octets shall be counted from the destination address (DA) up to the frame check sequence (FCS) and shall be compared with LE.

7.4 Token DLPDU

7.4.1 Asynchronous transmission

The format of the token DLPDU shall be as shown in Figure 30.



where

SYN	is the synchronization period, a minimum of 33 line idle bits
SD4	is the start delimiter, value: DCH
DA	is the destination address
SA	is the source address.

Figure 30 – Token DLPDU

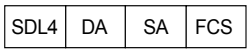
Transmission Rules

1) Line idle state shall correspond to signalling level binary "1".
2) Each token DLPDU shall be preceded by at least 33 line idle bits (synchronization time).
3) No idle states are permitted between a DLPDUs UART characters.
4) The receiver shall check:
   — per UART character: start bit, stop bit and parity bit (even),
   — per DLPDU: synchronization time, start delimiter and DA/SA.

   If the result of the check is negative, the whole DLPDU shall be discarded.

7.4.2 Synchronous transmission

The format of the token DLPDU shall be as shown in Figure 31.



where

SDL4	is the start delimiter 4 data link, code: DCH
DA	is the destination address
SA	is the source address
FCS	is the frame check sequence, 2 octets.

Figure 31 – Token DLPDU

Transmission Rules

The same transmission rules as for DLPDUs of fixed length with no data shall apply (see 7.1.2).

7.5 ASP DLPDU

The ASP DLPDU is generated by the DLE in the isochronous mode to produce bus activity in the spare time of an isochronous cycle to avoid a time-out (T_{TO}). The format of the ASP DLPDU is the same as described in 7.1, with the following restrictions:

— Control Octet (FC) is equal to Request DL Status with Reply (Code 49H)

— DA is equal to SA

— there is no response DLPDU by any DLE for this ASP DLPDU.

7.6 SYNCH DLPDU

The SYNCH DLPDU is generated by the DLE in the isochronous mode to mark the beginning of a new isochronous cycle. The format of the SYNCH DLPDU is the same as described in 7.3, with the following restrictions:

— FC, the Control Octet, is equal to Send Data with No Acknowledge high (Code 46H)

— DA is equal to 127

— SAE is equal to 62

— DAE is equal to 58

— the DATA_UNIT shall contain the value of the operating parameter SYNCHT

— there is no response DLPDU by any DLE for this SYNCH DLPDU.

7.7 Time Event (TE) DLPDU

The TE DLPDU is used to synchronize the clock at the time receiver. The format of the TE DLPDU is the same as described in 7.1, with the following restrictions:

— FC, the Control Octet, is equal to Time Event (Code 40H)

— DA is equal to 127

— there is no response DLPDU by any DLE for this TE DLPDU.

7.8 Clock Value (CV) DLPDU

The CV DLPDU is used to transmit the clock value to the time receivers. The format of the CV DLPDU is the same as described in 7.3, with the following restrictions:

— FC, the Control Octet, is equal to Clock Value (Code C0H)

— DA is equal to 127
— SAE and DAE are not used
— there is no response DLPDU by any DLE for this CV DLPDU.

7.9 Transmission procedures

7.9.1 Asynchronous transmission

Permissible DLPDU sequences (message transfer periods) for asynchronous transmission are described in Figure 32 through Figure 34. Error sequences and broadcast/multicast messages are excluded.

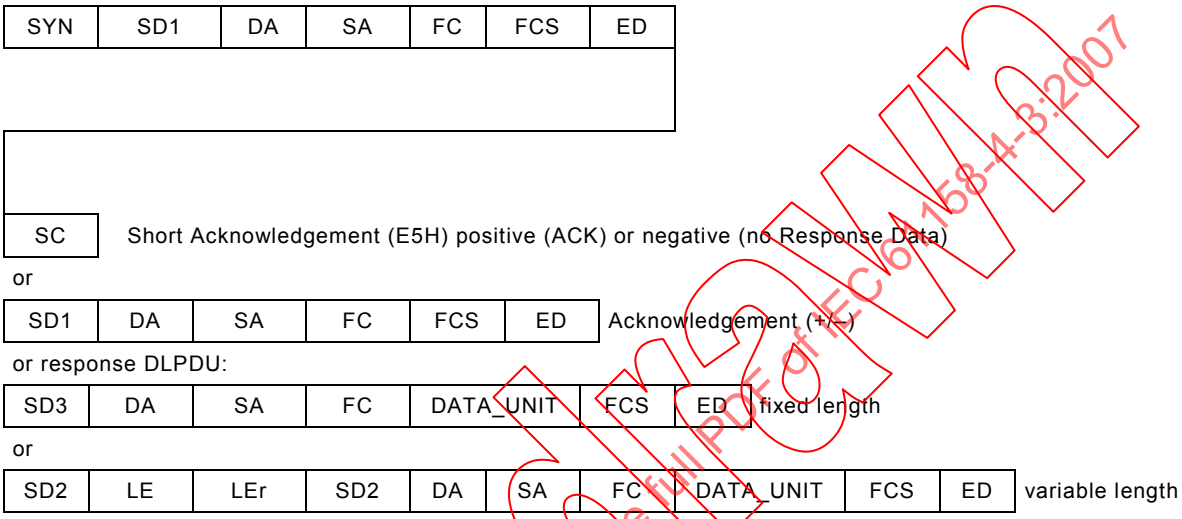


Figure 32 – Send/request DLPDU of fixed length with no data

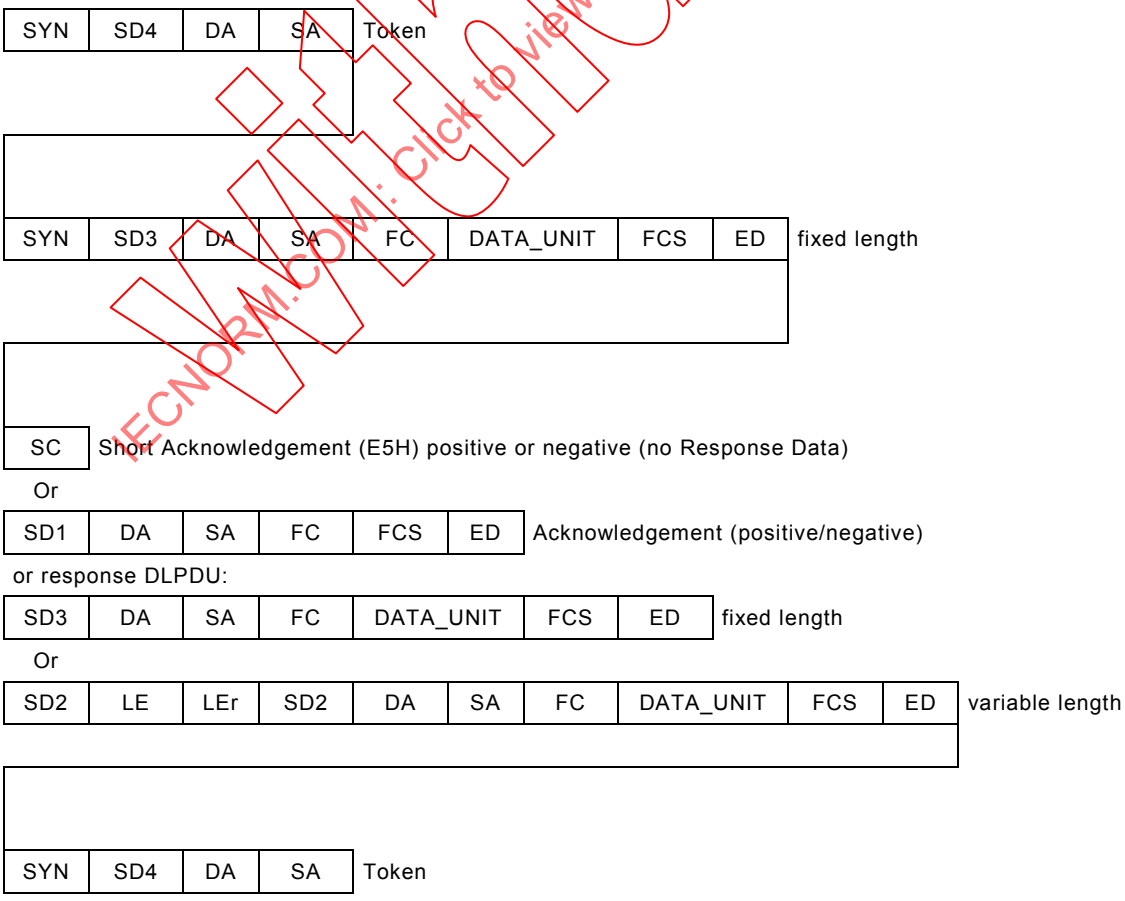


Figure 33 – Token DLPDU and send/request DLPDU of fixed length with data

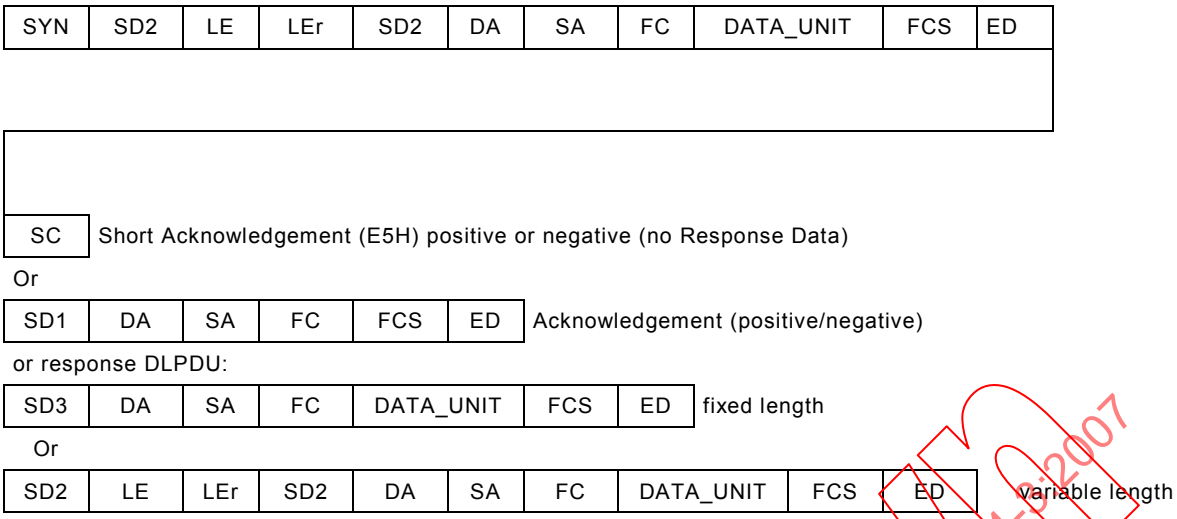


Figure 34 – Send/request DLPDU with variable data field length

7.9.2 Synchronous transmission

Permissible DLPDU sequences (message transfer periods) for synchronous transmission are described in Figure 35 through Figure 37. Error sequences and broadcast/multicast messages are excluded.

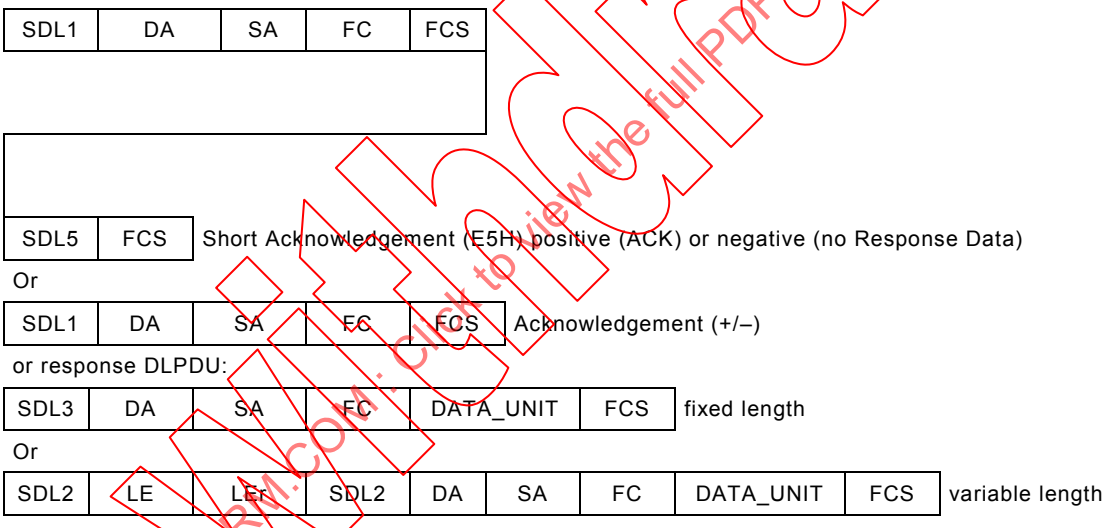


Figure 35 – Send/request DLPDU of fixed length with no data

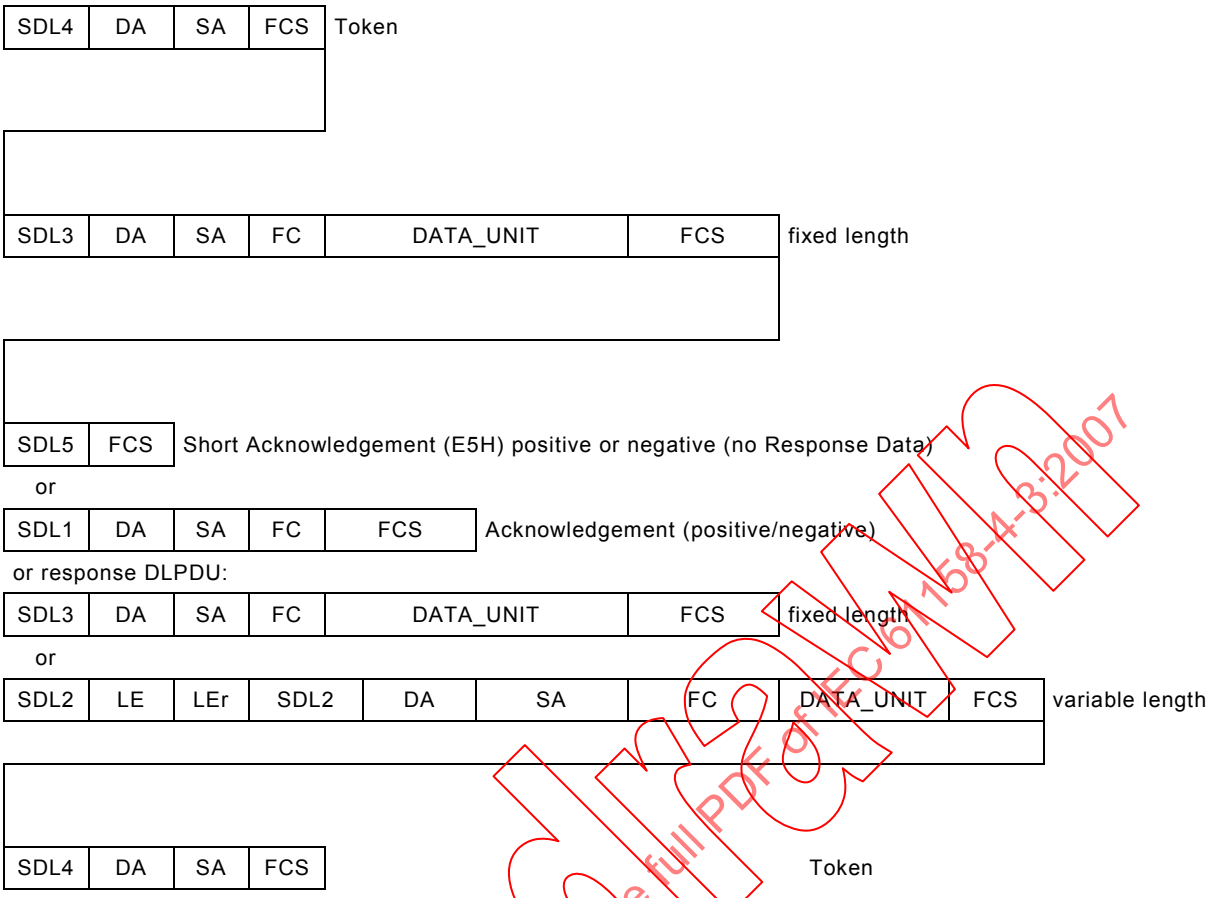


Figure 36 – Token DLPDU and send/request DLPDU of fixed length with data

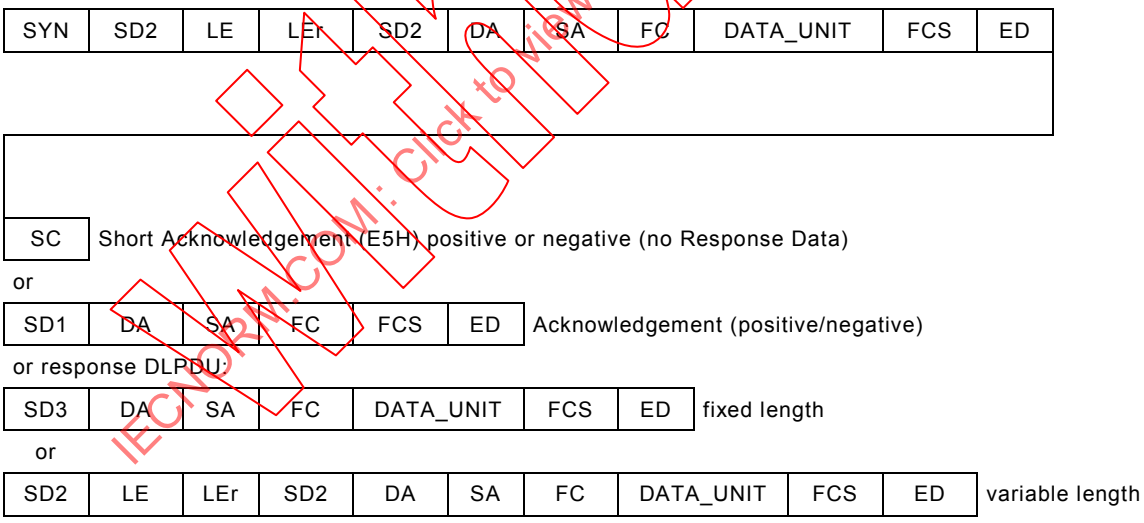


Figure 37 – Send/request DLPDU with variable data field length

8 Other DLE elements of procedure

NOTE Annex A specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. The specification of Annex A is complementary to the textual specification in this and related clauses in the body of this standard. In case of conflict the requirements of Annex A take precedence.

8.1 DL-entity initialization

After power on (PON) the DL-entity of Master and Slave stations shall enter the "Offline" state. Within this state, the DL-entity does not receive or transmit any signals (DLPDUs) from or to the bus.

The DL-entity may enter the "Passive_Idle" or "Listen_Token" state from the "Offline" state only, if the operating parameters (DL-variables) have been set for correct protocol handling (see IEC 61158-3-3). The operating parameters, shown in Table 5, shall be provided by the DLMS-user.

Table 5 – Operating parameters

Parameter number	Name
1	Station DL-address TS
2	Data rate (kbit/s)
3	Single/redundant media available
4	Hardware release
5	Software release
6	Slot time T_{SL}
7	Station delay time min T_{SDR}
8 (see Note 1)	Station delay time max T_{SDR}
9 (see Note 1)	Transmitter fall/Repeater switch time T_{QUI}
10 (see Note 1)	Setup time T_{SET}
11 (see Note 1)	Target rotation time T_{TR}
12 (see Note 1)	GAP update factor G
13 (see Note 1)	Master station enter/leave the logical ring (in_ring_desired)
14 (see Note 1)	Highest station address (HSA)
15 (see Note 1)	Maximum number of retries (max_retry_limit)
16 (see Note 2)	Clock synchronization interval T_{CSI}
17 (see Note 3)	Contents of the SYNCH User Data (SYNCHT)
18 (see Note 3)	Isochronous cycle time T_{CT}
19 (see Note 3)	Allowed maximal time shift T_{TS}
NOTE 1 This applies only to Master stations.	
NOTE 2 This applies only to stations able to support clock synchronization.	
NOTE 3 This applies only to stations able to work in isochronous mode.	

8.2 States of the media access control of the DL-entity

8.2.1 General

A Master station's media access control state machine (MAC) is described by means of 11 states and the transitions between them. A Slave station's MAC has 2 states.

Figure 38 shows as an overview the combined state diagram of the Master (state 0 to 9 and 11) and the Slave station (state 0 and 10).

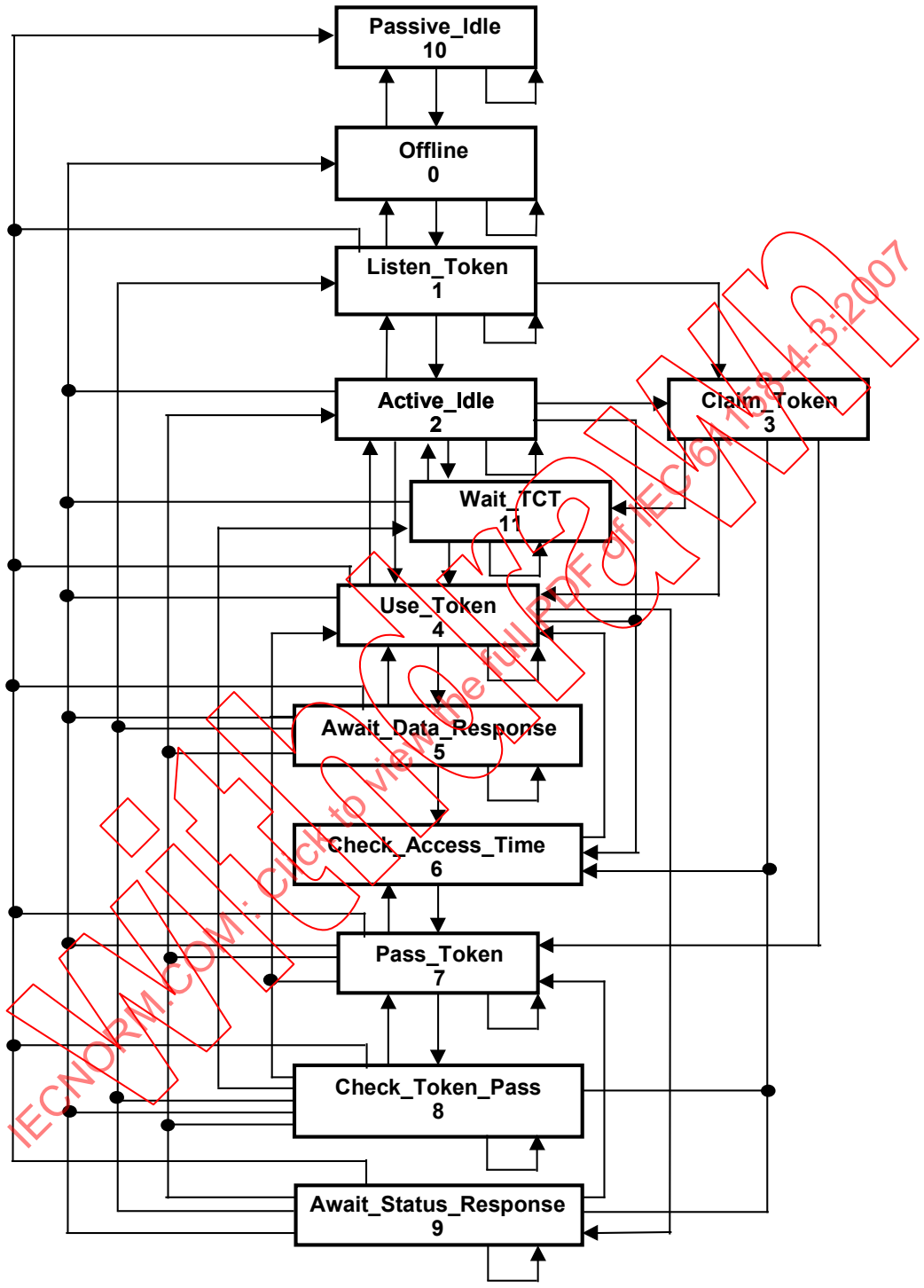


Figure 38 – DL-state-diagram

8.2.2 Offline

The "Offline" state shall be entered immediately after power on, after the DLM-RESET service, or after certain error conditions have been detected. After power on each station should perform a self-test. This internal self-test depends on the implementation and does not

influence the other stations. For this reason, the self-test procedure is not specified in this specification.

After terminating the power on sequence, the MAC remains in the "Offline" state until all required operating parameters (see 8.1) have been initialized. The DL-entity may only afterwards connect to the transmission medium, but without becoming active.

8.2.3 Passive_Idle

After its parameters are initialized, the Slave station's MAC or the Master station's MAC depending on the operating parameter "in_ring_desired" (in_ring_desired equal false) (see Table 5) shall enter the "Passive_Idle" state and listen to the line. If a plausible send/request DLPDU addressed to that station is received, the DL-entity shall acknowledge or respond as required, except for DLPDUs with global address (broadcast message, see 6.3.1) and token DLPDUs addressed to itself. The token DLPDU is discarded.

At the occurrence of the DLM-RESET service the MAC re-enters the "Offline" state.

8.2.4 Listen_Token

After its operating parameters have been initialized and the value of the parameter "in_ring_desired" is true, the Master station's MAC shall enter the "Listen_Token" state. In this state the Master station's DL-entity shall monitor the line in order to identify those Master stations which are already in the logical token ring. For that purpose, token DLPDUs are analysed and the station addresses contained in them are used to generate the list of Master stations (LMS). After listening to two complete token rotations, the MAC shall enter the state "Active_Idle".

During LMS generation a "Request DL Status with Reply" responds with "Master station not ready" after one token rotation is completed. All other DLPDUs are not processed in the "Listen_Token" state, that is, they are neither acknowledged/responded nor indicated to the local DLS-user.

If the MAC detects its own address as source address (SA) in two token DLPDUs when registering the Master stations, it shall assume that another Master station with the same address exists already in the ring. The MAC shall then re-enter the "Offline" state and report this event to the local DLMS-user.

If the MAC observes no bus activity for the time-out period, it shall assume that an initialization or a restoration of the logical token ring is necessary. The MAC shall enter the "Claim_Token" state.

8.2.5 Active_Idle

On leaving the "Listen_Token" state, the Master station's MAC shall enter the "Active_Idle" state and shall wait for received messages. If it receives an send/request DLPDU addressed to itself or as broadcast, it shall proceed, acknowledge or respond as required and shall unfreeze the token-rotation-timer if the token-rotation-timer is frozen.

The "Listen_Token" state shall be entered in the case of an error, if two token DLPDUs with SA = TS are received in immediate succession. This event shall be reported to the local DLMS-user.

If the MAC find out that it was taken from the logical token ring not on its own initiative, it also shall enter in the "Listen_Token" state and report this (Out_of_ring) to the local DLMS-user. The "Listen_Token" state shall also be entered in case of the MACs LMS is not ready, that is, the SA and DA in the token DLPDU are not compliant with the LMS entry for a number of Tokens receipts. Token DLPDUs, which indicate the addition and removal of a Master, shall not be counted as not compliant.

If the MAC observes no bus activity for the time-out period, it shall assume that a restoration of the logical token ring is necessary. The MAC shall try to claim the token and to (re)initialize the logical ring ("Claim_Token" state).

If the MAC is addressed by a "Request DL Status with Reply" transmitted by its predecessor (PS) then:

a) if the DL-address (TS) is contained in the LMS of the MAC, the MAC shall respond with "Master station in logical ring", or

b) if the DL-address (TS) is not contained in the LMS of the MAC, the MAC shall respond with "Master station ready to enter logical token ring".

After receiving a token DLPDU addressed to the MAC itself and the isochronous mode is not set, it shall

1) unfreeze the token-rotation-timer if this timer is frozen,

2) drop the token, if the token is not sent by the predecessor or if the DL-address (TS) is not contained in the LMS of the MAC. In these both cases, the MAC shall update the LMS,

3) otherwise, by processing the transition the T_{RR} (real rotation time) shall be calculated as the difference between target rotation time and the read value of the token-rotation-timer and the token-rotation-timer shall be restarted with target rotation time. The MAC shall enter:

   i) the "Use_Token" state, if a high priority message is available, or

   ii) the "Check_Access_Time" state if no high priority message is available.

After receiving a token DLPDU addressed to the MAC itself and the isochronous mode is set, it shall

1) send an ASP message, and

2) change into the "Wait_TCT" state.

If the MAC receives an ASP message and the isochronous mode is set, it shall freeze the token-rotation-timer.

8.2.6 Claim_Token

The MAC shall enter the "Claim_Token" state after the "Active_Idle" state or the "Listen_Token" state, when its time-out has expired. In this state it shall try to initialize the logical ring or to start an initialization.

When re-initializing and the MAC is not in the isochronous mode, the DL-entity address is contained in the LMS of the MAC, and thus the "Use_Token" state is entered immediately, if high priority message are pending. Otherwise the MAC shall enter the "Check_Access_Time" state.

When re-initializing and the MAC is in the isochronous mode, the DL-entity address is contained in the LMS of the MAC, and thus the MAC shall send an ASP message and shall enter the "Wait_TCT" state.

When initializing (the DL-address (TS) is not contained in the LMS of the MAC), at first the token shall be addressed twice to the own DL-entity, that is, NS = TS, namely in the "Pass_Token" state. This is necessary in order to cause an entry in the other Master stations' LMS.

8.2.7 Wait_TCT

This state shall be entered after the "Claim_Token" state, "Check_Token_Pass" state or "Active_Idle" state when a ASP message was passed if the Master is in the isochronous

mode. In this state the MAC shall pass as much ASP messages as spare time is available (T_{RCT} < (T_{CT}-T_{PSP})). If no time available to pass an ASP messages then the MAC shall

a) load the passive-spare-time-timer with T_{CT}–T_{RCT},

b) start the passive-spare-time-timer,

c) wait until the passive-spare-time-timer expires, and

d) pass a SYNCH message for synchronization purposes.

The MAC shall indicate the sending of the SYNCH message to the local DLMS-user by a Synch event. If the SYNCH message could not be sent within a defined maximum time shift (maxT_{SH}) then the MAC shall pass a Synch_Delay event to the local DLMS-user in addition.

8.2.8 Use_Token

The MAC enters the "Use_Token" state in order to process high priority or low priority message transfer periods.

The MAC shall enter the

a) "Await_Data_Response" state after each transmitted send/request DLPDU with acknowledgement or response and start the slot-timer (see 5.5.5), or

b) "Check_Access_Time" state in the case of a SDN or CS service is transmitted, or

c) "Await_Status_Response" state in the case of a "Request DL Status with Reply" is transmitted.

8.2.9 Await_Data_Response

This state shall be entered in the case of the transmission of requests with acknowledgement or response. The MAC waits **one** slot time for the receipt of the acknowledgement or response DLPDU.

The MAC shall enter the

a) "Check_Access_Time" state in order to check the available token holding time for processing potential further requests, if:

   1) a valid acknowledgement DLPDU or valid response DLPDU addressed to the request's initiator or

   2) after the retry (retries) no valid acknowledgement or response is received and no more retry is possible. In this case the MAC shall notify the DLS-user accordingly. Further requests to this station are not repeated in case of errors, until a correct message transfer period (send/request data) has been performed;

b) "Use_Token state, if:

   1) an invalid DLPDU is received (start-, end-, length-, FCS error; start-, stop-, parity-bit error or slip error) and a retry is possible, or

   2) slot time expires (see 5.5) and a retry is possible;

   and shall retry the transmission of the send/request DLPDU.

Receipt of another valid DLPDU indicates that an error has occurred. The MAC shall enter the "Active_Idle" state and discard the received DLPDU.

8.2.10 Check_Access_Time

In this state the available token holding time shall be computed (see 5.5.5). Only if there is still token holding time available, the MAC may re-enter the "Use_Token" state. Otherwise the MAC shall enter the "Pass_Token" state.

If token holding time is available then the MAC shall transmit:

a) if high priority message are pending a high prior send/request DLPDU and shall enter the "Use_Token" state, or

b) if low priority message are pending a low prior send/request DLPDU and shall enter the "Use_Token" state, or

c) when the GAP update time has expired, but there is still token holding time T_{TH} available, then the MAC shall try to record one possible new station in the GAP before passing the token, in order to include it in the logical ring, if necessary. For that purpose, the MAC transmits a "Request DL Status with Reply" and enters the "Use_Token" state.

8.2.11 Pass_Token

In the "Pass_Token" state the MAC shall try to pass the token to the next station (NS) in the logical ring. When transmitting the token DLPDU, the MAC shall check by simultaneous monitoring if the transceiver is working correctly. If it does not receive its own token DLPDU, there is a fatal error in the transmit or receive channel. The MAC shall stop its activity in the logical ring, enter the "Offline" state and notify the DLMS-user.

If the MAC receives its own token DLPDU corrupted, this may be caused by a temporarily defective transmitter, receiver, or by the bus line. This error condition does not result in stopping activities in the first instance, instead the MAC shall enter the "Check_Token_Pass" state as after receiving the token DLPDU correctly.

Only after the token DLPDU has been retransmitted (due to no reaction from NS) and monitored as incorrect, the MAC shall stop its activity in the logical ring, enter the "Offline" state and notify the DLMS-user.

If the MAC has sent the token successfully then it shall enter the "Check_Token_Pass" state.

Only if no successor is known, that is, this DL-entity is at the moment the only active station on the bus, it shall pass the token to itself and then re-enter:

a) the "Use_Token" state if a high priority message available and has been sent or

b) the "Check_Access_Time" state if no high priority message available;

8.2.12 Check_Token_Pass

"Check_Token_Pass" is the state in which the MAC waits **one** slot time for a reaction of the station to which it has passed the token. This waiting time allows for the delay between the receipt of the token DLPDU and the ensuing transmission reaction of the addressed station.

If the MAC detects a valid DLPDU header within a slot time, it shall assume that the token passing was successful. The DLPDU shall be processed as if it were received in the "Active_Idle" state.

If an invalid DLPDU is detected within the slot time, the MAC shall assume that another station is active and therefore also enter the "Active_Idle" state.

If the MAC does not receive any DLPDU within one slot time, it shall pass the token to the successor again and re-enter the "Pass_Token" state and react as described in 5.3.2.1.

If the MAC does not receive any DLPDU within one slot time and it was the second pass token to the same successor then the MAC shall remove its NS from the LMS, pass the token to the new successor (NS of NS) and re-enter the "Pass_Token" state and react as described in 5.3.2.1.

If the MAC receives a Token with DA equal TS and the isochronous mode is active, then the MAC shall send an ASP message and shall enter the "Wait_TCT" state.

8.2.13 Await_Status_Response

This state is entered from the "Use_Token" state during GAP maintenance. In this state the MAC shall wait **one** slot time for an acknowledgement DLPDU.

The MAC shall enter the "Check_Access_Time" state:

a) if nothing is received during the slot time, or

b) a corrupted DLPDU is received.

If an existing station does not answer it shall be deleted from the GAPL, that is, be marked as an unused address. Then the MAC shall enter the "Check_Access_Time" state.

If a status request is answered by a new Slave station or a Master station that does not want to be included in the ring, that station shall be entered in the GAPL.

If the MAC receives a valid acknowledgement with "Master station ready to enter logical token ring" then a new Master station acknowledges that it wants to be included in the logical ring. The MAC shall shorten the own GAP to the new NS and shall pass the token to this station and enter the "Pass_Token" state.

If the MAC receives any other DLPDU instead of an acknowledgement DLPDU (indicates that multiple tokens may exist), it shall enter the "Active_Idle" state.

8.3 Clock synchronization protocol

8.3.1 Overview

Synchronization of clocks between devices on a fieldbus segment and a time master application is provided in parallel with other communication functions. A time master is a fieldbus master device. The scheme used is a "backwards time based correction". This results in very few real-time constraints being imposed on a field device. There is no requirement for generating messages at precise instants. Instead, knowledge of when a special timer event message has been broadcasted is subsequently distributed and used to calculate appropriate clock adjustments.

There is no confirmation of correct reception at the remote DLE, as neither acknowledgements are given nor local retries take place. Once the data is sent it reaches all remote DLEs at the same time (not taking into account signal propagation time, Physical Layer delays and Data Link Layer delays). For the correct and secure time calculation the time transfer will take place in a sequence of two messages: the first transfer (Time Event) and the second transfer (Clock Value). By receiving the Timer Event, every remote DLE starts its receive-delay-timer. By receiving the clock value, the value of the receive-delay-timer is send within the indication. Each addressed remote DLE that has received the clock value error-free passes it to the DLS-user by means of a DL-CS-CLOCK-VALUE indication primitive (see IEC 61158-3-3 fieldbus connectionless-mode Data Link Service, Clock Synchronization service). If errors occur by a violation of the sequence (e. g. a Timer Event is received twice or expiration of the Clock Synchronization Interval Timer), this error is notified to the DLS-user.

8.3.2 State machine time master

The clock synchronization sequence consists of two messages broadcasted by the time master. When the first message, called Time Event, is broadcast, all of the DLEs receiving it start a receive-delay-timer. The time master then sends a second message, the Clock Value, which contains the actual time when the Time Event was sent. Upon reception of this message, the remote DLEs can calculate their receive delay time value. In conjunction with

the received clock value each DLS-users is able to back calculate the Time Event message reception time according to its own clock, and compare it with the value distributed in the clock message. Their clocks may then be adjusted to agree with the time master's. In order to support scheduling and to check that message pairs Time Event / Clock Value are matched, Clock Value also contain the time at which the last clock synchronization occurred, which is the time sent at the last Time Event.

Special DL services exist to support clock synchronization. A station's clock is adjusted by the DLS-user, but precise measurement of the time when sending or receiving the Time Event must be done in the DLE. Therefor the DLE uses special timers to measure the send delay respective receive delay.

On the sending side, the DLE in the time master starts its DL timer - the send-delay-timer - by receiving a DL-CS-TIME-EVENT request from the DLS-user. The DLE will send a TE DLPDU and stop the send-delay-timer when the last bit is sent. It will then deliver a confirmation to the DLS-user that the message was sent, including a service parameter which reports the calculated value of the send delay time. The DLS-user then adds this value to the time to determine when the Time Event was actually sent. A Clock Value DLPDU which includes this time is then broadcast to remote DLEs.

On the receiving side, the DLE starts its timer - the receive-delay-timer - when it receives the TE DLPDU. When the clock value is received (CV DLPDU), the receive-delay-timer is stopped and an indication is delivered to the DLS-user together with the calculated receive delay time to determine when the time message was received according to its clock.

The clock synchronization sequence supports optionally redundant time masters. Although only one is normally active at a given time, when switchover occurs more than one time master may be broadcasting the Time Event and Clock Value messages. Therefore, when the Time Event is received, the source station address is saved by the receiving DLEs. CV DLPDUs from other masters will be ignored.

Figure 39 illustrates an overview of the Clock Synchronization.

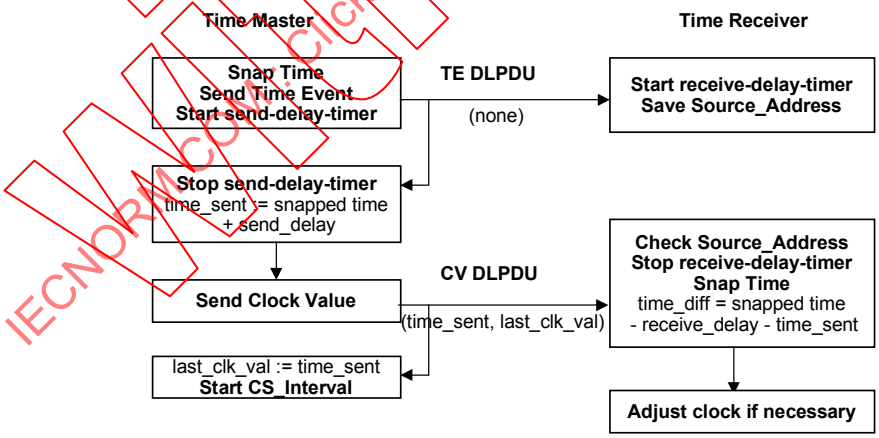


Figure 39 – Overview of clock synchronization

The clock synchronization protocol provides optionally redundant time masters, and a simple election procedure for determining the active time master. One time master on each link is presumed to be the primary time master. Other time masters are standby time masters that can act as sources of clock synchronization when the primary time master is not active. Standby time masters are not ranked in order of importance. The clock synchronization protocol will always select the primary time master as the clock synchronization source if it is present. The figure below is the state machine definition for the operation of a primary time master (TM) or a standby time master.

Standby time masters remain inactive as long as they receive clock synchronization messages from the active time master within a monitoring window. The monitoring window is restarted at each received clock synchronization message. The length of the monitoring window is set to 4 × T_{CSI}, a multiple of the Clock Synchronization Interval Time.

If no clock synchronization message is received during the monitoring window it is presumed that the active time master has failed. Standby time master then begin sending clock synchronization messages.

If a standby time master receives a clock synchronization message from another standby time master, then it will fall back to monitoring clock synchronization sequences. Messages received from the primary time master always result in the standby time master falling back to monitoring (see Figure 40).

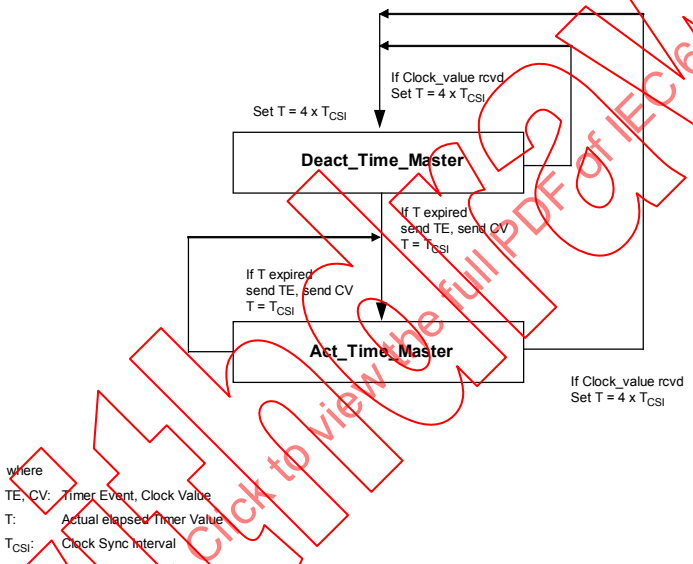


Figure 40 – Time master state machine

8.3.3 State machine time receiver

The receiver state machine is described in Figure 41. The receiver monitors clock messages within a time window of a multiple of the Clock Synchronization Interval Time 4 × T_{CSI}.

It adjusts, if necessary, its clock after reception of a correct clock synchronization sequence. No clock adjustment is made and errors are marked in following cases .

a) Different source addresses in Time Event and Clock Value

b) No clock synchronization message received within time window

c) Clock values not matched (different last_clk_val).

The handling of the Time Event, checking the source address of TE/CV DLPDU and the receive-delay-timer is part of the DLE.

With the reception of a valid Clock Value (SA is the same as the SA in the previously received Timer Event) , a DL service gives the received synchronization time and the receiver part of

the communication delay to the DLS-user. The receiver part of the communication delay includes the time past since the receive-delay-timer was started with the reception of a TE DLPDU till read from the DLE after the reception of a CV DLPDU.

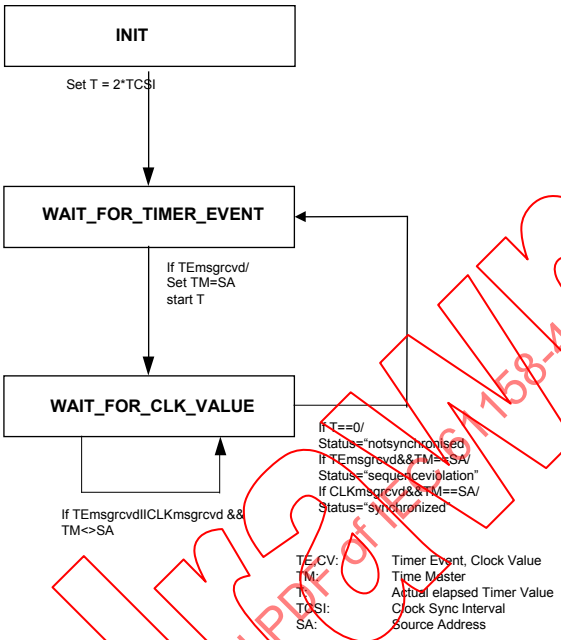


Figure 41 – Time receiver state machine

Figure 42 illustrates the Clock Synchronization sequences.

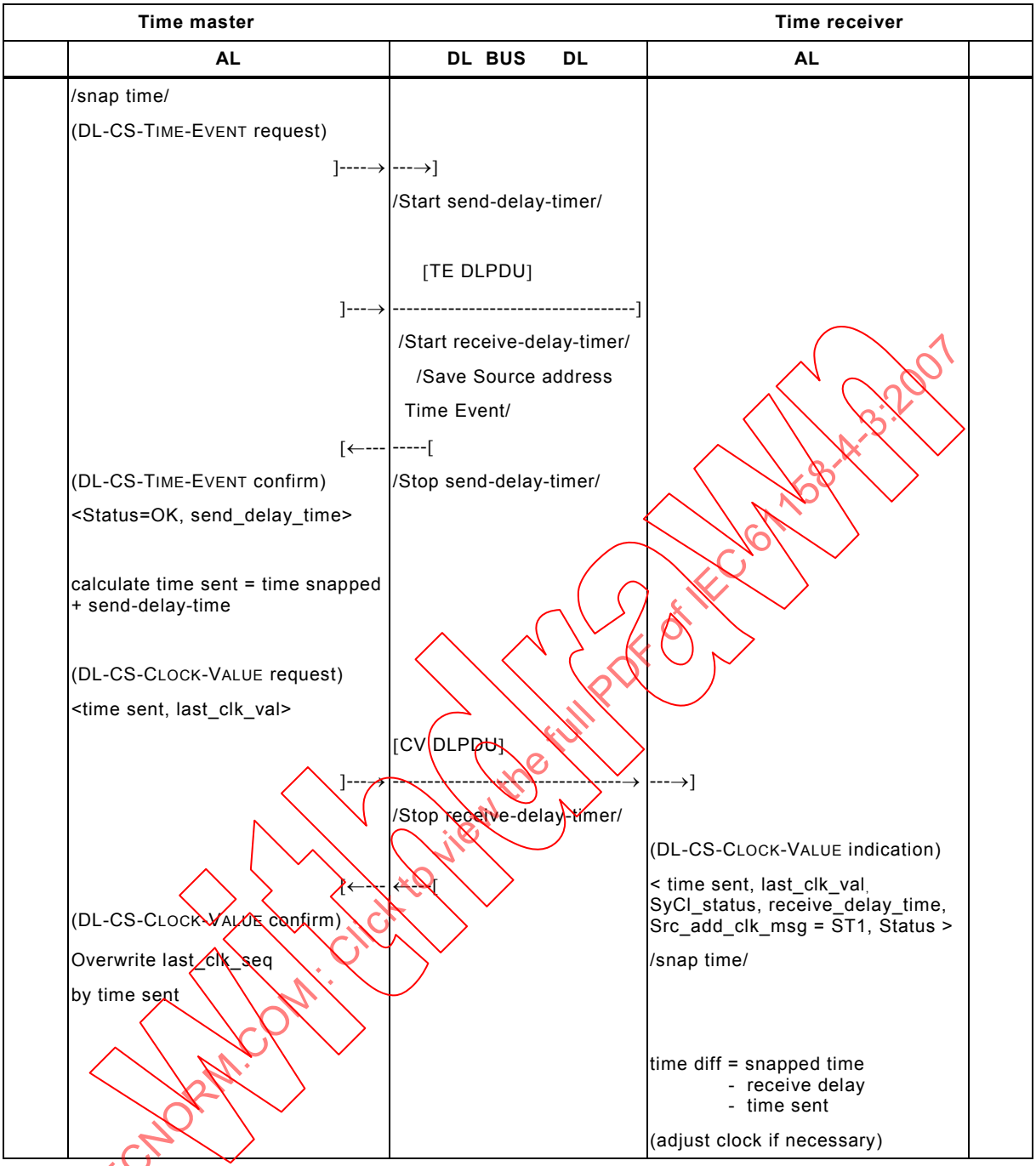


Figure 42 – Clock synchronization

Annex A
(normative)
–
DL-Protocol state machines

NOTE 1 This annex specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. This specification is complementary to the textual specification in the body of this standard; in case of conflict the requirements of this Annex take precedence.

NOTE 2 The finite state machine descriptions given here are necessarily less than a complete description of an implementation. Additional requirements and considerations are found in the textual specification.

A.1 Overall structure

The DL protocol of Type 3 consist of the following three parts:

- handling of the services invoked by Application Layer (DL- and DLM-services);
- media access control (token handling and service interactions);
- interface to physical layer with PDU assembling/disassembling.

The Interface between the service handler (FLC/DLM) and Media Access Control (MAC) consists of a set of Queues, a SAP-list and a DL-Data-Resource. (This will be used also by the Send-Receive Unit –SRU.)

The Interface between MAC and SRU consist of a set of services (see A.6.5).

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The textual listing of the state transitions is structured as follows:

The first row contains the name of the transition.

The second row in define the current state.

The third row contains an optional event followed by Conditions starting with a “/” as first line character and finally followed by the Actions starting with a “=>” as first line character.

The last row contains the next state.

If the event occurs and the conditions are fulfilled the transition fires, that is, the actions are executed and the next state is entered.

Additional conventions for state machines are given in IEC 61158-6-3.

Figure A.1 illustrates the structuring of the Protocol Machines.

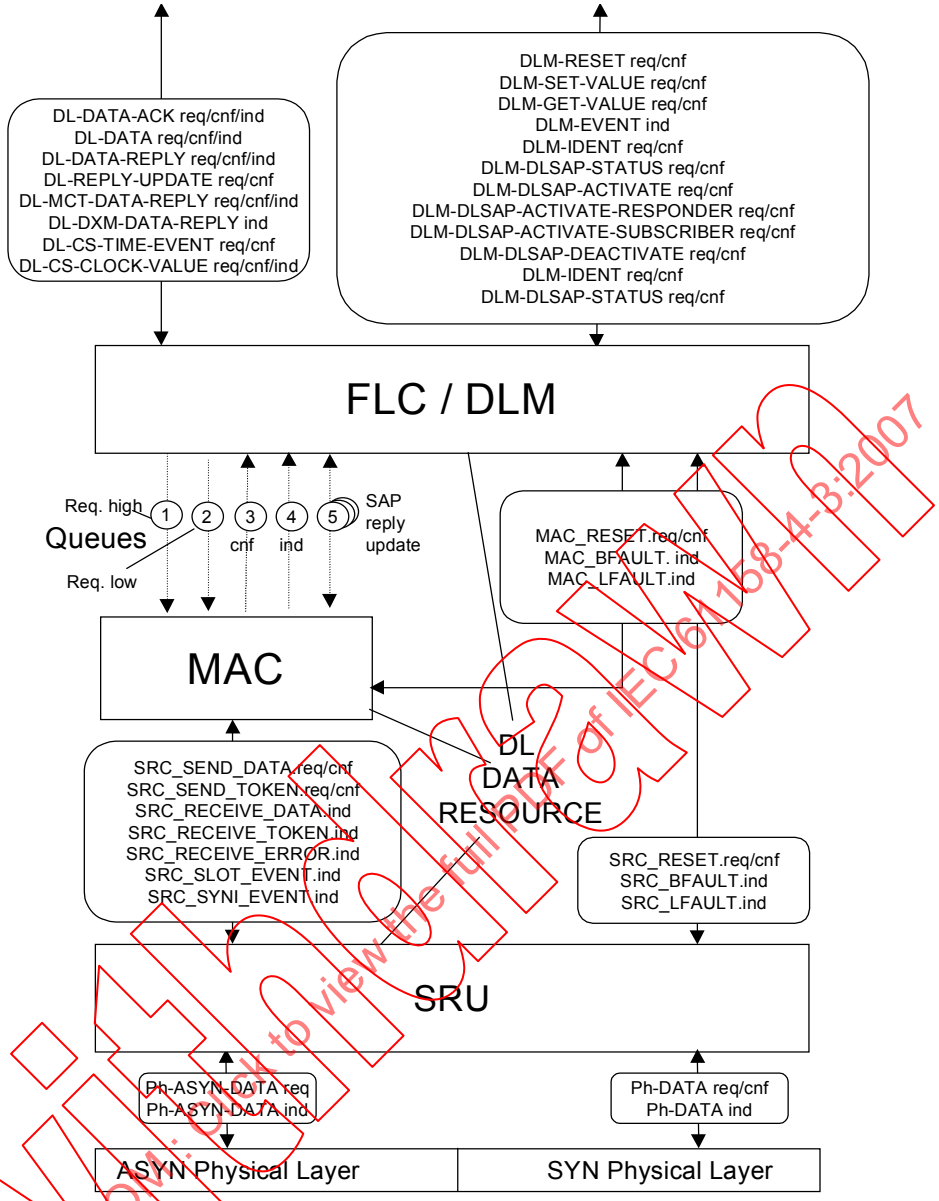


Figure A.1 – Structuring of the protocol machines

A.2 Variation of state machines in different devices

Table A.1 shows the assignment of State Machines parts to devices. The layout of the state machine tables is defined in IEC 61158-6-3, state machine conventions.

Table A.1 – Assignment of state machines

	Machine	Options	Remarks
Slave	FLC	SDN/SDA/SRD/CS/MSRD Indication SRD reply-update Req/Con	
		DLSAP-Activate/Deactivate	
	DLM	Limited set of FDL-variables No DLSAP-Activate-service	
	MAC	Only OFFLINE- and PASSIVE-IDLE-State	support for Data rate-detection
	SRU	No Token No Request (sending) No Response (receiving)	
Master	FLC	Full set of services	
	DLM	Full set of services	
	MAC	Full set of states	
	SRU	Full set of services/states	
Only Master	MAC	Passive idle omitted No active/passive switching	Other option: Passive but no switching

A.3 DL Data Resource

The DL Data Resource models a data interface for all state machines of the Data Link Layer. It contains queues and buffers for exchanging data between the sublayers of the DL. Moreover it contains management information, which have to be accessed by various sublayers of the DL. Table A.2 illustrates all Data Resources of the Data Link Layer.

Table A.2 – Data resource

Name	Struct element	Type	Range	Remark
FDL-Variables		S0		
SAP_List		[0..63, CS, NIL] of S1		
H_List		S2		
L_List		S2		
C_List		S3		
I_List		S3		
Ident_List		S12		
S0	**Variables**			
	TS	U8	0 to 126	DL-address of this station
	Data_rate		9,6; 19,2; 31,25; 45,45; 93,75; 187,5; 500; 1500, 3000; 6000; 12000 kBit/s and others	Data rate of this fieldbus
	Medium_ redundancy		single; redundant	Availability of redundant media
	HW-Release		LE_HR; hardware release identification	Hardware release number
	SW-Release		LE_SR; software release identification	Software release number
	SYNCHT		DSAP SSAP LSDU	Contents of the SYNCH DLPDU
	Tct	U32	1 to 2^{24}-1 (bit times)	Isochronous cycle time
	Isochronous_ mode		0 1 2 3	non Isochronous with cycle correction with cycle drop TTR timer stop
	maxTsh	U8	1 to 256 (bit times)	maximal time shift
	Tcsi	U32	1 to 2^{32}-1 (bit times)	Clock Synchronization Interval Time
	Preamble_length	U8	8 to 64 (bit)	Length of the preamble of physical frames at synchronous transmission
	Tsyn	U8	4 to 32 (bit times)	post transmission gap time of synchronous transmission
	Tsl	U16	52 to 2^{16}-1 (bit times)	Slot time
	minTsdr	U16	2^0 to 2^{16}-1 (bit times)	Smallest station delay time
	maxTsdr	U16	2^0 to 2^{16}-1 (bit times)	Largest station delay time
	Tqui	U8	0 to 2^8-1 (bit times)	Transmitter fall time (line state uncertain time) or repeater switch time
	Tset	U8	2^0 to 2^8-1 (bit times)	Setup time
	Ttr	U32	2^0 to 2^{24}-1 (bit times)	Target rotation time
	G	U8	1 to 100	GAP update factor
	in_ring_desired	Bool	true; false	Request entry into or exit out of the logical token ring

Name	Struct element	Type	Range	Remark
	HSA	U8	0 to 126	Highest station address on this fieldbus
	max_retry_limit	U8	0 to 15 (preferably 0)	Maximum number of retries
	DLPDU_sent_ count	U32	0 to 2^{32}-1	Number of DLPDUs sent
	Retry_count	U16	0 to 2^{16}-1	Number of DLPDU repeats
	DLPDU_sent_ count_sr	[0..126] of U32	max. 126 entries of 0 to 2^{32}-1	List of numbers of DLPDUs sent per station
	Error_count	[0..126] of U16	max. 126 entries of 0 to 2^{16}-1	List of numbers of no or erroneous responses per station
	SD_count	U32	0 to 2^{32}-1	Number of correct start delimiters
	SD_error_count	U16	0 to 2^{16}-1	Number of defective start delimiters
	Trr	U32	2^0 to 2^{24}-1 (bit times)	Real rotation time
	LMS	[0..126] of U8	up to 127 DL-addresses	List of Master stations in the logical ring
	GAPL	[0..126] of DLE status	max. 126 DL-addresses (0 to 126) inclusive DLE status	List all of stations in the own GAP
	Cycle_violation_ count	U32		Number of cycle violations which occurred since the start of isochronous mode
	Tid1	U16	33 bit + 2 bit + 2 × Tset + Tqui	Deduced Variables
	Tid2	U16	max (Tid1, maxTsdr)	
S1	**SAP_List**			
	Access	U8	0..126, NIL	
	Function_List_R	List of		SDN_H/L,SDA_H/L,SRD_H/L, MSRD_H/L, CS(TE/CV)
	Function_List_I	List of		SDN_H/L, SDA_H/L,SRD_H/L, MSRD_H/L, CS(TE/CV)
	LenList	[0..16] of U8	0..246	List of permitted DLSDU lengths depending on FC.Function
	Indication_mode		ALL/DATA/ UNCHANGED	
	Publisher_ enabled		TRUE/FALSE	
	Ibuffer	S7		Indication Buffer
	Ubuffer	S8		Update Buffer
	Sbuffer	S7		Subscriber Buffer
S2	**H_List/L_List**			
	Num_entry	U16		
	First_entry	Ref to S9		
	Last_entry	Ref to S9		
	Insert()			
	Remove()			

Name	Struct element	Type	Range	Remark
S3	**C_List/I_List**			
	Num_entry	U16		
	First_entry	Ref to S10		
	Last_entry	Ref to S10		
	Insert()			
	Remove()			
S4	**REQM**			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	Serv_class		high/low	
	Conf	Bool		
S5	**RESM**			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
S6	**FC**			
	Function		req rsp	SDN_H/L, SDA_H/L, SRD_H/L, MSRD_H/L, Ident, LSAP_Status, CS(TE/CV)
	Frame_type		req/rsp	
	FCB	U8	0,1	
	FCV	U8	0,1	
	Stn-type		Slave, M_n_rdy, M_rdy, M_in_ring	
S7	**Ibuffer**			
	Num_entry	U16		
	First_entry	Ref to S10		
	Last_entry	Ref to S10		
	Insert()			
	Remove()			
S8	**Ubuffer**			
	High_reference	U32		
	High_buffer	S11		
	High_transmit		SINGLE/MULTIPLE	
	Low_reference	U32		
	Low_buffer	S11		
	Low_transmit		SINGLE/MULTIPLE	

Name	Struct element	Type	Range	Remark
S9	**H/L_List entry**			
	Next entry	Ref to S9		
	DA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
S10	**C/I_List entry**			
	Next entry	Ref to S10		
	DA	U8	0..127	
	SA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
	R_Status	NO,DL,DH,OK,RS, RR,UE,NR,RDH, RDL,NA,DS		
	Reference	U32		
S11	**DLSDU**			
	Len	U8	0..246	
	Data	[0..246] of U8		
S12	**Ident_List**			
	Vendor_Name	[0..32] of U8		
	Model_Name	[0..32] of U8		
	HW_Release	[0..32] of U8		
	SW_Release	[0..32] of U8		

A.4 FLC / DLM

A.4.1 Primitive definitions

A.4.1.1 Primitive exchanged between DL-User and FLC

Table A.3 shows all primitives issued by DL-User to the FLC. See IEC 61158-3-3 for a description of the functions.

Table A.3 – Primitives issued by DL-User to FLC

Primitive name	Associated parameters
DL-DATA-ACK request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA-REPLY request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-MCT-DATA-REPLY request	Service_class, D_addr D_SAP_index, S_SAP_index, DLSDU,
DL-REPLY-UPDATE request	Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference
DL-CS-TIME-EVENT request	D_addr, D_SAP_index, S_SAP_index
DL-CS-CLOCK-VALUE request	D_addr, D_SAP_index, S_SAP_index, DLSDU

Table A.4 shows all primitives issued by the FLC to the DL-User. See IEC 61158-3-3 for a description of the functions.

Table A.4 – Primitives issued by FLC to DL-User

Primitive name	Associated parameters
DL-DATA-ACK confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status

Primitive name	Associated parameters
DL-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-REPLY-UPDATE confirm	Service_class, S_SAP_index, DL_status
DL-MCT-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-CS-TIME-EVENT confirm	D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status
DL-CS-CLOCK-VALUE confirm	D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA-ACK indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference
DL-MCT-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference
DL-DXM-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-CS-CLOCK-VALUE indication	D_addr, D_SAP_index, S-addr, S_SAP_index, DLSDU, Receive_delay_time CS_status

A.4.1.2 Primitive exchanged between DL-User and DLM

Table A.5 shows all primitives issued by the DL-User to the DLM. See IEC 61158-3-3 for a description of the functions.

Table A.5 – Primitives issued by DL-User to DLM

Primitive name	Associated parameters
DLM-RESET request	(none)
DLM-SET-VALUE request	Variable_name(1 to n), Index(1 to k), Desired_value (1 to n)
DLM-GET-VALUE request	Variable_name(1 to n), Index(1 to k)
DLM-DLSAP-ACTIVATE request	S_SAP_index, Access, Service_list
DLM-DLSAP-ACTIVATE-RESPONDER request	S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled
DLM-DLSAP-ACTIVATE-SUBSCRIBER request	S_SAP_index, DLSDU_length_list
DLM-DLSAP-DEACTIVATE request	S_SAP_index
DLM-IDENT request	DL_addr
DLM-DLSAP-STATUS request	D_SAP_index, DL_addr

Table A.6 shows all primitives issued by the DLM to the DL-User. See IEC 61158-3-3 for a description of the functions.

Table A.6 – Primitives issued by DLM to DL-User

Primitive name	Associated parameters
DLM-RESET confirm	DLM_Status
DLM-SET-VALUE confirm	Current_value(1 to n), DLM_status(1 to n)
DLM-GET-VALUE confirm	Desired_value(1 to n), DLM_status(1 to n)
DLM-EVENT indicate	Event/Fault TSH
DLM-DLSAP-ACTIVATE confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-RESPONDER confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-SUBSCRIBER confirm	S_SAP_index, DLM_status
DLM-DLSAP-DEACTIVATE confirm	S_SAP_index, DLM_status
DLM-IDENT confirm	DL-addr, Ident_list, DLM_status
DLM-DLSAP-STATUS confirm	D_SAP_index, DL-addr, Access, Service_type(1 to n), Role_in_service_list(1 to n), DLM_status

A.4.1.3 Parameters of FLC Primitives

Table A.7 shows all parameters used with primitives between the DL-User and the FLC.

Table A.7 – Parameters used with primitives exchanged between DL-User and FLC

Parameter name	Description
S_SAP_index	Identifier of the local Service Access Point
D_SAP_index	Identifier of a remote Service Access Point
D_addr	Station address of the receiving DLE
S_addr	Station address of the sending DLE
Service_class	Indicates priority of the service
DL_status	Status of the service execution
DLSDU	Data Unit from or to the DLS-user
Transmit_strategy	Operation mode of the update buffer (single = buffer is transmitted once, multiple = buffer is transmitted repeated)
Reference	Reference to identify the DLSDU passed to the local DLE
Update_status	Indicates the presence of a DLSDU from a remote DLE in the update buffer
Send_delay_time	Protocol delay time between the invocaton of the Clock Synchronization service request primitive and the transmission of the DLPDU
Receive_delay_time	Protocol delay time between the reception of the previous and the current DLPDU of Clock Synchronization
CS_status	Status of the Clock Synchronization sequence

A.4.1.4 Parameters of DLM primitives

Table A.8 shows all parameters used with primitives between the DL-User and the DLM.

Table A.8 – Parameters used with primitives exchanged between DL-User and DLM

Parameter name	Description
S_SAP_index	Identifier of the local Service Access Point
D_SAP_index	Identifier of a remote Service Access Point
Variable_name(1 to n)	List of DL variable names
Desired_value (1 to n)	List of DL variable values
Current_value (1 to n)	List of DL variable values
Access	Permission to use of a local SAP by one or several remote DLEs
Service_list	Type of services accepted at a local SAP
DLSDU_length_list	Maximum length of incoming or outgoing DLSDU at a local SAP
Indication_mode	Mode for Indication of received DLPDUs without user data
DLM_status	Status of the service execution
Event/Fault	Indicates the cause of an event or a fault
DL-addr	Station address of the local or a remote DLE
Ident_list	Specifies Vendor Name, Controller Type; HW/SW-Release
Service_type(1 to n)	Specifies the DL-services activated
Role_in_service_list(1 to n)	Specifies the role in service (initiator or responder or both)

A.4.2 State machine description

The FLC forms the interface between DL Protocol Machines and DL-User for SDA, SDN, SRD, MSRD and CS services.

The DLM forms the interface between DLM Protocol Machines and DLM-User for management services.

The FLC services are processed in the same way as the DLM services. For this reason the DLM module is described in the same state machine as the FLC module and is contained in the following description.

The services are all handled in the READY-State.

The service request will be validated first. A negative validation will result in a negative confirmation. Otherwise the service will be put into the high or low priority service queue according to the parameter service class. Services executed by MAC will be moved from the request queues to the confirmation queues. All valid incoming services will be put into the indication queue.

The set of SAP (de-)activate services are used to organise the SAP list. The Reply Update services are used to put reply data in the service list.

Local Variables

The FLC / DLM does not use local variables. All variables which have to be accessed by the FLC / DLM are contained in the DL Data Resource.

State Table Nomenclature

The standard suffixes ".req", ".cnf" and ".ind" are used to indicate the request, confirm and indication primitives, respectively. Service primitive names are entirely upper-case.

A.4.3 FLC / DLM Table

The FLC and DLM State Table is shown in Table A.9.

Table A.9 – FLC/DLM state table

No.	Current state	Event /condition ⇒action	Next state
1	READY	DLM-RESET.req ⇒ MAC_reset := FALSE SRC_reset := FALSE MAC_RESET.req SRC_RESET.req	WAIT_RESET_CNF
2	WAIT_RESET_CNF	MAC_RESET.cnf /SRC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
3	WAIT_RESET_CNF	MAC_RESET.cnf /!SRC_reset ⇒ MAC_reset := TRUE	WAIT_RESET_CNF
4	WAIT_RESET_CNF	SRC_RESET.cnf /MAC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
5	WAIT_RESET_CNF	SRC_RESET.cnf /!MAC_reset ⇒ SRC_reset := TRUE	WAIT_RESET_CNF
6	READY	DLM-GET-VALUE.req (Variable_name((1 to n), Index(1 to k))) /!CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k))) ⇒ DLM_status (1 to n) := IV Current_value (1 to n) := NIL DLM-GET-VALUE.cnf (Current_value(1 to n),DLM_status(1 to n))	READY
7	READY	DLM-GET-VALUE.req (Variable_name((1 to n), Index(1 to k))) /CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k))) ⇒ Check_variable_names (Variable_name((1 to n), Index(1 to k))) Set_current_values (Variable_name((1 to n), Index(1 to k))) DLM-GET-VALUE.cnf (Current_value(1 to n),DLM_status(1 to n))	READY
8	READY	DLM-SET-VALUE.req (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) /!CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) ⇒ DLM_status (1 to n) := IV DLM-SET-VALUE.cnf (DLM_status (1 to n))	READY
9	READY	DLM-SET-VALUE.req (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) /CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) ⇒ Set_variable_list (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) DLM-SET-VALUE.cnf (DLM_status (1 to n))	READY

No.	Current state	Event /condition ⇒action	Next state
10	READY	MAC_BFAULT.ind (Fault_type, Tsh) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault, Tsh)	READY
11	READY	MAC_BFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY
12	READY	MAC_LFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY
13	READY	SRC_BFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
14	READY	SRC_LFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
15	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /!CHECK_PAR_DLSAP ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
16	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
17	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && !ACTIVATED (S_SAP_index) ⇒ SET_SAP_LIST (S_SAP_index, Access, Service_list) DLM_status := OK DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
18	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /!CHECK_PAR_DLSAP_RES ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
19	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
20	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY

No.	Current state	Event /condition ⇒action	Next state
21	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !RACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
22	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
23	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && !CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
24	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /!CHECK_PAR_DLSAP_SUB ⇒ DLM_status := IV DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
25	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && SACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
26	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && !SACTIVATED (S_SAP_index) ⇒ SET_SSAP_LIST (S_SAP_index, DLSDU_length_list) DLM_status := OK DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
27	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /!CHECK_PAR_DLSAP_DEACT ⇒ DLM_status := IV DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
28	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && !ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
29	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && ACTIVATED (S_SAP_index) ⇒ RESET_SAP_LIST (S_SAP_index) DLM_status := OK DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY

No.	Current state	Event /condition ⇒action	Next state
30	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /!CHECK_PAR_STATUS ⇒ Access := NIL Service_list (1 to n) := NIL Role_in_service_list (1 to n) := NIL DLM_status := IV DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
31	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List[D_SAP_index] = NIL ⇒ Access := NIL Service_list (1 to n) := NIL Role_in_service_list (1 to n) := NIL DLM_status := LR DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
32	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List ≠ NIL[D_SAP_index] ⇒ Write_SAPstatus_list() DLM_status := OK DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
33	READY	DLM-IDENT.req (D_addr) /!CHECK_PAR_IDENT ⇒ Ident_list := NIL DLM_status := IV DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
34	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List = NIL ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
35	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List ≠ NIL ⇒ Write_ident_list() DLM_status := OK DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
36	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = TRUE && D_addr ≠ Variables.TS ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY

No.	Current state	Event /condition ⇒action	Next state
37	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = FALSE && D_addr ≠ Variables.TS ⇒ Ident_Pending := TRUE PUT_LREQ (IDENT)	READY
38	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_SDN ⇒ DL_status := IV DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
39	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && !CHECK_SAP(Service_class, S_SAP_index, SDN) ⇒ DL_status := LS DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
40	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
41	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SDN_H)	READY
42	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SDN_L)	READY
43	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_SDA ⇒ DL_status := IV DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
44	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && !CHECK_SAP(Service_class, S_SAP_index, SDA) ⇒ DL_status := LS DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
45	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
46	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = high ⇒ PUT_HREQ (SDA_H)	READY
47	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = low ⇒ PUT_LREQ (SDA_L)	READY

No.	Current state	Event /condition ⇒action	Next state
48	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_REPLY ⇒ DL_status := IV DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
49	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
50	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
51	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SRD_H)	READY
52	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SRD_L)	READY
53	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_REPLY ⇒ DL_status := IV DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
54	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
55	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
56	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (MSRD_H)	READY
57	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_HREQ (MSRD_L)	READY

No.	Current state	Event /condition ⇒action	Next state
58	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /!CHECK_PAR_UPDATE ⇒ DL_status := IV DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
59	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
60	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) / CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
61	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ DL_status = OK PUT_HUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
62	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ DL_status = OK PUT_LUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
63	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = IDENT ⇒ Ident_pending := FALSE GET_IDENT_CNF() DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
64	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = DLSAP-STATUS ⇒ Status_pending := FALSE GET_DLSAPSTATUS_CNF() DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
65	READY	/C_List.Num_entry≠0 && C_List.First_entry.Function = SDA_H || C_List.First_entry.Function = SDA_L ⇒ GET_SDA/SDN_CNF() DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
66	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SDN ⇒ GET_SDA/SDN_CNF() DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
67	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SRD ⇒ GET_SRD_CNF() DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY

No.	Current state	Event /condition ⇒action	Next state
68	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = MSRD ⇒ GET_SRD_CNF() DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
69	READY	/I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = SDA_H ‖ I_List.First_entry.FC.Function = SDA_L ⇒ GET_SDA/SDN_IND() DL-DATA-ACK.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY
70	READY	/I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function= SDN_H ‖ I_List.First_entry.FC.Function = SDN_L ⇒ GET_SDA/SDN_IND() DL-DATA.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY
71	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA ≠ 127 && I_List_entry.First_entry.FC.Function= SRD_H ‖ I_List.First_entry.FC.Function = SRD_L ⇒ GET_SRD_IND() DL-DATA-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference)	READY
72	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA = 127&& I_List_entry.First_entry.FC.Function= SRD_H ‖ I_List.First_entry.FC.Function = SRD_L ⇒ GET_DXM_IND() DL-DXM-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY
73	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA ≠ 127 && I_List_entry.First_entry.FC.Function= MSRD_H ‖ I_List.First_entry.FC.Function = MSRD_L ⇒ GET_SRD_IND() DL-MCT-DATA-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference)	READY
101	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &!CHECK_PAR_TE ⇒ DL_status:=IV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
102	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && !CHECK_SAP(High, CS, TE/CV) ⇒ DL_status:=LS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
103	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && CHECK_SAP(High, CS, TE/CV) && !RESOURCE(CS) ⇒ DL_status:=LR DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY

No.	Current state	Event /condition ⇒action	Next state
104	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && CHECK_SAP(High, CS, TE/CV) && RESOURCE(CS) ⇒ TSDT.start(2*Tcsi); PUT_HREQ (TE) Tm_State:=W_STE	READY
105	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==STE ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
106	READY	/Tm_State==STE &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=CONFLICT	READY
107	READY	/Tm_State==W_STE &C_List.Num_entry≠0 && C_List.First_entry.Service=TE && C_List.First_entry.R_status=OK ⇒ Send_delay_time := 2*Tcsi-TSDT.cv; TSDT.stop; GET_TE_CNF(); Status:=OK DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=SCV	READY
108	READY	/Tm_State==W_STE &C_List.Num_entry≠0 && C_List.First_entry.Service=TE && C_List.First_entry.R_status≠OK ⇒ Send_delay_time := 2*Tcsi-TSDT.cv; TSDT.stop; GET_TE_CNF(); Status:=DS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=STE	READY
109	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==W_STE ⇒ Send_delay_time := 0 DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=W_STE	READY
110	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==W_STE ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=W_STE	READY
111	READY	/Tm_State==W_STE &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=W_CC	READY

No.	Current state	Event /condition ⇒action	Next state
112	READY	/Tm_State==W_STE &TSDT expired ⇒ TSDT.start(4*Tcsi); Tm_State:=W_DS	READY
113	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &!CHECK_PAR_CV ⇒ DL_status:=IV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
114	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && !CHECK_SAP(High, CS, TE/CV) ⇒ DL_status:=LS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
115	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && CHECK_SAP(High, CS, TE/CV) && !RESOURCE(CS) ⇒ DL_status:=LR DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
116	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && CHECK_SAP(High, CS, TE/CV) && RESOURCE(CS) ⇒ DLSDU:=CS_list; TSDT.start(2*Tcsi); PUT_HREQ (CV) Tm_State:=W_SCV	READY
117	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==SCV ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=SCV	READY
118	READY	/Tm_State==SCV &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=CONFLICT	READY
119	READY	/Tm_State==W_SCV &C_List.Num_entry≠0 && C_List.First_entry.Service=CV && C_List.First_entry.R_status=OK ⇒ GET_CV_CNF(); Status:=OK DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY

No.	Current state	Event /condition ⇒action	Next state
120	READY	/Tm_State==W_SCV &C_List.Num_entry≠0 && C_List.First_entry.Service=CV && C_List.First_entry.R_status≠OK ⇒ GET_CV_CNF(); Status:=DS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
121	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==W_SCV ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=W_SCV	READY
122	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==W_SCV ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=W_SCV	READY
123	READY	/Tm_State==W_SCV &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=W_CC	READY
124	READY	/Tm_State==W_STE &TSDT expired ⇒ TSDT.start(4*Tcsi); Tm_State:=W_DS	READY
125	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==CONFLICT ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=CONFLICT	READY
126	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==CONFLICT ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=CONFLICT	READY
127	READY	/Tm_State==CONFLICT &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi) Tm_State:=CONFLICT	READY
128	READY	/Tm_State==CONFLICT &TSDT expired ⇒ Tm_State:=STE	READY

No.	Current state	Event /condition ⇒action	Next state
129	READY	/Tm_State==W_CC &C_List.Num_entry≠0 && C_List.First_entry.Service=TE ⇒ Send_delay_time := 0; GET_TE_CNF(); Status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=CONFLICT	READY
130	READY	/Tm_State==W_CC &C_List.Num_entry≠0 && C_List.First_entry.Service=CV ⇒ GET_CV_CNF(); Status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=CONFLICT	READY
131	READY	/Tm_State==W_DS &C_List.Num_entry≠0 && C_List.First_entry.Service=TE ⇒ GET_TE_CNF(); Status:=DS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
132	READY	/Tm_State==W_DS &C_List.Num_entry≠0 && C_List.First_entry.Service=CV ⇒ GET_CV_CNF(); Status:=DS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
201	READY	/Tr_State==W_TE &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE ⇒ TRDT.start(2*Tcsi); GET_TE/CV_IND(); TM:=S_addr Tr_State:=W_CV	READY
202	READY	/Tr_State==W_TE &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV ⇒ GET_TE/CV_IND() Tr_State:=W_TE	READY
203	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE && I_List_entry.First_entry.SA ≠ TM ⇒ GET_TE/CV_IND() Tr_State:=W_CV	READY
204	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE && I_List_entry.First_entry.SA = TM ⇒ TRDT.stop; Receive_delay_time := 2*Tcsi – TRDT.cv; CS_list := DLSDU; CS_status:=SV; GET_TE/CV_IND() DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY

No.	Current state	Event /condition ⇒action	Next state
205	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV && I_List_entry.First_entry.SA ≠ TM ⇒ GET_TE/CV_IND() Tr_State:=W_CV	READY
206	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV && I_List_entry.First_entry.SA = TM ⇒ TRDT.stop; Receive_delay_time := 2*Tcsi – TRDT.cv; CS_list := DLSDU; CS_status:=SV; GET_TE/CV_IND() DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY
207	READY	/Tr_State==W_CV &TCSI expired ⇒ Receive_delay_time := 0; CS_list := NULL; CS_status:=SV DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY

A.4.4 Functions

The FLC and DLM Functions are summarized in Table A.10.

Table A.10 – FLC / DLM function table

Function name	Operations
ACTIVATED (S_SAP_index)	Check that SAP is activated by setting its entry to valid value. SAP_List(S_SAP_index) ≠ NIL
RACTIVATED (S_SAP_index)	Check that SAP is activated as Responder.
SACTIVATED (S_SAP_index)	Check that SAP is activated as Subscriber.
CHECK_PAR_DLSAP	Check that all parameters of Service DLM-DLSAP-ACTIVATE.req are valid. Number of paramters must be 3. S_SAP_index: 0..63, NIL Access: 1..127 Service_list-Service_list_length: 1..(1 + 4 × 3) Service_list.n-th service_activate: “SDA” or “SDN” or “SRD” Service_list.n-th role_in_service: “INITIATOR” or “RESPONDER” (SDA, SDN only) or “BOTH” (SDA, SDN only) Service_list. nth DLSDU_length_list (1 to 4): 0..246
CHECK_PAR_DLSAP_DEACT	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 1. S_SAP_index: 0..63, NIL
CHECK_PAR_DLSAP_SUB	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 2. S_SAP_index: 0..62, NIL DLSDU_length_list (1 to 2): 0..246
CHECK_PAR_DLSAP_RES	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 4. S_SAP_index: 0..62, NIL Access: 1..127 DLSDU_length_list (1 to 4): 0..246 Indication_mode: “ALL or “DATA” or “UNCHANGED”
CHECK_PAR_IDENT	Check that all parameters of Service DLM-IDENT.req are valid. Number of paramters must be 1. D_addr: 0..126

Function name	Operations
CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k)))	Check that the requested management variable names are valid. Possible values for MAC sublayer are: in_ring_desired Data_rate TS Ttr G HSA max_retry_limit SYNCHT Tct Isochronous_mode maxTsh Tcsi HW_Release SW_Release Trr LMS GAPL Possible values for SRC sublayer are: minTsdr maxTsdr Tsl Tqui Tset Medium_redundancy DLPDU_sent_count Retry_count DLPDU_sent_count_sr(1 to n) Error_count(1 to n) SD_count SD_error_count
CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)))	Check that the requested management variable names are valid. Possible values for MAC sublayer are: in_ring_desired Data_rate TS Ttr G HSA max_retry_limit SYNCHT Tct Isochronous_mode maxTsh Tcsi HW_Release SW_Release Possible values for SRC sublayer are: minTsdr maxTsdr Tsl Tqui Tset Medium_redundancy DLPDU_sent_count Retry_count DLPDU_sent_count_sr(1 to n) Error_count(1 to n) SD_count SD_error_count
CHECK_PAR_SDA	Check that all parameters of Service DL-DATA-ACK.req are valid. Number of paramters must be 5. Service_class: “high” or “low” D_addr: 0..126 D_SAP_index: 0..62, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 1..246 DLSDU.Data: according to DLSDU.Len

Function name	Operations
CHECK_PAR_SDN	Check that all parameters of Service DL-DATA.req are valid. Number of paramters must be 5. Service_class: “high” or “low” D_addr: 0..127 D_SAP_index: 0..63, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 1..246 DLSDU.Data: according to DLSDU.Len
CHECK_PAR_STATUS	Check that all parameters of Service DLM-DLSAP-STATUS.req are valid. Number of paramters must be 2. DLSAP: 0..63, NIL, CS D_addr: 0..126
CHECK_PAR_REPLY	Check that all parameters of Service DL-DATA-REPLY.req are valid. Number of paramters must be 5. Service_class: “high” or “low” D_addr: 0..126 D_SAP_index: 0..62, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 0..246 DLSDU.Data: according to DLSDU.Len
CHECK_PAR_UPDATE	Check that all parameters of Service DL-REPLY-UPDATE.req are valid. Number of paramters must be 4. Service_class: “high” or “low” S_SAP_index: 0..62, NIL DLSDU.Len: 0..246 DLSDU.Data: according to DLSDU.Len Transmit_strategy: “SINGLE” or “MULTIPLE”
CHECK_PAR_TE	Check that all parameters of Service DL-CS-TIME-EVENT.req are valid. Number of paramters must be 3. D_addr = 127 D_SAP_index = CS S_SAP_index = CS
CHECK_PAR_CV	Check that all parameters of Service DL-CS-CLOCK-VALUE.req are valid. Number of paramters must be 4. D_addr = 127 D_SAP_index = CS S_SAP_index = CS Length of DLSDU: 18
CHECK_SAP (Service_class, S_SAP_index, Function)	Check that the SAP is activated to perform the current service request. SAP_List(S_SAP_index) ≠ NIL SAP_List(S_SAP_index).Function_List_I contains combination (Service_class, Function)
CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list)	Check that the parameters of the service primitive conform to the settings of the SAP. DLSDU_length_list.Max_DLSDU_length_req_low = SAP_List(S_SAP_index).Ubuffer.Low_buffer.DLSDU.Len) DLSDU_length_list.Max_DLSDU_length_req_high = SAP_List(S_SAP_index).Ubuffer.High_buffer.DLSDU.Len) DLSDU_length_list.Max_DLSDU_length_ind_low = SAP_List(S_SAP_index).Ibuffer.Low_len) DLSDU_length_list.Max_DLSDU_length_ind_high = SAP_List(S_SAP_index).Ibuffer.High_len)
Check_variable_names (Variable_name((1 to n), Index(1 to k)))	Check that the management variables are set to valid values. LOOP for i from 1 to n for all variable names: if Variables.Variable_name (i) = NIL DLM_status (i) := NO else DLM_status (i) := OK

Function name	Operations
GET_DLSAPSTATUS_CNF ( )	Get a management service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP retrieve Access from C_List.First_Entry.DLSDU.Data retrieve Service_type (1 to n) from C_List.First_Entry.DLSDU.Data retrieve Role_in_service_list (1 to n) from C_List.First_Entry.DLSDU.Data DLM_status := C_List.First_Entry.R_Status C_List.Remove()
GET_IDENT_CNF ( )	Get a management service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA Ident_list := C_List.First_Entry.DLSDU.Data DLM_status := C_List.First_Entry.R_Status C_List.Remove()

GET_SDA/SDN_CNF
(
)
	Get a service confirmation from the confirmation queue to MAC.
	C_List.Num_entry--
	if (C_List.First_Entry.FC.Function = "SDA_H" or "SDN_H") Service_class := high
	if (C_List.First_Entry.FC.Function = "SDA_L" or "SDN_L") Service_class := low
	D_addr := C_List.First_Entry.DA
	D_SAP_index := C_List.First_Entry.DSAP
	S_SAP_index := C_List.First_Entry.SSAP
	DL_status := C_List.First_Entry.R_Status
	C_List.Remove()

GET_SDA/SDN_IND
(
)
	Get a service indication from the indication queue to MAC.
	I_List.Num_entry--
	if (I_List.First_Entry.FC.Function = "SDA_H" or "SDN_H") Service_class := high
	if (I_List.First_Entry.FC.Function = "SDA_L" or "SDN_L") Service_class := low
	D_addr := I_List.First_Entry.DA
	D_SAP_index := I_List.First_Entry.DSAP
	S_addr := I_List.First_Entry.SA
	S_SAP_index := I_List.First_Entry.SSAP
	DLSDU := I_List.First_Entry.DLSDU
	I_List.Remove()
	SAP_List(S_SAP_index).Ibuffer.Insert()
	SAP_List(S_SAP_index).Ibuffer.Num_entry++

GET_SRD_CNF
(
)
	Get a service confirmation from the confirmation queue to MAC.
	C_List.Num_entry--
	if (C_List.First_Entry.FC.Function = "SRD_H") Service_class := high
	if (C_List.First_Entry.FC.Function = "SRD_L") Service_class := low
	D_addr := C_List.First_Entry.DA
	D_SAP_index := C_List.First_Entry.DSAP
	S_SAP_index := C_List.First_Entry.SSAP
	DLSDU := C_List.First_Entry.DLSDU
	DL_status := C_List.First_Entry.R_Status
	C_List.Remove()

GET_SRD_IND
(
)
	Get a service indication from the indication queue to MAC.
	I_List.Num_Entry--
	if (I_List.First_Entry.FC.Function = "SRD_H") Service_class := high
	if (I_List.First_Entry.FC.Function = "SRD_L") Service_class := low
	D_addr := I_List.First_Entry.DA
	D_SAP_index := I_List.First_Entry.DSAP
	S_addr := I_List.First_Entry.SA
	S_SAP_index := I_List.First_Entry.SSAP
	DLSDU := I_List.First_Entry.DLSDU
	R_Status := I_List.First_Entry.Update_status
	Reference := I_List.First_Entry.Reference
	I_List.Remove()
	SAP_List(S_SAP_index).Ibuffer.Insert()
	SAP_List(S_SAP_index).Ibuffer.Num_entry++

Function name	Operations
GET_DXM_IND ( )	Get a service indication from the indication queue to MAC. I_List.Num_entry-- if (I_List.First_Entry.FC.Function = "DH") Service_class := high else Service_class := low D_addr := I_List.First_Entry.DA D_SAP_index := I_List.First_Entry.DSAP S_addr := I_List.First_Entry.SA S_SAP_index := I_List.First_Entry.SSAP DLSDU := I_List.First_Entry.DLSDU I_List.Remove() SAP_List(S_SAP_index).Sbuffer.Insert() SAP_List(S_SAP_index).Sbuffer.Num_entry++
GET_TE_CNF	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_CV_CNF	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_TE/CV_IND()	Get a service indication from the indication queue to MAC. I_List.Num_entry-- D_addr := I_List.First_Entry.DA D_SAP_index := CS S_addr := I_List.First_Entry.SA S_SAP_index := CS DLSDU := I_List.First_Entry.DLSDU I_List.Remove() SAP_List(S_SAP_index).Ibuffer.Insert() SAP_List(S_SAP_index).Ibuffer.Num_entry++ Get a service indication from the indication queue to MAC.
PUT_HREQ (Function)	Put a service request to the high prior service queue to MAC. H_List.Insert() H_List.First_Entry.DA := D_addr H_List.First_Entry.DSAP := D_SAP_index H_List.First_Entry.SSAP := S_SAP_index H_List.First_Entry.FC.Frame_type := req H_List.First_Entry.FC.Function := Function H_List.First_Entry.DLSDU := DLSDU H_List.Num_entry++
PUT_HUBUFFER (S_SAP_index)	Put a service request to the high prior update buffer of the LSAP. SAP_List(S_SAP_index).Ubuffer.High_reference := Reference SAP_List(S_SAP_index).Ubuffer.High_transmit := Transmit_strategy SAP_List(S_SAP_index).Ubuffer.High_buffer := DLSDU
PUT_LREQ (Function)	Put a service request to the low prior service queue to MAC. L_List.Insert() L_List.First_Entry.DA := D_addr L_List.First_Entry.DSAP := D_SAP_index L_List.First_Entry.SSAP := S_SAP_index L_List.First_Entry.FC.Frame_type := req L_List.First_Entry.FC.Function := Function L_List.First_Entry.DLSDU := DLSDU L_List.Num_entry++
PUT_LUBUFFER (S_SAP_index)	Put a service request to the low prior update buffer of the LSAP. SAP_List(S_SAP_index).Ubuffer.Low_reference := Reference SAP_List(S_SAP_index).Ubuffer.Low_transmit := Transmit_strategy SAP_List(S_SAP_index).Ubuffer.Low_buffer := DLSDU

Function name	Operations
RESET_LIST	Reset all activated SAP by setting its entry to invalid value and reset all services queued in the data interface to MAC sublayer. SAP_List (0..63, NIL).I/Ubuffer.Remove() until empty SAP_List (0..63, NIL) := NIL I_List.Num_entry-- I_List.Remove() until empty C_List.Num_entry-- C_List.Remove() until empty
RESET_SAP_LIST (S_SAP_index)	Reset an activated SAP by setting its entry to invalid value. SAP_List(S_SAP_index) := NIL
RESOURCE (S_SAP_index, DLSDU.Len)	Check that local SAP resources are available to handle the requested data SAP_List(S_SAP_index) ≠ NIL SAP_List(S_SAP_index).Function_List_I contains combination (Service_class, Function) DLSDU.Len ≤ DLSDU_length_list-entry of combination (Service_class, Function)
Set_current_values (Variable_name((1 to n), Index(1 to k)))	Set the management variables for confirmation to DLMS-user. LOOP for i from 1 to n for all variable names: Current_value (i) := Variables.Variable_name (i)
SET_RSAP_LIST (S_SAP_index, DLSDU_length_list)	Activate a SAP as SRD responder by setting its entries to the requested values. SAP_List(S_SAP_index).Sbuffer.Low_len) := DLSDU_length_list.Max_DLSDU_DXM_length_ind_low SAP_List(S_SAP_index).Sbuffer.High_len) := DLSDU_length_list.Max_DLSDU_DXM_length_ind_high
SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled)	Activate a SAP as SRD responder by setting its entries to the requested values. SAP_List(S_SAP_index).Access := Access SAP_List(S_SAP_index).Indication_mode := Indication_mode SAP_List(S_SAP_index).Publisher_enabled := Publisher_enabled SAP_List(S_SAP_index).Ubuffer.Low_buffer.DLSDU.Len) := DLSDU_length_list.Max_DLSDU_length_req_low SAP_List(S_SAP_index).Ubuffer.High_buffer.DLSDU.Len) := DLSDU_length_list.Max_DLSDU_length_req_high SAP_List(S_SAP_index).Ibuffer.Low_len) := DLSDU_length_list.Max_DLSDU_length_ind_low SAP_List(S_SAP_index).Ibuffer.High_len) := DLSDU_length_list.Max_DLSDU_length_ind_high
SET_SSAP_LIST (S_SAP_index, Access, Service_list)	Activate a SAP by setting its entries to the requested values. SAP_List(S_SAP_index).Access := Access compose SAP_List(S_SAP_index).Function_list_I based on Service_list compose SAP_List(S_SAP_index).Function_list_R based on Service_list

Function name	Operations

Set_variable_list
(
Variable_name ((1 to n),
Index(1 to k)),
Desired_value (1 to n)
)
	Check that the requested management variable names are inside their ranges and set them to the list of variables and set the corresponding DLM_status:
	LOOP for i from 1 to n for all variable names:
	vn := Variable_name(i)
	if (LOWLIM(vn) ≤ Desired_value(i) ≤ HIGHLIM(vn))
	Variables.vn := Desired_value(i)
	DLM_status(i) := OK
	else
	DLM_status(i) := IV
	
	with following LOWLIM and HIGHLIM for variables of MAC sublayer:
	in_ring_desired = 0, 1
	data_rate = 9,6 … 12000 // kbit/s
	TS = 0 … 126
	Ttr = 1 … 2^24-1
	G = 1 … 100 with Tgud = G * Ttr < 2°24
	HSA = TS … 126
	max_retry_limit = 1 … 8
	SYNCHT = Octetstring with length 0 or 2
	Tct = 1..2^32-1
	maxTsh = 1..256
	Tcsi = 1 … 2^32-1
	HW_Release = Visible String with length 0 … 32
	SW_Release = Visible String with length 0 … 32
	with following LOWLIM and HIGHLIM for variables of SRC sublayer:
	minTsdr = 1 … 2^16-1
	maxTsdr = minTdsr … 2^16-1
	Tsl = maxTsdr … 2^16-1
	Tqui = 0 … 255
	Tset = 1 … 255
	DLPDU_sent_cou nt = 0
	Retry_count = 0
	DLPDU_sent_count_sr(1 to n) = 0
	Error_count(1 to n) = 0
	SD_count = 0
	SD_error_count = 0
	with following LOWLIM and HIGHLIM for variables of PHY sublayer:
	Medium_redundancy = 0, 1
	Interface_mode = "FULL_DUPLEX" or "HALF_DUPLEX"
	Loop_back_mode = "DISABLED" or "in MDS" or "in MAU"
	Preamble_extension = 0..7
	Tptg = 0..7
	Tics = 0..7
	Transmitter_output_channel (1 to 8) = "ENABLED" or "DISABLED"
	Receiver_inpiut_channel (1 to 8) = "ENABLED" or "DISABLED"
	Preferred_receive_channel = "NONE" or 1..8

Write_ident_list ( )	Compose parameter Ident_list for confirmation to DLMS user. Ident_list := Ident_List.Vendor_name + Ident_List.Model_name + Ident_List.HW_release + Ident_List.SW_release
Write_SAPstatus_list ( )	Compose parameters Access, Service_type, Role_in_service_list for confirmation to DLMS user. Access := SAP_List(D_SAP_index).Access compose Service_type(1 to n) based on SAP_List(D_SAP_index).Sevice_listI/R compose Role_in_service_list(1 to n) based on SAP_List(D_SAP_index).Sevice_listI/R

A.5 MAC

A.5.1 Primitive definitions

A.5.1.1 Primitives exchanged between DLM and MAC

Table A.11 shows the primitives issued by the DLM to the MAC.

Table A.11 – Primitives issued by DLM to MAC

Primitive name	Associated parameters
MAC_RESET.req	none

Table A.12 shows the primitives issued by the MAC to the DLM.

Table A.12 – Primitives issued by MAC to DLM

Primitive name	Associated parameters
MAC_RESET.cnf	none
MAC_LFAULT.ind	Fault_type
MAC_BFAULT.ind	Fault_type Tj

A.5.1.2 Parameters of MAC primitives

Table A.13 shows the parameters used with primitives exchanged between the DLM and the MAC.

Table A.13 – Parameters used with primitives exchanged between DLM and MAC

Parameter name	Description
Fault_type	This parameter contains the error type. Possible values: State_conflict, Faulty_transceiver Double_token, Duplicate_address, Not_synchronized, Out_of_ring, Time_out, Hsa_error, In_ring
Tj	Jitter time in Isochronous Mode

A.5.2 State machine description

The Media Access Control is responsible for the message exchange and control of the various components using the media.

For a token based system, a main focus of this state machine is the token handling in all situations (see 5.3.2). This will be done in the states LISTEN-TOKEN, CLAIM-TOKEN, ACTIVE-IDLE (token-receipt) and PASS-TOKEN. To identify new stations ready to enter the ring, a FDL-Status request will be executed periodically. The reply will be checked in the state AWAIT-STATUS-RESPONSE. The state WAIT-TCT is used in isochronous mode to keep the interval between SYNCH requests constant.

The other main task is the correct execution of service sequences (Request, Response) including retries and duplication detection. This will be done by checking the high and low priority request queues when the station is token holder (state USE-TOKEN). In the state AWAIT-DATA-RESPONSE the reply to the processed service should be received. The state CHECK-TOKEN-PASS is used for checking the Token-hold-time.

The incoming service indications are processed in the states ACTIVE-IDLE, PASSIVE-IDLE and CHECK-TOKEN-PASS. All valid services received will be put into the indication queue and the appropriate reply activity will be invoked.

All local variables of the MAC are shown in Table A.14.

Table A.14 – Local MAC variables

Struct element	Type	Range	Remark
FCB	[0..126] of U8	0,1	
FCV	[0..126] of U8	0,1	
REQM	S4		
RESM	S5		
Cnf	Bool		Single variables
Dup_add_count	U8	0, 1	
Gap_lp_cnt	U16		
Gap_to_do	Bool		
Gud_timer	U32		
Isochronous_Start	Bool		
LMS_cnt	U16		
Req_h_cnt	U16		
Retry_cnt	U8	0..15	
Second	U8	0, 1	
Token_holder	U8	0..126, NIL	
Tok_cnt	U8		
Tok_err_cnt	U8		
Trr	U24		
T_cnt	U16	0..259	
TRT			Token Rotation Timer
cv	U24		Current Value of
Start(Ttr)	Trigger		Starting count down (from Ttr)

A.5.3 MAC state table

The MAC state table is shown in Table A.15.

Table A.15 – MAC state table

No.	Current state	Event /condition ⇒action	Next state
1	OFFL	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	OFFL
2	OFFL	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	OFFL
3	OFFL	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
4	OFFL	/Data_Rate ≠ NIL && In_ring_desired ⇒ INIT_FCBV_LIST, RESM := empty, INIT_LMS, Isochronous_Start = TRUE,T_cnt:=0	LISTEN

No.	Current state	Event /condition ⇒action	Next state
5	OFFL	/Data_Rate ≠ NIL && !In_ring_desired ⇒ RESM := empty	PASSIVE_I
6	OFFL	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
7	OFFL	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
8	OFFL	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
9	OFFL	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
10	OFFL	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
11	OFFL	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
12	OFFL	SRC_SYNI_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
13	LISTEN	/!In_ring_desired ⇒	PASSIVE_I
14	LISTEN	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	LISTEN
15	LISTEN	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	LISTEN
16	LISTEN	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
17	LISTEN	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ T_cnt := 0	LISTEN
18	LISTEN	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0	LISTEN
19	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /SA = TS || DA = TS && Dup_add_count > 0 ⇒ Fault_type := Duplicate_address, T_cnt := 0, Dup_add_count := 0 MAC_BFAULT.ind (Fault_type)	OFFL
20	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA = TS || SA = TS && Dup_add_count = 0 ⇒ Dup_add_count++, Tok_cnt--,T_cnt := 0	LISTEN

No.	Current state	Event /condition ⇒action	Next state
21	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /(DA ≠ TS && SA ≠ TS) && (DA > HSA || SA > HSA) && Dup_hsa_count > 0 ⇒ Fault_type := Hsa_error, Dup_hsa_count := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	OFFL
22	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /(DA ≠ TS && SA ≠ TS) && (DA > HSA || SA > HSA) && Dup_hsa_count = 0 ⇒ Dup_hsa_count++, Tok_cnt--,T_cnt := 0	LISTEN
23	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First = NIL ⇒ BUILD_LMS(DA), Token_holder := DA, Tok_cnt--,T_cnt := 0,First:=SA	LISTEN
24	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First ≠ NIL && (SA ≠ Token_holder || Tok_cnt=0) ⇒ INIT_LMS, BUILD_LMS(DA), Token_holder := DA,Tok_cnt--, T_cnt := 0, First:=DA	LISTEN
25	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First ≠ NIL && SA = Token_holder && Tok_cnt > 0 && DA ≠ First && LMS_cnt = 0 ⇒ LMS_UPDATE(DA, SA), Token_holder := DA ,Tok_cnt--, T_cnt := 0	LISTEN
26	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First ≠ NIL && SA = Token_holder && Tok_cnt > 0 && DA = First && LMS_cnt = 0 ⇒ LMS_UPDATE(DA, SA), Token_holder := DA, LMS_cnt := 1 , Tok_cnt--, T_cnt := 0	LISTEN
27	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First ≠ NIL && SA = Token_holder && LMS[SA]=DA && Tok_cnt > 0 && DA ≠ First && LMS_cnt = 1 ⇒ Token_holder := DA, Tok_cnt--, T_cnt := 0	LISTEN
28	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA =< HSA && SA =< HSA && First ≠ NIL && SA = Token_holder && LMS[SA]=DA && Tok_cnt > 0 && DA = First && LMS_cnt = 1 ⇒ Tok_cnt--, T_cnt := 0, LMS_cnt++, RECV_ERR_cnt,SER_ACK_cnt := 0	ACTIVE_I
29	LISTEN	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
30	LISTEN	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
31	LISTEN	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ T_cnt ++	LISTEN
32	LISTEN	SRC_SLOT_EVENT.ind /TIME_OUT ⇒ Fault_type := Time_out, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	CLAIM_T

No.	Current state	Event /condition ⇒action	Next state
33	LISTEN	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
34	ACTIVE_I	/!In_ring_desired ⇒	PASSIVE_I
35	ACTIVE_I	/LMS[TS] = NIL && H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	ACTIVE_I
36	ACTIVE_I	/LMS[TS] = NIL && L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	ACTIVE_I
37	ACTIVE_I	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
38	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode>0 ⇒ TRT_OFF	ACTIVE_I
39	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) = NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_rdy, FC.Function := OK , RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
40	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) ≠ NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_in_ring, FC.Function := OK, RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
41	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, IDENT, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
42	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = DLSAP_status ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
43	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0,NO), T_cnt := 0	ACTIVE_I
44	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && (!SAP_CHECK(DSAP) || !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
45	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function =TE || FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
46	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = CV || FC.Function = TE) && (!SAP_CHECK(CS) || !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
47	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
48	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && (!SAP_CHECK(DSAP) || !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
49	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H || FC.Function = SRD_L || FC.Function = SRD_H || FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
50	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
51	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
52	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL || Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤> SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
53	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY,DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:⇒ SSAP, DLSDU := R_SDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
54	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
55	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
56	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L || FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && !SAP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
57	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA≠127 || (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) || FC.Frame ≠ req || INVALID_FUNCTION) && !(DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode=0) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
58	ACTIVE_I	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0, TRT_ON	ACTIVE_I
59	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count > 0 ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Dup_add_count := 0, Fault_type := Duplicate_address, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
60	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count = 0 ⇒ Dup_add_count++, TOK_CNT_UPD, Second := 0, RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
61	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt > (Limit - 2) ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Fault_type := Out_of_ring, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
62	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt ≤ (Limit - 2) ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Tok_err_cnt++, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
63	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] ≠ DA && DA ≠ TS ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
64	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] = DA && DA ≠ TS ⇒ RESM := empty, TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
65	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry ≠ 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
66	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second = 0 ⇒ RESM := empty, Second++, T_cnt := 0, TRT_ON	ACTIVE_I
67	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct = 0 && H_list.Num_entry ≠ 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
68	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) || (LMS[SA] ≤ SA && ((SA < TS) || (TS < LMS[SA])))) ⇒ RESM := empty, TTH_INIT, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, GAP_INIT(0), Fault_Type := In_ring, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
69	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry = 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
70	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0&&Tct=0 && H_list.Num_entry = 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
71	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
72	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0&& Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL,T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT

No.	Current state	Event /condition ⇒action	Next state
73	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && (!LMS[LMS[SA]]=DA && !(LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) || (LMS[SA] ≤ SA && ((SA < TS) || (TS < LMS[SA]))))) ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
74	ACTIVE_I	SRC_SEND_DATA.cnf ⇒	ACTIVE_I
75	ACTIVE_I	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
76	ACTIVE_I	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ T_cnt ++, TRT_ON	ACTIVE_I
77	ACTIVE_I	SRC_SLOT_EVENT.ind /TIME_OUT ⇒ RESM := empty, Fault_type := Time_out, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	CLAIM_T
78	ACTIVE_I	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
79	CLAIM_T	/LMS[TS] ≠ NIL && H_list.Num_entry ≠ 0 && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, Isochronous_Start = TRUE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
80	CLAIM_T	/LMS[TS] ≠ NIL && H_list.Num_entry = 0 && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, Isochronous_Start = TRUE	CHECK_A
81	CLAIM_T	/LMS[TS] ≠ NIL && Tct > 0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
82	CLAIM_T	/LMS[TS] = NIL ⇒ INIT_LMS, BUILD_LMS(TS), Second := 0, Fault_Type := In_ring, Retry_cnt := 0, DA := TS, SA := TS, TTH_INIT, if (NS := (TS+1) mod (HSA+1)) Gap_to_do := FALSE else Gap_to_do := TRUE, Gap_Ip_cnt := 0 MAC_BFAULT.ind (Fault_type), SRC_SEND_TOKEN.req (DA, SA)	PASS_T
83	WAIT_TCT	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
84	WAIT_TCT	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
85	WAIT_TCT	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL

No.	Current state	Event /condition ⇒action	Next state
86	WAIT_TCT	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
87	WAIT_TCT	SRC_SEND_DATA.cnf /Isochronous_Start ⇒ Tct:= TCT.cv, TCT.start(0), Fault_type:=SYNCH, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON Isochronous_Start = FALSE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind (Fault_type)	USE_T
88	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=1 && TCT.cv ≥ Tct ⇒ Tsh :=TCT.cv-Tct, TCT.start(2*Tct-TCT.cv), Fault_type:=Synch_Delay, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind (Fault_type,Tsh)	USE_T
89	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=2 && TCT.cv ≥ Tct ⇒ Tsh :=Tct, TCT.start(2*Tct-TCT.cv), Fault_type:=SynchDelay, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind(Fault_Type,Tsh)	WAIT_TCT
90	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=0 && TCT.cv ≥ Tct && H_list.Num_entry ≠ 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, SETUP_HREQ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
91	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=0 && TCT.cv ≥ Tct && H_list.Num_entry = 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry	CHECK_A
92	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Tct-Tpsp ≤ TCT.cv < Tct ⇒ TPSP.start(Tct-TCT.cv)	WAIT_TCT
93	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && TCT.cv < Tct-Tpsp ⇒ DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT

No.	Current state	Event /condition ⇒action	Next state
94	WAIT_TCT	TPSP expired /Isochronous_mode>0 ⇒ TCT.start(2*Tct-TCT.cv), Fault_type := SYNCH, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind(Fault_Type)	USE_T
95	WAIT_TCT	TPSP expired / (Isochronous_Mode=0 || Isochronous_Mode=3) && H_list.Num_entry ≠ 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, SETUP_HREQ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
96	WAIT_TCT	TPSP expired / (Isochronous_Mode=0 || Isochronous_Mode=3) && H_list.Num_entry = 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, T_cnt := 0	CHECK_A
97	WAIT_TCT	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
98	WAIT_TCT	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
99	WAIT_TCT	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
100	USE_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
101	USE_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
102	USE_T	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
103	USE_T	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
104	USE_T	SRC_SEND_DATA.cnf /Cnf && !Gap_in_action ⇒ DLPDU_sent_count++ DLPDU_sent_count_sr[DA]++ SER_ACK_cnt++, if (SER_ACK_cnt := SER_ACK_limit) SER_ACK_cnt,RECV_ERR_cnt := 0	AW_DATA
105	USE_T	SRC_SEND_DATA.cnf /!Cnf && !Synch ⇒	CHECK_A

No.	Current state	Event /condition ⇒action	Next state
106	USE_T	SRC_SEND_DATA.cnf /!Cnf && Synch && TCT.cv >143+ maxTsh ⇒ Synch := False,Fault_type:=SynchDelay Tsh := TCT.cv-143 MAC_BFAULT.ind (Fault_type,Tsh)	CHECK_A
107	USE_T	SRC_SEND_DATA.cnf /!Cnf && Synch&& TCT.cv =<143+ maxTsh ⇒ Synch = False	CHECK_A
108	USE_T	SRC_SEND_DATA.cnf /Cnf && Gap_in_action ⇒ Gap_in_action := FALSE, SER_ACK_cnt++, if (SER_ACK_cnt := SER_ACK_limit) SER_ACK_cnt,RECV_ERR_cnt := 0	AW_STATUS
109	USE_T	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
110	USE_T	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
111	USE_T	SRC_SYNI_EVENT.ind /Gap_in_action ⇒ Gap_in_action := FALSE, Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
112	USE_T	SRC_SYNI_EVENT.ind /!Gap_in_action ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I
113	AW_DATA	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
114	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(FC.Frame = req || (FC.Function ≠ SC && DA ≠ TS)) ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
115	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /FC.Frame = rsp && DA = TS && FC.Function ≠ SC && (REQM.DA≠SA || (DSAP,SSAP ≠ Nil && (REQM.DSAP ≠ SSAP || REQM.SSAP ≠ DSAP))) ⇒ FCV_CLEAR, SETUP_CONM(DS), T_cnt := 0	ACTIVE_I
116	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / FC.Frame = rsp && DA = TS && FC.Function ≠ SC && REQM.DA=SA && (DSAP,SSAP =Nil || (REQM.DSAP = SSAP && REQM.SSAP = DSAP)) ⇒ SETUP_CON(FC, DLSDU), T_cnt := 0, FCB_UPDATE	CHECK_A
117	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / FC.Frame = rsp && FC.Function =SC ⇒ SETUP_CON(FC, DLSDU), T_cnt := 0, FCB_UPDATE	CHECK_A
118	AW_DATA	SRC_RECEIVE_ERROR.ind /RECV_ERR_cnt ≥ RECV_ERR_limit ⇒ FCV_CLEAR, SETUP_CONM(DS), T_cnt := 0	LISTEN

No.	Current state	Event /condition ⇒action	Next state
119	AW_DATA	SRC_RECEIVE_ERROR.ind /!(Isochronous_mode=2 && REQM.serv_class high) && Retry_cnt > 0 && FCV = 1 && RECV_ERR_cnt < RECV_ERR_limit ⇒ Retry_Count--, Error_count[SA]--, DA := REQM.DA, SA := TS, FC := REQM.FC, DSAP := REQM.DSAP, SSAP := REQM.SSAP, DLSDU := REQM.DLSDU, Retry_cnt--, T_cnt := 0, RECV_ERR_cnt++ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
120	AW_DATA	SRC_RECEIVE_ERROR.ind /!(Isochronous_mode=2 && REQM.serv_class = high) && (Retry_cnt = 0 || FCV = 0) && RECV_ERR_cnt < RECV_ERR_limit ⇒ FCV_CLEAR, SETUP_CONM(NA), T_cnt := 0	CHECK_A
121	AW_DATA	SRC_RECEIVE_ERROR.ind /Isochronous_mode=2 && REQM.serv_class = high && RECV_ERR_cnt < RECV_ERR_limit ⇒ SETUP_CONM(NA), T_cnt := 0	CHECK_A
122	AW_DATA	SRC_RECEIVE_TOKEN.ind (DA, SA) /In_ring_desired ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
123	AW_DATA	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
124	AW_DATA	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
125	AW_DATA	SRC_SLOT_EVENT.ind /!(Isochronous_mode=2 && REQM.serv_class = high) && Retry_cnt > 0 && FCV = 1 ⇒ Retry_Count--, Error_count[SA]--, DA := REQM.DA, SA := TS, FC := REQM.FC, DSAP := REQM.DSAP, SSAP := REQM.SSAP, DLSDU := REQM.DLSDU, Retry_cnt-- SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
126	AW_DATA	SRC_SLOT_EVENT.ind /!(Isochronous_mode=2 && REQM.serv_class = high) && Retry_cnt = 0 || FCV = 0 ⇒ FCV_CLEAR, SETUP_CONM(NA)	CHECK_A
127	AW_DATA	SRC_SLOT_EVENT.ind /Isochronous_mode=2 && REQM.serv_class = high ⇒ SETUP_CONM(NA)	CHECK_A
128	AW_DATA	SRC_SYNI_EVENT.ind /In_ring_desired ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I
129	CHECK_A	/ (TTH_AVAILABLE || Isochronous_Mode=2 || Isochronous_Mode=3) && Req_h_cnt ≠ 0 ⇒ SETUP_HREQ, Req_h_cnt--, SA := TS SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T

No.	Current state	Event /condition ⇒action	Next state
130	CHECK_A	/TTH_AVAILABLE && Req_h_cnt = 0 && Gap_lp_cnt = 0 ⇒ FC.Frame := req, FC.FCB := 0, FC.FCV :=0, FC.Function := FDL_status, DA := Gap_address, SA := TS, DLSDU,SSAP,DSAP:=NIL, Cnf := TRUE, Gap_in_action := TRUE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
131	CHECK_A	/TTH_AVAILABLE && Req_h_cnt = 0 && Gap_lp_cnt ≠ 0 && L_list.Num_entry ≠ 0 ⇒ SETUP_LREQ, SA := TS SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
132	CHECK_A	/!((TTH_AVAILABLE || Isochronous_Mode=2 || Isochronous_Mode=3) && Req_h_cnt <> 0) || !TTH_AVAILABLE || (Req_h_cnt = 0 && Gap_lp_cnt <> 0 && L_list.Num_entry = 0) ⇒ Second := 0, DA := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
133	PASS_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
134	PASS_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
135	PASS_T	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
136	PASS_T	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
137	PASS_T	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
138	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = token_pass_failed && Second = 0 ⇒ Second++, DS := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
139	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = token_pass_failed && Second > 0 ⇒ Second := 0, Fault_type := Faulty_transceiver MAC_BFAULT.ind (Fault_type)	OFFL
140	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = no_token_pass ⇒ Second := 0, Fault_type := Faulty_transceiver MAC_BFAULT.ind (Fault_type)	OFFL
141	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && !In_ring_desired ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T

No.	Current state	Event /condition ⇒action	Next state
142	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && In_ring_desired && Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
143	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && H_list.Num_entry ≠ 0 && In_ring_desired && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
144	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && H_list.Num_entry = 0 && In_ring_desired && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry	CHECK_A
145	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS ≠ TS && Status = OK ⇒	CHECK_T
146	PASS_T	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
147	PASS_T	SRC_SYNI_EVENT.ind /In_ring_desired ⇒ Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I
148	CHECK_T	/LMS[TS] = NIL && H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	CHECK_T
149	CHECK_T	/LMS[TS] = NIL && L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	CHECK_T
150	CHECK_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
151	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = SA && FC.Frame = req && FC.Function = FDL_status && Tct > 0 ⇒ TRT_OFF	ACTIVE_I
152	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) = NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_rdy, FC.Function := OK , RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
153	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) ≠ NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_in_ring, FC.Function := OK , RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
154	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, IDENT, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
155	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = LSAP_status ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
156	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0,NO), T_cnt := 0	ACTIVE_I
157	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && (!SAP_CHECK(DSAP) || !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
158	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function =TE || FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
159	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = CV || FC.Function = TE) && (!SAP_CHECK(CS) || !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
160	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
161	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && (!SAP_CHECK(DSAP) || !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
162	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H || FC.Function = SRD_L || FC.Function = SRD_H || FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
163	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
164	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
165	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL || Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤> SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
166	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY,DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:⇒ SSAP, DLSDU := R_SDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
167	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
168	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
169	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L || FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && !SAP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
170	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA≠127 || (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) || FC.Frame ≠ req || INVALID_FUNCTION) && !(DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
171	CHECK_T	SRC_RECEIVE_ERROR.ind / In_ring_desired ⇒ T_cnt := 0	ACTIVE_I
172	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count > 0 && In_ring_desired ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Dup_add_count := 0, Fault_type := Duplicate_address, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN

No.	Current state	Event /condition ⇒action	Next state
173	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count = 0 && In_ring_desired ⇒ Dup_add_count++, TOK_CNT_UPD, Second := 0, RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
174	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && Tok_err_cnt > (Limit - 2) && In_ring_desired ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Fault_type := Out_of_ring, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
175	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt ≤ (Limit - 2) && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Tok_err_cnt++, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
176	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] ≠ DA && DA ≠ TS && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
177	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] = DA && DA ≠ TS && In_ring_desired ⇒ RESM := empty, TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
178	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry ≠ 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
179	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second = 0 && In_ring_desired ⇒ RESM := empty, Second++, T_cnt := 0, TRT_ON	ACTIVE_I
180	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct=0 && H_list.Num_entry ≠ 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
181	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) || (LMS[SA] ≤ SA && ((SA < TS) || (TS < LMS[SA])))) && In_ring_desired ⇒ RESM := empty, TTH_INIT, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, GAP_INIT(0), Fault_Type := In_ring, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
182	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry = 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A

No.	Current state	Event /condition ⇒action	Next state
183	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct=0 && H_list.Num_entry = 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
184	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct > 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
185	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0&& Tct > 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL,T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
186	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && (!LMS[LMS[SA]]=DA && !(LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) || (LMS[SA] ≤ SA && ((SA < TS) || (TS < LMS[SA]))))) && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
187	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!In_ring_desired ⇒ Fault_Type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I
188	CHECK_T	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
189	CHECK_T	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
190	CHECK_T	SRC_SLOT_EVENT.ind /Retry_cnt < 2 ⇒ DS := NS(TS), SA := TS, Retry_cnt++ SRC_SEND_TOKEN.req (DA, SA)	PASS_T
191	CHECK_T	SRC_SLOT_EVENT.ind /Retry_cnt = 2 ⇒ EX_LMS (NS), Retry_cnt := 0, DS := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
192	CHECK_T	SRC_SYNI_EVENT.ind / In_ring_desired ⇒ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I

No.	Current state	Event /condition ⇒action	Next state
193	AW_STATUS	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
194	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(FC.Frame = req || (FC.Function ≠ SC && DA ≠ TS)) ⇒ Fault_type := Double_token, NEXT_GAP, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
195	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /FC.Frame = rsp && DA = TS && SA=GAP_address && FC.Function = OK(&& FC.Stn-Type = M_rdy ⇒ DA := SA, SA := TS, LMS_UPDATE(DA, SA), GAP_INIT(NIL), T_cnt := 0 SRC_SEND_TOKEN.req (DA, SA)	PASS_T
196	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /FC.Frame = rsp && (DA = TS || FC.Function = SC) && (FC.Function ≠ OK || FC.Stn-Type ≠ M_rdy) ⇒ NEXT_GAP , T_cnt := 0 SRC_SEND_TOKEN.req (DA, SA)	CHECK_A
197	AW_STATUS	SRC_RECEIVE_ERROR.ind /RECV_ERR_cnt ≥ RECV_ERR_limit ⇒ T_cnt := 0	LISTEN
198	AW_STATUS	SRC_RECEIVE_ERROR.ind /RECV_ERR_cnt < RECV_ERR_limit ⇒ NEXT_GAP , T_cnt := 0	CHECK_A
199	AW_STATUS	SRC_RECEIVE_TOKEN.ind (DA, SA) /In_ring_desired ⇒ Fault_type := Double_token, NEXT_GAP, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
200	AW_STATUS	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
201	AW_STATUS	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
202	AW_STATUS	SRC_SLOT_EVENT.ind ⇒ NEXT_GAP	CHECK_A
203	AW_STATUS	SRC_SYNI_EVENT.ind /In_ring_desired ⇒ Fault_type := Not_synchronized, NEXT_GAP MAC_BFAULT.ind (Fault_type)	ACTIVE_I
204	PASSIVE_I	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	PASSIVE_I
205	PASSIVE_I	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	PASSIVE_I
206	PASSIVE_I	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL

No.	Current state	Event /condition ⇒action	Next state
207	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := Slave, FC.Function := OK , RESM := empty, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
208	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, IDENT, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
209	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = LSAP_status ⇒ DA := SA, SA := TS, DSAP <:⇒ SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
210	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0,NO), T_cnt := 0	PASSIVE_I
211	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =127) && FC.Frame = res && (!SAP_CHECK(DSAP) || !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
212	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function =TE || FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	PASSIVE_I
213	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = CV || FC.Function = TE) && (!SAP_CHECK(CS) || !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	PASSIVE_I
214	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0	PASSIVE_I
215	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA =TS || DA =127) && FC.Frame = req && (FC.Function = SDN_H || FC.Function = SDN_L) && (!SAP_CHECK(DSAP) || !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
216	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H || FC.Function = SRD_L || FC.Function = SRD_H || FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
217	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0,NO), DA := NIL, SA := NIL, FC.Function := OK, DLSDU,SSAP,DSAP:=NIL, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I

No.	Current state	Event /condition ⇒action	Next state
218	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L || FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
219	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL || Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤> SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
220	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY,DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:⇒ SSAP, DLSDU := R_SDU, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
221	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
222	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
223	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H || FC.Function = SRD_BCT || FC.Function = SRD_L || FC.Function = SDA_H || FC.Function = SDA_L) && !RETRY && !SAP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
224	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA≠127 || (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) || (FC.Frame ≠ req && (DA ≠ 127) || INVALID_FUNCTION) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
225	PASSIVE_I	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0	PASSIVE_I
226	PASSIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I

No.	Current state	Event /condition ⇒action	Next state
227	PASSIVE_I	SRC_SEND_DATA.cnf ⇨	PASSIVE_I
228	PASSIVE_I	SRC_SEND_TOKEN.cnf(Status) ⇨ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
229	PASSIVE_I	SRC_SLOT_EVENT.ind /!TIME_OUT ⇨ T_cnt ++	PASSIVE_I
230	PASSIVE_I	SRC_SLOT_EVENT.ind /TIME_OUT ⇨ Fault_type := Time_out, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I
231	PASSIVE_I	SRC_SYNI_EVENT.ind ⇨ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I

A.5.4 Functions

All functions of the MAC are summarized in Table A.16.

Table A.16 – MAC function table

Function name	Operations
IDENT	- Prepares Ident-Response send Ident data, if Ident buffer available send SC, if no Ident buffer
LSAP_STATUS	- Prepares Ident-Response send LSAP data, if LSAP-status buffer available send SC, if no LSAP-status buffer
INIT_FCBV_LIST	FCV[0..126]:= 0, FCB[0..126]:= 1
INIT_LMS	LMS[0..126]:= NIL Tok_err_cnt:=0, Tok_cnt:=255 First :=NIL, LMS_cnt:=0
BUILD_LMS(Ad)	LMS[0..126]:= NIL, LMS[Ad]:=Ad
LMS_UPDATE	if(DA < LMS[SA] ≤ SA || SA < DA < LMS[SA] || LMS[SA] ≤ SA < DA) LMS[DA] := LMS[SA], LMS[SA] := DA else if (DA = LMS[LMS[SA]]) LMS[LMS[SA]] := NIL, LMS[SA] :=DA else Tok_err_cnt++
EX_LMS (Ad)	LMS[TS]:= LMS[Ad], LMS[Ad]:=NIL
TOK_CNT_UPD	if (Tok_cnt--=0) Tok_cnt:=255, Tok_err_cnt:=0, Dup_add_cnt:=0
DUPLICATE_ADDRESS	SA = TS || SA > HSA || DA > HSA
TOKEN_ERROR	LMS[SA] := NIL
FCB_UPDATE	FCB[DA]:= not FCB[DA], FCV[DA]:=1
FCV_CLEAR	FCV[DA]:=0, FCB[DA]:=1
SETUP_STN_TYPE	if (In_ring_desired = FALSE) FC.Stn-Type:=Slave else if (LMS[TS] = NIL) FC.Stn-Type:=M_rdy else FC.Stn-Type:=M_in_ring
R_FUNCTION	if (Upd_sts=DL) FC.Function:=RDL else FC.Function:=RDH
SETUP_RESM	RESM.DA := DA RESM.DSAP := DSAP RESM.SSAP := SSAP RESM.FC := FC RESM.DLSDU := DLSDU
RETRY	RESM≠NIL && RESM.SA=SA && DA=TS && FCV[DA]=1 && FC.FCV=1 && FCB[DA]=FC.FCB

Function name	Operations
SETUP_HREQ	H_List.Num_entry-- REQM.DA,DA := H_List.First_Entry.DA REQM.DSAP,DSAP := H_List.First_Entry.DSAP REQM.SSAP,SSAP := H_List.First_Entry.SSAP FC := H_List.First_Entry.FC REQM.DLSDU,DLSDU := H_List.First_Entry.DLSDU Cnf := H_List.First_Entry.conf REQM.serv_class := high SA:=TS if (FC.Function≠ FDL_Status, Ident, LSAP_Status, SDN_H, SDN_L) (FC.FCB:=FCB[DA], FC.FCV:=FCV[DA]) else (FC.FCB:=0, FC.FCV:=0) FC.Frame:=req REQM.FC:=FC H_List.Remove()
SETUP_LREQ	L_List.Num_entry-- REQM.DA,DA := L_List.First_Entry.DA REQM.DSAP,DSAP := L_List.First_Entry.DSAP REQM.SSAP,SSAP := L_List.First_Entry.SSAP FC := L_List.First_Entry.FC REQM.DLSDU,DLSDU := L_List.First_Entry.DLSDU Cnf := L_List.First_Entry.conf REQM.serv_class := low SA:=TS if (FC.Function≠ FDL_Status, Ident, LSAP_Status, SDN_H, SDN_L) (FC.FCB:=FCB[DA], FC.FCV:=FCV[DA]) else (FC.FCB:=0, FC.FCV:=0) FC.Frame:=req REQM.FC:=FC L_List.Remove()
SETUP_CONM(Err_type)	C_List.Insert() C_List.Last_Entry.DA := REQM.DA C_List.Last_Entry.DSAP := REQM.DSAP C_List.Last_Entry.SSAP := REQM.SSAP C_List.Last_Entry.serv_class := REQM.serv_class C_List.Last_Entry.FC := Err_type C_List.Last_Entry.DLSDU := NIL C_List.Num_entry++
RESET_HL_LIST	while (H_List.Num_entry ≠ 0) SETUP_HCON_DS while (L_List.Num_entry ≠ 0) SETUP_LCON_DS

Function name	Operations
SETUP_HCON_DS	H_List.Num_entry-- C_List.Insert() C_List.Last_Entry.DA := H_List.First_Entry.DA C_List.Last_Entry.DSAP := H_List.First_Entry.DSAP C_List.Last_Entry.SSAP := H_List.First_Entry.SSAP C_List.Last_Entry.serv_class := H_List.First_Entry.serv_class C_List.Last_Entry.FC:= H_List.First_Entry.FC C_List.Last_Entry.Status:= DS C_List.Num_entry++ H_List.Remove()
SETUP_LCON_DS	L_List.Num_entry-- C_List.Insert() C_List.Last_Entry.DA := L_List.First_Entry.DA C_List.Last_Entry.DSAP := L_List.First_Entry.DSAP C_List.Last_Entry.SSAP := L_List.First_Entry.SSAP C_List.Last_Entry.serv_class := L_List.First_Entry.serv_class C_List.Last_Entry.FC:= L_List.First_Entry.FC C_List.Last_Entry.Status:= DS C_List.Num_entry++ L_List.Remove()
SETUP_CON(FC,DLSDU)	C_List.Insert() C_List.Last_Entry.DA := REQM.DA C_List.Last_Entry.DSAP := REQM.DSAP C_List.Last_Entry.SSAP := REQM.SSAP C_List.Last_Entry.serv_class := REQM.serv_class C_List.Last_Entry.FC:= REQM.FC C_List.Last_Entry.DLSDU := DLSDU C_List.Last_Entry.Status:= FC C_List.Num_entry++
SETUP_IND(Ref,Upd_sts)	SAP_List[DSAP].Ibuffer.Num_entry-- I_List.Insert() I_List.Last_Entry.DA := DA I_List.Last_Entry.SA := SA I_List.Last_Entry.DSAP := DSAP I_List.Last_Entry.SSAP := SSAP I_List.Last_Entry.FC := FC I_List.Last_Entry.DLSDU := DLSDU I_List.Last_Entry. Status := Upd_sts I_List.Last_Entry.Reference := Ref I_List.Num_entry++ SAP_List[DSAP].Ibuffer.Remove()

Function name	Operations
SETUP_SIND(Ref,Upd_sts)	SAP_List[DSAP].Sbuffer.Num_entry-- I_List.Insert() I_List.Last_Entry.DA := DA I_List.Last_Entry.SA := SA I_List.Last_Entry.DSAP := DSAP I_List.Last_Entry.SSAP := SSAP I_List.Last_Entry.FC := FC I_List.Last_Entry.DLSDU := DLSDU I_List.Last_Entry. Status := OK I_List.Num_entry++ SAP_List[DSAP].Sbuffer.Remove()
SAP_CHECK (DSAP)	(SA= SAP_List[DSAP].Access || SAP_List[DSAP].Access=NIL) && (FC.Function is in SAP_List[DSAP].Function_List_R) && DLSDU.Len ≤ SAP_List[DSAP].LenList[FC.Function]
I_BUF(DSAP)	SAP_List[DSAP]. Ibuffer.First_Entry ≠ NIL
B_BUF(DSAP)	SAP_List[DSAP]. Sbuffer.First_Entry ≠ NIL
U_BUF(DSAP)	SAP_List[DSAP]. Ubuffer.High_buffer ≠ NIL || SAP_List[DSAP]. Ubuffer.Low_buffer ≠ NIL

SETUP_REPLY	if (FC.Function = SRD_BCT) R_DA=127
	else R_DA=SA
	If (SAP_List[DSAP]. Ubuffer.High_buffer.Len ≠ 0)
	(Upd_sts := DH, Ref := SAP_List[DSAP]. Ubuffer.High_reference,
	R_SDU := SAP_List[DSAP]. Ubuffer.High_buffer,
	if(SAP_List[DSAP].Ubuffer.High_transmit = SINGLE) SAP_List[DSAP]. Ubuffer.High_buffer=NIL)
	else(Upd_sts := DL, Ref := SAP_List[DSAP]. Ubuffer.Low_reference,
	R_SDU := SAP_List[DSAP]. Ubuffer.Low_buffer,
	if(SAP_List[DSAP].Ubuffer.Low_transmit = SINGLE) SAP_List[DSAP]. Ubuffer.Low_buffer=NIL)

DECODE(func)	if (func = SDN_H) (Service := SDN, Serv_class := High)
	if (func = SDA_H) (Service := SDA, Serv_class := High)
	if (func = SRD_H) (Service := SRD, Serv_class := High)
	if (func = SDN_L) (Service := SDN, Serv_class := Low)
	if (func = SDA_L) (Service := SDA, Serv_class := Low)
	if (func = SRD_L) (Service := SRD, Serv_class := Low)

TIME_OUT	(In_ring_desired=true && T_cnt = 2*TS+5) || (In_ring_desired=false && T_cnt = 259)
INVALID_ FUNCTION	FC.Function = 0 || FC.Function = 1 || FC.Function = 2 || FC.Function = 7 || FC.Function = 8 || FC.Function = 10 || FC.Function = 11
GAP_INIT	Gap_to_do := true GAP_address := (TS + 1) mod (HSA + 1) if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud

Function name	Operations
NEXT_GAP	GAP_Ip_cnt := NIL GAP_address := (GAP_address + 1) mod (HSA + 1) if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud

GAP_UPDATE	Gud_timer := Gud_timer - Ttr,
	if (GAP_to_do && GAP_Ip_cnt = NIL) GAP_Ip_cnt := L_list.Num_entry,
	if (!GAP_to_do && Gud_timer<0)
	(GAP_address := (TS + 1) mod (HSA + 1)
	if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud

TTH_INIT	Trr := Ttr, TRT.start(Ttr)
TTH_UPDATE	Trr :=Ttr-TRT.cv, TRT.start(Ttr)
TTH_AVAILABLE	Trr < TRT.cv
TRT_ON	if (TRT.stopped) TRT.start(Trr)
TRT_OFF	TRT.stop, Trr := TRT.cv, TRT.stopped:= TRUE

A.6 SRU

A.6.1 Overview

SRU contains the following parts:

a) Message oriented Main-SM (SRC)

b) Character Receive SM (CRX)

c) Timer-SM (TIM)

d) Character Send SM (CTX).

Machine internal communication is done without any time delay.

The CTX and CRX machines are used only in asynchrounous mode to transfer a bit oriented stream in a character oriented stream. This task can be done with standard components (called Universal Asynchrounous Receiver / Transmitter) and are not described further. The CRX (together with the physical receive units) shall allow a distortion of the input signal of +/- 10 %.

The timer module is also a standard component that just implements the required timers. The Wait-service selects the specific idle-timers. The respective timer is started with StartIdle and Start. The other timer services are used to indicate the expiration of timers.

The SRC is responsible for coding/encoding of DLPDUs. This module is reponsible for the transmission procedures as well (see clause 21 for further description for send receive procedures).

Both the asynchronous and the synchronous interfaces are shown in Figure A.2.

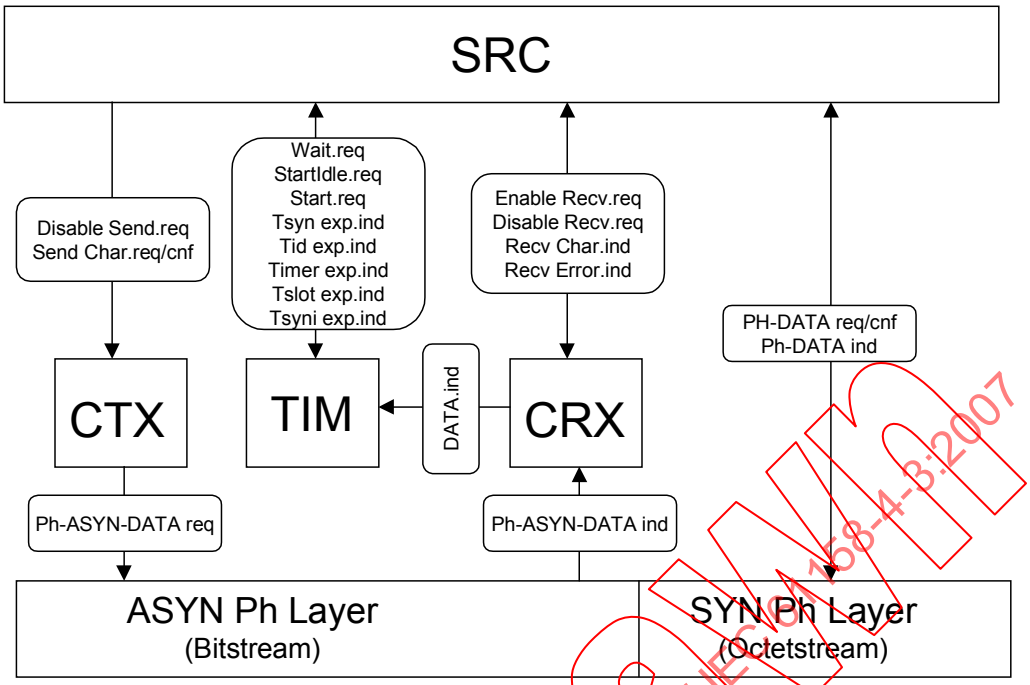


Figure A.2 – Structure of the SRU Machine

A.6.2 Character send SM(CTX)

This state machine implements an UART which has the ability to send without gaps between stop-bit and preceding start-bit. In conjunction with the underlying physical layer it produces a maximum of less than 0,3 % deviation of the nominal data rate. For data rates of 1 500 kbit/s and higher the maximum clock drift of ± 0,03 % should not be exceeded.This model describes a SRC which gets an indication after a complete character has been send (this model implies a zero delay between SRC and CTX). Real devices may implement a double buffering which has influence on the termination of a send sequence (a sender empty indication is required).

Interface to SRC:

— Send_Char.req/.cnf

— Disable_Send.req

A.6.3 Character receive SM (CRX)

This state machine implements an UART which has the ability to receive without gaps between stop-bit and preceding start-bit. It has to detect stop-bit and parity errors. Together with the underlying physical layer it has to detect at least signals with 10 % distortion correctly. If there is a signal change detectible by the underlying physical layer within 10 % and 90 % of the signal, the resulting bit-value is undefined.

When a change to start signal without an valid character is detected with enabled receiver the CRX has to report an error to the SRC. It has be at least as sensitive as the Timer-SM (a trigger for the timer must trigger the receiver as well).

Interface to SRC:

— Receive_Char.ind

— Enable_Recv.req

— Disable_Recv.req

— Receive_Error.ind

A.6.4 Timer-SM (TIM)

The timer module implements three different timers.

One Timer is used for one shot events and will be started by the SRC with a start.req primitive.

Two Timers start when the receiver is disabled or the bus changes from active (logical 0) to inactive (logical 1).

Timer for Tsyn/Tid

This timer is set to 0 when receiver is enabled and the signal is ZERO. If a synchronous Ph Layer is used the timer is loaded with StartIdle.req.

Two thresholds will produce indications:

— the threshold Tsyn is a fixed value;
— the threshold Tid will be loaded according to Tid1/Tid2 with the wait.req primitive.

Timer for Tsyni/Tslot

This timer is reloaded with Tslot -Tsyn when Tsyn-timer expires
This timer is reloaded with Tslot when this timer expires with Tslot
This timer is reloaded with Tsyn, when the receiver becomes active

Interface to SRC:

— Tsyn_exp.ind
— Tid_exp.ind
— Tslot_exp.ind
— Tsyni_exp.ind
— Timer_exp.ind
— Wait.req
— Start.req
— StartIdle.req

CTX, CRX and TIM are not described as state machines.

A.6.5 Primitive definition of SRC

A.6.5.1 Primitives exchanged between DLM and SRC

This interface is not worked out further. The primitives issued by the DLM to the SRC are shown in Table A.17.

Table A.17 – Primitives issued by DLM to SRC

Primitive name	Associated parameters
SRC_RESET.req	(none)

Primitives Exchanged between SRC and DLM.

This interface is not described in detail. The primitives issued by the SRC to the DLM are shown in Table A.18.

Table A.18 – Primitives issued by SRC to DLM

Primitive name	Associated parameters
SRC_RESET.cnf	(none)
MAC_LFAULT.ind	Fault_type
MAC_BFAULT.ind	Fault_type

A.6.5.2 Primitives exchanged between MAC and SRC

The primitives issued by the MAC to the SRC are shown in Table A.19.

Table A.19 – Primitives issued by MAC to SRC

Primitive name	Associated parameters
SRC_SEND_TOKEN.req	DA, SA
SRC_SEND_DATA.req	DA, SA, FC, DSAP, SSAP, DLSDU

The primitives issued by the SRC to the MAC are shown in Table A.20.

Table A.20 – Primitives issued by SRC to MAC

Primitive name	Associated parameters
SRC_SEND_TOKEN.cnf	Status
SRC_SEND_DATA.cnf	(none)
SRC_RECEIVE_DATA.ind	DA, SA, FC, DSAP, SSAP, DLSDU
SRC_RECEIVE_TOKEN.ind	DA, SA
SRC_RECEIVE_ERROR.ind	(none)
SRC_SLOT_EVENT.ind	(none)
SRC_SYNI_EVENT.ind	(none)

A.6.5.3 Parameters of SRC Primitives

All parameters used with primitives exhcanged between the MAC and the SRC are shown in Table A.21.

Table A.21 – Parameters used with primitives exchanged between MAC and SRC

Parameter name	Description
DA	Station address of the receiving unit
SA	Station address of the sending unit
FC	The FC structure is described in Table A.22
DSAP	Identifier of a remote Service Access Point
SSAP	Identifier of the local Service Access Point
DLSDU	Data unit of a DLS-user
Status	Status of the service execution

Table A.22 – FC structure

FC.frame	FC.function	FCB,FCV	FC.stn-type
req	SDN_H,SDN_L, SDA_H, SDA_L, SRD_H, SRD_L, TE, CV, FDL_Status, LSAP_Status, Ident	0,0: all services 0,1: SYN for SRD,SDA 1,0: normal Op. SDA,SRD 1,1: normal Op. SDA,SRD	
rsp	OK, RS, RR, DL, DH, NR, RDL, RDH, SC		Slave, M_n_rdy, M_rdy, M_in_ring

A.6.6 State machine description

The Send Receive Control forms messages out of a stream of data offered by the Physical Layer and vice versa. The SRC supports both Synchronous and Asynchronous Physical layers.

The local variables of the SRC are shown in Table A.23.

Table A.23 – Local variables of SRC

Name	Type	Range	Remark
SDNM	Bool	—	SDN Marker = SDN (was) executed
SType	Enum	(I/R)	Send type (Initiator/Response)
DAE	U8	0,0x80	Address Extension
SAE	U8	0,0x80	Address Extension
F	U8	—	FC-Field
Fcs	U8	—	Checksum
Char	U8	—	—
I	U8	—	—
j	U8	—	—
TOK	Bool	—	Marker Token was sent
RxSD	Bool	—	Marker SD was received
SDdata	U8	—	Storage of SD
C1,C2	U8	—	FCS components

A.6.7 SRC state table

The state table of the SRC is shown in Table A.24.

Table A.24 – SRC state table

No.	Current state	Event /condition ⇒action	Next state
1	any-state	SRC_RESET.req /Data_rate = NIL ⇒ SRC_RESET.cnf	OFFL
2	any-state	SRC_RESET.req /!PHSYN && Data_rate ≠ NIL ⇒ SRC_RESET.cnf	Idle
3	any-state	SRC_RESET.req /PHSYN && Data_rate ≠ NIL ⇒ SRC_RESET.cnf, StartIdle.req	AW_SD
4	Idle	Tsyn_exp.ind ⇒ Enable_Recv.req	AW_SD
5	Idle	Tid_exp.ind ⇒	Idle
6	Idle	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind	Idle
7	Idle	SRC_SEND_TOKEN.req (DA,SA) ⇒ SDNM := FALSE Wait.req (Tid1)	SND_SD4
8	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = FALSE ⇒ SET_SDNM(FC.Function) Wait.req (Tid1)	ISND_SD1
9	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = req && SDNM = FALSE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Wait.req (Tid1)	ISND_SD2
10	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = TRUE ⇒ SET_SDNM(FC.Function) Wait.req (Tid2)	ISND_SD1
11	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = req && SDNM = TRUE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Wait.req (Tid2)	ISND_SD2
12	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = rsp && FC.Function ≠ SC ⇒ SDNM := FALSE	RSND_SD1

No.	Current state	Event /condition ⇒action	Next state
13	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = rsp ⇒ SET_DAE, SET_SAE, SDNM := FALSE , j:=(SAE+DAE)/0x80+3	RSND_SD2
14	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /FC.Frame = rsp && FC.Function = SC ⇒ SDNM := FALSE	SND_SC
15	AW_SD	SRC_SEND_TOKEN.req (DA,SA) ⇒ SDNM := FALSE Wait.req (Tid1)	SND_SD4
16	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = FALSE ⇒ SET_SDNM(FC.Function) Disable_Recv.req Wait.req (Tid1)	ISND_SD1
17	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = req && SDNM = FALSE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Disable_Recv.req Wait.req (Tid1)	ISND_SD2
18	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = TRUE ⇒ SET_SDNM(FC.Function) Disable_Recv.req Wait.req (Tid2)	ISND_SD1
19	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = req && SDNM = TRUE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Disable_Recv.req Wait.req (Tid2)	ISND_SD2
20	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = rsp && FC.Function ≠ SC ⇒ SDNM := FALSE Disable_Recv.req	RSND_SD1
21	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 || DSAP ≠ NIL || SSAP ≠ NIL) && FC.Frame = rsp ⇒ SET_DAE, SET_SAE, SDNM := FALSE , j:=(SAE+DAE)/0x80+3 Disable_Recv.req	RSND_SD2
22	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /FC.Frame = rsp && FC.Function = SC ⇒ SDNM := FALSE Disable_Recv.req	SND_SC
23	AW_SD	RX_DATA(data) /data = SD1 ⇒ SD_count++, DLSDU.Len := 0, CHECKINI(data), i:=0	AW_DA

No.	Current state	Event /condition ⇒action	Next state
24	AW_SD	RX_DATA(data) /data = SD2 ⇒ SD_count++, CHECKINI(data)	AW_LE
25	AW_SD	RX_DATA(data) /data = SD3 ⇒ SD_count++, DLSDU.Len := 8, CHECKINI(data)	AW_DA
26	AW_SD	RX_DATA(data) /!PHSYN && data = SC ⇒ SD_count++, FC.Function := SC, FC.Frame := rsp, i:=0 SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Disable_Recv.req	Idle
27	AW_SD	RX_DATA(data) /PHSYN && data = SC ⇒ SD_count++, FC.Function := SC, FC.Frame := rsp, i:=0, TOK := FALSE	AW_CRC1
28	AW_SD	RX_DATA(data) /data = SD4 ⇒ SD_count++	AW_DAT
29	AW_SD	RX_DATA(data) /data ≠ SC, SD1, SD2, SD3, SD4 ⇒ SD_error_count++ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
30	AW_SD	Recv_Error.ind ⇒ SD_error_count++ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
31	AW_SD	Tslot_exp.ind ⇒ SRC_SLOT_EVENT.ind	AW_SD
32	AW_SD	Tsyn_exp.ind /!PHSYN ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req, Enable_Recv.req	AW_SD
33	AW_SD	Tsyn_exp.ind /PHSYN ⇒	AW_SD
34	AW_SD	Ph-DATA.ind(SOD,data) ⇒	AW_SD
35	AW_LE	RX_DATA(data) /DLSDU.Len > 3 && DLSDU.Len < 250 ⇒ DLSDU.Len := (data - 3), i:=0	AW_LER
36	AW_LE	RX_DATA(data) /DLSDU.Len < 4 || DLSDU.Len > 249 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
37	AW_LER	RX_DATA(data) /data = DLSDU.Len + 3 ⇒	AW_SDR
38	AW_LER	RX_DATA(data) /data ≠ (DLSDU.Len + 3) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle

No.	Current state	Event /condition ⇒action	Next state
39	AW_SDR	RX_DATA(data) /data = SD2 ⇒	AW_DA
40	AW_SDR	RX_DATA(data) /data ≠ SD2 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
41	AW_DA	RX_DATA(data) ⇒ DA := data AND 0x7f, DAE := data AND 0x80, CHECKS(data), DSAP := NIL	AW_SA
42	AW_SA	RX_DATA(data) ⇒ SA := data AND 0x7f, SAE := data AND 0x80, CHECKS(data), SSAP := NIL	AW_FC
43	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE ≠ 0 ⇒ CHECKS(data), SETUP_FC	AW_DSAP
44	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE = 0 && SAE ≠ 0 ⇒ CHECKS(data), SETUP_FC	AW_SSAP
45	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_DATA
46	AW_FC	RX_DATA(data) /!PHSYN && DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_FCS
47	AW_FC	RX_DATA(data) /PHSYN && DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_CRC1
48	AW_FC	RX_DATA(data) /DLSDU.Len = 0 && (DAE ≠ 0 || SAE ≠ 0) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
49	AW_DSAP	RX_DATA(data) /data && 0xc0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
50	AW_DSAP	RX_DATA(data) /DLSDU.Len > 1 && SAE ≠ 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_SSAP
51	AW_DSAP	RX_DATA(data) /DLSDU.Len > 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_DATA
52	AW_DSAP	RX_DATA(data) /!PHSYN && DLSDU.Len = 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_FCS
53	AW_DSAP	RX_DATA(data) /PHSYN && DLSDU.Len = 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_CRC1

No.	Current state	Event /condition ⇒action	Next state
54	AW_DSAP	RX_DATA(data) /DLSDU.Len = 1 && SAE ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
55	AW_SSAP	RX_DATA(data) /data && 0xc0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
56	AW_SSAP	RX_DATA(data) /DLSDU.Len > 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_DATA
57	AW_SSAP	RX_DATA(data) /!PHSYN && DLSDU.Len = 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_FCS
58	AW_SSAP	RX_DATA(data) /PHSYN && DLSDU.Len = 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_CRC1
59	AW_DATA	RX_DATA(data) /DLSDU.Len > i + 1 ⇒ DLSDU.Data[i] := data, i := i + 1, CHECKS(data)	AW_DATA
60	AW_DATA	RX_DATA(data) /!PHSYN && DLSDU.Len = i + 1 ⇒ DLSDU.Data[i] := data, CHECKS(data)	AW_FCS
61	AW_DATA	RX_DATA(data) /PHSYN && DLSDU.Len = i + 1 ⇒ DLSDU.Data[i] := data, CHECKS(data), TOK := FALSE	AW_CRC1
62	AW_FCS	RX_DATA(data) /data = Fcs mod 256 ⇒	AW_ED
63	AW_FCS	RX_DATA(data) /data ≠ (Fcs mod 256) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
64	AW_ED	RX_DATA(data) /data = ED && DA ≠ 0x7f ⇒ SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), Disable_Recv.req	Idle
65	AW_ED	RX_DATA(data) /data = ED && DA = 0x7f ⇒ SDNM := TRUE SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), Disable_Recv.req	Idle
66	AW_ED	RX_DATA(data) /data ≠ ED ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
67	AW_CRC1	RX_DATA(data) /data = C1 ⇒	AW_CRC2

No.	Current state	Event /condition ⇒action	Next state
68	AW_CRC1	RX_DATA(data) /data ≠ C1 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
69	AW_CRC2	RX_DATA(data) /data = C2 ⇒	AW_EODA
70	AW_CRC2	RX_DATA(data) /data ≠ C2 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
71	AW_EODA	RX_DATA(data) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
72	AW_EODA	Ph-DATA.ind(EODA,data) /!TOK && DA ≠ 0x7f ⇒ SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), StartIdle.req	AW_SD
73	AW_EODA	Ph-DATA.ind(EODA,data) /TOK && DA ≠ 0x7f ⇒ SRC_RECEIVE_TOKEN.ind (DA, SA), StartIdle.req	AW_SD
74	AW_EODA	Ph-DATA.ind(EODA,data) /DA = 0x7f ⇒ SDNM := TRUE SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU),StartIdle.req	AW_SD
75	AW_EODAF	Ph-DATA.ind(DATA,data) ⇒	AW_EODAF
76	AW_EODAF	Ph-DATA.ind(EODA,data) ⇒ StartIdle.req	AW_SD
77	AW_DAT	RX_DATA(data) /(data && 0x80) = 0 ⇒	AW_SAT
78	AW_DAT	RX_DATA(data) /(data && 0x80) ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
79	AW_SAT	RX_DATA(data) /!PHSYN && (data && 0x80) = 0 ⇒ SRC_RECEIVE_TOKEN.ind (DA, SA), Disable_Recv.req	Idle
80	AW_SAT	RX_DATA(data) /PHSYN && (data && 0x80) = 0 ⇒ TOK := TRUE	AW_CRC1
81	AW_SAT	RX_DATA(data) /(data && 0x80) ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
82	SND_SD4	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
83	SND_SD4	Tsyn_exp.ind ⇒	SND_SD4

No.	Current state	Event /condition ⇒action	Next state
84	SND_SD4	Timer_exp.ind ⇒	SND_SD4
85	SND_SD4	Tid_exp.ind /!PHSYN ⇒ data:=SD4 Enable_Recv.req, TX_DATA(data)	SND_DAT
86	SND_SD4	Tid_exp.ind /PHSYN ⇒ CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAT
87	SND_SOAT	Ph-DATA.cnf ⇒ data := SD4 Ph-DATA.req(DATA,data)	SND_DAT
88	SND_DAT	TX_DATA_CNF ⇒ CHECKS(DA), data:=DA TX_DATA(data)	SND_SAT
89	SND_SAT	TX_DATA_CNF /!PHSYN ⇒ CHECKS(DA), data:=SA TX_DATA(data)	SND_TOK
90	SND_SAT	TX_DATA_CNF /PHSYN ⇒ CHECKS(DA), data:=SA TX_DATA(data)	ST_SY_C1
91	SND_TOK	TX_DATA_CNF /!RxSD ⇒ Status := no_token_pass SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW_QUI
92	SND_TOK	TX_DATA_CNF /RxSD && SDdata≠SD4 ⇒ Status := token_pass_failed SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW_QUI
93	SND_TOK	TX_DATA_CNF /RxSD && SDdata=SD4 ⇒ Status := token_pass_ok SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW_QUI
94	ST_SY_C1	Ph-DATA.cnf ⇒ data := Crc1 Ph-DATA.req(DATA,data)	ST_SY_C2
95	ST_SY_C2	Ph-DATA.cnf ⇒ data := Crc2 Ph-DATA.req(DATA,data)	ST_SY_EO

No.	Current state	Event /condition ⇒action	Next state
96	ST_SY_EO	Ph-DATA.cnf ⇒ Ph-DATA.req(EODA,data)	ST_SY_END
97	ST_SY_END	Ph-DATA.cnf /PHSYN && !RxSD ⇒ Status := no_token_pass SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW_QUI
98	ST_SY_END	Ph-DATA.cnf /PHSYN && RxSD && SDdata≠SD4 ⇒ Status := token_pass_failed SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW_QUI
99	ST_SY_END	Ph-DATA.cnf /PHSYN && RxSD && SDdata=SD4 ⇒ Status := token_pass_ok SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW_QUI
100	ISND_SD1	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
101	ISND_SD1	Tsyn_exp.ind ⇒	ISND_SD1
102	ISND_SD1	Timer_exp.ind ⇒	ISND_SD1
103	ISND_SD1	Tid_exp.ind /SA ≠ DA || FC.Function ≠ 9 ⇒ SType:=I data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
104	ISND_SD1	Tid_exp.ind /!PHSYN && SA = DA && FC.Function = 9 ⇒ SType:=R data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
105	ISND_SD1	Tid_exp.ind /SA ≠ DA || FC.Function ≠ 9 ⇒ SType:=I CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
106	ISND_SD1	Tid_exp.ind /SA = DA && FC.Function = 9 ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
107	ISND_SD2	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
108	ISND_SD2	Tsyn_exp.ind ⇒	ISND_SD2
109	ISND_SD2	Timer_exp.ind ⇒	ISND_SD2

No.	Current state	Event /condition ⇒action	Next state
110	ISND_SD2	Tid_exp.ind /!PHSYN ⇒ SType:=I data:=SD2 CHECKINI(data) TX_DATA(data)	SND_LE
111	ISND_SD2	Tid_exp.ind /PHSYN ⇒ SType:=I CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAL
112	RSND_SD1	Tsyn_exp.ind ⇒	RSND_SD1
113	RSND_SD1	Timer_exp.ind /!PHSYN ⇒ SType:=R data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
114	RSND_SD1	Timer_exp.ind /PHSYN ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
115	RSND_SD2	Tsyn_exp.ind ⇒	RSND_SD2
116	RSND_SD2	Timer_exp.ind /!PHSYN ⇒ SType:=R data:=SD2 CHECKINI(data) TX_DATA(data)	SND_LE
117	RSND_SD2	Timer_exp.ind /PHSYN ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAL
118	SND_SC	Timer_exp.ind /!PHSYN ⇒ SType:=R, data:=SC, CHECKINI(data) TX_DATA(data)	SND_ENDD
119	SND_SC	Timer_exp.ind /PHSYN ⇒ SType:=R, CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAS
120	SND_SOAL	Ph-DATA.cnf ⇒ data := SD2 Ph-DATA.req(DATA,data)	SND_LE

No.	Current state	Event /condition ⇒action	Next state
121	SND_LE	TX_DATA_CNF ⇒ data := DLSDU.Len + j, CHECKSYN(data) TX_DATA(data)	SND_LER
122	SND_LER	TX_DATA_CNF ⇒ CHECKSYN(data) TX_DATA(data)	SND_SD2R
123	SND_SD2R	TX_DATA_CNF ⇒ data := SD2, CHECKSYN(SD2) TX_DATA(data)	SND_DAD
124	SND_SOAD	Ph-DATA.cnf ⇒ data := SD1 Ph-DATA.req(DATA,data)	SND_DAD
125	SND_DAD	TX_DATA_CNF ⇒ data := DA + DAE, CHECKS(data) TX_DATA(data)	SND_SAD
126	SND_SAD	TX_DATA_CNF ⇒ data := SA + SAE, CHECKS(data) TX_DATA(data)	SND_FC
127	SND_FC	TX_DATA_CNF /DAE ≠ 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_DSAP
128	SND_FC	TX_DATA_CNF /DAE = 0 && SAE ≠ 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_SSAP
129	SND_FC	TX_DATA_CNF /DLSDU.Len > 0 && DAE = 0 && SAE = 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_DATA
130	SND_FC	TX_DATA_CNF /DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_FCS
131	SND_DSAP	TX_DATA_CNF /SAE ≠ 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_SSAP
132	SND_DSAP	TX_DATA_CNF /DLSDU.Len>0 && SAE = 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_DATA
133	SND_DSAP	TX_DATA_CNF /DLSDU.Len = 0 && SAE = 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_FCS

No.	Current state	Event /condition ⇒action	Next state
134	SND_SSAP	TX_DATA_CNF /DLSDU.Len > 0 ⇒ data := SSAP, CHECKS(data) TX_DATA(data)	SND_DATA
135	SND_SSAP	TX_DATA_CNF /DLSDU.Len = 0 ⇒ data := SSAP, CHECKS(data) TX_DATA(data)	SND_FCS
136	SND_DATA	TX_DATA_CNF /DLSDU.Len > i ⇒ data:=DLSDU.Data[i], i := i + 1, CHECKS(data) TX_DATA(data)	SND_DATA
137	SND_DATA	TX_DATA_CNF /!PHSYN && DLSDU.Len = i ⇒ data:=DLSDU.Data[i], CHECKS(data) TX_DATA(data)	SND_FCS
138	SND_DATA	TX_DATA_CNF /PHSYN && DLSDU.Len = i ⇒ data:=DLSDU.Data[i], CHECKS(data) Ph-DATA.req(DATA,data)	SND_SY_C1
139	SND_FCS	TX_DATA_CNF ⇒ data := (Fcs mod 256) TX_DATA(data)	SND_ED
140	SND_ED	TX_DATA_CNF ⇒ data := ED TX_DATA(data)	SND_ENDD
141	SND_ENDD	TX_DATA_CNF ⇒ SRC_SEND_DATA.cnf, Disable_Send.req, Start.req(Tqui)	AW_QUI
142	SND_SOAS	Ph-DATA.cnf ⇒ data := SC Ph-DATA.req(DATA,data)	SD_SY_C1
143	SD_SY_C1	Ph-DATA.cnf ⇒ data := Crc1 Ph-DATA.req(DATA,data)	SD_SY_C2
144	SD_SY_C2	Ph-DATA.cnf ⇒ data := Crc2 Ph-DATA.req(DATA,data)	SD_SY_EO
145	SD_SY_EO	Ph-DATA.cnf ⇒ Ph-DATA.req(EOA,data)	SD_SY_END
146	SD_SY_END	Ph-DATA.cnf ⇒ SRC_SEND_DATA.cnf, Start.req(Tqui)	AW_QUI
147	AW_QUI	Timer_exp.ind /PHSYN ⇒ StartIdle.req	AW_SD

No.	Current state	Event /condition ⇒action	Next state
148	AW_QUI	Timer_exp.ind /!PHSYN && SType=I && !SDNM ⇒ Enable_Recv.req	AW_SD
149	AW_QUI	Timer_exp.ind /!PHSYN && SType=R || SDNM ⇒	idle
150	S-State	RX_DATA /!RxSD ⇒ SDdata := data, RxSD :=TRUE	S-State
151	S-State	RX_DATA /RxSD ⇒	S-State
152	S-State	Ph-DATA.ind(EOA,Data) ⇒	S-State
153	S-State	Ph-DATA.ind(EOAD,Data) ⇒	S-State
154	AW-State	Recv_Error.ind ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
155	AW-State	Tsyn_exp.ind ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req, Enable_Recv.req	AW_SD
156	AW-State	Ph-DATA.ind(EOA,Data) ⇒ SRC_RECEIVE_ERROR.ind, StartIdle.req	AW_SD

A.6.8 Functions

All function of the SRC are shown in Table A.25.

Table A.25 – SRC functions

Function	Description
PHSYN	Data_rate = 31,25 kbit/s
CHECKINI(data)	if (PHSYN) then (Init CRC(C1,C2)) else (Fcs := 0)
CHECKS(data)	if (PHSYN) then (Update CRC(C1,C2)) else (Fcs := 0)
CHECKSYN(data)	Update CRC(C1,C2)
RX_DATA(data)	if (PHSYN) then (Ph-DATA.ind(DATA,data)) else (Recv_Char.ind(data))
TX_DATA(data)	if (PHSYN) then (Ph-DATA.req(DATA,data)) else (Send_Char.req(data))
TX_DATA_CNF	if (PHSYN) then (Ph-DATA.cnf) else (Send_Char.cnf)
SET_SDNM	(if (FC.Function = SDN_L, SDN_H,TE,CV) SDNM = TRUE else SDNM = FALSE)
SETUP_FC	(FC.Function = Char AND 0x0f, if (Char AND 0x40) (FC.FCB = Char AND 0x20, FC.FCV = Char AND 0x10, FC.Frame_type =req) else (FC.Stn-Type= Char AND 0x30, FC.Frame_type =rsp))
SET_F	(F = FC.Function, if (FC.Frame_type =req) (0x80 + FC.FCB * 0x20 + FC.FCV * 0x10) else (F = F + FC.Stn-Type* 0x30))

Annex B
(informative)
–
Type 3 (synchronous): exemplary FCS implementations

This annex provides an example implementation of FCS generation and FCS syndrome checking for Type 3 when used with a synchronous PhL.

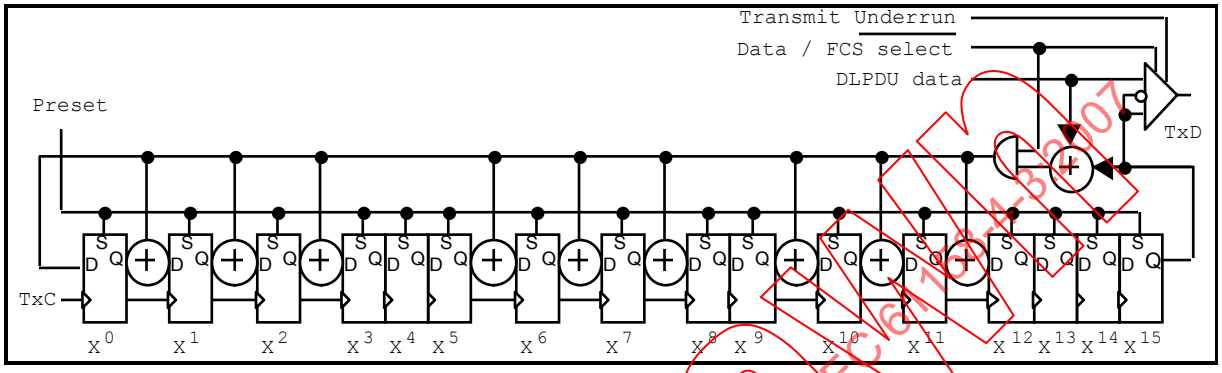


Figure B.1 – Example of FCS generation for Type 3 (synchronous)

In this example, shown in Figure B.1, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial preset of the register before transmission serves to include the initial L(X) term in the FCS computation. Feedback is disabled during transmission of the FCS itself, and the FCS is transmitted complemented to provide the final L(X) inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

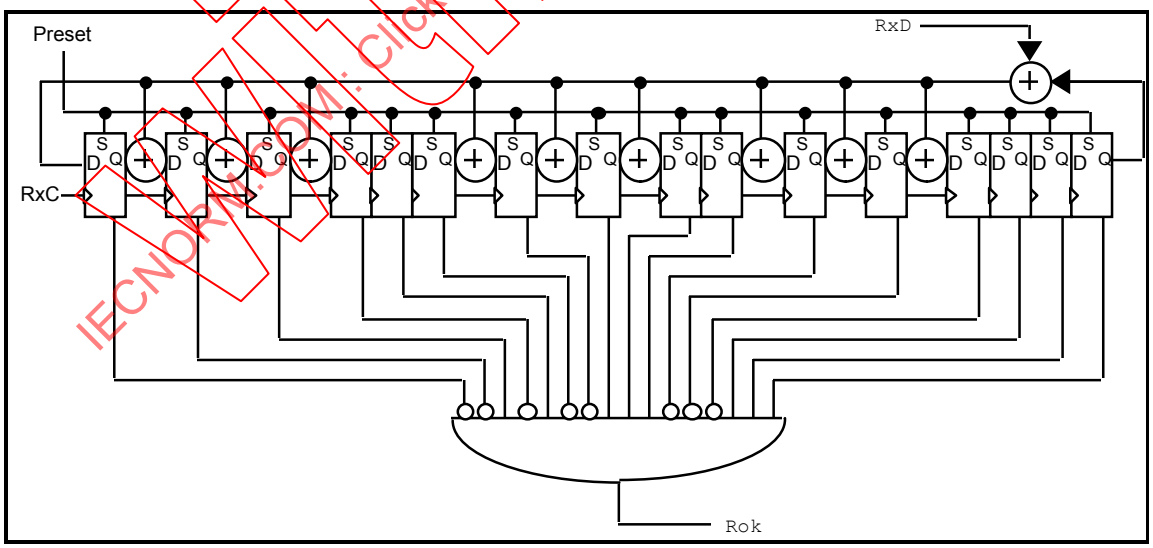


Figure B.2 – Example of FCS syndrome checking on reception for Type 3 (synchronous)

In this example, shown in Figure B.2, the residual FCS is computed in a similar register. The Q outputs of the 16 flip-flops are compared to the expected residual value by the 16-input “and” gate, half of whose inputs are complemented.

In this example, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial pre-set of the register before transmission serves to include the initial L(X) term in the FCS computation. Feedback is disabled during transmission of the FCS it self, and the FCS is transmitted complemented to provide the final L(X) inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

Annex C
(informative)
–
Type 3: Exemplary token procedure and message transfer periods

C.1 Procedure of token passing

For explanatory purposes, the description of the token rotation time in 5.3.2.6 and message priorities in 5.3.2.7 are amended. The timers are presented as functions of the time t.

The token rotation time is measured by using the token-rotation-timer. At the beginning (t_{i-1} in Figure C.1) the token-rotation-timer is loaded with the target rotation time T_{TR} and decrement each bit time.

After receiving a token, the DL-entity may carry out high priority and low priority message transfer periods, according to the following rules.

— On reception of the token (t_i) the T_{RR} (real rotation time for the last token cycle) is derived from the token-rotation-timer by reading its current value that represents T_{TH} and calculating $T_{RR(i)} = T_{TR} - T_{TH}$. The timer is loaded with the value T_{TR} and restarted.

— **One** high priority message transfer period is always possible at this moment (just after token reception).

— A **further** high priority or low priority message transfer period or generally a low priority message transfer period may only be carried out if time to hold the token is still available at the instant of execution, that means that the actual value of T_{RR} of the last, just ending token cycle (from the viewpoint of this station) is less than the current value of the token-rotation-timer.

In order to avoid unnecessary timers in an implementation, the increase of T_{RR} during token usage is measured indirectly by use of the token-rotation-timer, which is measuring the token hold time of the just started actual token cycle. As known from the timer concepts of IEC 61131-3-3, one timer can be used for measurement of parallel but shifted times and therefore the availability of the token holding time is derived from the token-rotation-timer. Thus, a message transfer period can be executed, if at the instant of execution, the read value of the token-rotation-timer is greater than T_{RR}, as shown in Figure C.1.

Figure C.1 shows the relationship of T_{TR}, T_{RR}, T_{TH}.

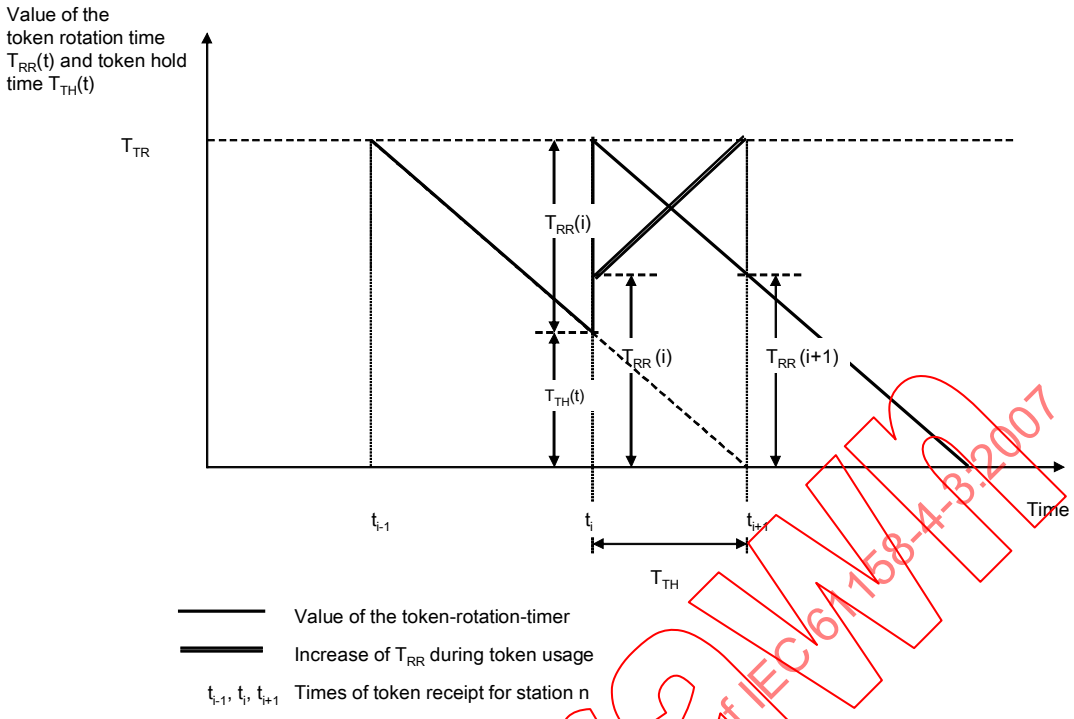


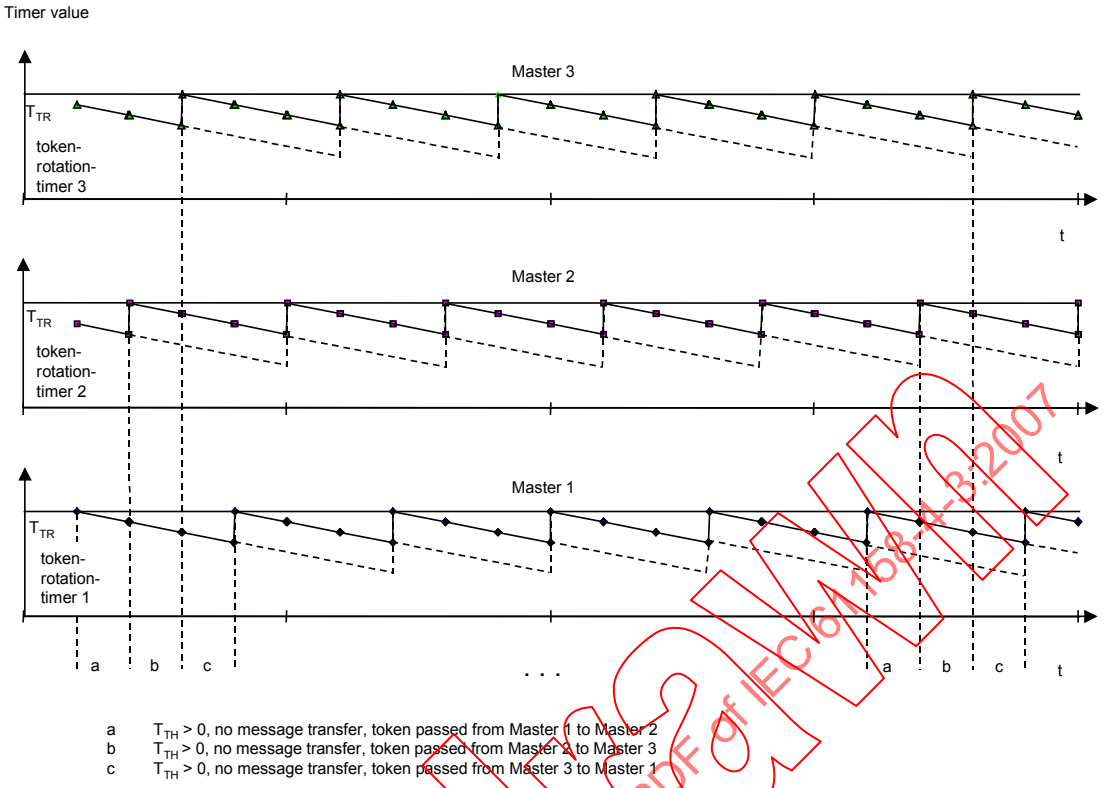
Figure C.1 – Derivation of the token holding time (T_{TH})

C.2 Examples for token passing procedure

The consumption of available token holding time depends of the actual working load. In the following examples the different working load situations are demonstrated together with the usage of the token holding time.

The figures illustrate the behavior of the token passing in cases of varying load. The used time scale only demonstrates the relationship without any relation to real-time. Below are listed some enumerated cases to explain the Token Passing procedure.

Case one: A startup situation is depicted in Figure C.2. The token is passed by the token holder without usage of the available token holding time (T_{TH}). During the start up phase no messages are sent. As depicted in Figure C.2, the timer value of each token holder is always greater than zero in case of token receipt, for example, the read value from token-rotation-timer is greater than zero. Thus T_{TH} is still available.

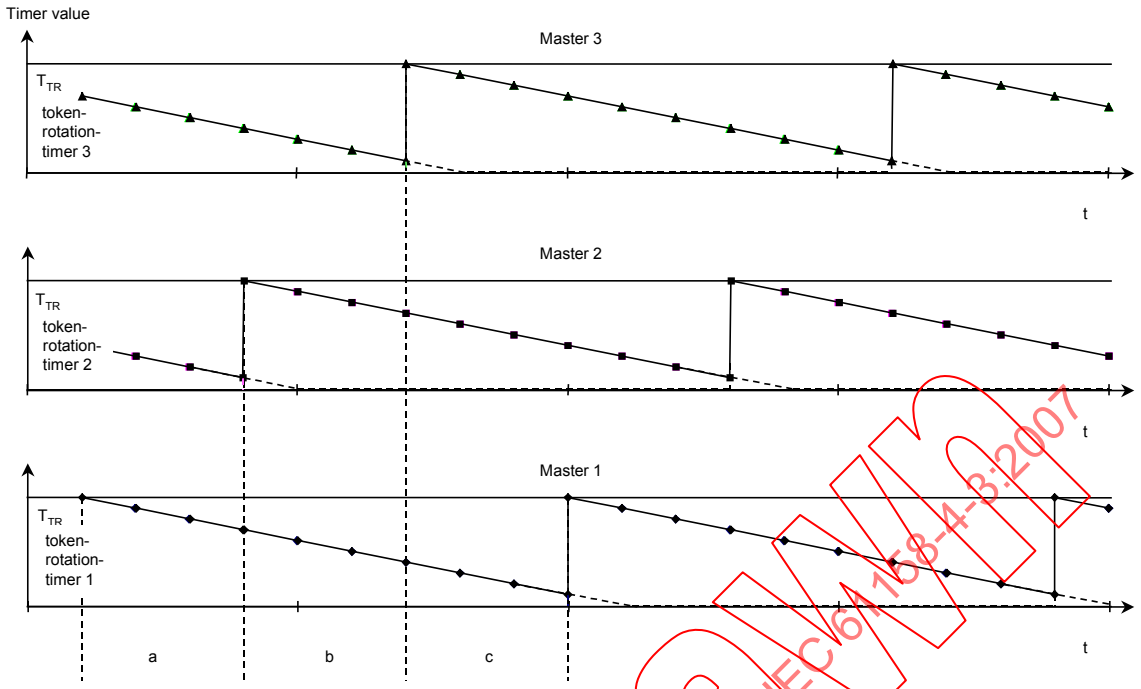


a $T_{TH} > 0$, no message transfer, token passed from Master 1 to Master 2
b $T_{TH} > 0$, no message transfer, token passed from Master 2 to Master 3
c $T_{TH} > 0$, no message transfer, token passed from Master 3 to Master 1

Figure C.2 – No usage of token holding time (T_{TH})

Case two: The bus transfer time is shared fairly between all token holders (see Figure C.3). Each token holder only uses a part of the token holding time (T_{TH}). The actual token holder only sends one message in each token cycle. After that it passes the token to the next token holder. The T_{TH} used by a token holder has a constant (for that token holder) value (see Figure C.3: a, b, c).

As depicted in Figure C.3 some T_{TH} time is still available in each station at the time of token receipt because the read value of the token-rotation-timer is greater than zero at that instant.



a $T_{TH} > 0$, message transfer by Master 1, token passed from Master 1 to Master 2
b $T_{TH} > 0$, message transfer by Master 2, token passed from Master 2 to Master 3
c $T_{TH} > 0$, message transfer by Master 3, token passed from Master 3 to Master 1

Figure C.3 – Usage of token holding time (T_{TH}) for message transfer (equivalence between T_{TH} of each Master station)

Case three: This case is characterized by a load change, from no load to a maximum load, that is a worst-case situation. This means that each token holder has a maximum amount of messages to send after a period of no message requests. That leads to the fact that each token holder tries to use the whole available token holding time for that token rotation. The Token Passing procedure guarantees a fair access to the medium for all token holders even in this situation.

In case of using the whole available bus transfer time (T_{TR}) by one token holder within one token cycle, no local T_{TH} time is available on token arrival in any of the other token holders for that cycle, so they are only allowed to send one high priority message. Within the next token cycle the token holder that had used the whole T_{TR} previously is limited to one high priority message, after which it immediately passes the token. This situation is depicted in Figure C.4. After a start up phase in which the token is passed by the token holders, Master station 1 uses the whole available bus transfer time to send messages (see Figure C.4: d). In this case Master station 2 and Master station 3 have no T_{TH}, indicated by a read value of zero of the token-rotation-timer at token receipt. Because Master station 3 has to transfer a high priority message it sends this message after receiving the token (see Figure C.4: f).

In the next token cycle (g), Master station 1 passes the token immediately after token receipt as its read timer value has reached zero and it has no high priority messages waiting. Now Master station 2 has the possibility to use the whole bus transfer time for message transfer within this token cycle (h). Master station 2 also uses all the available T_{TH} time because of its high working load, after which it is forced to pass the token to Master station 3.

Master station 3 is now limited by its local T_{TH} that has reached zero, and as it has no high priority messages, it passes the token immediately after receipt (see Figure C.4: i). In the

following token cycle no T_{TH} is available for Master station 1 or Master station 2. They could send one high priority message, but in this example they pass the token immediately without message transfer (see Figure C.4: j and k). At this point in the cycle the read value of the token-rotation-timer in Master station 3 is greater than zero, so this token holder can use the available T_{TH} time for message transfer (see Figure C.4: l).

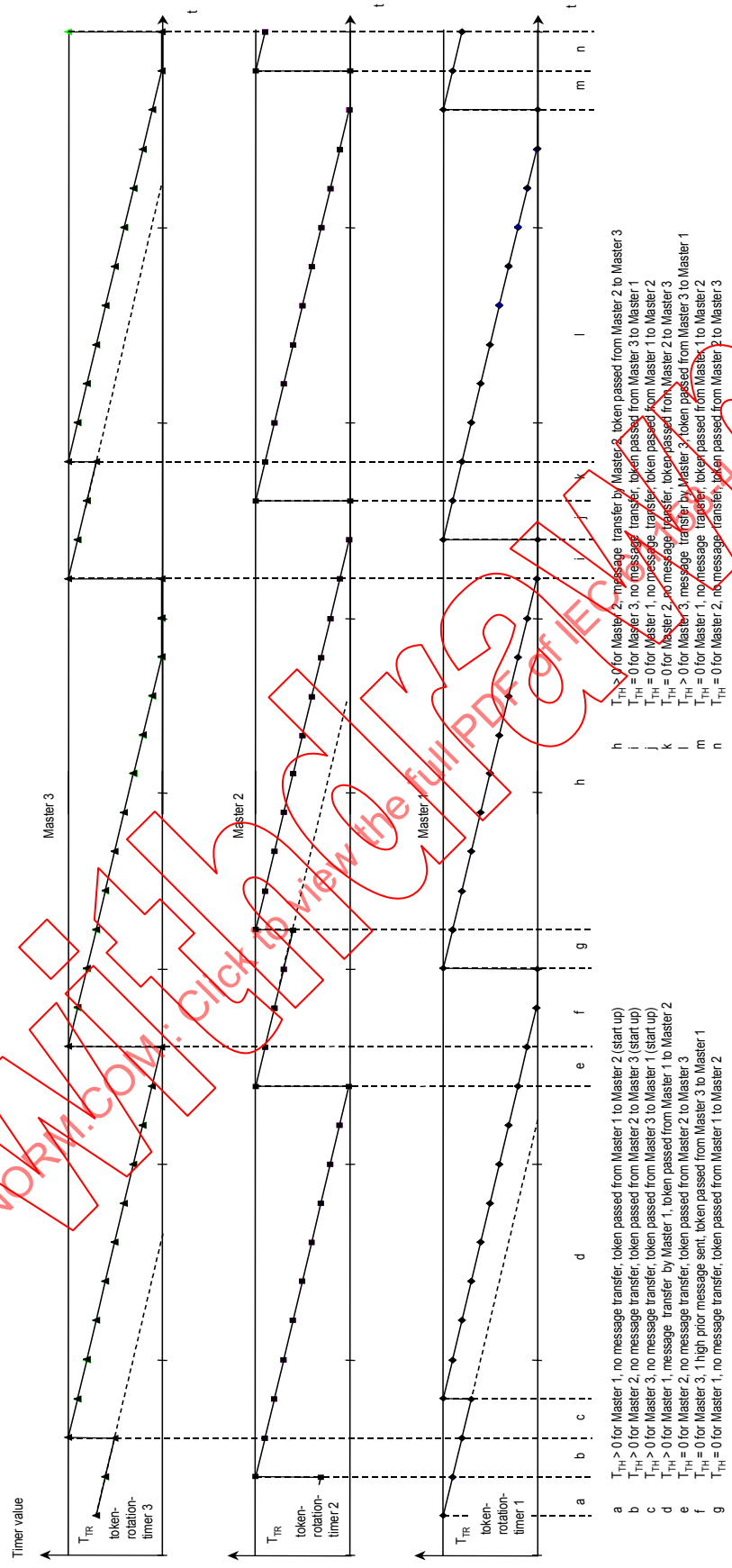


Figure C.4 – Usage of token holding time (T_{TH}) in different working load situations

C.3 Examples for message transfer periods – asynchronous transmission

The following examples give a detailed overview about the possible message transfer periods for asynchronous transmission according to typical messages used by the DLS-user:

Symbols used in the following examples:

T_{SC}	bit time to transmit a single character SC; (11 bits)
T_{SAP}	bit time to transmit D_SAP_index and S_SAP_index; (22 bits)
T_{SD1}	bit time to transmit a DLPDU with start delimiter SD1; (66 bits)
T_{SD2}	bit time to transmit a DLPDU with start delimiter SD2 (without DLSDU and Address-Extensions); (99 bits)
T_{SD3}	bit time to transmit a DLPDU with start delimiter SD3, DLPDU format of fix length with data field; (154 bits)
Req-PDU	number of octets in the DATA_UNIT
Res-PDU	number of octets in the DATA_UNIT

EXAMPLES

a) GAP request with reply (T_{GAP1})

$$T_{GAP1} = T_{ID1} + T_{SDR} + 2 \times T_{SD1} + 2 \times T_{TD} \quad \text{(C.1)}$$

b) GAP request with no reply (T_{GAP2})

$$T_{GAP2} = T_{ID1} + T_{SD1} + T_{SL} \quad \text{(C.2)}$$

c) Cyclic message transfer (T_{CY})

$$T_{CY1} = T_{ID1} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 11 \times (\text{Req-PDU} + \text{Res-PDU}) \quad \text{(C.3)}$$

$$T_{CY2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SC} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad \text{(C.4)}$$

$$T_{CY3} = T_{ID1} + T_{SD2} + T_{SL} + 11 \times \text{Req-PDU} \quad \text{(C.5)}$$

d) Acyclic message transfer with response (T_{ACR})

$$T_{ACR1} = T_{ID1} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 2 \times T_{SAP} + 11 \times \text{Res-PDU} \quad \text{(C.6)}$$

$$T_{ACR2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SD3} + 2 \times T_{TD} + T_{SAP} \quad \text{(C.7)}$$

e) Acyclic message transfer with acknowledgement (T_{ACY})

$$T_{ACY1} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SC} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad \text{(C.8)}$$

$$T_{ACY2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SD1} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad \text{(C.9)}$$

f) Broadcast message transfer (T_{BC})

$$T_{BC} = T_{ID2} + T_{SD2} + T_{SAP} + T_{TD} + 11 \times \text{Req-PDU} \quad \text{(C.10)}$$

C.4 Examples for message transfer periods – synchronous transmission

The following examples give a detailed overview about the possible message transfer periods for synchronous transmission according to typical messages used by the DLS-user:

Symbols used in the following examples:

T_{SC}	bit time to transmit a single character SC including CRC; (24 bits)
T_{SAP}	bit time to transmit D_SAP_index and S_SAP_index; (16 bits)
T_{SD1}	bit time to transmit a DLPDU with start delimiter SD1; (48 bits)
T_{SD2}	bit time to transmit a DLPDU with start delimiter SD2 (without DLSDU and Address-Extensions); (72 bits)
T_{SD3}	bit time to transmit a DLPDU with start delimiter SD3, DLPDU format of fix length with data field; (112 bits)
T_{PhL}	bit time to transmit preamble, physical start delimiter and end delimiters as well as post-transmission gap time; unit in bit (40 bits: 16 bits preamble, 8 bits start delimiter and 8 bits end delimiter, 8 bits post-transmission gap time)
Req-PDU	number of octets in the DATA_UNIT
Res-PDU	number of octets in the DATA_UNIT

EXAMPLES

a) GAP request with reply (T_{GAP1})

$$T_{GAP1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD1} + 2 \times T_{TD} \quad \text{(C.11)}$$

b) GAP request with no reply (T_{GAP2})

$$T_{GAP2} = T_{PhL} + T_{SD1} + T_{SL} \quad \text{(C.12)}$$

c) Cyclic message transfer (T_{CY})

$$T_{CY1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 8 \times (\text{Req-PDU} + \text{Res-PDU}) \quad \text{(C.13)}$$

$$T_{CY2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SC} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad \text{(C.14)}$$

$$T_{CY3} = T_{PhL} + T_{SD2} + T_{SL} + 8 \times \text{Req-PDU} \quad \text{(C.15)}$$

d) Acyclic message transfer with response (T_{ACR})

$$T_{ACR1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 2 \times T_{SAP} + 8 \times \text{Res-PDU} \quad \text{(C.16)}$$

$$T_{ACR2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SD3} + 2 \times T_{TD} + T_{SAP} \quad \text{(C.17)}$$

e) Acyclic message transfer with acknowledgement (T_{ACY})

$$T_{ACY1} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SC} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad \text{(C.18)}$$

$$T_{ACY2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SD1} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad \text{(C.19)}$$

f) Broadcast message transfer (T_{BC})

$$T_{BC} = T_{PhL} + T_{SD2} + T_{SAP} + T_{TD} + 8 \times \text{Req-PDU} \quad \text{(C.20)}$$

Bibliography

IEC 60870-5-1, *Telecontrol equipment and systems – Part 5: Transmission protocols – Section One: Transmission frame formats*

IEC/TR 61158-1 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-5-3, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5-3: Application layer service definition – Type 3 elements*

IEC 61158-6-3, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 6-3: Application layer protocol specification – Type 3 elements*

IEC 61784-1 (Ed.2.0), *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

ISO/IEC 3309, *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure.*

ISO/IEC 8802 (all parts), *Information technology – Telecommunications and information exchange between systems – Local and Metropolitan area networks*

ISO/IEC TR 8802-1, *Specific requirements – Part 1: Overview of Local Area Network Standards*

ISO/IEC 8802-2, *Specific requirements – Part 2: Logical link control*

ISO/IEC 8802-3, *Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

ISO/IEC 8802-5, *Specific requirements – Part 5: Token ring access method and physical layer specifications*

ISO/IEC 8802-4:1990, *Information processing systems – Local area networks – Part 4: Token-passing bus access method and physical layer specifications*

ISO/IEC 9314-2, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 2: Token Ring Media Access Control (MAC)*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 4-3: Spécification des protocoles des couches de liaison de données – Éléments de Type 3

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.

NOTE L'utilisation de certains des types de protocoles est restreinte par les détenteurs des droits de propriété intellectuelle correspondants. Quoi qu'il en soit, l'engagement pris par les détenteurs, quant à une diffusion limitée desdits droits de propriété intellectuelle, permet d'utiliser un type particulier de protocole de Couche Liaison de données avec des protocoles de Couche Physique et de Couche Application dans les combinaisons de types explicitement spécifiées dans la série CEI 61784. L'utilisation de divers types de protocoles dans d'autres combinaisons peut nécessiter l'autorisation de leurs détenteurs de droits de propriétés intellectuelle respectifs.
La Commission Electrotechnique Internationale (CEI) attire l'attention sur le fait qu'il est déclaré que la conformité avec la présente norme peut impliquer l'utilisation de brevets présentés ci-après, dans lesquels la notation [xx] indique le détenteur du brevet:

Type 3 et autres types éventuels:

DE 36 43 979 C2	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung
DE 36 43 979 A1	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung

La CEI ne prend aucunement position en ce qui concerne la démonstration, la validité et l'étendue de ces droits de propriété.

Les détenteurs de ces droits de propriété ont donné l'assurance à la CEI qu'ils consentent à négocier des licences avec des demandeurs du monde entier, en des termes et à des conditions raisonnables et non discriminatoires. A

ce propos, la déclaration des détenteurs des droits de propriétés est enregistrée à la CEI. Des informations peuvent être obtenues auprès de:

[SI]: SIEMENS AG
Ludwig Winkel
Siemensallee 84
D-76181 Karlsruhe
Allemagne

L'attention est attirée sur le fait que certains des éléments de la présente norme peuvent faire l'objet de droits de propriété industrielle distincts de ceux mentionnés ci-dessus. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété ou de ne pas avoir signalé leur existence.

La Norme internationale CEI 61158-4-3 a été établie par le sous-comité 65C: Réseaux de communication industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette première édition et ses parties d'accompagnement de la sous-série CEI 61158-4 annulent et remplacent la CEI 61158-4:2003. La présente édition de cette partie constitue une révision éditoriale.

Cette édition de la CEI 61158-4 inclut les modifications majeures suivantes par rapport à l'édition précédente:

a) suppression du précédent bus de terrain de Type 6 et de la référence à une couche de liaison de données de bus de terrain de Type 5, en raison du manque d'adéquation au marché;

b) ajout de nouveaux types de bus de terrain;

c) division de cette partie en parties multiples numérotées -4-1, -4-2, ..., -4-19.

La présente version bilingue (2014-08) correspond à la version anglaise monolingue publiée en 2007-12.

Le texte anglais de cette norme est issu des documents 65C/474/FDIS et 65C/485/RVD.

Le rapport de vote 65C/485/RVD donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous http://webstore.iec.ch dans les données relatives à la publication recherchée. A cette date, la publication sera:

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

NOTE La révision de la présente norme sera synchronisée avec les autres parties de la série CEI 61158.

La liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, est disponible sur le site web de la CEI.

INTRODUCTION

Cette partie de la CEI 61158 fait partie d'une série élaborée pour faciliter l'interconnexion des composants de systèmes d'automatisation. Elle est apparentée à d'autres normes de cet ensemble, comme défini par le modèle de référence de bus de terrain “à trois couches” décrit dans la CEI/TR 61158-1.

Le protocole de liaison de données assure le service de liaison de données en utilisant les services disponibles à partir de la couche physique. Le principal objectif de la présente norme est de fournir un ensemble de règles de communication exprimées en termes de procédures à appliquer par des entités de liaison de données (DLE) homologues au cours de la communication. Ces règles de communication sont destinées à fournir une base saine de développement, de manière à satisfaire à divers objectifs:

a) servir de guide pour les ingénieurs d'application et les concepteurs;

b) être utilisées pour les essais et l'acquisition d'équipements;

c) servir de base, dans le cadre d'un accord donné, à l'admission de systèmes dans l'environnement OSI;

d) approfondir les connaissances en matière de communications critiques du point de vue temporel (à priorité stricte) dans le cadre de l'OSI.

La présente norme couvre notamment la communication et l'interaction de capteurs, organes terminaux et autres dispositifs d'automatisation. L'utilisation de la présente norme, associée à d'autres normes qui font partie des modèles de référence OSI ou bus de terrain, permet de combiner et de faire fonctionner ensemble des systèmes qui seraient autrement incompatibles.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 4-3: Spécification des protocoles des couches de liaison de données – Éléments de Type 3

1 Domaine d'application

1.1 Généralités

La couche de liaison de données permet la communication de messages de base, critiques du point de vue temporel, entre dispositifs dans un environnement d'automatisation.

Ce protocole donne les moyens de communiquer à un sous-ensemble "maître" présélectionné d'entités de liaison de données de manière asynchrone cyclique, séquentiellement pour chacune de ces entités de liaison de données. D'autres entités de liaison de données communiquent uniquement si elles sont autorisées et déléguées par ces entités de liaison de données "maîtres".

Les communications d'un maître avec d'autres entités de liaison de données peuvent être cycliques ou acycliques avec accès selon un ordre de priorité, ou une combinaison des deux.

Ce protocole est un moyen de partager de manière équitable les ressources de communication disponibles. Il comporte des dispositions de fonctionnement synchrone et isochrone.

1.2 Spécifications

La présente norme spécifie :

a) des procédures de transfert en temps opportun de données et d'informations de commande d'une entité utilisateur de liaison de données à une entité utilisateur homologue ainsi qu'entre entités de liaison de données qui constituent le fournisseur de services distribués de la liaison;

b) la structure des DLPDU de bus de terrain utilisées pour le transfert des données et les des informations de commande par le protocole objet de la présente norme, ainsi que leur représentation en tant qu'unité de données d'interface de couche physique.

1.3 Procédures

Les procédures sont définies en termes:

a) d'interactions entre entités DL (DLE) homologues par échange de DLPDU de bus de terrain;

b) d'interactions entre un fournisseur de services DL (DLS) et un utilisateur DLS au sein du même système, par l'échange de primitives DLS;

c) d'interactions entre un fournisseur DLS et un fournisseur de services Ph au sein du même système, par l'échange de primitives de services Ph.

1.4 Applicabilité

Ces procédures sont applicables à des instances de communication entre systèmes qui prennent en charge des services de communication à priorité stricte au sein de la couche de liaison de données de l'OSI ou des modèles de référence des bus de terrain et qui

nécessitent la faculté de s'interconnecter dans un environnement OSI (Interconnexion de systèmes ouverts).

Les profils constituent un moyen simple, à attributs multiples, qui permet de résumer les capacités d'une mise en œuvre et par conséquent, son applicabilité à divers besoins de communication à priorité stricte.

1.5 Conformité

La présente Norme spécifie également les conditions de conformité des systèmes mettant en œuvre ces procédures. La présente norme ne fournit pas d'essais destinés à démontrer la conformité à ces exigences.

2 Références normatives

Les documents de référence suivants sont indispensables à l'application de la présente norme. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les amendements).

CEI 61158-2 (Ed.4.0), *Communications numériques pour les systèmes de mesure et de commande – Bus de terrain utilisé dans les systèmes de contrôle industriels – Partie 2: Spécification de couche physique et définition des services*

IEC 61158-3-3, *Industrial communication networks – Fieldbus specifications – Part 3-3: Data-link layer service definition – Type 3 elements* (disponible uniquement en anglais)

ISO/CEI 2022, *Technologies de l'information – Structure de code de caractères et techniques d'extension*

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le Modèle de base*

ISO/CEI 7498-3, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Dénomination et adressage*

ISO/CEI 10731, *Technologies de l'information – interconnexion de systèmes ouverts – modèle de référence de base – conventions pour la définition des services OSI*

ISO 1177, *Traitement de l'information – Structure des caractères pour la transmission arythmique et synchrone orientée caractère*

3 Termes, définitions, symboles et abréviations

Pour les besoins de la présente norme, les termes, définitions, symboles et abréviations suivants s'appliquent.

3.1 Termes et définitions du modèle de référence

La présente norme repose en partie sur les concepts développés dans l'ISO/CEI 7498-1 et l'ISO/CEI 7498-3; elle utilise les termes suivants qui y sont définis.

3.1.1 adresse DL appelée [7498-3]

3.1.2 adresse DL appelante [7498-3]

3.1.3 connexion centralisée à points d'extrémité multiples [7498-1]

3.1.4 (N)-entités correspondantes
entités DL correspondantes (N=2)
entités Ph correspondantes (N=1) [7498-1]

3.1.5 démultiplexage [7498-1]

3.1.6 adresse DL [7498-3]

3.1.7 mise en correspondance d'adresses DL [7498-1]

3.1.8 connexion DL [7498-1]

3.1.9 point d'extrémité de connexion DL [7498-1]

3.1.10 identifiant de point d'extrémité de connexion DL [7498-1]

3.1.11 émission en mode connexion DL [7498-1]

3.1.12 émission en mode sans connexion DL [7498-1]

3.1.13 destination de données DL [7498-1]

3.1.14 origine de données DL [7498-1]

3.1.15 transmission en duplex DL [7498-1]

3.1.16 fonctionnalité DL [7498-1]

3.1.17 vue locale DL [7498-3]

3.1.18 nom DL [7498-3]

3.1.19 protocole DL [7498-1]

3.1.20 identifiant de connexion de protocole DL [7498-1]

3.1.21 information de contrôle de protocole DL [7498-1]

3.1.22 unité de données de protocole DL [7498-1]

3.1.23 identifiant de version de protocole DL [7498-1]

3.1.24 relais DL [7498-1]

3.1.25 identifiant de connexion de service DL [7498-1]

3.1.26 unité de données de service DL [7498-1]

3.1.27 transmission simplex DL [7498-1]

3.1.28 sous-système DL [7498-1]

3.1.29 données utilisateur DL [7498-1]

3.1.30 contrôle de flux [7498-1]

3.1.31 gestion de couche [7498-1]

3.1.32 multiplexage [7498-3]

3.1.33 autorité de nommage (adressage) [7498-3]

3.1.34 domaine de nommage (adressage) [7498-3]

3.1.35 sous-domaine de nommage (adressage) [7498-3]

3.1.36 (N)-entité [7498-1]
entité DL
entité Ph

3.1.37 (N)- unité de données d'interface [7498-1]
unité de données de service DL (N=2)
unité de données d'interface Ph (N=1)

3.1.38 (N)-couche [7498-1]
couche DL (N=2)
couche Ph (N=1)

3.1.39 (N)-service [7498-1]
service DL (N=2)
service Ph (N=1)

3.1.40 (N)-point d'accès de service [7498-1]
point d'accès de service DL (N=2)
point d'accès de service Ph (N=1)

3.1.41 (N)-adresse de point d'accès de service [7498-1]
adresse de point d'accès de service DL (N=2)
adresse de point d'accès de service Ph (N=1)

3.1.42 entités homologues [7498-1]

3.1.43 informations de commande d'interface Ph [7498-1]

3.1.44 données d'interface Ph [7498-1]

3.1.45 nom de primitive [7498-3]

3.1.46 réassembler [7498-1]

3.1.47 recombiner [7498-1]

3.1.48 réinitialiser [7498-1]

3.1.49 adresse DL répondante [7498-3]

3.1.50 acheminement [7498-1]

3.1.51 segmenter [7498-1]

3.1.52 séquencer [7498-1]

3.1.53 subdiviser [7498-1]

3.1.54 nom synonyme [7498-3]

3.1.55 gestion systèmes [7498-1]

3.2 Termes et définitions de convention de service

La présente norme utilise également les termes suivants définis dans l'ISO/CEI 10731 dans la mesure où ils s'appliquent à la couche liaison de données:

3.2.1 accepteur

3.2.2 service asymétrique

3.2.3 confirmation (primitive de);
requestor.deliver (remise au demandeur) (primitive de)

3.2.4 remise (primitive de)

3.2.5 fonctionnalité confirmée DL

3.2.6 fonctionnalité DL

3.2.7 vue locale DL

3.2.8 fonctionnalité obligatoire DL

3.2.9 fonctionnalité non confirmée DL

3.2.10 fonctionnalité lancée par le fournisseur DL

3.2.11 fonctionnalité facultative de fournisseur DL

3.2.12 primitive de service DL;
primitive

3.2.13 fournisseur de service DL

3.2.14 utilisateur de service DL

3.2.15 fonctionnalité facultative d'utilisateur DL

3.2.16 indication (primitive de)
acceptor.deliver (remise.accepteur) (primitive de)

3.2.17 multi-homologue

3.2.18 demande (primitive);
requestor.submit (soumission au demandeur) (primitive)

3.2.19 demandeur

3.2.20 réponse (primitive);
acceptor.submit (soumission à l'accepteur) (primitive)

3.2.21 soumission (primitive)

3.2.22 service symétrique

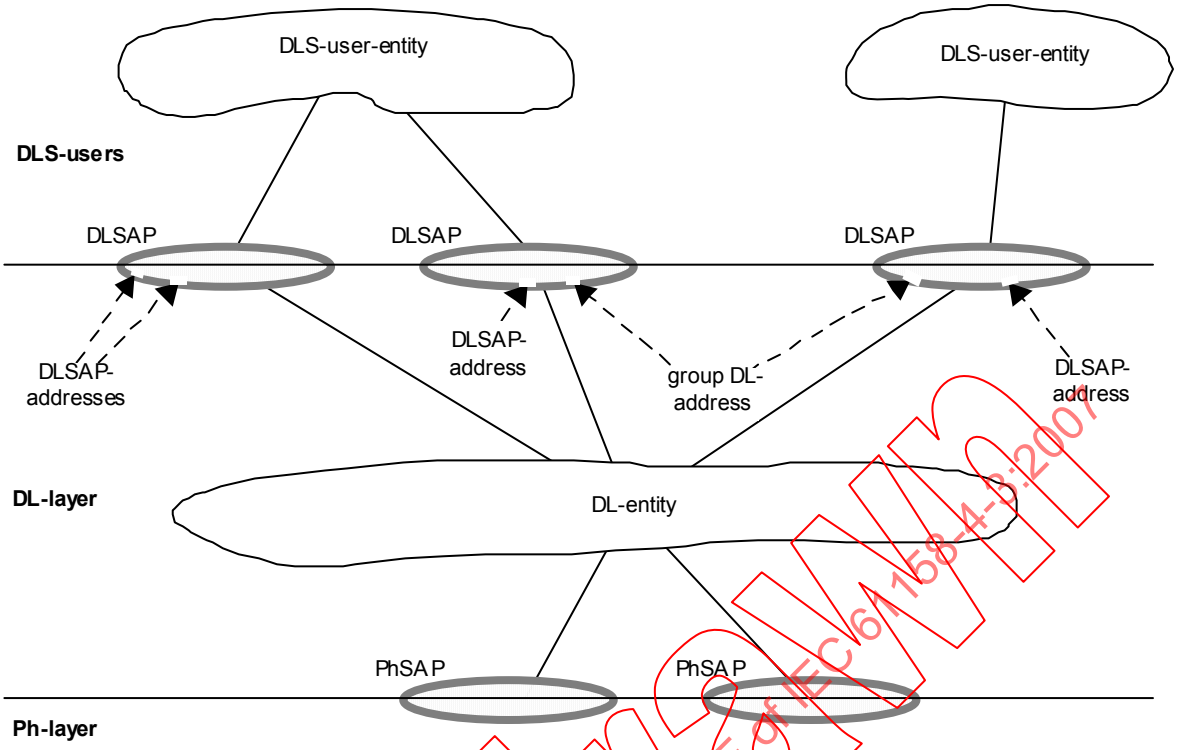
3.3 Termes et définitions communs

NOTE De nombreuses définitions sont communes à plusieurs types de protocoles; elles ne sont pas nécessairement utilisées par tous les types de protocoles.

3.3.1
segment, liaison, liaison locale DL
sous-réseau DL simple dans lequel toute DLE connectée peut directement communiquer, sans aucune intervention de relais DL, lorsque chacune de ces DLE participant à une instance de communication est simultanément attentive au sous-réseau DL au cours de la (ou des) période(s) de tentative de communication

3.3.2
DLSAP
point distinctif où des services DL sont fournis par une entité DL simple à une entité simple de couche supérieure

NOTE Cette définition, tirée de l'ISO/CEI 7498-1, est ici réitérée pour faciliter la compréhension de l'importante distinction qui existe entre les DLSAP et leurs adresses DL. (Voir la Figure 1.)



Légende

Anglais	Français
DLS user entity	Entité utilisateur DLS
DLS users	Utilisateurs DLS
DLSAP addresses	Adresses DLSAP
Group DL address	Adresse de groupe DL
DL layer	Couche DL
DL entity	Entité DL
Ph layer	Couche Ph

NOTE 1 Les DLSAP et les PhSAP sont décrits comme des ovales qui recouvrent la frontière entre deux couches adjacentes.

NOTE 2 Les adresses DL sont décrites comme désignant de petits intervalles (points d'accès) dans la partie DLL d'un DLSAP.

NOTE 3 Une entité DL simple peut avoir plusieurs adresses DLSAP et adresses DL de groupe, associées à un DLSAP simple.

Figure 1 – Relations entre DLSAP, adresses DLSAP et adresses DL de groupe

3.3.3
adresse DL(SAP)
il s'agit soit d'une adresse DLSAP individuelle, désignant un simple DLSAP d'un simple utilisateur DLS, soit d'une adresse DL de groupe désignant potentiellement plusieurs DLSAP appartenant chacun à un simple utilisateur DLS

NOTE Cette terminologie est choisie parce que l'ISO/CEI 7498-3 ne permet pas d'utiliser le terme adresse DLSAP pour désigner plusieurs DLSAP simples au niveau d'un utilisateur DLS simple.

3.3.4
adresse DLSAP (individuelle)
adresse DL qui désigne un seul DLSAP au sein de la liaison étendue

NOTE Une entité DL simple peut avoir plusieurs adresses DLSAP, associées à un DLSAP simple.

3.3.5
liaison étendue
sous-réseau DL, constitué d'un ensemble maximal de liaisons interconnectées par des relais DL, partageant un espace de nom DL simple (adresse DL) dans lequel chacune des entités DL connectées peut communiquer avec l'autre, soit directement, soit au moyen d'une ou de plusieurs entités de relais DL intervenantes

NOTE Une liaison étendue peut être constituée d'une seule liaison uniquement.

3.3.6
trame
synonyme discrédité de DLPDU

3.3.7
adresse DL de groupe
adresse DL qui désigne potentiellement plusieurs DLSAP au sein de la liaison étendue. Une entité DL simple peut avoir plusieurs adresses DL de groupe, associées à un DLSAP simple. Une entité DL simple peut également avoir une seule adresse DL de groupe, associée à plusieurs DLSAP.

3.3.8
nœud
entité DL simple telle qu'elle apparaît sur une liaison locale

3.3.9
utilisateur DLS de réception
utilisateur de services DL qui agit comme destinataire de données utilisateur DL

NOTE Un utilisateur de services DL peut être en même temps un utilisateur DLS d'émission et de réception.

3.3.10
utilisateur DLS d'émission
utilisateur de services DL qui agit comme origine de données utilisateur DL

3.4 Définitions supplémentaires de Type 3

3.4.1
DLPDU d'acquittement
DLPDU de réponse qui ne contient aucune DLSDU

3.4.2
extension d'adresse
adresse DLSAP ou adresse de région/segment

3.4.3
temps binaire
temps nécessaire pour transmettre un bit

3.4.4
échange de messages confirmé
transfert de données complet avec DLPDU de demande et d'acquittement ou de réponse

3.4.5
controller_type (type de contrôleur)
classe de matériel de l'entité de communication

3.4.6
maître courant
détenteur du jeton

3.4.7
DLPDU de données
DLPDU transportant une DLSDU d'un utilisateur DLS local à un utilisateur DLS distant

3.4.8
DL_status (état DL)
état qui indique le résultat de l'exécution de la demande correspondante

3.4.9
GAP (intervalle)
plage d'adresses DL de stations (DLE) de cette station (TS) à la station suivante (NS) sur l'anneau à jeton logique; à l'exclusion des stations au-dessus de l'adresse de station la plus élevée (HSA)

3.4.10
maintenance d'intervalle GAP
enregistrement de nouvelles stations maîtres et esclaves

3.4.11
mode isochrone
mode opérationnel particulier qui implique à la fois un cycle constant (isochrone) avec une planification fixe de messages de haute et basse priorité et la synchronisation des utilisateurs DLS sur ce cycle constant (isochrone)

3.4.12
utilisateur DLS local
utilisateur DLS qui lance le service courant

3.4.13
échange de messages
transfert de données complet, confirmé ou non confirmé

3.4.14
adresse de région/segment
extension d'adresse qui identifie un sous-réseau de bus de terrain particulier

NOTE La prise en charge de l'acheminement DL entre bus de terrain est assurée.

3.4.15
données de demande
DLSDU fournie par l'utilisateur DLS distant à l'utilisateur DLS local

3.4.16
DLE distante
DLE à laquelle s'adresse une demande de service (c'est-à-dire la DLE de réception prévue de toute DLPDU d'envoi/demande résultante)

3.4.17
utilisateur DLS distant
utilisateur DLS auquel s'adresse la demande de service (c'est-à-dire le destinataire prévu de toute primitive d'indication résultante)

3.4.18
DLPDU de réponse
DLPDU transmise d'une DLE distante vers la DLE émettrice (locale) et éventuellement d'autres DLE

NOTE Lorsque la DLE distante est un éditeur, la DLPDU de réponse peut également être envoyée à plusieurs DLE distantes.

3.4.19
DLPDU de réponse
DLPDU de réponse transportant une DLSDU d'un utilisateur DLS distant à un utilisateur DLS local

3.4.20
données d'émission
DLSDU fournie par un utilisateur DLS local à un utilisateur DLS distant

3.4.21
DLPDU d'envoi/demande
DLPDU transportant soit une demande de données, soit une DLSDU ou les deux à la fois, d'un utilisateur DLS local à un utilisateur DLS distant

3.4.22
maître d'horloge
dispositif capable d'envoyer des DLPDU de synchronisation d'horloge

NOTE Les dispositifs de liaison incluent une fonctionnalité de maître d'horloge.

3.4.23
récepteur d'horloge
dispositif qui peut être synchronisé par un maître d'horloge

3.4.24
détenteur de jeton
station maître qui commande l'accès au bus

3.4.25
passage de jeton
méthode d'accès au support dans laquelle le droit d'émettre est transmis d'une station maître à une station maître dans un anneau logique

3.5 Symboles et abréviations communs

3.5.1 Unités de données

3.5.1.1	**DLPDU**	unité de données de protocole DL
3.5.1.2	**DLSDU**	unité de données de service DL
3.5.1.3	**PhIDU**	unité de données d'interface Ph
3.5.1.4	**PhPDU**	unité de données de protocole Ph

3.5.2 Divers

3.5.2.1	**DL-**	couche liaison de données (utilisée comme préfixe)
3.5.2.2	**DLCEP**	point d'extrémité de connexion DL
3.5.2.3	**DLE**	DLE (l'instance active locale de la couche liaison de données)
3.5.2.4	**DLL**	couche DL
3.5.2.5	**DLM**	gestion DL (utilisée comme préfixe)
3.5.2.6	**DLMS**	service de gestion DL
3.5.2.7	**DLS**	service DL

3.5.2.8	**DLSAP**	point d'accès de service DL
3.5.2.9	**FIFO**	premier entré, premier sorti (méthode de mise en file d'attente)
3.5.2.10	**LLC**	contrôle de liaison logique
3.5.2.11	**MAC**	contrôle d'accès au support physique
3.5.2.12	**OSI**	interconnexion de systèmes ouverts
3.5.2.13	**Ph**	couche physique (utilisée comme préfixe)
3.5.2.14	**PhE**	entité Ph (l'instance active locale de la couche physique)
3.5.2.15	**PhL**	couche physique
3.5.2.16	**PhS**	service Ph
3.5.2.17	**PhSAP**	point d'accès de service Ph
3.5.2.18	**QS**	qualité de service

3.6 Symboles et abréviations de Type 3

3.6.1 ACK	DLPDU d'acquittement
3.6.2 ASM	message de temps de réserve active
3.6.3 ASP	période de temps de réserve active
3.6.4 BUS ID (ID de bus)	identification d'un bus, une extension d'adresse (adresse de région/segment DL) qui identifie un bus particulier comme prenant en charge l'acheminement entre des segments DL
3.6.5 CRX	exécution de réception de caractères
3.6.6 CS	synchronisation d'horloge
3.6.7 CTX	exécution d'émission de caractères
3.6.8 DA	adresse de destination d'une DLPDU
3.6.9 DAE	extension(s) d'adresse de destination d'une DLPDU, qui achemine D_SAP_index et/ou un ID de bus de destination
3.6.10 D_SAP	point d'accès de service de destination, le DLSAP associé à l'utilisateur DLS distant
3.6.11 D_SAP_index	indice de point d'accès de service de destination – la composante d'une adresse DLSAP – qui désigne un DLSAP et l'utilisateur DLS distant au sein de la DLE distante
3.6.12 DXM	échange des données en multidiffusion
3.6.13 ED	délimiteur de fin d'une DLPDU
3.6.14 EOA	END-OF-ACTIVITY (fin d'activité)
3.6.15 EOD	END-OF-DATA (fin de données)
3.6.16 EODA	END-OF-DATA-AND-ACTIVITY (fin de données et d'activité)
3.6.17 EXT	bit d'extension d'adresse d'une DLPDU

3.6.18 FC champ de contrôle de trame (type de trame) d'une DLPDU

3.6.19 FCB bit de nombre de trames d'une DLPDU (champ FC) utilisé pour éliminer des DLPDU perdues ou dupliquées

3.6.20 FCV bit de validité d'un bit de nombre de trames d'une DLPDU, indique si le FCB doit être évalué

3.6.21 FCS séquence de contrôle de trame (synchrone) ou somme de contrôle de trame (asynchrone)

3.6.22 FLC contrôle de liaison de bus de terrain

3.6.23 G facteur de mise à jour du GAP (intervalle), le nombre de rotations de jeton entre cycles de maintenance d'intervalle (mise à jour)

3.6.24 GAPL liste GAP contenant l'état de toutes les stations dans le GAP de cette station

3.6.25 IsoM mode isochrone

3.6.26 Hd distance de Hamming, une mesure d'intégrité de DLPDU, le nombre minimal d'erreurs binaires qui peuvent donner lieu à l'acceptation d'une DLPDU dégradée

3.6.27 HSA adresse de station la plus élevée montée (configurée) sur ce bus de terrain

3.6.28 L longueur du champ informations, la partie d'une DLPDU vérifiée par la FCS

3.6.29 LE champ qui donne la longueur d'une DLPDU au-delà de la partie fixe

3.6.30 LEr champ qui répète la longueur pour améliorer l'intégrité de la DLPDU

3.6.31 LMS liste de stations maîtres

3.6.32 LR ressource locale non disponible ou insuffisante (état DL/DLM de la primitive de service)

3.6.33 LS service local non activé au point d'accès de service DL ou DLSAP local non activé (état DL/DLM de la primitive de service)

3.6.34 lsb bit de poids faible d'un champ ou d'un octet

3.6.35 max la fonction maximum en arithmétique

3.6.36 MCT multidiffusion

3.6.37 MP périodes de nouvelles tentatives de transfert de messages

3.6.38 mr nombre de nouvelles tentatives

3.6.39 msb bit de poids fort d'un champ ou d'un octet

3.6.40 MSRD DLS: Emettre et demander données avec réponse multidestinataire

3.6.41 mt nombre de nouvelles tentatives par rotation de jeton

3.6.42 n nombre de stations

3.6.43 NA pas d'acquittement/réponse (état DL/DLM de la primitive de service)

3.6.44 na nombre de stations actives

3.6.45 NIL (NULLE) valeur déterminée localement

3.6.46 NON non Ok (état DL/DLM de la primitive de service)

3.6.47 np nombre de stations passives

3.6.48 NR pas de réponse, acquittement négatif des données DL et envoi de données Ok (état DL de la primitive de service)

3.6.49 NRZ non-retour à zéro (PhL), technique de codage où les transitions n'ont lieu que lorsque des bits de données successifs ont des valeurs différentes

3.6.50 NS station suivante, la station à laquelle ce maître passera le jeton

3.6.51 OK service achevé conformément aux règles (état DL/DLM de la primitive de service)

3.6.52 PhICI informations de commande d'interface PhL [ISO/CEI 7498-1]

3.6.53 PhPCI Informations de commande de protocole PhL [ISO/CEI 7498-1]

3.6.54 PON transition à la mise sous tension à une station donnée

3.6.55 PS station précédente, la station qui passe le jeton à cette station maître

3.6.56 PSP période de temps de réserve passive

3.6.57 RDH état haut de données de réponse DL et pas de ressource pour l'envoi de données (état DL/DLM de la primitive de service)

3.6.58 RDL état bas de données de réponse DL/DLM et pas de ressource pour l'envoi de données (état DL/DLM de la primitive de service)

3.6.59 Res réservé

3.6.60 RET nouvelle tentative

3.6.61 RR pas de ressource pour l'envoi de données et pas de données DL de réponse disponibles (acquittement négatif) (état DL/DLM de la primitive de service)

3.6.62 RS pas de service, ou aucun service activé au DLSAP distant (acquittement négatif) (état DL/DLM de la primitive de service)

3.6.63 RSYS fréquence de messages système (en périodes de transfert de message par seconde) des échanges de messages DL confirmés

3.6.64 SA adresse d'origine d'une DLPDU

3.6.65 SAE extension(s) d'adresse d'origine d'une DLPDU, acheminant S_SAP_index et/ou l'ID de bus d'origine

3.6.66 SC DLPDU d'acquittement à un seul caractère

3.6.67 SD1 à SD4 délimiteurs de début d'émission de DLPDU asynchrone

3.6.68 SDA émettre données avec acquittement (service DL)

3.6.69 SDA_H/L état haut/bas d'émission de données avec acquittement (fonction de DLPDU)

3.6.70 SDX délimiteur de début d'une émission de DLPDU asynchrone ou synchrone

3.6.71 SDL1 à SDL5 délimiteurs de début d'émission de DLPDU synchrone

3.6.72 SDN émission de données sans acquittement (service DL)

3.6.73 SDN_H/L état haut/bas d'émission de données sans acquittement (fonction de DLPDU)

3.6.74 SM Diagramme d'états

3.6.75 SOA START-OF-ACTIVITY (démarrage d'activité)

3.6.76 SRC commande d'émission/réception

3.6.77 SRD émettre et demander des données avec réponse (service DL)

3.6.78 SRD_H/L état haut/bas d'émission et de demande de données avec réponse (fonction de DLPDU)

3.6.79 SRU unité d'émission/réception

3.6.80 S_SAP point d'accès de service d'origine, le DLSAP associé à l'utilisateur DLS local initiateur

3.6.81 S_SAP_index indice de point d'accès de service d'origine – une composante d'adresse DLSAP – au sein de la DLE d'où la transaction est lancée

3.6.82 Stn station, un dispositif mettant en œuvre une DLE avec une adresse DL de bus de terrain

3.6.83 SYN bits de synchronisation d'une DLPDU (période de repos), qui garantit l'intégrité de la DLPDU spécifiée et permet la synchronisation du récepteur

3.6.84 TA/R temps nécessaire à l'émission d'une DLPDU d'acquittement/réponse

3.6.85 TASM temps de message ASM

3.6.86 t_{BIT} temps binaire

3.6.87 TCSI temps d'intervalle de synchronisation d'horloge

3.6.88 T_{CT} durée de cycle isochrone

3.6.89 TGUD temps de mise à jour GAP (intervalle)

3.6.90 T_{ID} temps au repos

3.6.91 TIM diagramme d'états de temporisateurs

3.6.92 T_{MP} période de transfert de message

3.6.93 TP période de transfert de jeton

3.6.94 TPSP temps de réserve passive

3.6.95 TPTG période de gap post-émission

3.6.96 TQUI temps de silence

3.6.97 TRCT durée de cycle isochrone réelle

3.6.98 T_{RD} temps de retard de réception

3.6.99 TRDY temps à l'état prêt

3.6.100 TRES temps de réserve

3.6.101 T_{RR} temps réel de rotation

3.6.102 TS cette station

3.6.103 TS/R temps de DLPDU d'envoi/demande

3.6.104 T_{SD} temps de retard d'émission

3.6.105 TSDI retard station de l'initiateur

3.6.106 TSDR retard station du répondeur

3.6.107 TSET temps de montée

3.6.108 T_{SH} décalage temporel

3.6.109 T_{SL} durée de créneau

3.6.110 T_{SM} temps de marge de sécurité

3.6.111 T_{SR} temps de réaction système

3.6.112 TSYN temps de synchronisation

3.6.113 TSYNI temps d'intervalle de synchronisation

3.6.114 T_{TC} durée de cycle de jeton

3.6.115 T_{TD} temps de retard d'émission

3.6.116 T_{TF} temps de DLPDU de jeton

3.6.117 T_{TH} temps de conservation de jeton

3.6.118 T_{TO} temps de dépassement de délai; durée du temps imparti

3.6.119 T_{TP} période de transfert de jeton

3.6.120 T_{TR} temps de rotation cible

3.6.121 UART émetteur-récepteur asynchrone universel

3.6.122 UC caractère UART

3.6.123 UE acquittement négatif, erreur d'interface d'utilisateur distant (état DL/DLM de la primitive de service)

4 Éléments communs de protocole DL

4.1 Séquence de contrôle de trame

4.1.1 Généralités

Toute référence au bit *K* de l'octet est une référence au bit dont le poids, dans un entier non signé d'un octet, est de 2^K.

NOTE 1 Ceci est parfois appelé numérotation binaire "petit-boutiste".

Comme dans d'autres normes internationales (voir Note 2), la détection d'erreurs au niveau DLPDU est assurée en calculant et en annexant aux autres champs de la DLPDU une séquence de contrôle de trame (FCS) sur plusieurs bits, au cours de l'émission, de manière à constituer un "mot de code systématique"[1)] d'une longueur ***n*** constituée de ***k*** bits de message de DLPDU suivis par ***n – k*** bits redondants et en calculant au cours de la réception que le message et le FCS concaténés constituent un mode de code licite (*n*,*k*). Le mécanisme correspondant à cette vérification est le suivant:

NOTE 2 Pour exemple, voir l'ISO/CEI 3309, l'ISO/CEI 8802 et l'ISO/CEI 9314-2.

La forme générique du polynôme générateur pour cette construction de FCS est spécifiée dans l'équation (4) et le polynôme de la partie résiduelle prévue dans le récepteur est spécifié dans l'équation (9). Les polynômes spécifiques pour le protocole DL de Type 3 sont spécifiés dans le Tableau 1. L'exemple d'application est présenté à l'Annexe B.

Tableau 1 – Longueur, polynômes et constantes de FCS pour une transmission synchrone de Type 3

Point	Valeur
n-k	16
G(x)	$X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$ (voir Notes 1, 2, 3)
R(x)	$X^{15} + X^{14} + X^{13} + X^9 + X^8 + X^7 + X^4 + X^2$ (voir Note 4)

NOTE 1 Les mots de code D(X) construits à partir de ce polynôme G(X) ont une distance de Hamming de 4 pour des longueurs ≤ 344 octets et une distance de Hamming de 5 pour des longueurs ≤ 15 octets.

NOTE 2 Ce polynôme G(X) est primordial par rapport à tous les autres et par conséquent, il ne peut être compromis par aucun des polynômes communément utilisés dans des DCE (modems): le polynôme de codage différentiel $1 + X^{-1}$ ainsi que tous les polynômes de brouillage de primitives de forme $1 + X^{-j} + X^{-k}$.

NOTE 3 Le polynôme G(X) est le polynôme 16 bits optimal pour la détection d'erreurs en paquets sur des DLPDU de 300 octets ou moins lorsque la distribution statistique du paquet d'erreurs suit une loi de Poisson (comme c'est en général le cas).

NOTE 4 Il convient que le reste R(x) soit 1110 0011 1001 0100 (X^{15} à X^0, respectivement) en l'absence d'erreurs.

4.1.2 Au niveau de la DLE d'émission

Le message initial (c'est-à-dire la DLPDU sans FCS), la FCS et le mot de code de message composé (la FCS et la DLPDU concaténées) doivent être considérés comme des vecteurs M(X), F(X) et D(X), de dimension *k*, *n – k* et *n*, respectivement, dans un champ d'extension sur Champ Galois Base (2). Si les bits du message sont m_1 à m_k et si les bits FCS sont f_{n-k-1} à f_0, où

m_1 à m_8 constituent le premier octet envoyé,

m_{8N-7} à m_{8N} constituent le *Nème* octet envoyé,

f_7 à f_0 constituent le dernier octet envoyé, et

1) W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes* (2nd edition), MIT Press, Cambridge, 1972.

m_1 est envoyé par le(s) premier(s) symbole(s) PhL du message et f_0 est envoyé par le(s) dernier(s) symbole(s) PhL du message (sans compter les informations de structure de trame PhL),

NOTE 1 Cet ordre “conforme à l'émission” est critique pour les propriétés de détection d'erreurs de la FCS.

dans ce cas, le vecteur du message M(X) doit être considéré comme étant

$$M(X) = m_1X^{k-1} + m_2X^{k-2} + \ldots + m_{k-1}X^1 + m_k \quad (1)$$

et le vecteur FCS F(X) doit être considéré comme étant

$$F(X) = f_{n-k-1}X^{n-k-1} + \ldots + f_0 \quad (2)$$
$$= f_{15}X^{15} + \ldots + f_0$$

Le vecteur composé D(X), pour la DLPDU complète, doit être construit comme étant la concaténation des vecteurs message et FCS

$$D(X) = M(X)\,X^{n-k} + F(X) \quad (3)$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \ldots + f_0$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{16} + f_{15}X^{15} + \ldots + f_0$$

La DLPDU présentée à la PhL doit être constituée d'une séquence d'octets dans l'ordre spécifié.

Les bits de contrôle redondant f_{n-k-1} à f_0 de la FCS doivent être les coefficients du reste F(X), après division par G(X), de $L(X)\,(X^k + 1) + M(X)\,X^{n-k}$

où G(X) est le polynôme générateur de degré *n-k* pour les mots de code

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \ldots + 1 \quad (4)$$
$$= X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^3 + X^2 + X + 1$$

et L(X) est le polynôme de poids maximal ("tous à 1") de degré *n-k-1*

$$L(X) = \frac{X^{n-k} + 1}{X + 1} = X^{n-k-1} + X^{n-k-2} + \ldots + X + 1 \quad (5)$$
$$= X^{15} + X^{14} + X^{13} + X^{12} + \ldots + X^2 + X + 1$$

En d'autres termes,

$$F(X) = L(X)\,(X^k + 1) + M(X)\,X^{n-k} \ \text{(modulo } G(X)) \quad (6)$$

NOTE 2 Les termes L(X) sont inclus dans le calcul afin de détecter la troncation ou l'extension du message initial ou terminal en ajoutant à la FCS un facteur en fonction de la longueur.

NOTE 3 Dans une application type, lorsque n-k = 16, le reste initial de la division est préréglé sur "tous à 1". Le train binaire du message transmis est multiplié par X^{n-k} et divisé (modulo 2) par le polynôme générateur G(X), spécifié dans l'équation (4). Le complément à "1" du reste résultant est transmis comme la FCS à (n-k)-bit, le coefficient de X^{n-k-1} étant transmis en premier.

4.1.3 Au niveau de la DLE de réception

La séquence d'octets indiquée par la PhE doit être concaténée dans la DLPDU reçue et dans la FCS, et être considérée comme un vecteur V(X) de dimension ***u***

$$V(X) = v_1X^{u-1} + v_2X^{u-2} + \ldots + v_{u-1}X + v_u \quad (7)$$

NOTE 1 Du fait des erreurs, **u** peut être différent de **n**, la dimension du vecteur de code transmis.

Un reste R(X) doit être calculé pour V(X), la DLPDU reçue et la FCS reçue, par une méthode similaire à celle utilisée par la DLE d'émission (voir 4.1.2) dans le calcul de F(X).

$$R(X) = L(X)\,X^u + V(X)\,X^{n-k} \ \text{(modulo } G(X)) \quad (8)$$
$$= r_{n-k-1}X^{n-k-1} + \ldots + r_0$$

Définir E(X) comme étant le vecteur de code d'erreur des différences additives (modulo 2) entre le vecteur de code émis D(X) et le vecteur reçu V(X) résultant des erreurs rencontrées (dans le fournisseur de PhS et dans les ponts) entre DLE d'émission et de réception.

$$E(X) = D(X) + V(X) \quad (9)$$

Si aucune erreur n'est rencontrée, de sorte que E(X) = 0, R(X) sera égal à un polynôme à reste constant différent de zéro

$$R_{ok}(X) = L(X)\, X^{n-k} \text{ (modulo } G(X)) \quad (10)$$

dont la valeur est indépendante de D(X). Malheureusement, R(X) sera aussi égal à $R_{ok}(X)$ dans le cas où E(X) est un multiple exact différent de zéro de G(X), auquel cas il y a des erreurs "indétectables". Dans tous les autres cas, R(X) ne sera pas égal à $R_{ok}(X)$; ces DLPDU sont erronées et doivent être ignorées sans autre forme d'analyse.

NOTE 2 Dans une application type, le reste initial de la division est préréglé sur "tous à 1". Le train binaire reçu est multiplié par X^{n-k} et divisé (modulo 2) par le polynôme générateur G(X), spécifié dans l'équation (4).

5 Aperçu général du protocole DL

NOTE L'Annexe A décrit un certain nombre de diagrammes d'états finis utilisés par la DLE pour fournir ses fonctions de protocole de niveau bas et de niveau haut. La spécification de l' Annexe A complète la spécification textuelle du présent exposé et des Articles correspondants dans le corps de la présente norme. En cas de divergence, les exigences de l' Annexe A prévalent.

5.1 Généralités

A partir des exigences et des divers domaines d'application, tels que par exemple le contrôle de processus, l'automatisation d'usine, la distribution d'énergie, l'automatisation des bâtiments ou les industries de transformation primaires, etc., on obtient les fonctions caractéristiques suivantes du protocole de liaison de données de bus de terrain (voir le Tableau 2).

Tableau 2 – Fonctionnalités caractéristiques du protocole de liaison de données de bus de terrain

Fonctionnalité	Description
Types de stations	Maîtres (stations actives, avec contrôle d'accès au bus); esclaves (stations passives, sans contrôle d'accès au bus); de préférence 32 maîtres au plus, jusqu'à 127 optionnellement, si les applications ne sont pas à priorité stricte
Adressage des stations	0 à 127 (127 = adresses globales de messages de diffusion et de multidiffusion), extension d'adresse pour adresse de région/segment et adresse d'accès de service (DLSAP), de 6 bits chacune
Accès au bus	Hybride: décentralisé et centralisé; "Passage de jeton" entre stations maîtres et "Maître-Esclave" entre stations maîtres et esclaves
Services de transfert de données	Emission de données avec/sans acquittement Emettre et demander données avec réponse Emettre et demander données avec réponse multidestinataire Synchronisation d'horloge
Longueur de DLPDU	255 octets max. par DLPDU, 0 à 246 octets pour chaque unité de données sans extension d'adresse
Vitesses de transmission	En fonction de la topologie du réseau et des longueurs de câbles, par exemple, par paliers de 9,6 à 12 000 kbits/s
Caractéristique de transmission	Transmission en semi duplex, asynchrone et synchrone
Transmission asynchrone d'intégrité des données	Messages avec une distance de Hamming (Hd) = 4, détection de décalage de synchronisation, séquence spéciale pour éviter la perte ou la duplication des données
Transmission synchrone d'intégrité des données	Distance de Hamming (Hd) = 4 pour des longueurs de DLPDU inférieures à 255 octets et Hd = 5 pour des longueurs inférieures à 15 octets, séquence spéciale pour éviter la perte ou la duplication des données

NOTE 1 Le format de DLPDU pour la transmission asynchrone est le format FT 1.2 (transmission asynchrone avec synchronisation du démarrage et de l'arrêt) pour des matériels et systèmes de téléconduite, comme spécifié dans la CEI 60870-5-1.

NOTE 2 Le format de DLPDU pour la transmission synchrone repose sur des caractères représentés par des octets.

5.2 Aperçu du contrôle d'accès au support physique (MAC : Medium Access Control) et du protocole de transmission

La couche liaison de données du bus de terrain utilise un accès commandé au support physique, réalisé par une méthode d'accès hybride: une méthode décentralisée, selon le principe du **passage de jeton**, reposant sur une méthode centralisée selon le principe **maître-esclave**. Chaque station maître (station active) peut exercer le contrôle d'accès au support physique. Le passage de jeton est caractérisé par les fonctionnalités suivantes:

a) Le passage de jeton permet un accès équitable au support pour tous les détenteurs de jeton.

EXEMPLE: Lorsque quatre détenteurs de jetons produisent la même quantité de données de priorité similaire, ils partagent les supports de façon à ce qu'en moyenne, chacun d'entre eux puisse utiliser 25% du temps disponible pour le transfert de messages. La procédure de passage de jeton comporte des règles permettant de partager le temps de transfert de messages entre détenteurs de jetons sans aucune discrimination. S'il est fait usage de l'ensemble du temps disponible de conservation de jeton par un détenteur donné au cours d'une rotation de jeton, ce détenteur de jeton est limité à un message prioritaire au cours de la rotation de jeton suivante et il doit ensuite transférer immédiatement le jeton.

b) Le passage de jeton garantit des temps de réaction courts.

EXEMPLE: Pour des messages urgents, il est spécifié un délai maximal de remise des informations. Ainsi, une durée configurable (T_{TR}) spécifie le temps de rotation cible. Quel que soit le nombre des détenteurs de jetons et la quantité de messages à envoyer, au moins un message urgent peut être envoyé par chaque détenteur de jeton. Il est donc garanti un temps de réaction court pour message urgent en provenance de chaque station détentrice de jeton au cours de chacune des périodes de détention de jeton.

c) Le passage de jeton permet une (re-)configuration souple.

EXEMPLE: A la mise sous tension ou hors-tension d'un détenteur de jeton, il sera automatiquement inclus ou exclu de l'anneau logique. Au cours de la rotation de jeton suivante, après détection et inclusion du nouveau détenteur de jeton dans l'anneau à jeton logique, il dispose des mêmes droits d'émission de messages que les autres détenteurs de jetons. Le partage du temps de transfert de message du bus est automatiquement organisé sans modification des paramètres des autres détenteurs de jetons.

Le contrôle d'accès au support physique peut être exercé par chaque station maître (station active) si cette station détient le jeton. Le jeton est passé d'une station maître à une autre station maître dans un anneau logique et ainsi, il détermine l'instant où une station maître peut accéder au support. Ce passage de jeton contrôlé est géré par chaque station qui connaît son prédécesseur (station précédente: PS), c'est-à-dire la station dont il a reçu le jeton. Par ailleurs, chaque station connaît son successeur (station suivante: NS), c'est-à-dire la station à laquelle le jeton est transmis ainsi que son adresse propre (cette station: TS). Chaque station maître détermine les adresses PS et NS après la première initialisation des paramètres de fonctionnement puis, par la suite, de manière dynamique, selon l'algorithme décrit en 5.3.2.4.

Le principe maître-esclave présente les caractéristiques suivantes:

La communication est toujours lancée par une station maître qui a la permission d'accès au support, à savoir le jeton. Les stations esclaves (stations passives) agissent en toute neutralité pour ce qui concerne l'accès au support, ce qui signifie qu'elles ne transmettent pas de manière indépendante mais uniquement sur demande. Si l'anneau logique est constitué d'un seul maître et de plusieurs stations esclaves, il s'agit d'un système maître-esclave pur.

Le système traite également des conditions d'erreurs, des exceptions et des états opérationnels suivants:

— des jetons multiples,
— un jeton perdu,
— une erreur de passage de jeton,
— des adresses de station en double,
— des stations ayant un émetteur ou un récepteur défectueux,
— l'ajout et le retrait de stations en cours de fonctionnement,
— toute combinaison de stations maîtres et esclaves.

5.3 Modes de transmission et entité DL

5.3.1 Aperçu général

L'échange des messages est effectué avec ou sans confirmation. Un échange de message confirmé est constitué d'une DLPDU d'envoi/demande de la station maître et de la DLPDU d'acquittement ou de réponse correspondante d'un maître ou d'une station esclave. Des données utilisateur peuvent être transmises dans la DLPDU d'envoi/demande ainsi que dans la DLPDU de réponse. La DLPDU d'acquittement ne contient pas de données utilisateur (pour les formats de DLPDU, voir l'article 7).

Un échange de message non confirmé a lieu en cas de transmission du jeton et en cas de transmission des données sans acquittement (nécessaire par exemple pour des messages de diffusion). Dans les deux modes de fonctionnement, il n'y a pas d'acquittement. Dans le cas de messages de diffusion, une station maître (initiatrice) s'adresse à toutes les autres stations simultanément au moyen d'une adresse globale (adresse de station la plus élevée, tous les bits d'adresse étant des valeurs binaires "1").

Toutes les stations, à l'exception du détenteur de jeton (initiateur) concerné, doivent en général surveiller toutes les demandes. Les stations renvoient un acquittement ou une réponse uniquement si elles sont concernées. L'acquittement ou la réponse doit arriver au cours d'une durée prédéfinie, appelée durée de créneau; dans le cas contraire, l'initiateur réitère la demande s'il ne s'agit pas d'une "première demande" (voir 6.4, FCB). Une nouvelle

demande ou un jeton ne doivent être émis par l'initiateur qu'après expiration d'une période d'attente et après réception d'un acquittement ou d'une réponse, cette période étant appelée temps au repos (voir 5.5).

Si le répondeur ne réagit pas en envoyant un acquittement de réponse après un nombre prédéfini de nouvelles tentatives (voir le Tableau 5), il est marqué comme "non-opérationnel". Si un répondeur est "non-opérationnel", toute demande ultérieure aboutissant à un échec ne sera pas répétée.

Les modes de fonctionnement en émission définissent la séquence des périodes de transfert des messages. On distingue trois modes:

1) Traitement des jetons.

2) Opération d'envoi.

3) Opération d'envoi et de demande.

5.3.2 Procédures relatives au jeton

5.3.2.1 Circulation du jeton

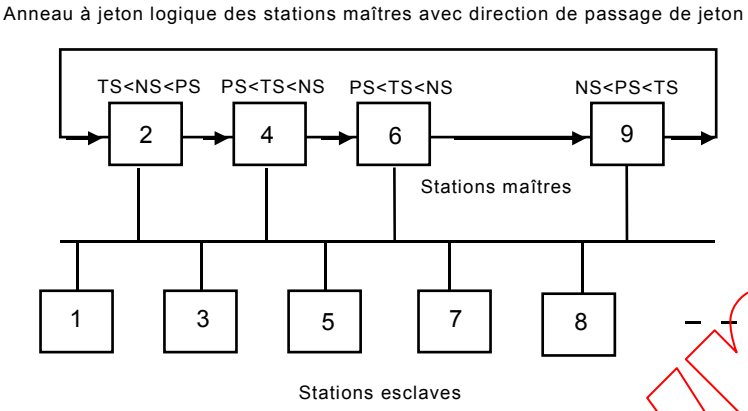
Le jeton est passé d'une station maître à une autre station maître dans l'ordre numérique ascendant des adresses de station, en utilisant la DLPDU de jeton (voir 7.4). Pour fermer l'anneau à jeton logique, la station ayant l'adresse la plus élevée passe le jeton à la station ayant l'adresse la plus basse; voir la Figure 2.

5.3.2.2 Réception du jeton

Si une station maître (TS) reçoit une DLPDU de jeton qui lui est adressée à partir d'une station enregistrée comme étant une station précédente (PS) dans sa liste de stations maîtres (LMS), elle **possède le jeton** et peut alors lancer des périodes de transfert de messages. La LMS est générée dans une station maître qui est à l'état "Listen_Token" (Ecouter jeton) (voir 8.2.4) après mise sous tension; si nécessaire elle est ensuite mise à jour et corrigée à chaque réception d'une DLPDU de jeton.

Si l'émetteur du jeton n'est pas la PS enregistrée, le destinataire considère qu'il y a une erreur et ne doit pas accepter le jeton. C'est seulement suite à une nouvelle tentative de la même PS que le jeton est reçu et accepté car, dans ce cas, le récepteur du jeton doit à présent considérer qu'il y a eu modification de l'anneau logique. Il remplace, dans la LMS, la PS initialement enregistrée par la nouvelle PS.

La Figure 2 illustre l'anneau logique de passage de jeton.



où

TS est cette station (adresse);

PS est la station précédente (adresse);

NS est la station suivante (adresse);

Les stations maîtres sont 2, 4, 6 et 9 (adresses prises pour exemple);

Les stations esclaves sont 13, 5, 7 et 8 (adresses prises pour exemple).

Figure 2 – Anneau logique de passage de jeton

5.3.2.3 Transmission du jeton

Une fois que la station maître est parvenue à la fin de ses périodes de transfert de messages – y compris la maintenance conditionnelle de la liste de station GAP (GAPL, voir 5.3.2.4) elle passe le jeton à son successeur (NS) en émettant la DLPDU de jeton. La fonctionnalité de cet émetteur-récepteur est vérifiée par une surveillance simultanée (voir 8.2.11, état "Pass_Token" ("Passer Jeton").

Si, après émission de la DLPDU de jeton et après expiration du temps de synchronisation au sein de la durée de créneau (voir 5.5), l'émetteur de jeton reçoit une DLPDU valide, c'est-à-dire un en-tête de DLPDU sans aucune erreur, il considère que sa NS détient le jeton et lance des périodes de transfert de messages. Si l'émetteur de jeton reçoit une DLPDU non valide, il considère qu'une autre station maître est en cours d'émission. Dans les deux cas, il cesse de surveiller le passage de jeton et se retire, ce qui signifie qu'il passe à l'état "Active_Idle" ("Actif au Repos") (voir 8.2.5).

Si l'émetteur de jeton ne reconnaît aucune activité de bus au cours de la durée de créneau, il répète la DLPDU de jeton et attend pendant une autre durée de créneau. Il se retire ensuite s'il reconnaît une activité de bus pendant la seconde durée de créneau. Dans le cas contraire, il répète une dernière fois la DLPDU de jeton destinée à sa NS. Si, après cette seconde tentative, il reconnaît une activité de bus au cours de la durée de créneau, il se retire.

Si, après la seconde tentative il n'y a aucune activité de bus, l'émetteur de jeton tente de passer le jeton à l'avant-dernière station maître (NS de sa NS). Il continue à répéter cette procédure jusqu'à ce qu'il trouve un successeur dans sa LMS. En cas d'échec, l'émetteur de jeton considère qu'il est seul sur l'anneau à jeton logique et transmet le jeton à lui-même. Si, dans un enregistrement ultérieur de station, il retrouve une NS, il tente de nouveau de passer le jeton.

5.3.2.4 Ajout et retrait de stations

Des stations maîtres et esclaves peuvent être connectées ou déconnectées du support de transmission à tout moment. Chaque station maître dans l'anneau à jeton logique est

responsable de l'ajout de nouvelles stations dans la plage de sa propre adresse de station (TS) à la station suivante (NS), à l'exclusion de TS et NS. Cette plage d'adresses est appelée GAP (intervalle) et représentée dans la liste GAP (GAPL), à l'exception de la plage d'adresses comprise entre l'adresse de station la plus élevée (HSA: 2 à 126) et 127, qui n'appartient pas au GAP.

Chaque station maître dans l'anneau à jeton logique examine sa plage d'adresses (toutes les adresses du GAP) après expiration du temps de mise à jour du GAP (voir 5.5.3.11) pour déceler les éventuelles modifications concernant les stations maîtres et esclaves. Ceci est réalisé en examinant **une** adresse par réception de jeton, en utilisant la DLPDU de demande "Request DL Status with Reply" ("Demande d'état DL avec réponse") (voir Tableau 3, b7=1, Code-No 9: Format 7.1.1 a) et 7.1.2 a)).

Après réception du jeton, la maintenance de GAP commence immédiatement, dès achèvement des périodes de transfert de messages en file d'attente, s'il y a encore du temps de transmission disponible (voir 5.3.2.6). Dans le cas contraire, la maintenance du GAP démarre dès réception du jeton suivant ou consécutif, après les périodes de transfert de messages prioritaires et d'autres services non prioritaires invoqués avant que ne soit effectuée la maintenance du GAP précédente (voir 5.3.2.7). Dans les réalisations pratiques, il est important de s'assurer que la maintenance du GAP et les périodes de transfert de messages non prioritaires ne se bloquent pas l'une l'autre.

Les adresses de GAP sont examinées par ordre ascendant, à l'exception du GAP qui prime sur la HSA. Par exemple, si la HSA et l'adresse 0 ne sont pas utilisées par une station maître, la station maître qui a l'adresse la plus élevée examine l'adresse 0 après vérification de la HSA.

Si une station renvoie un acquittement positif avec l'état DL "Master station not ready to enter logical token ring" ("Station maître non prête à entrer dans l'anneau à jeton logique") ou "Slave station" ("Station esclave") (voir le Tableau 3, b7=0, Code-No 0, pas de SC, et voir la Figure 20), elle est marquée en conséquence dans la liste GAPL et l'adresse suivante est vérifiée. Si une station répond avec l'état "Master station ready to enter logical token ring" ("Station maître prête à entrer dans l'anneau à jeton logique"), le détenteur de jeton modifie ses listes GAPL et LMS et passe le jeton à la nouvelle NS. Cette station, nouvellement admise dans l'anneau à jeton logique, a déjà constitué sa propre LMS, lorsqu'elle était en l'état "Listen_Token" ("Ecouter jeton"), de sorte qu'elle est capable de déterminer sa plage GAP et sa NS.

Si une station répond avec l'état DL "Master station in logical token ring" ("Station maître dans l'anneau à jeton logique"), le détenteur de jeton ne modifie pas sa GAP et passe le jeton à la NS indiquée dans la LMS. Ainsi, la station maître ignorée peut se retirer du bus dans le cas où il y a non réception permanente du jeton. Dans une telle situation, la station maître doit passer à l'état "Listen_Token". Lorsqu'elle est dans cet état, elle génère une nouvelle LMS et reste dans cet état jusqu'à ce qu'il lui soit de nouveau adressé une "Request DL Status with Reply" ("Demande d'état DL avec réponse") transmise par la station qui la précède (PS).

Les stations qui étaient enregistrées dans la GAPL et qui ne répondent pas à une "Request DL Status with Reply" sont retirées de la GAPL et sont enregistrées comme étant des adresses de stations non utilisées.

Pour les questions de performance, les réitérations d'états "Request DL Status with Reply" ne sont pas souhaitables.

5.3.2.5 (Ré)initialisation de l'anneau à jeton logique

L'initialisation est principalement un cas particulier de mise à jour des listes LMS et GAPL. Si après mise sous tension (PON) d'une station maître, il est rencontré une expiration de délai à l'état "Listen_Token" (aucune activité de bus au cours du délai T_{TO} (voir 5.5)), la station

maître doit revendiquer le jeton (état "Claim_Token" ("Revendiquer jeton")) et lancer l'initialisation.

Lorsque l'ensemble du système de bus de terrain a démarré, la station maître ayant l'adresse de station la plus basse lance l'initialisation. En émettant deux DLPDU de jeton adressées à elles-mêmes (DA = SA = TS) elle informe toute autre station maître (en saisissant une NS dans leur LMS) qu'elle est à présent la seule station dans l'anneau à jeton logique. Ensuite, elle transmet une DLPDU "Request DL Status with Reply" à chaque station dans une séquence d'adresses incrémentielle, afin d'enregistrer les autres stations. Si une station répond avec "Master station not ready to enter logical token ring" ou avec "Slave station" elle est saisie dans la GAPL. La première station maître qui répond avec "Master station ready to enter logical token ring" est enregistrée comme NS dans la liste LMS et ferme ainsi la plage GAP du détenteur de jeton. Le détenteur de jeton passe ensuite le jeton à sa NS.

Une réinitialisation devient nécessaire après perte du jeton, ce qui entraîne une expiration de délai (pas d'activité de bus). Dans ce cas, une séquence d'initialisation de l'ensemble du bus n'est pas nécessaire car les listes LMS et GAPL existent déjà dans les stations maîtres. L'expiration du délai touche en premier lieu la station maître ayant l'adresse la plus basse. Celle-ci prend le jeton et lance des périodes normales de transfert de messages ou passe le jeton à sa NS.

5.3.2.6 Temps de rotation de jeton

Après réception d'un jeton, l'entité DL peut lancer des périodes de transfert de messages prioritaires et non prioritaires en fonction du temps de rotation du jeton. Le temps de rotation de jeton est mesuré au moyen du temporisateur de rotation du jeton. Au départ, le temporisateur de rotation du jeton est chargé avec le temps de rotation cible (T_{TR}) et il est décrémenté à chaque temps binaire. Le temporisateur de rotation du jeton s'arrête lorsque la valeur zéro est atteinte.

Le temps de rotation de jeton doit être mesuré conformément aux règles suivantes:

— immédiatement après réception du jeton, la station maître doit lire la valeur courante du temporisateur de rotation du jeton (temps de conservation de jeton: T_{TH}) pour calculer le temps réel de rotation T_{RR} (différence entre le temps de rotation cible et la valeur courante du temporisateur de rotation du jeton) et doit ensuite lancer le temporisateur de rotation du jeton avec la valeur temps de rotation cible (T_{TR})

— le temps T_{RR} représente toujours le temps de rotation de jeton de la dernière rotation de jeton (du point de vue de cette station)

Un temps de rotation minimal du système est fonction du nombre de stations maîtres et ainsi de la période de transfert de jeton (T_{TP}) et de la durée des périodes de transfert de messages prioritaires (T_{MP} haut). Le temps de rotation cible T_{TR} prédéfini doit également être suffisant pour assurer des périodes de transfert de messages non prioritaires ainsi qu'une marge de sécurité pour les éventuelles nouvelles tentatives.

Pour rester au sein du temps de réaction système requis par le champ d'application, le temps de rotation cible T_{TR} du jeton dans l'anneau logique doit être spécifié.

Le temps de réaction système est défini comme étant l'intervalle de temps maximal (dans le cas le plus défavorable) entre deux périodes consécutives de transfert de messages prioritaires d'une station maître, mesuré à l'interface de l'utilisateur DLS à la charge maximale du bus.

Pour obtenir un temps de rotation cible le plus court possible, il est recommandé que l'utilisateur DLS (voir la CEI 61158-5-3) déclare uniquement les données importantes comme étant des périodes de transfert de messages prioritaires et qu'il restreigne leur longueur (par exemple, ≤ 32 octets pour la DLSDU, voir 6.6).

Si les périodes de transfert définies en 5.6 (équations (52) et (59) ainsi que (53) et (60)) sont incluses et que de nouvelles tentatives sont prises en compte, le paramètre de fonctionnement "temps de rotation cible T_{TR}" (voir 5.6 et le Tableau 5), nécessaire à l'initialisation (T_{TR} min), est calculé de la manière suivante pour l'utilisateur DLS:

$$\min T_{TR} = na \times T_{TP} + (na + 1) \times \text{high } T_{MP} + k \times \text{low } T_{MP} + mt \times T_{RMP} \quad (11)$$

où

na est le nombre de stations maîtres;

k est le nombre estimé de périodes de transfert de messages non prioritaires par rotation de jeton;

T_{TP} est la période de transfert de jeton (voir 5.6.1.1 et 5.6.2.1);

T_{MP} est la période de transfert de messages, en fonction de la longueur des DLPDU (voir 5.6.1.2 et 5.6.2.2);

mt est le nombre de périodes de nouvelles tentatives de transfert de messages par rotation de jeton;

T_{RMP} est la période de nouvelle tentative de transfert de messages.

Le premier terme contient une période de transfert de jeton par station maître. Le deuxième terme contient **une** période de transfert de messages prioritaires pour *N+1* stations maîtres. Le troisième terme contient le nombre estimé de périodes de transferts de messages non prioritaires par rotation de jeton. Le quatrième terme est utilisé comme marge de sécurité pour les éventuelles nouvelles tentatives. Le temps de réaction maximal pour des périodes de transfert de messages prioritaires est garanti à toutes les charges de bus.

5.3.2.7 Priorités des messages

Dans le paramètre service_class (Classe de Service) des services DL, l'utilisateur DLS (Couche Application) a le choix entre deux priorités: basse et haute. La priorité est passée à la DLE avec la demande de service.

Après réception du jeton, chaque station maître peut toujours lancer **une** période de transfert de messages prioritaires, y compris les nouvelles tentatives en cas d'erreur, indépendamment du temps de conservation de jeton T_{TH}.

En outre, des périodes de transfert de messages prioritaires ou non prioritaires peuvent être lancées en fonction des règles suivantes si T_{TH} est toujours disponible (voir 5.5.5).

— Les messages prioritaires ou non prioritaires peuvent être envoyés si le temps réel de rotation calculé (T_{RR}) est inférieur à la valeur courante du temporisateur de rotation du jeton avant le moment de l'exécution de l'émission du message.

— Lorsqu'elle est lancée, une période de transfert de messages prioritaires ou non prioritaires est toujours achevée, y compris la ou les éventuelles nouvelles tentatives requises, même si le temporisateur de rotation du jeton a une valeur inférieure ou égale à celle de T_{RR} au cours de l'exécution.

— Si le temps T_{TH} est écoulé (T_{RR} est supérieur à la valeur courante du temporisateur de rotation du jeton avant le moment d'exécution de l'émission du message), le jeton doit être immédiatement passé à la NS.

NOTE 1 La prolongation du temps de conservation de jeton T_{TH} entraîne automatiquement un raccourcissement du temps d'émission pour des périodes de transfert de messages à la réception de jeton suivante.

NOTE 2 Pour plus d'explications, voir l'Annexe A et l'Annexe C.

5.3.3 Mode émission ou émission/demande

En mode émission ou émission/réception, des périodes de transfert de messages uniques sont lancées sporadiquement. L'entité DL de la station maître lance ce mode à la demande d'un utilisateur local sur réception du jeton. S'il y a plusieurs demandes, ce mode de fonctionnement peut se poursuivre jusqu'à ce que le temps de conservation de jeton maximal autorisé expire.

5.4 Service pris en charge à partir de la PhL

5.4.1 Transmission asynchrone

5.4.1.1 Services PhS d'émission et de réception

Le service de données PhL comprend deux primitives de service. Une primitive de demande utilisée pour demander un service par l'entité DL; et une primitive d'indication utilisée pour indiquer une réception à l'entité DL. Les primitives correspondantes portent les désignations suivantes:

Ph- ASYN-DATA request (demande de données d'émission asynchrone Ph)

Ph-ASYN-DATA indication (Indication de données asynchrones Ph)

La relation temporelle entre primitives est présentée en Figure 3.

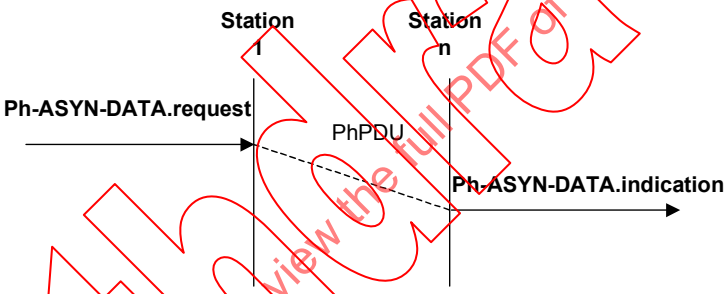


Figure 3 – Service de données PhL pour transmission asynchrone

5.4.1.2 Spécification détaillée du service et échanges de messages

Ce paragraphe décrit en détail les primitives de service et les paramètres correspondants en mode résumé. Les paramètres contiennent l'information nécessaire à l'entité PhL.

Paramètres des primitives:

Demande Ph-ASYN-DATA (DL_symbol : symbole DL)

Le paramètre DL_symbol doit prendre l'une des valeurs suivantes:

a) ZERO qui correspond à un "0" binaire;

b) UN qui correspond à un "1" binaire;

c) SILENCE qui désactive l'émetteur lorsque aucun symbole DL valide ne doit être transmis.

La primitive de demande Ph-ASYN-DATA est transmise de l'entité DL à l'entité PhL afin de demander que le symbole indiqué soit envoyé au support de bus de terrain.

La réception de cette primitive doit engendrer une tentative de codage et d'émission du symbole DL par l'entité PhL.

La demande Ph-ASYN-DATA est une primitive qui ne doit être générée qu'une fois par période de symbole DL (t_{BIT}). L'entité PhL peut confirmer cette primitive au moyen d'une primitive de confirmation définie localement.

Indication Ph-ASYN-DATA (DL_symbol)

Le paramètre DL_symbol doit prendre l'une des valeurs suivantes:

— ZERO qui correspond à un "0" binaire,

— UN qui correspond à un "1" binaire.

La primitive d'indication Ph-ASYN-DATA est transmise de l'entité PhL à l'entité DL pour indiquer qu'un symbole DL a été reçu du support de bus de terrain.

L'indication Ph-ASYN-DATA est une primitive qui ne doit être générée qu'une fois par période de symbole DL reçue (t_{BIT}).

5.4.2 Transmission synchrone

5.4.2.1 Généralités

Ce paragraphe définit les primitives de services physiques (PhS) prises en charge et les contraintes liées à leur utilisation par la DLE.

NOTE Une organisation appropriée des couches exige qu'une entité de couche (N+1) ne soit pas concernée par les moyens utilisés par une couche (N) pour fournir ses services (N) et qu'une interface de services (N) ne restreigne pas exagérément ces moyens. Ainsi, l'interface de service Ph n'exige pas que les DLE soient informées des détails internes de la PhE (par exemple préambule, postambule et configuration des signaux délimiteurs de DLPDU, nombre de bits par baud) et il convient que la PhE ne soit pas empêchée d'utiliser des avancées technologiques appropriées.

5.4.2.2 Primitives du PhS prises en charge

La granularité d'émission dans le protocole de bus de terrain est d'un octet. Il s'agit de la granularité des données utilisateur PhS et des informations de synchronisation échangées au niveau de l'interface PhL-DLL.

5.4.2.2.1 Service de compte-rendu de caractéristiques PhS

Le PhS est supposé fournir la primitive de service suivante afin de rendre compte des caractéristiques essentielles du PhS utilisées dans les activités d'émission, de réception et de planification de la DLL:

Indication Ph-CHARACTERISTICS (débit binaire minimal, surdébit de trame)

où

débit binaire minimal – spécifie le débit minimal d'acheminement des données effectif, en bits par seconde, y compris d'éventuelles tolérances temporelles, de toute PhE sur la liaison locale.

NOTE 1 Une PhE ayant un débit de données nominal de 1 Mbits/s ± 0,01 % indiquerait un débit de données minimal de 0,9999 Mbits/s.

surdébit de trame – le nombre maximal de périodes binaires, où
la période = 1/débit de données,
utilisées dans toute émission de PhPDU qui n'acheminent pas directement des données (par exemple des PhPDU qui acheminent un préambule, des délimiteurs de DLPDU, un postambule, des "silences" entre DLPDU, et ainsi de suite).

NOTE 2 Si le surdébit de trame est F et si deux longueurs de messages DL sont L_1 et L_2, le temps nécessaire pour envoyer deux messages immédiatement consécutifs de longueurs L_1 et L_2 sera au moins égal au temps nécessaire pour l'envoi d'un message de longueur $L_1 + F + L_2$.

Si ce surdébit de trame est supérieur aux DLE configurées pour chaque surcharge de PhL de DLPDU V(PhLO), la DLE doit rendre compte de cette anomalie à la gestion DL et ne

doit pas émettre de demande Ph-DATA tant que l'anomalie existe.

NOTE 3 Cette restriction interdit la transmission des DLE en présence de cette anomalie. La gestion de la station locale des DLE peut remédier à cette anomalie en reconfigurant V(PhLO) à une valeur supérieure.

5.4.2.2.2 Services PhS d'émission et de réception

Le PhS est supposé fournir les primitives de service suivantes pour l'émission et la réception:

demande Ph-DATA (classe, données);

indication Ph-DATA (classe, données);

confirmation Ph-DATA (état)

où

classe – spécifie la composante Ph-interface-control-information ("informations de commande d'interface de couche physique") (PhICI) de l'unité de données d'interface de couche physique (PhIDU). Pour une demande Ph-DATA, les valeurs possibles sont:

START-OF-ACTIVITY (démarrage d'activité) – il convient que commence l'émission des PhPDU qui précèdent les données d'utilisateur Ph;

DATA (données) – il convient que la valeur d'octet unique du paramètre de données correspondant soit transmise comme partie d'une émission continue correctement formée;

END-OF-DATA-AND-ACTIVITY (fin de données et d'activité) – il convient que les PhPDU qui terminent les données d'utilisateur Ph soient émises après le dernier octet précédent des données d'utilisateur Ph, aboutissant à la cessation de l'émission active.

Pour une indication Ph-DATA, les valeurs possibles sont:

START-OF-ACTIVITY – la réception d'une émission apparente d'une ou de plusieurs PhE a commencé;

DATA – le paramètre de données correspondant a été reçu comme partie d'une réception continue correctement formée;

END-OF-DATA (fin de données) – la réception continue correctement formée des données d'utilisateur Ph en cours s'est terminée par la réception correcte des PhPDU impliquant la fin des données;

END-OF-ACTIVITY (fin d'activité) – la réception en cours (d'une émission apparente provenant d'une ou de plusieurs PhE) s'est achevée avec aucune preuve supplémentaire d'émission PhE;

END-OF-DATA-AND-ACTIVITY – occurrence simultanée de END-OF-DATA et de END-OF-ACTIVITY;

data – spécifie la composante données d'interface Ph (PhID) de la PhIDU. Elle est constituée d'un octet de données utilisateur Ph à émettre (demande Ph-DATA) ou qui a été reçu avec succès (indication Ph-DATA).

status (état) – spécifie soit le succès, soit la raison localement détectée qui a donné lieu à l'échec.

La primitive de confirmation Ph-DATA fournit le retour d'informations critiques de temporisation physique nécessaires pour empêcher la DLE de lancer une seconde émission avant que la première ne soit terminée. La confirmation Ph-DATA finale d'une émission ne doit pas être initiée avant que la PhE n'ait terminé l'émission en cours.

5.4.2.3 Notification de caractéristiques PhS

Il incombe à la PhE d'informer la DLE des caractéristiques du PhS qui sont importantes pour le fonctionnement de la DLE. Cette notification est assurée par la PhE par l'émission d'une primitive unique d'indication Ph-CHARACTERISTICS à chaque PhSAP de PhE, au démarrage de la PhE.

5.4.2.4 Emission de données utilisateur Ph

La PhE détermine la synchronisation de toutes les émissions. Lorsqu'une DLE a une DLPDU à émettre et que le protocole DL donne à cette DLE le droit d'émettre, la DLE doit envoyer la DLPDU, y compris une FCS concaténée, en réalisant une séquence bien formée de demandes Ph-DATA, constituée d'une seule demande spécifiant START-OF-ACTIVITY, suivie de 3 à 300 (au maximum) demandes consécutives qui spécifient les DATA (données), et se terminant par une demande unique spécifiant END-OF-DATA-AND-ACTIVITY.

La PhE signale l'achèvement de chaque demande Ph-DATA et indique qu'elle est prête à accepter une nouvelle demande Ph-DATA au moyen d'une primitive de confirmation Ph-DATA; le paramètre d'état de la primitive de confirmation Ph-DATA achemine le résultat, succès ou échec, de la demande Ph-DATA correspondante. Il convient qu'une seconde demande Ph-DATA ne soit lancée qu'après que la confirmation Ph-DATA correspondant à la première demande ait été reçue de la PhE.

5.4.2.5 Réception de données utilisateur Ph

La PhE rend compte d'une émission reçue par une séquence bien formée d'indications Ph-DATA, qui doit être constituée :

a) soit d'une indication unique spécifiant START-OF-ACTIVITY, suivie par des indications consécutives spécifiant les DATA, suivie par une indication unique spécifiant END-OF-DATA; et terminée par une indication unique spécifiant END-OF-ACTIVITY ;

b) soit d'une indication unique spécifiant START-OF-ACTIVITY; suivie par des indications consécutives spécifiant les DATA et suivie par une indication unique spécifiant END-OF-DATA-AND-ACTIVITY;

c) soit d'une indication unique spécifiant START-OF-ACTIVITY; suivie optionnellement d'une ou de plusieurs indications consécutives spécifiant les DATA; et terminée par une indication unique spécifiant END-OF-ACTIVITY.

NOTE Cette dernière séquence indique une réception incomplète ou incorrecte. La détection d'une erreur dans la séquence de PhPDU reçues, ou dans le processus de réception des PhE, désactive des indications Ph-DATA ultérieures avec un paramètre de classe spécifiant les DATA, END-OF-DATA ou END-OF-DATA-AND-ACTIVITY jusqu'à ce que la fin de la période d'activité courante et le début d'une période d'activité suivante aient été notifiés par des indications Ph-DATA spécifiant respectivement END-OF-ACTIVITY et START-OF-ACTIVITY.

Dans les deux premiers cas, la DLE effectue la concaténation des données reçues et tente de les analyser dans une DLPDU suivie par une FCS concaténée. Dans le dernier cas, la DLE rejette toutes les données notifiées et rend compte de l'événement à la gestion DL.

5.5 Éléments opérationnels

5.5.1 Aperçu général

Les temps T définis ci-après sont mesurés en bits. Par conséquent, un temps t exprimé en secondes (s) doit être divisé par le temps binaire t_{BIT}.

5.5.2 Temps binaire t_{BIT}

Le temps binaire t_{BIT} est la durée qui s'écoule pendant l'émission d'un bit. Cette durée est équivalente à la valeur inverse du débit de données:

$$t_{BIT} = \frac{1}{\text{data rate}} \ [\text{s bit}^{-1}] \quad (12)$$

5.5.3 Transmission asynchrone

5.5.3.1 Temps de synchronisation (T_{SYN})

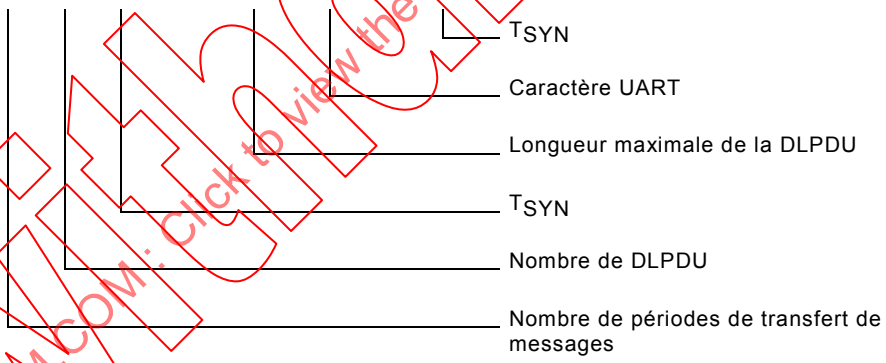
Le temps de synchronisation T_{SYN} est l'intervalle de temps minimal au cours duquel chaque station doit recevoir l'état au repos (repos = binaire "1") du support de transmission avant qu'il ne puisse accepter le début d'une DLPDU d'envoi/demande ou d'une DLPDU de jeton. La valeur du temps de synchronisation doit être réglée comme suit:

$$T_{SYN} = 33 \text{ bit} \quad (13)$$

5.5.3.2 Temps d'intervalle de synchronisation (T_{SYNI})

Le temps d'intervalle de synchronisation T_{SYNI} est l'intervalle de temps maximal admissible entre deux temps de synchronisation consécutifs (T_{SYN}), afin de détecter des “émetteurs permanents”. La valeur du temps d'intervalle de synchronisation doit être réglée comme suit:

$$T_{SYNI} = 2 \times (2 \times (33 \text{ bit} + 255 \times 11 \text{ bit})) + 33 \text{ bit} = 11\,385 \text{ bit} \quad (14)$$



Cette valeur correspond à deux périodes de transfert de messages complets, chacune étant constituée de deux DLPDU de longueur maximale et des temps de synchronisation correspondants. Une perturbation de la transmission est admise au cours d'un de ces temps de synchronisation.

5.5.3.3 Temps de retard de station (T_{SDx})

Le temps de retard de station T_{SDx} est la période qui peut s'écouler entre l'émission ou la réception du dernier bit d'une DLPDU et l'émission ou la réception du premier bit de la DLPDU suivante (en référence au support de transmission, c'est-à-dire y compris le récepteur et l'émetteur de la ligne). Les trois retards de station suivants sont définis:

a) Retard station de l'initiateur (station qui émet la DLPDU de demande ou de jeton):

   TSDI

b) Retard station minimal des répondeurs (stations qui envoient l'acquittement ou la réponse):

   T_{SDR} min

c) Retard station maximal des répondeurs:

T_{SDR} max

5.5.3.4 Temps de silence (T_{QUI})

Lors de la transposition de signaux NRZ en un codage de signal différent, le temps de décroissance de l'émetteur, après mise hors-tension de l'émetteur (au niveau de l'initiateur), doit être pris en compte s'il est supérieur à T_{SDR}.

Au cours de ce temps de silence T_{QUI}, l'émission et la réception des DLPDU doivent être désactivées. Ceci doit également être pris en compte lorsqu'il est utilisé des répéteurs autonomes dont le temps de commutation doit être pris en compte. L'application doit s'assurer que la condition suivante est remplie:

$$T_{QUI} < \min T_{SDR} \quad (15)$$

Pour remplir cette condition, il peut être nécessaire d'allonger T_{SDR} min.

5.5.3.5 Temps à l'état prêt (T_{RDY})

Le temps à l'état prêt T_{RDY} est la durée pendant laquelle une station maître doit être prête à recevoir un acquittement ou une réponse après envoi d'une demande. L'application doit s'assurer que la condition suivante est remplie:

$$T_{RDY} < \min T_{SDR} \quad (16)$$

Pour remplir cette condition, il peut être nécessaire d'allonger T_{SDR} min.

Lors de la transposition de signaux NRZ en un codage de signal différent, le temps de silence doit également être pris en compte lorsque l'émetteur est mis hors tension. La désactivation du récepteur ne doit pas avoir lieu avant cet instant:

$$T_{QUI} < T_{RDY} \quad (17)$$

Pour remplir cette condition, il peut être nécessaire d'allonger T_{RDY} et par conséquent T_{SDR} min.

5.5.3.6 Marge de sécurité (T_{SM})

L'intervalle de temps suivant est spécifié comme étant la marge de sécurité T_{SM}:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} + T_{QUI} \quad (18)$$

T_{SET} est le temps de montée qui s'écoule entre l'occurrence d'un événement (par exemple, l'interruption du dernier bit d'une DLPDU envoyée ou l'expiration du temps de synchronisation) et la réaction nécessaire correspondante (par exemple, démarrage du temps de synchronisation ou activation du récepteur).

5.5.3.7 Temps au repos (T_{IDx})

Le temps au repos T_{IDx} est le temps écoulé au niveau de l'initiateur, soit entre la réception du dernier bit d'une DLPDU (mesuré au niveau du récepteur de la ligne) à l'état repos = binaire "1" sur le support de transmission et l'émission du premier bit d'une nouvelle DLPDU sur le support (y compris l'émetteur de la ligne), soit entre l'émission du dernier bit d'une DLPDU sans acquittement et l'émission du premier bit de la DLPDU suivante. Le temps au

repos doit être au moins égal au temps de synchronisation plus le temps de la marge de sécurité T_{SM} (voir la Figure 4, la Figure 5 et la Figure 6 cas a)).

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (19)$$

A des débits de données élevés (voir la Figure 4, cas b) et c) et la Figure 5 cas b)) le temps de synchronisation est très court, et par conséquent, les retards de station deviennent significatifs et doivent être pris en compte.

On distingue deux temps au repos. Après une DLPDU d'acquittement, de réponse ou de jeton, le temps au repos doit être calculé de la manière suivante:

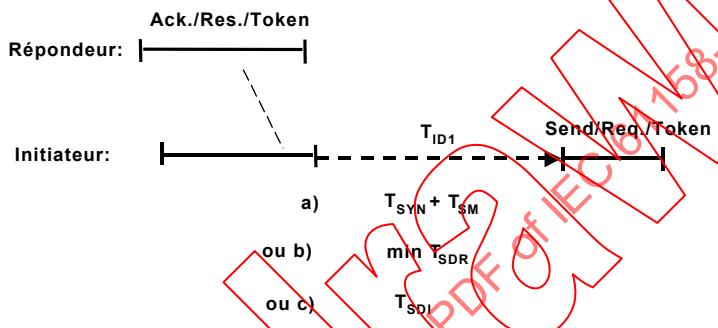
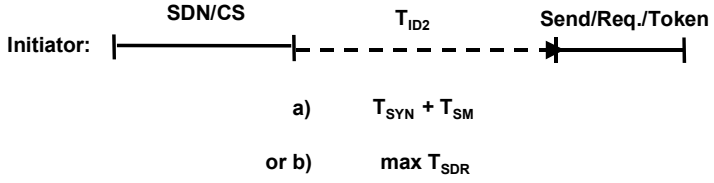


Figure 4 – Temps au repos T_{ID1}

$$T_{ID1} = \max((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (20)$$

On doit s'assurer que la durée nécessaire à la mise à jour de la LMS n'est pas supérieure à T_{ID1}. Ceci peut être réalisé en prolongeant T_{SET} ou T_{SDR} min. Si l'allongement de durée nécessaire ne peut être obtenu dans la plage de valeurs de T_{SET}, on doit augmenter la durée de T_{SDR} min.

Après une DLPDU d'émission **sans** acquittement (SDN ou CS), le temps au repos doit être calculé comme présenté dans la Figure 5 et dans l'équation 34.



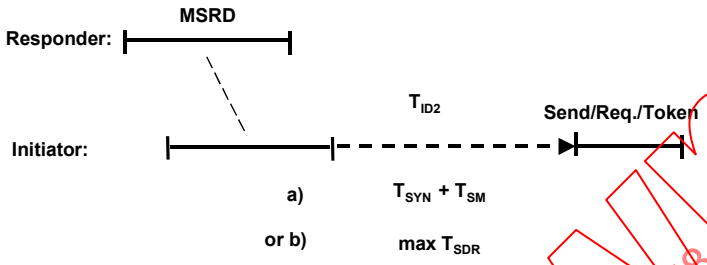
Légende

Anglais	Français
initiator	initiateur
or	ou

Figure 5 – Temps au repos T_{ID2} (SDN, CS)

$$T_{ID2} = \max ((T_{SYN} + T_{SM}) , (\max T_{SDR})) \quad (21)$$

De la même manière, après une DLPDU de réponse (MSRD), le temps au repos doit être calculé comme présenté dans la Figure 6 et dans l'équation 34.



Légende

Anglais	Français
initiator	initiateur
or	ou
responder	répondeur

Figure 6 – Temps au repos T_{ID2} (MSRD)

5.5.3.8 Temps de retard d'émission (T_{TD})

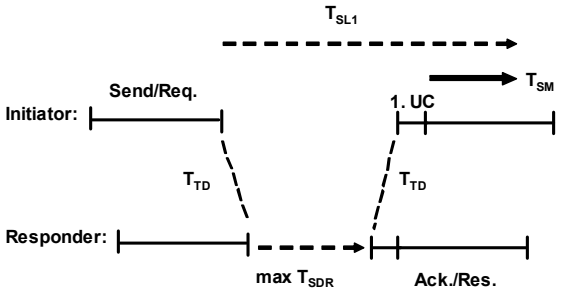
Le temps de retard d'émission T_{TD} est la durée maximale qui s'écoule sur le support de transmission, entre l'émetteur et le récepteur, lorsqu'une DLPDU est transmise. Le calcul correspondant doit, si nécessaire, tenir compte des temps de retard des répéteurs.

EXEMPLE de calcul du temps de retard d'émission: étant donné une longueur de ligne de 200 m sans répéteurs, t_{TD} est environ 1 µs et ainsi, à 500 kbits/s:

T_{TD} = (1µs × 500 kbits/s) = 0,5 bit.

5.5.3.9 Durée de créneau (T_{SL})

La durée de créneau T_{SL} est la durée maximale pendant laquelle l'initiateur doit attendre la réception complète du premier caractère (voir 6.1.1, caractère UART,1 UC = 11 bits) de la DLPDU d'acquittement ou de réponse immédiate, après émission du dernier bit d'une DLPDU d'envoi/demande (y compris l'émetteur de la ligne) (voir la Figure 7). F Par la suite, T_{SL} est la durée maximale pendant laquelle l'initiateur attend le premier caractère UART (1.UC) du destinataire de jeton, après émission d'une DLPDU de jeton (voir la Figure 8). En théorie, on distingue deux durées de créneau. Après une DLPDU d'envoi/demande, la durée de créneau doit être calculée de la manière suivante:



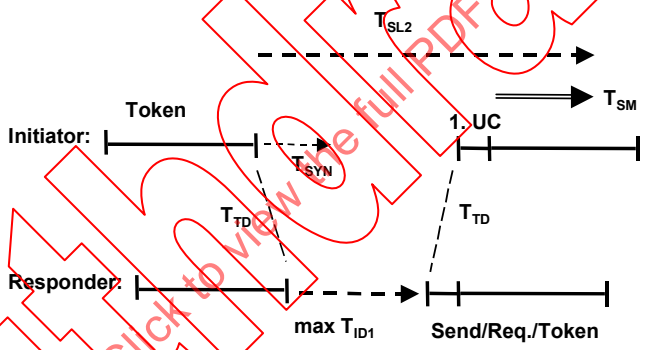
Légende

Anglais	Français
responder	Répondeur
Initiator	initiateur

Figure 7 – Durée de créneau T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + 11 \text{ bit} + T_{SM} \quad (22)$$

Après une DLPDU de jeton, la durée de créneau doit être calculée de la manière suivante:



Légende

Anglais	Français
initiator	initiateur
token	jeton
responder	répondeur

Figure 8 – Durée de créneau T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + 11 \text{ bit} + T_{SM} \quad (23)$$

Pour simplifier la réalisation, il est admis d'utiliser dans le système **une** seule durée de créneau: la plus longue. Ceci n'a pas d'effet négatif sur le temps de réaction du système car la durée de créneau est un temps de pure surveillance.

$$T_{SL} = \max (T_{SL1} , T_{SL2}) \quad (24)$$

5.5.3.10 Durée du temps imparti (T_{TO})

La durée du temps imparti T_{TO} sert à surveiller le temps d'activité et de repos du bus pour les stations maîtres et esclaves. La surveillance doit commencer soit immédiatement après PON, à l'état "Listen_Token" ou "Passive_Idle", soit plus tard après réception du dernier bit d'une

DLPDU. Elle doit s'achever après réception du premier bit de la DLPDU suivante. Si la phase de non activité du bus expire, le bus doit être considéré comme inactif. (Il s'agit d'un cas d'erreur dû, par exemple, à la perte d'une DLPDU de jeton). Cet intervalle de temps doit être réglé comme suit:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (25)$$

Pour des stations maîtres: n = adresse de station (0 à 126)

Pour des stations esclaves: n = 130, indépendamment de l'adresse de station

Le premier terme permet de s'assurer qu'il existe une différence suffisante par rapport au temps au repos maximal admissible entre deux DLPDU. Le second terme permet de s'assurer que toutes les stations maîtres ne revendiquent pas le jeton au même moment après apparition d'une erreur.

5.5.3.11 Temps de mise à jour GAP (T_{GUD})

Le temps de mise à jour GAP (intervalle) T_{GUD} est utilisé pour initialiser la maintenance du GAP par la station maître. Après génération de la première GAPL, la mise à jour de l'image GAP est initialisée de manière cyclique après chaque intervalle T_{GUD}. L'initialisation a lieu à la réception de jeton suivante s'il reste suffisamment de temps de conservation de jeton disponible après les périodes normales de transfert de DLPDU ou au cours des dernières phases de détention du jeton. Le temps de mise à jour du GAP est un multiple du temps de rotation cible T_{TR} et doit être réglé de la manière suivante:

$$T_{GUD} = G \times T_{TR} \quad (\text{où } 1 \leq G \leq 100) \quad (26)$$

T_{TR} est le temps de rotation cible (voir 5.3.2.6).

5.5.3.12 Mode isochrone

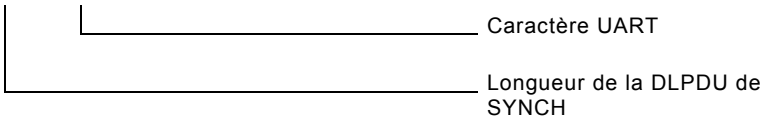
5.5.3.12.1 Durée de cycle isochrone (T_{CT})

La durée de cycle isochrone T_{CT} est utilisée pour superviser le cycle isochrone. Le premier cycle démarre par l'envoi de la DLPDU de SYNCH après réception de la DLPDU de jeton.

5.5.3.12.2 Temps de DLPDU de synchronisation IsoM (T_{SYNCH})

Le temps de DLPDU de synchronisation en mode isochrone décrit la durée nécessaire pour envoyer la DLDPU de SYNCH au début d'un nouveau cycle IsoM. La valeur du temps de message de synchronisation IsoM doit être réglée comme suit:

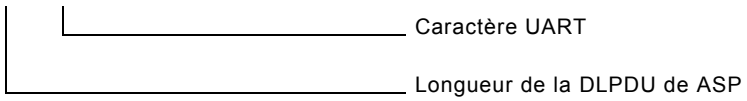
$$T_{SYNCH} = 13 \times 11 \text{ bit} = 143 \text{ bit} \quad (27)$$



5.5.3.12.3 Temps de DLPDU temporelle de réserve active (T_{ASM})

Le temps de DLPDU temporelle de réserve active décrit la durée nécessaire pour envoyer une DLPDU temporelle de réserve active (ASP).

$$T_{ASM} = T_{ID1} + (6 \times 11 \text{ bit}) \quad (28)$$



5.5.3.12.4 Durée de cycle isochrone réel (T_{RCT})

La durée de cycle isochrone réel est la valeur lue du temporisateur de cycle isochrone.

5.5.3.12.5 Temps de réserve (T_{RES})

Au cours du cycle isochrone, le jeton est passé par le maître IsoM après la transmission de tous les messages prioritaires et du nombre de messages non prioritaires configurés. Après réception du jeton suivant adressé au maître IsoM, la station lit la valeur de la durée de cycle isochrone (T_{RCT}) afin de calculer le temps restant avant la DLPDU de SYNCH suivante à envoyer.

$$T_{RES} = T_{CT} - T_{RCT} \quad (29)$$

Au cours du temps de réserve, au moins une DLPDU temporelle de réserve active doit être envoyée. Pour d'autres temps de réserve active, les DLPDU doivent être valides.

$$T_{RES} > T_{ASM} + T_{ID1} \quad (30)$$

5.5.3.12.6 Temps de réserve passive (T_{PSP})

Le temps de réserve passive représente la partie du cycle isochrone au cours de laquelle le maître IsoM ne présente aucune activité de bus. Il doit être inférieur à la durée minimale du temps imparti T_{TO} en tenant compte du retard possible dans les stations et pendant l'émission. Le temps de réserve passive est défini comme suit:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (31)$$

$$T_{PSP} > T_{ID1} + T_{ASM} + \max T_{SDR} \quad (32)$$

5.5.3.12.7 Décalage temporel (T_{SH})

Le décalage temporel est la différence entre la durée de cycle mesurée et la durée de cycle calculée à l'expiration du temporisateur de réserve passive.

$$T_{SH} = T_{RCT} - T_{CT} \quad (33)$$

5.5.3.13 Temps de retard d'émission (T_{SD})

Le temps de retard d'émission est la durée qui s'écoule entre la réception d'une primitive DL-CS-TIME-EVENT.request (demande d'événement temporel de CS de DL) au niveau de la DLE du maître d'horloge et du dernier bit d'une DLPDU d'événement temporel TE résultant (voir 7.7) transmise par le maître d'horloge.

5.5.3.14 Temps de retard de réception (T_{RD})

Le temps de retard de réception est la durée qui s'écoule au niveau d'un récepteur d'horloge, entre la réception du dernier bit d'une DLPDU de TE (voir 7.7) et la réception du dernier bit d'une DLPDU de valeur d'horloge (voir 7.8).

5.5.3.15 Temps d'intervalle de synchronisation d'horloge (T_{CSI})

Le temps d'intervalle de synchronisation d'horloge est utilisé pour surveiller les séquences de synchronisation au sein de la DLE.

5.5.4 Transmission synchrone

5.5.4.1 Temps de synchronisation (T_{SYN})

Le temps de synchronisation est l'intervalle de temps minimal au cours duquel chaque station ne doit recevoir aucune activité du support de transmission avant qu'il ne puisse accepter le début d'une DLPDU d'envoi/demande ou d'une DLPDU de jeton.

Le temps de synchronisation doit correspondre au temps d'intervalle (de GAP) post-émission (T_{PTG}) défini en 9.2.8 de la CEI 61158-2. Sa valeur doit être réglée à au moins 4 bits et peut être augmentée par l'utilisateur DLMS jusqu'à 32 bits.

$$T_{SYN} = T_{PTG} = T_{QUI} \quad (34)$$

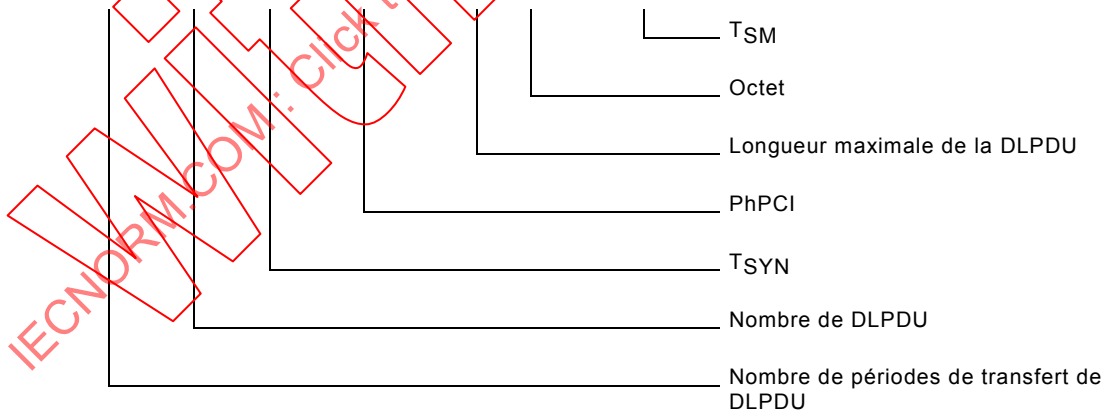
$$T_{SYN} = 4 \text{ à } 32 \text{ bit} \quad (35)$$

5.5.4.2 Temps d'intervalle de synchronisation (T_{SYNI})

Le temps d'intervalle de synchronisation T_{SYNI} est l'intervalle de temps maximal admissible entre deux temps de synchronisation consécutifs (T_{SYN}), afin de détecter des “émetteurs permanents”.

La valeur de T_{SYNI} doit être réglée comme suit:

$$T_{SYNI} = 2 \times (2 \times (32 \text{ bit} + 80 \text{ bit} + 255 \times 8 \text{ bit})) + 64 \text{ bit} = 8\,672 \text{ bit} \quad (36)$$



Cette valeur correspond à deux séquences de messages complètes dont chacune est constituée de deux DLPDU de longueur maximale et du nombre maximal d'informations de commande de protocole PhL (PhPCI: préambule, délimiteur de début, délimiteur de fin) et du temps maximal de synchronisation (gap post-émission). Une perturbation de la transmission est admise au cours d'un de ces temps de synchronisation.

5.5.4.3 Temps de retard de station (T_{SDx})

Le temps de retard de station T_{SDx} est la période qui peut s'écouler entre l'émission ou la réception du dernier octet d'une DLPDU et l'émission ou la réception du premier octet de la DLPDU suivante (en référence au support de transmission, c'est-à-dire y compris le récepteur et l'émetteur de la ligne). Les trois retards de station suivants sont définis:

a) Retard station de l'initiateur (station qui émet la DLPDU de demande ou de jeton)

T_{SDI}

b) Retard station minimal des répondeurs (station qui envoie l'acquittement ou la réponse):

T_{SDR} min

c) Retard station maximal des répondeurs

max T_{SDR}

5.5.4.4 Temps de silence (T_{QUI})

Le temps de décroissance de l'émetteur ou le temps de commutation du répéteur correspond au temps d'intervalle (de gap) post-émission (T_{SYN}). Les points suivants doivent être respectés:

$$T_{QUI} = T_{PTG} = T_{SYN} \quad (37)$$

5.5.4.5 Temps à l'état prêt (T_{RDY})

Le temps à l'état prêt T_{RDY} est la durée pendant laquelle une station maître doit être prête à recevoir un acquittement ou une réponse après envoi d'une demande. L'application doit s'assurer que la condition suivante est remplie:

$$T_{RDY} < \min T_{SDR} \quad (38)$$

Pour remplir cette condition, il peut être nécessaire d'allonger T_{SDR} min.

Au cours de ce temps de silence TQUI, l'émission et la réception des DLPDU doivent être désactivées. L'application doit s'assurer que la condition suivante est remplie:

$$T_{QUI} = T_{PTG} = T_{SYN} < T_{RDY} \quad (39)$$

Pour remplir cette condition, T_{SDR} min doit être, si nécessaire, augmenté conformément à l'équation (38).

5.5.4.6 Marge de sécurité (T_{SM})

La marge de sécurité T_{SM} est définie comme étant l'intervalle de temps:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} \quad (40)$$

T_{SET} est le temps de montée qui s'écoule entre l'occurrence d'un événement (par exemple, l'interruption du dernier octet d'une DLPDU envoyée ou l'expiration du temps de synchronisation) et l'exécution de la réaction requise.

5.5.4.7 Temps au repos (T_{IDx})

Le temps au repos T_{IDx} est la durée qui s'écoule au niveau de l'initiateur entre une primitive d'indication Ph-DATA et l'envoi d'une nouvelle DLPDU avec une primitive de demande Ph-DATA ou entre l'envoi d'une primitive de demande Ph-DATA avec une primitive de confirmation Ph-DATA pour l'émission d'une DLPDU sans acquittement et l'envoi d'une nouvelle primitive de demande Ph-DATA pour l'émission de la DLPDU suivante. Le temps au repos doit être au moins égal au temps de synchronisation plus le temps de la marge de sécurité T_{SM}.

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (41)$$

On distingue deux temps au repos (voir la description du temps au repos en 5.5.3.7, dans la Figure 4, la Figure 5 et la Figure 6). Après une DLPDU d'acquittement, de réponse ou de jeton, le temps au repos doit être calculé comme suit:

$$T_{ID1} = \max ((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (42)$$

On doit s'assurer que la durée nécessaire à la mise à jour de la LMS n'est pas supérieure à T_{ID1}. Ceci peut être réalisé en prolongeant T_{SET} ou T_{SDR} min. Si l'allongement de durée nécessaire ne peut être obtenu dans la plage de valeurs de T_{SET}, on doit augmenter la durée de T_{SDR} min.

Après une DLPDU d'émission sans acquittement (SDN ou CS), le temps au repos doit être calculé comme suit:

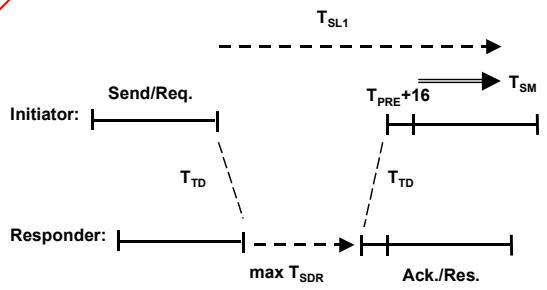
$$T_{ID2} = \max ((T_{SYN} + T_{SM}), (\max T_{SDR})) \quad (43)$$

5.5.4.8 Temps de retard d'émission (T_{TD})

Le temps de retard d'émission T_{TD} est la durée maximale qui s'écoule sur le support de transmission, entre l'émetteur et le récepteur, lorsqu'une DLPDU est transmise. Le calcul correspondant doit, si nécessaire, tenir compte des temps de retard des répéteurs. Ce temps de retard doit également se conformer aux éventuelles restrictions qui lui sont imposées par la CEI 61158-2, où il est appelé “délai de propagation”.

5.5.4.9 Durée de créneau (T_{SL})

La durée de créneau T_{SL} est la durée maximale pendant laquelle l'initiateur doit attendre après l'envoi d'une primitive de demande Ph-DATA pour l'émission d'une DLPDU de demande entre la primitive de confirmation Ph-DATA et la réception de la première primitive d'indication Ph-DATA qui signale la réception immédiate de l'acquittement ou de la réponse (voir Figure 9). Par ailleurs, T_{SL} est la durée maximale pendant laquelle l'initiateur doit attendre une primitive d'indication Ph-DATA après la DLPDU de jeton, comme réaction à la réception du premier octet de DLPDU du destinataire du jeton. En théorie, on distingue deux durées de créneau (voir la Figure 10). Après une DLPDU d'envoi/demande, la durée de créneau doit être calculée de la manière suivante:



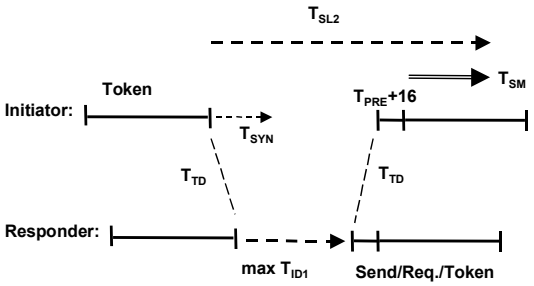
Légende

Anglais	Français
initiator	initiateur
responder	répondeur

Figure 9 – Durée de créneau T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (44)$$

Après une DLPDU de jeton, la durée de créneau doit être calculée de la manière suivante:



Légende

Anglais	Français
initiator	initiateur
responder	répondeur
token	jeton

Figure 10 – Durée de créneau T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (45)$$

où T_{PRE} est la période de préambule (voir la CEI 61158-2)

Pour simplifier la réalisation, seule la durée de créneau la plus longue peut être utilisée dans le système. Ceci n'a pas d'effet négatif sur le temps de réaction du système car la durée de créneau est tout simplement un temps de surveillance.

$$T_{SL} = \max (T_{SL1}, T_{SL2}) \quad (46)$$

5.5.4.10 Durée du temps imparti (T_{TO})

La durée du temps imparti T_{TO} sert à surveiller le temps d'activité et de repos du bus pour les stations maîtres et esclaves. La surveillance doit commencer soit immédiatement après PON, à l'état "Listen_Token" ou "Passive_Idle", soit plus tard après réception d'une primitive d'indication Ph-DATA. Cette durée doit se terminer dès qu'une primitive d'indication Ph-DATA est reçue pour la réception du premier octet de la DLPDU suivante. Si la phase de non activité du bus expire, le bus doit être considéré comme inactif. (Il s'agit d'un cas d'erreur dû, par exemple, à la perte d'une DLPDU de jeton). Cet intervalle de temps imparti doit être réglé comme suit:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (47)$$

Pour des stations maîtres: n = adresse de station (0 à 126)

Pour des stations esclaves: n = 130, indépendamment de l'adresse de station

5.5.4.11 Temps de mise à jour GAP (T_{GUD})

Les dispositions du 5.5.3.11 doivent s'appliquer.

5.5.4.12 Mode isochrone

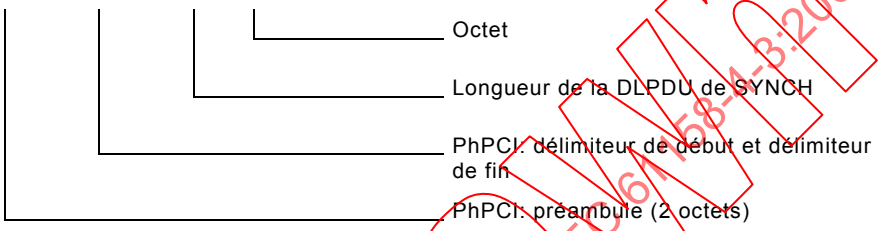
5.5.4.12.1 Durée de cycle isochrone (T_{CT})

Les dispositions du 5.5.3.12.1 doivent s'appliquer.

5.5.4.12.2 Temps de DLPDU de synchronisation IsoM (T_{SYNCH})

Le temps de DLPDU de synchronisation en mode isochrone décrit la durée nécessaire à l'envoi de la DLDPU de SYNCH au début d'un nouveau cycle IsoM. La valeur du temps de message de synchronisation IsoM doit être réglée comme suit:

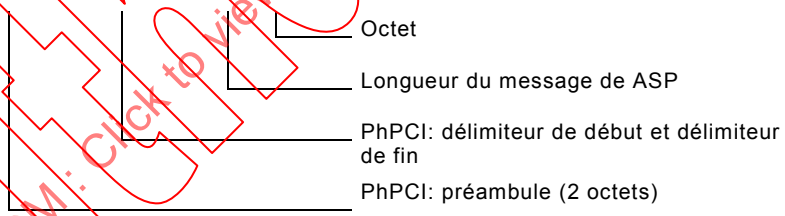
$$T_{SYNCH} = 16 \text{ bit} + 16 \text{ bit} + 13 \times 8 \text{ bit} = 136 \text{ bit} \quad (48)$$



5.5.4.12.3 Temps de DLPDU temporelle de réserve active (T_{ASM})

Le temps de DLPDU temporelle de réserve active décrit la durée nécessaire pour envoyer une DLPDU temporelle de réserve active (ASP).

$$T_{ASM} = T_{PTG} + 16 \text{ bit} + 16 \text{ bit} + (6 \times 8 \text{ bit}) \quad (49)$$



5.5.4.12.4 Durée de cycle isochrone réel (T_{RCT})

Les dispositions de 5.5.3.12.4 doivent s'appliquer.

5.5.4.12.5 Spare time (T_{RES})

Les dispositions du 5.5.3.12.5 doivent s'appliquer.

5.5.4.12.6 Temps de réserve passive (T_{PSP})

Le temps de réserve passive représente la partie du cycle isochrone au cours de laquelle le maître IsoM ne présente aucune activité de bus. Il doit être inférieur à la durée minimale du temps imparti T_{TO} en tenant compte du retard possible dans les stations et pendant l'émission. Le temps de réserve passive est défini comme suit:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (50)$$

$$T_{PSP} > T_{ID1} + T_{ASM} + \max T_{SDR} \quad (51)$$

5.5.4.12.7 Décalage temporel (T_{SH})

Les dispositions du 5.5.3.12.6 doivent s'appliquer.

5.5.4.13 Temps de retard d'émission (T_{SD})

Le temps de retard d'émission est la durée qui s'écoule entre une primitive de demande DL-CS-TIME-EVENT envoyée à la DLE du maître d'horloge et le dernier octet d'une DLPDU de TE résultant (voir 7.7) transmise par le maître d'horloge.

5.5.4.14 Temps de retard de réception (T_{RD})

Le temps de retard de réception est la durée qui s'écoule au niveau d'un récepteur d'horloge, entre la réception du dernier octet d'une DLPDU de TE (voir 7.7) et le dernier octet d'une DLPDU de CV (voir 7.8).

5.5.4.15 Temps d'intervalle de synchronisation d'horloge (T_{CSI})

Les dispositions du 5.5.3.15 doivent s'appliquer.

5.5.5 Temporisateurs et compteurs

5.5.5.1 Transmission asynchrone

5.5.5.1.1 Temporisateurs

Pour mesurer le temps de rotation de jeton et appliquer les durées de surveillance; les temporisateurs suivants doivent être mis en œuvre:

le temporisateur de rotation du jeton, le temporisateur de repos, le temporisateur de créneau, le temporisateur de temps imparti, le temporisateur d'intervalle de synchronisation, le temporisateur de mise à jour du GAP, le temporisateur de cycle isochrone, le temporisateur de réserve passive, le temporisateur de retard d'émission et le temporisateur de retard de réception.

Temporisateur de rotation du jeton: lorsqu'une station maître reçoit le jeton, ce temporisateur est chargé avec le temps de rotation cible T_{TR} et décrémenté à chaque temps binaire. Lorsque la station reçoit de nouveau le jeton, la valeur du temporisateur, c'est-à-dire le temps restant ou le temps de conservation du jeton T_{TH}, est lue et le temporisateur est rechargé avec T_{TR}. Le temps réel de rotation T_{RR} est obtenu en calculant la différence $T_{TR} - T_{TH}$.

Le temporisateur de rotation du jeton peut être lu à tout moment et représente toujours la valeur de la rotation réelle du jeton. Des périodes de transfert de messages non prioritaires peuvent être traitées si à ce moment-là, le temps de rotation réel du jeton est inférieur à la valeur de la rotation réelle du jeton.

Temporisateur de repos: ce temporisateur surveille l'état au repos (binaire "1"), c'est-à-dire le temps de synchronisation immédiat sur la ligne de bus. Le temps de synchronisation qui précède chaque demande est nécessaire pour une synchronisation des récepteurs sans aucune ambiguïté. Le temporisateur de repos des stations esclaves et des stations maîtres "sans jeton" est chargé avec T_{SYN} après émission ou réception du dernier bit des DLPDU puis décrémenté à chaque temps binaire. Le récepteur doit être activé immédiatement après expiration du temporisateur. Le temporisateur d'une station maître "avec jeton" est chargé conformément au service émission de données avec T_{ID1} ou T_{ID2} (voir 5.5.3). Une nouvelle DLPDU de demande ou de jeton ne peut être transmise qu'après expiration du temporisateur. Lorsque le niveau du signal est binaire "0", le temporisateur est toujours rechargé.

Temporisateur de créneau: ce temporisateur, dans une station maître, après envoi d'une demande ou passage du jeton, surveille la réponse de la station de réception ou devient actif au cours du temps prédéfini T_{SL} (durée de créneau). Après émission du dernier bit des DLPDU, ce temporisateur est chargé avec T_{SL} et décrémenté à chaque temps binaire dès que le récepteur est activé. Si le temporisateur expire avant réception du premier bit de DLPDU, une erreur a eu lieu. Dans ce cas, une nouvelle tentative ou une nouvelle période de transfert de messages doit être lancée.

Temporisateur de temps imparti: ce temporisateur surveille les activités du bus dans des stations maîtres et esclaves. Après émission ou réception du dernier bit des DLPDU, ce temporisateur est chargé avec un multiple de la durée de créneau (voir 5.5.3) et décrémenté à chaque temps binaire tant qu'aucune nouvelle DLPDU n'est reçue. Si le temporisateur expire, une erreur fatale est survenue, et dans ce cas, la station maître provoque une (ré)initialisation. L'utilisateur DLMS des stations esclaves et maîtres reçoit une notification d'expiration du délai.

Temporisateur d'intervalle de synchronisation: les stations maîtres et esclaves utilisent ce temporisateur pour surveiller le support de transmission afin d'établir si, au cours de TSYNI, il y a synchronisation du récepteur (T_{SYN}, état au repos, repos = binaire "1"). Chaque fois que le récepteur est synchronisé, le temporisateur est chargé avec T_{SYNI} (voir 5.5.3). Dès le début d'une DLPDU (premier bit de départ), le temporisateur est décrémenté à chaque temps binaire tant qu'aucun nouveau T_{SYN} n'est détecté. Si ce temporisateur expire, une erreur a eu lieu sur le support de transmission, par exemple, bloqué à "0" ou fronts "0" / "1" permanents. L'utilisateur DLMS est informé en conséquence.

Temporisateur de mise à jour du GAP: seules les stations maîtres nécessitent ce temporisateur. Son expiration indique l'instant de maintenance du GAP. Après une vérification complète du GAP qui peut durer plusieurs rotations de jeton (segmentées; voir 5.3.2.4), le temporisateur est chargé avec un multiple (G) du temps de rotation cible T_{TR} (voir 5.3.2.6).

NOTE Pour faciliter la mise en œuvre, ce temporisateur peut être un compteur décrémenté à chaque réception de jeton.

Temporisateur de cycle isochrone: ce temporisateur surveille le cycle isochrone. Il est chargé d'une valeur égale à 0 au point de départ du mode isochrone (après réception du premier jeton) et incrémenté à chaque temps binaire. Ce temporisateur est lisible à tout moment. Ce temporisateur est lancé soit par le chargement d'une valeur égale à 0 dès le commencement du cycle isochrone, soit après expiration du temporisateur de réserve passive. Si le temporisateur est lancé trop tard, il est chargé d'une valeur égale à la différence temporelle entre la durée de cycle isochrone réel du dernier cycle et la durée de cycle isochrone.

Temporisateur de réserve passive: ce temporisateur surveille le temps au repos avant d'envoyer un message SYNCH. Il doit être réglé en fonction de la différence temporelle entre T_{CT} et la valeur courante du temporisateur de cycle isochrone T_{RCT} à la fin du dernier message ASP si cette différence est supérieure à 0. L'émission du dernier bit du dernier message ASP au cours d'un cycle isochrone lance ce temporisateur. En mode isochrone, après expiration du temporisateur de réserve passive, un message SYNCH doit être envoyé et un événement de notification SYNCH doit être envoyé à l'utilisateur DLMS.

Temporisateur de retard d'émission: lorsque le maître d'horloge reçoit une primitive de demande DL-CS-TIME-EVENT, de la part de l'utilisateur DLS local, ce temporisateur est lancé avec la valeur = $2 \times T_{CSI}$ et il est décrémenté à chaque temps binaire. Le temporisateur est arrêté après confirmation de l'émission de la dernière portion d'une DLPDU de TE (voir 7.7). La valeur du temps de retard d'émission est calculée comme étant la différence entre la valeur de départ ($2 \times T_{CSI}$) et la valeur relevée du temporisateur de retard d'émission. Le temps de retard d'émission calculé est transmis à l'utilisateur DLS local. En cas d'expiration du temporisateur, l'utilisateur DLS est avisé d'une violation de séquence de synchronisation d'horloge.

Temporisateur de retard de réception: lorsque le récepteur d'horloge reçoit le dernier bit des DLPDU d'une DLPDU de TE (voir 7.7), ce temporisateur est lancé avec la valeur = $2 \times T_{CSI}$ et décrémenté à chaque temps binaire. Ce temporisateur s'arrête après réception du dernier bit d'une DLPDU de CV (voir 7.8). La valeur du temps de retard de réception est calculée comme étant la différence entre la valeur de départ ($2 \times T_{CSI}$) et la valeur relevée du temporisateur de retard de réception. Le temps de retard de réception calculé est transféré à l'utilisateur DLS local. En cas d'expiration du temporisateur, l'utilisateur DLS est avisé d'une violation de séquence de synchronisation d'horloge.

Lorsqu'une station maître passe à l'état "Listen_Token", le temporisateur de repos est chargé avec T_{SYN}, le temporisateur de temps imparti est chargé avec T_{TO}, le temporisateur d'intervalle de synchronisation est chargé avec T_{SYNI} et les autres temporisateurs sont remis à zéro. Lorsqu'une station esclave passe à l'état "Passive_Idle", le temporisateur de temps imparti est chargé avec T_{TO} et le temporisateur d'intervalle de synchronisation est chargé avec T_{SYNI}.

5.5.5.1.2 Compteurs

Pour l'installation et la maintenance, les paires suivantes de compteurs (variables DL) peuvent être utilisées:

Pour les stations maîtres:

— compteur des DLPDU émises (DLPDU_sent_count – nombre de DLPDU envoyées), sauf pour les services SDN et demande d'état DL avec réponse

— compteur de nouvelles tentatives d'émission de DLPDU (Retry_count – nombre de nouvelles tentatives)

Des compteurs supplémentaires peuvent être prévus pour chaque station distante particulière:

— compteur des DLPDU émises (DLPDU_sent_count_sr – nombre de DLPDU SR envoyées), sauf pour les services SDN et demande d'état DL avec réponse ainsi que pour les DLPDU de jeton

— compteur des DLPDU émises sans réponse ou avec une réponse erronée (Error_count – nombre d'erreurs), sauf pour les services SDN et demande d'état DL avec réponse ainsi que pour les DLPDU de jeton

Pour les stations esclaves et maîtres:

— compteur des délimiteurs de début valides reçus (SD_count – nombre de SD)

— compteur des délimiteurs de début invalides reçus (SD_error_count – nombre d'erreurs SD)

Lorsqu'une station passe à l'état "Listen_Token" ou "Passive_Idle"", les compteurs sont remis à zéro et activés. Si un compteur atteint sa valeur maximale, le comptage ainsi que celui du compteur comparatif correspondant s'arrête. Lorsqu'un compteur est remis à zéro, le compteur comparatif correspondant est également remis à zéro et ils sont de nouveau activés. L'utilisateur DLMS peut accéder à ces compteurs en utilisant les services Set/Read Value (établir/lire valeur).

5.5.5.2 Transmission synchrone

5.5.5.2.1 Temporisateurs

Comme déjà indiqué en 5.5.5.1.1, les temporisateurs suivants doivent être mis en œuvre pour mesurer le temps de rotation de jeton et assurer la surveillance:

Le temporisateur de rotation du jeton, le temporisateur de repos, le temporisateur de créneau, le temporisateur de temps imparti, le temporisateur d'intervalle de synchronisation, le temporisateur de mise à jour du GAP, le temporisateur de cycle isochrone, le temporisateur de réserve passive, le temporisateur de retard d'émission et le temporisateur de retard de réception.

Temporisateur de rotation du jeton: La fonctionnalité de ce temporisateur est définie en 5.5.5.1.1.

Temporisateur de repos: Ce temporisateur surveille l'état au repos $T_{PTG} = T_{SYN} = T_{QUI}$ sur la ligne de bus. Le temporisateur de repos dans la station maître qui détient le jeton est chargé avec T_{ID1} ou T_{ID2} en fonction du service d'émission de données (voir 5.5.4). Le temporisateur est décrémenté à chaque temps binaire, lorsque après une primitive de demande Ph-DATA (informations PhICI spécifiant END-OF-DATA-AND-ACTIVITY), est transférée: soit la primitive de confirmation Ph-DATA au niveau de l'émetteur, soit la primitive d'indication Ph-DATA (informations PhICI spécifiant END-OF-ACTIVITY ou END-OF-DATA-AND-ACTIVITY) au niveau du récepteur. Une nouvelle DLPDU de demande ou jeton ne peut être émise qu'après expiration de ce temporisateur.

Temporisateur de créneau: Après une demande ou un transfert de jeton par une station maître, ce temporisateur de la station maître surveille la réponse de la station de réception ou son passage à l'état actif au cours de la durée de créneau TSL définie. Ce temporisateur est initialisé avec T_{SL} et décrémenté à chaque temps binaire après chaque émission d'une DLPDU. Cette émission de DLPDU est indiquée par une primitive de confirmation Ph-DATA après transfert d'une primitive de demande Ph-DATA (informations PhICI spécifiant END-OF-DATA-AND-ACTIVITY). Si le temporisateur expire avant qu'une DLPDU ait été reçue, comme indiqué par une primitive d'Indication Ph-DATA (informations PhICI spécifiant START-OF-ACTIVITY), une erreur a eu lieu. De ce fait, une nouvelle tentative ou une nouvelle période de transfert de messages est lancée.

Temporisateur de temps imparti Ce temporisateur surveille les activités du bus dans des stations maître et esclave. Après transfert de la dernière primitive de demande Ph-DATA avec des informations PhICI spécifiant END-OF-DATA-AND-ACTIVITY et retour de la primitive correspondante de confirmation Ph-DATA, ou après réception d'une primitive d'indication Ph-DATA avec des informations PhICI spécifiant END-OF-ACTIVITY ou END-OF-DATA-AND-ACTIVITY, ce temporisateur est chargé avec un multiple de la durée de créneau (voir 5.5.4) et décrémenté à chaque temps binaire tant qu'aucune primitive d'indication Ph-DATA avec des informations PhICI spécifiant START-OF-ACTIVITY n'a été reçue. Si le temporisateur expire, une erreur fatale est survenue, et dans ce cas, la station maître provoque une (ré)initialisation. L'utilisateur DLMS de la station maître ou esclave, respectivement, reçoit une notification d'expiration du délai.

Temporisateur d'intervalle de synchronisation: Les stations maîtres et esclaves utilisent ce temporisateur pour surveiller la présence d'"émetteurs permanents" sur le support de transmission. Après chaque primitive d'indication Ph-DATA avec des informations PhICI spécifiant START-OF-ACTIVITY, ce temporisateur est chargé avec la valeur T_{SYNI} (voir 5.5.4) et décrémenté à chaque temps binaire, tant qu'aucune primitive d'indication Ph-DATA avec des informations PhICI spécifiant END-OF-ACTIVITY ou END-OF-DATA-AND-ACTIVITY, n'a été reçue. Si ce temporisateur expire, une erreur du support de transmission a eu lieu. L'utilisateur DLMS reçoit une notification correspondante.

Temporisateur de mise à jour du GAP Ce temporisateur fonctionne comme décrit en 5.5.5.1.1.

Temporisateur de cycle isochrone: La fonctionnalité de ce temporisateur est définie en 5.5.5.1.1.

Temporisateur de réserve passive Ce temporisateur surveille le temps au repos avant émission d'un message SYNCH. Il doit être réglé en fonction de la différence temporelle entre T_{CT} et la valeur courante du temporisateur de cycle isochrone T_{RCT}, à la fin du dernier message ASP si cette différence est supérieure à 0. La transmission du dernier bit du dernier message ASP au cours d'un cycle isochrone lance ce temporisateur. En mode isochrone, après expiration du temporisateur de réserve passive, un message SYNCH doit être émis et un événement de notification Synch doit être envoyé à l'utilisateur DLMS.

Temporisateur de retard d'émission: Lorsque le maître d'horloge reçoit une primitive de demande DL-CS-TIME-EVENT, de la part de l'utilisateur DLS local, ce temporisateur est lancé avec la valeur = $2 \times T_{CSI}$ et il est décrémenté à chaque temps binaire. Le temporisateur s'arrête après confirmation de l'émission de la dernière portion d'une DLPDU de TE (voir 7.7). La valeur du temps de retard d'émission est calculée comme étant la différence entre la valeur de départ ($2 \times T_{CSI}$) et la valeur relevée du temporisateur de retard d'émission. Le temps de retard d'émission calculé est transmis à l'utilisateur DLS local. En cas d'expiration du temporisateur, l'utilisateur DLS est avisé d'une violation de la séquence de synchronisation d'horloge.

Temporisateur de retard de réception: Lorsque le récepteur d'horloge reçoit le dernier bit d'une DLPDU de TE (voir 7.7) indiqué par une primitive d'indication Ph-DATA (informations PhICI spécifiant END-OF-ACTIVITY ou END-OF-DATA-AND-ACTIVITY), ce temporisateur démarre avec la valeur = $2 \times T_{CSI}$ et il est décrémenté à chaque temps binaire. Ce temporisateur s'arrête après réception du dernier bit des DLPDU d'une DLPDU de CV (voir 7.8) indiqué par une primitive d'indication Ph-DATA (informations PhICI spécifiant END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY). La valeur du temps de retard de réception est calculée comme étant la différence entre la valeur de départ ($2 \times T_{CSI}$) et la valeur relevée du temporisateur de retard de réception. Le temps de retard de réception calculé est transmis à l'utilisateur DLS local. En cas d'expiration du temporisateur, l'utilisateur DLS est avisé d'une violation de séquence de synchronisation d'horloge.

5.5.5.2.2 Compteurs

Les spécifications données en 5.5.5.1 doivent s'appliquer aux compteurs facultatifs.

5.6 Cycle et temps de réaction système

5.6.1 Transmission asynchrone

5.6.1.1 Période de transfert de jeton

La charge de base d'un système comportant plusieurs stations maîtres, c'est-à-dire la charge du bus due aux commandes d'accès au support physique (DLPDU de jeton) et non à des périodes de transfert de messages ordinaires, est déterminée par la période de jeton TP. La charge de base totale par rotation de jeton résulte de *na* (nombre de stations maîtres) cycles de jeton. La période de transfert T_{TP} est constituée du temps de DLPDU de jeton T_{TF}, du temps de retard d'émission T_{TD} et du temps au repos T_{ID1}. T_{ID1} résulte respectivement du temps de retard de station T_{SDI} ou du temps de synchronisation T_{SYN}. T_{TP} est mesuré en bits (voir la Figure 11).

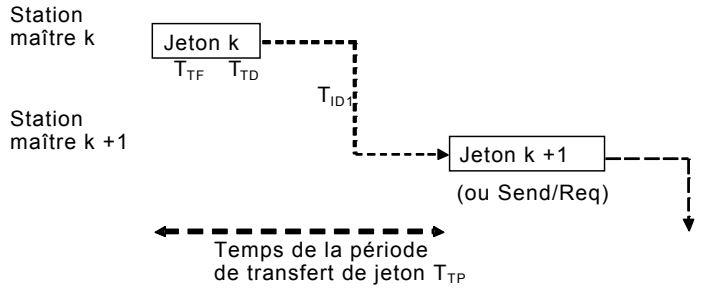


Figure 11 – Période de transfert de jeton

$$T_{TP} = T_{TF} + T_{TD} + T_{ID1} \quad (52)$$

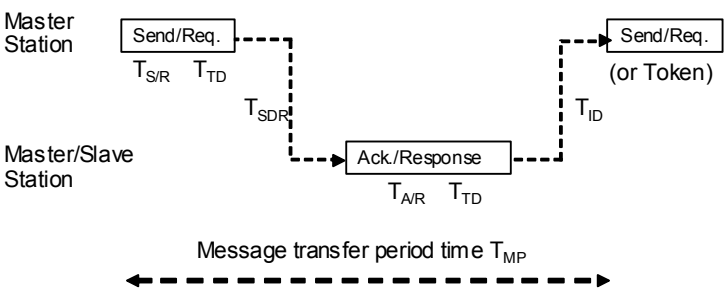
Le temps de DLPDU de jeton T_{TF} est déterminé par le nombre de caractères UART (UC) dans la DLPDU de jeton. Un caractère UART est toujours constitué de 11 bits (voir 6.1.1) et par conséquent la DLPDU de jeton comprend en tout 33 bits. Le temps de retard d'émission T_{TD} dépend de la longueur de ligne (environ 5 ns/m sans répéteur) et il est substantiellement inférieur aux autres temps. Le temps au repos T_{ID1} qui s'écoule entre les DLPDU de jeton, contient le temps de retard de station T_{SDI} du récepteur de jeton d'une part et le temps de synchronisation d'autre part; $T_{SYN} + T_{SM}$ doivent être utilisés si cette somme est supérieure à T_{SDI} (voir 5.5.3). Ceci a généralement lieu à de faibles débits de données (< 100 kbits/s).

5.6.1.2 Période de transfert de message

Une période de transfert de messages MP est constituée de la DLPDU d'envoi/demande et de la DLPDU d'acquittement ou de réponse. La période de transfert est constituée des temps d'émission de DLPDU, des temps de retard d'émission et des temps de retard de station.

Le temps de retard de station T_{SDR} s'écoule entre la demande et l'acquittement ou la réponse. Ce temps est nécessaire au décodage de la demande et à l'assemblage de la DLPDU d'acquittement ou de réponse. Il dépend du protocole mis en œuvre dans la station et il est substantiellement supérieur au temps de retard d'émission T_{TD}. Le temps au repos T_{ID}, qui s'écoule entre l'acquittement ou la réponse et une nouvelle demande, inclut également le temps de retard de station (voir 5.5.3). Cependant, si la somme de $T_{SYN} + T_{SM}$ est supérieure à T_{SDI}, elle doit être utilisée.

La Figure 12 illustre les périodes de transfert de messages.



Légende

Anglais	Français
master station	station maître

Anglais	Français
or token	ou jeton
master/slave station	station maître/esclave
message transfer period time	durée de la période de transfert de message

Figure 12 – Période de transfert de messages

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{ID} + 2 \times T_{TD} \quad (53)$$

Si l'initiateur ne reçoit pas une DLPDU de réponse valide, du fait de perturbations au niveau de la station ou du support, il relancera la DLPDU d'envoi/demande en fonction de sa limite de nouvelle tentative et de l'état de la station destinataire. Pour cette période de nouvelle tentative de transfert de messages (RMP), il est appliqué une autre période de transfert T_{RMP}. T_{RMP} est constitué du temps d'émission de DLPDU d'envoi/demande et de la durée de créneau.

$$T_{RMP} = T_{S/R} + T_{SL} \quad (54)$$

A des débits de données < 100 kbits/s, le décodage et l'évaluation des DLPDU peuvent en partie suivre le rythme de leur réception. Les temps de retard de station deviennent alors beaucoup plus courts.

Les temps d'émission de DLPDU ($T_{S/R}$, $T_{A/R}$) sont déterminés par le nombre de caractères UART (UC). Ils sont calculés de la manière suivante:

$T_{S/R}$ = a × 11 bits, où “a” est le nombre de UC dans une DLPDU d'envoi/demande

$T_{A/R}$ = b × 11 bits, où “b” est le nombre de UC dans une DLPDU d'acquittement/réponse

EXEMPLE a = 6, pour la DLPDU de demande: $T_{S/R}$ = 66 bits; b = 59, pour la DLPDU de réponse: $T_{A/R}$ = 649 bits (50 DATA_UNIT – unités de données – en octets)

5.6.1.3 Temps de réaction système

La fréquence de messages système R_{SYS} est égale au nombre possible de périodes de transfert de messages par seconde:

$$R_{SYS} = 1 / t_{MP}; t_{MP} = T_{MP} \times t_{BIT} \quad (55)$$

Le temps maximal de réaction système T_{SR} dans un système qui comporte **une** station maître et n stations esclaves (système maître-esclave) est calculé à partir de la période de transfert de messages et du nombre de stations esclaves. Si de nouvelles tentatives de transfert de messages sont autorisées, T_{SR} est calculé comme suit:

$$T_{SR} = np \times T_{MP} + mp \times T_{RMP} \quad (56)$$

où

np est le nombre de stations esclaves

mp est le nombre de périodes de nouvelles tentatives de transfert de messages.

T_{RMP} est la période de nouvelle tentative de transfert de messages

Le temps maximal de réaction système dans un système qui comporte **plusieurs** stations maîtres et esclaves est égal au temps de rotation cible:

$T_{SR} = T_{TR}$ (voir 5.3.2.6) (57)

5.6.1.4 Durée de cycle isochrone

La durée de cycle isochrone commence avec l'émission de la DLPDU de SYNCH par le maître IsoM. Le maître IsoM émet tous les messages prioritaires et le nombre configuré de messages non prioritaires. Pour permettre l'utilisation d'un environnement multi-maître, le jeton est passé à la station maître suivante, ce qui laisse ainsi aux autres stations maîtres le temps nécessaire pour entreprendre leurs actions.

$$T_{CT} = T_{SYNCH} + N \times T_{TP} + \sum_{i=1}^{N} P_i + T_{RES} \quad (58)$$

où

N est le nombre de stations maîtres dans l'anneau à jeton

P_i = le temps de DLPDU par station maître I (y compris les DLPDU cycliques du maître IsoM).

La différence positive entre la durée de cycle réelle (T_{RCT}) et la durée de cycle calculée (T_{CT}) représente le temps de réserve T_{RES}. En fonction de la durée disponible, un certain nombre de messages de temps de réserve active (ASM) est envoyé. Il doit être envoyé au moins un message ASM. Après chaque envoi du message ASM, le temps de réserve doit être calculé. L'émission du dernier ASM est suivie par le démarrage du temporisateur de réserve passive si la différence entre la durée de cycle réelle (T_{RCT}) et la durée de cycle calculée (T_{CT}) est supérieure à zéro. L'envoi d'une nouvelle DLPDU de SYNCH commence après expiration du temporisateur de réserve passive. Si la différence entre la durée de cycle réelle (T_{RCT}) et la durée de cycle calculée (T_{CT}) est inférieure ou égale à zéro, l'envoi d'une nouvelle DLPDU de SYNCH commence immédiatement. L'émission d'une nouvelle DLPDU de SYNCH marque le début du cycle suivant. Le démarrage du cycle suivant est notifié à l'utilisateur DLMS par une notification de Synch.

Parallèlement, la valeur du temporisateur de cycle isochrone est relevée et comparée à la durée de cycle calculée. Si le résultat est égal à zéro ou s'il s'inscrit dans le décalage temporel autorisé ($maxT_{SH}$), aucun message n'est envoyé à l'utilisateur DLMS. Si le résultat ne s'inscrit pas dans la plage autorisée de décalage temporel, un événement Synch_Delay comportant la valeur de la différence entre les durées de cycle réelle et calculée, est envoyé à l'utilisateur DLMS.

5.6.2 Transmission synchrone

5.6.2.1 Période de transfert de jeton

Comme pour la transmission asynchrone, les règles suivantes doivent s'appliquer à la période de transfert de jeton T_{TP}:

$$T_{TP} = T_{TF} + T_{TD} \quad (59)$$

Du fait de la structure de la DLPDU (voir 6.1.2) et de la modification du format de la DLPDU (voir 7.4.2), le temps de DLPDU de jeton T_{TF} est de 80 bits avec un préambule de 16 bits et un temps d'intervalle post-émission de 8 bits.

NOTE Seule la vitesse de transmission de 31,25 kbits/s peut être choisie.

5.6.2.2 Période de transfert de message

Les règles suivantes doivent s'appliquer à la période de transfert de messages T_{MP}:

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{PTG} + 2 \times T_{TD} \quad (60)$$

Les spécifications de la période de transfert de messages stipulées en 5.6.1.2 doivent s'appliquer, à l'exception des cas suivants:

Les temps d'émission de PDU ($T_{S/R}$, $T_{A/R}$) sont déterminés par le nombre d'octets PhL. Ils sont calculés de la manière suivante:

$T_{S/R}$ = a, où “a” est le nombre d'octets dans une DLPDU de demande (≤ 255)

$T_{A/R}$ = b, où “b” est le nombre d'octets dans une DLPDU d'acquittement/réponse (≤ 255).

EXEMPLE a = 10, pour la DLPDU de demande: $T_{S/R}$ = 80 bits (outre les informations données en 5.6.1.2: un préambule de 2 octets et un délimiteur de début/fin de PhL de 2 octets); b = 63, pour la DLPDU de réponse: $T_{A/R}$ = 504 bits (outre les informations données en 5.6.1.2: un préambule de 2 octets et un délimiteur de début/fin de PhL de 2 octets)

5.6.2.3 Temps de réaction système

Voir 5.6.1.3.

5.6.2.4 Durée de cycle isochrone

Voir 5.6.1.4.

6 Structure générale et codage des DLPDU et éléments de procédure correspondants

6.1 Granularité des DLPDU

6.1.1 Transmission asynchrone – caractère UART

6.1.1.1 Généralités

Chaque DLPDU doit être constituée d'un certain nombre de caractères UART (UC). Chaque caractère UART est un caractère de départ-arrêt pour la transmission asynchrone, structuré comme illustré en Figure 13.

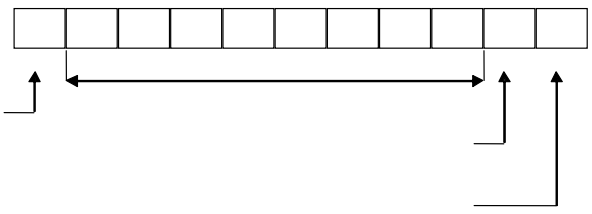


Figure 13 – Caractère UART

La présentation des caractères UART se fonde sur l'ISO/CEI 1177 et sur l'ISO/CEI 2022.

6.1.1.2 Règle d'émission

Chaque caractère UART doit être constitué de 11 bits: un bit de départ (ST) qui doit toujours être un binaire "0", 8 bits d'information (I) qui peuvent être un binaire "0" ou un binaire "1", ou même un bit de parité (P) qui peut être un binaire "0" ou un binaire "1" et un bit d'arrêt (SP) qui doit toujours être un binaire "1".

6.1.1.3 Synchronisation binaire

La synchronisation des bits du récepteur doit toujours commencer par le front descendant du bit de départ, c'est-à-dire au niveau de la transition du binaire "1" au binaire "0". Le bit de départ ainsi que tous les bits consécutifs doivent être explorés au milieu du temps binaire. Le bit de départ doit être un binaire "0" au milieu du bit, dans le cas contraire, on doit considérer qu'il y a échec de la synchronisation et elle doit donc s'arrêter. La synchronisation du caractère UART se termine par le bit d'arrêt qui est un binaire "1". S'il est rencontré un bit de valeur binaire "0" au lieu du bit d'arrêt, on doit supposer qu'il y a eu une erreur de synchronisation ou une erreur de caractère UART et notifier cette erreur; on doit attendre le front montant suivant d'un bit de départ.

On ne doit pas dépasser un écart maximal de ± 0,3 % du débit de données nominal (période binaire) d'émission/réception pour des débits de données inférieurs à 1 500 kbits/s. On ne doit pas dépasser un écart maximal de ± 0,03 % du débit de données nominal (période binaire) pour des débits de données de 1500 kbits/s et plus.

6.1.2 Transmission synchrone

Chaque octet de la DLPDU doit être structuré comme illustré en Figure 14 pour la transmission synchrone.

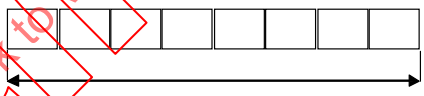
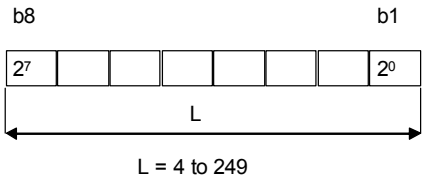


Figure 14 – Structure d'octet

6.2 Octet de longueur (LE, LEr)

Les deux octets de longueur, de valeur identique dans l'en-tête de la DLPDU du format pour une longueur de données variable, doivent contenir le nombre d'octets d'informations dans le corps de la DLPDU. Ces informations sont: DA, SA, FC et DATA_UNIT. La valeur doit couvrir les plages 4 à 249, de sorte qu'il soit possible d'émettre au maximum 246 octets dans une DATA_UNIT de DLPDU (voir 6.6). Une valeur < 4 n'est pas admise, car une DLPDU contient au moins DA, SA, FC et **un** octet de DONNÉES. La DLPDU la plus longue peut contenir au total 255 octets. Le codage de l'octet de longueur est illustré en Figure 15.



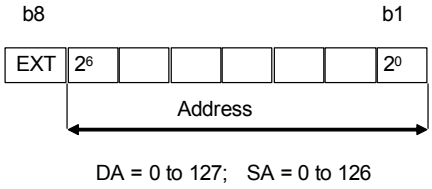
4 to 249 4 à 249

Figure 15 – Codage d'octet de longueur

6.3 Octet d'adresse

6.3.1 Adresses de stations de destination et d'origine (DA et SA)

Les deux octets d'adresse dans l'en-tête de la DLPDU (de demande, d'acquittement et de réponse) doivent contenir l'adresse de la station de destination (DA) et l'adresse de la station d'origine (SA). La DLPDU d'acquittement courte ne contient pas ces deux octets d'adresse. Le codage de l'octet d'adresse est illustré en Figure 16.



address adresse
to à

Figure 16 – Codage d'octet d'adresse

L'adresse 127 (b1 à b7 = 1) est réservée comme adresse globale des messages de diffusion et de multidiffusion obtenus (DLPDU destinées à toutes les stations ou à un groupe de stations sélectionné au moyen d'un point d'accès de service DL; elle ne peut apparaître que dans SDN et CS, ainsi que dans la DLPDU de réponse de MSRD).

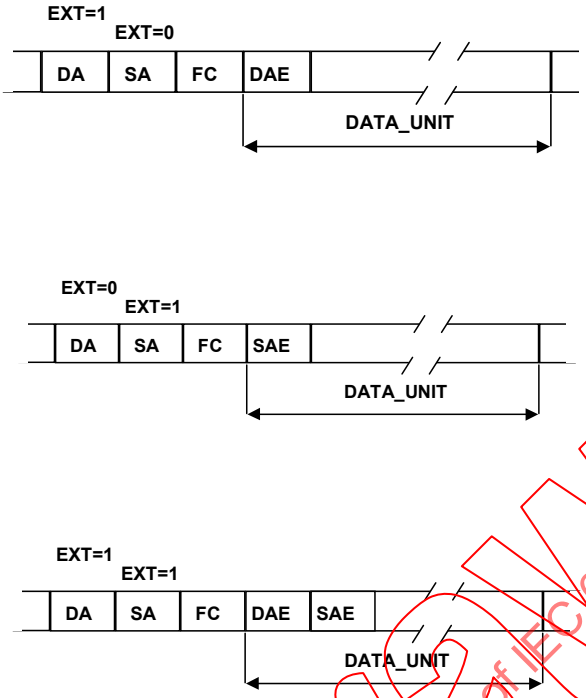
Ainsi, 127 adresses de stations (0 à 126) sont disponibles pour les stations maîtres et esclaves, dont de préférence 32 adresses au maximum peuvent être occupées par des stations maîtres. Pour des applications qui ne sont pas critiques du point de vue temporel, il peut y avoir jusqu'à 127 stations maîtres. Sachant qu'au moins une station maître est requise, au maximum 126 adresses sont admises pour des stations esclaves.

Sauf pour MSRD, les octets d'adresse de DLPDU d'envoi/demande doivent être renvoyés en miroir dans la DLPDU d'acquittement ou de réponse. En d'autres termes, l'adresse SA de la DLPDU d'acquittement ou de réponse doit contenir l'adresse de la station de destination et l'adresse DA doit contenir l'adresse de la station origine de la DLPDU d'envoi/demande. Pour MSRD, l'adresse DA de la DLPDU de réponse doit contenir l'adresse 127 et l'adresse SA de la DLPDU de réponse doit contenir l'adresse de la station de destination de la DLPDU d'envoi/demande.

6.3.2 Extension d'adresse (EXT)

Les extensions d'adresse s'appliquent uniquement aux DLPDU ayant des DATA_UNIT. Le bit EXT (extension) doit indiquer une extension d'adresse de destination et/ou d'origine (DAE, SAE) (voir la Figure 17), qui doit suivre immédiatement l'octet FC dans DATA_UNIT. On peut faire la distinction entre une adresse d'accès (point d'accès de service DL, DLSAP, voir 6.3.4) et l'adresse de région/segment DL. Ces deux types d'adresses peuvent aussi apparaître simultanément, étant donné que chaque extension d'adresse contient de nouveau un bit EXT (voir bit b8 en Figure 18).

Les extensions d'adresse de la DLPDU d'envoi/demande doivent être renvoyées en miroir dans la DLPDU de réponse.

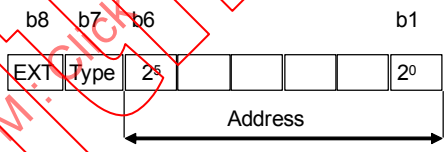


où

EXT = 0: Aucune extension d'adresse dans DATA_UNIT

EXT = 1: L'extension d'adresse suit dans DATA_UNIT.

Figure 17 – Octet DAE/SAE dans la DLPDU



address adresse

où

le bit 8 (EXT) indique une extension d'adresse supplémentaire:

0: Aucun octet d'extension d'adresse supplémentaire

1: Un octet d'extension d'adresse supplémentaire suit immédiatement. On doit se conformer à l'ordre suivant:

Premier octet: adresse de région/segment DL avec b7=1, b8=1

Second octet: DLSAP avec b7=0, b8=0.

b7 représente le type:

0: point d'accès de service DL (DLSAP) sur 6 bits: DAE = 0 à 63; SAE = 0 à 62

1: adresse de région/segment DL sur 6 bits pour réaliser des systèmes hiérarchiques; les valeurs correspondantes ne sont pas spécifiées dans la présente norme.

Figure 18 – Octet d'extension d'adresse

6.3.3 Vérification des adresses

Un récepteur doit vérifier les informations d'adresse de destination dans une DLPDU qu'il s'adresse à lui-même en tant que TS, conformément aux règles suivantes.

a) S'il n'y a pas d'adresse de région/segment DL dans la DLPDU existante (DA[b8=0] ou DAE[b7=0]), il doit uniquement vérifier l'équivalence entre DA et TS.

b) S'il y a une adresse de région/segment DL dans la DLPDU existante (DAE[b7=1]), il doit vérifier l'équivalence de la DAE et de la DA avec l'adresse de région/segment DL et le TS de la station destinataire.

c) Si le récepteur de la station, qui est concerné par l'adresse de région/segment DL, ne contient pas une adresse de région/segment DL, la DLPDU n'est pas destinée à cette station.

6.3.4 Point d'accès de service DL (DLSAP)

A l'interface DLS (voir la CEI 61158-3-3), un (ou plusieurs) service(s) de transmission de données est (sont) traité(s) via un point d'accès de service DL (DLSAP). Il peut y avoir plusieurs DLSAP dans des stations maîtres et esclaves. L'adresse DLSAP correspondante doit être transmise, avec le message, à l'exception des adresses DLSAP de valeur NIL et CS.

Les extensions d'adresse DAE et SAE doivent être utilisées pour la transmission des DLSAP. L'indice de point d'accès de service d'origine (S_SAP_index), qui représente l'adresse d'accès de l'utilisateur local à la DL, doit être transmis dans l'octet SAE. L'indice de point d'accès de service de destination (D_SAP_index), qui représente une ou toutes les adresses d'accès de l'utilisateur distant à la DL, doit être transmis dans l'octet DAE. Il peut être choisi des valeurs 0 à 62 pour S_SAP_index et des valeurs 0 à 63 pour D_SAP_index. La valeur 63 de D_SAP_index représente l'adresse d'accès globale. Cet indice D_SAP_index peut uniquement être utilisé pour le service émission de données sans acquittement.

NOTE Pour des DLPDU disposant d'une fonction = SDA_H/L ou SRD_H/L (voir le Tableau 3), un indice D_SAP_index de 63 dans la DLPDU d'envoi/demande donne lieu à un résultat d'acquittement négatif (RS).

Si l'émission des adresses DLSAP est omise pour des raisons d'efficacité des DLPDU, les services d'émission de données doivent être traités par l'intermédiaire du **DLSAP par défaut** au niveau de l'initiateur ou du répondeur ou des deux à la fois. Si tel est le cas, toutes les DLPDU doivent être émises sans l'extension d'adresse correspondante (SAE ou DAE ou les deux). Au niveau de l'interface DLS, le DLSAP par défaut est obligatoire et il est adressé avec la valeur NIL.

Pour la même raison, la CS d'adresse DLSAP utilisée pour les services de synchronisation d'horloge, valeur d'horloge et événement temporel est omise dans les DLPDU correspondantes.

6.4 Octet de contrôle (FC)

6.4.1 Généralités

L'octet de contrôle dans l'en-tête de la DLPDU doit indiquer le type de DLPDU, comme DLPDU d'envoi/demande et DLPDU d'acquittement ou de réponse. En outre, l'octet de contrôle doit contenir le N° de code de fonction et les informations de commande qui permettent d'éviter la perte et la multiplication des messages, ou le type de station avec l'état DL. Le codage de l'octet FC est illustré en Figure 19 et en Figure 20.

b8	b7	b6	b5	b4	b3	b2	b1
0/1	1	FCB	FCV	2^3	N° de code de fonction		2^0

où

b8 = 0/1, b7 = 1: DLPDU d'envoi/demande

FCB: est le bit de nombre de trames: 0 ou 1, alternativement

FCV: est le bit de validité du bit de nombre de trames :

0: la fonction alternative du FCB n'est pas valide

1: la fonction alternative du FCB est valide

N° de code de fonction: il doit être tel que décrit dans le Tableau 3.

Figure 19 – Codage de l'octet FC pour des DLPDU d'envoi/demande

b8	b7	b6	b5	b4	b3	b2	b1
Res	0	Type de station et état DL		2^3	N° de code de fonction		2^0

où

DLPDU de type b7 = 0: est une DLPDU d'acquittement ou de réponse

b6 et b5 sont le type de station et l'état DL combinés

b6 b5

0 0 Station esclave

0 1 Station maître non prête à entrer dans l'anneau à jeton logique

1 0 Station maître prête à entrer dans l'anneau à jeton logique

1 1 Station maître dans l'anneau à jeton logique

Res: signifie réservé (l'émetteur doit l'établir sur binaire "0"; le récepteur n'a pas à interpréter ce bit)

N° de code de fonction: il doit être tel que décrit dans le Tableau 3.

Figure 20 – Codage de l'octet FC pour des DLPDU d'acquittement ou de réponse

Le Tableau 3 présente le N° de code de fonction de l'octet de contrôle pour les DLPDU d'envoi/demande et les DLPDU d'acquittement ou de réponse.

Tableau 3 – Code de fonction de transmission

Code	Fonction	Format décrit au paragraphe	Possible pour les stations suivantes	
	DLPDU de type b8 = 1 & b7 = 1		Initiateur	Récepteur/ Répondeur
0	Valeur d'horloge	7.3.1 , 7.3.2	M	M et S
1…15	Réservé			
	DLPDU de type b8 = 0 & b7 = 1			
0	Événement temporel	7.1.1 , 7.1.2	M	M et S
1,2	Réservé			
3	Etat bas d'émission de données avec acquittement	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M	M et S
4	Etat bas d'émission de données sans acquittement		M	M et S
5	Etat haut d'émission de données avec acquittement		M	M et S
6	Etat haut d'émission de données sans acquittement		M	M et S
7	Emettre et demander données en multidiffusion	7.2.1 , 7.3.1 , 7.2.2 , 7.3.2	M	M et S
8	Réservé			
9	Demande d'état DL avec réponse	7.1.1 , 7.1.2	M	M et S
10, 11	Réservé			
12	Etat bas d'envoi et de demande de données	7.1.1 , 7.2.1 , 7.3.1	M	M et S
13	Etat haut d'envoi et de demande de données	7.1.2 , 7.2.2 , 7.3.2	M	M et S
14	Demande d'identifiant avec réponse	7.1.1 , 7.1.2	M	M et S
15	Réservé			
	DLPDU de type b7 = 0			
0	Acquittement positif (OK)	7.1.1 , 7.1.2 [a]	M et S	M
1	Acquittement négatif Erreur d'utilisateur (UE)	7.1.1 , 7.1.2	M et S	M
2	Acquittement négatif pas de ressource pour l'envoi de données (& pas de données DL de réponse) (RR)		M et S	M
3	Acquittement négatif aucun service activé (RS)		M et S	M
4 à 7	Réservé		M et S	M
8	Etat bas de données de réponse DL/DLM (& envoi de données Ok) (DL)	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M et S	M
9	Acquittement négatif pas de données de réponse DL/DLM, (& envoi de données Ok) (NR)	7.1.1 , 7.1.2 [a]	M et S	M
10	Etat haut de données de réponse DL (& envoi de données Ok) (DH)	7.2.1 , 7.3.1 7.2.2 , 7.3.2	M et S	M
11	Réservé		M et S	M
12	Etat bas de données de réponse DL, pas de ressource pour l'envoi de données (RDL)	7.2.1 , 7.3.1	M et S	M
13	Etat haut de données de réponse DL, pas de ressource pour l'envoi de données (RDH)	7.2.2 , 7.3.2	M et S	M
14, 15	Réservé			

où M: Maître, S: Esclave

	b4	b1
N° de code de fonction	0:	0 0 0 0
N° de code de fonction	15:	1 1 1 1

( ) Valeur du paramètre d'état L/M des primitives de service (voir la CEI 61158-3-3).

[a] Un acquittement court est également possible, SC = E5H; Exception: pour une demande d'état DL avec réponse, aucun SC n'est admis.

6.4.2 Bit de nombre de trames

Le bit de nombre de trames FCB (b6) permet d'éviter la duplication des messages au niveau du répondeur et la perte de messages au niveau de l'initiateur. Cependant, "Emission de données sans acquittement" ("Send Data with No Acknowledge") (SDN), "Demande d'état DL avec réponse" ("Request DL Status with Reply"), "Demande d'identifiant avec réponse" ("Request Ident with Reply"), “Événement temporel” (“Time Event”) et “Valeur d'horloge” (“Clock Value”) en sont exclus.

Pour gérer la séquence sécuritaire, l'initiateur doit transmettre un FCB pour chaque répondeur. Lorsqu'une DLPDU d'envoi/demande est transmise à un répondeur pour la première fois ou à un répondeur qui est actuellement marqué comme "non opérationnel", le FCB correspondant doit être établi sans aucune ambiguïté. L'initiateur doit réaliser cela par une DLPDU d'envoi/demande avec FCV=0 et FCB=1. Le répondeur doit classer une telle DLPDU comme première période de transfert de messages et stocker en mémoire FCB=1 avec l'adresse de l'initiateur (SA et SAE facultative [b7=1]) (voir le Tableau 4). Cette période de transfert de messages n'est pas répétée par l'initiateur.

Si un répondeur prend en charge l'adressage de région/segment DL et si la DLPDU d'envoi/demande contient une adresse de région/segment DL d'origine (SAE[b7=1]), qui n'est pas égale à sa propre adresse de région/segment DL (DAE[b7=1]), le répondeur doit garder en mémoire la SAE [b7=1] avec la SA.

Dans les DLPDU d'envoi/demande suivantes destinées au même répondeur, l'initiateur doit régler FCV=1 et basculer FCB à chaque nouvelle DLPDU d'envoi/demande. Le répondeur doit évaluer le FCB lorsqu'il reçoit une DLPDU d'envoi/demande adressée à lui-même avec FCV=1. Un FCB modifié en comparaison à la DLPDU d'envoi/demande précédente du même initiateur (même SA et même SAE facultative [b7=1]) doit être considéré comme une confirmation de l'achèvement correct de la période de transfert de message précédente. Si la DLPDU d'envoi/demande provient d'un initiateur différent (SA différente ou SAE facultative différente [b7=1]), il ne doit y avoir aucune évaluation du FCB. Dans les deux cas, le répondeur doit mettre en mémoire le FCB avec l'adresse d'origine (SA et SAE facultative [b7=1]) jusqu'à ce qu'il reçoive une nouvelle DLPDU adressée à lui-même.

Si une DLPDU d'acquittement ou de réponse est manquante ou altérée, le FCB **ne doit pas** être modifié par l'initiateur lors de la nouvelle tentative; ceci indique l'échec de la période de transfert de message précédente. Si un répondeur reçoit une DLPDU d'envoi/demande avec FCV=1 et le même FCB que dans la DLPDU d'envoi/demande immédiatement précédente du même initiateur (même SA et même SAE facultative [b7=1]), une nouvelle tentative doit être lancée. En conséquence, le répondeur doit réémettre la DLPDU d'acquittement ou de réponse qui était prête.

Le répondeur doit garder prête à l'envoi la DLPDU d'acquittement ou de réponse pour une éventuelle nouvelle tentative, jusqu'à ce qu'il détecte une confirmation de l'achèvement correct de la période de transfert de message précédente, comme décrit ci-dessus; ou jusqu'à réception d'une DLPDU de jeton ou d'une DLPDU dont l'adresse a été modifiée (SA ou DA ou SAE facultative [b7=1] ou DAE facultative [b7=1]).

Pour "Emission de données sans acquittement", "Demande d'état DL avec réponse", "Demande d'identifiant avec réponse", “Événement temporel” et “Valeur d'horloge”, les bits FCV et FCB sont tous les deux à zéro pour les types de DLPDU suivants; il n'est pas nécessaire que le répondeur analyse le FCB:

Tableau 4 – FCB et FCV dans le répondeur

b6	b5	←- position du bit		
FCB	FCV	Condition	Signification	Action
0	0	DA = TS/127	Demande sans acq Demande d'état DL avec réponse Demande d'identifiant avec réponse Événement temporel Événement d'horloge	La dernière DLPDU d'acquittement ou de réponse peut être supprimée
0/1	0/1	DA ≠ TS	Demande à un autre répondeur	La dernière DLPDU d'acquittement ou de réponse peut être supprimée
1	0	DA = TS	Première demande	FCBM := 1 SAM := SA La dernière DLPDU d'acquittement ou de réponse peut être supprimée
0/1	1	DA = TS SA = SAM FCB ≠ FCBM	Nouvelle demande	Supprimer dernière DLPDU d'acquittement ou de réponse FCBM := FCB Avoir une DLPDU d'acquittement ou de réponse prête pour une nouvelle tentative
0/1	1	DA = TS SA = SAM FCB = FCBM	Nouvelle tentative de demande	FCBM := FCB Répéter DLPDU d'acquittement ou de réponse et la garder prête
0/1	1	DA = TS SA ≠ SAM	Nouvel initiateur	FCBM := FCB SAM := SA Avoir une DLPDU d'acquittement ou de réponse prête pour une nouvelle tentative
—	—	DLPDU de jeton	—	La dernière DLPDU d'acquittement ou de réponse peut être supprimée
NOTE FCBM est le FCB enregistré en mémoire et SAM est la SA enregistrée en mémoire.				

6.5 Détection d'erreur de contenu de DLPDU

6.5.1 Transmission asynchrone – somme de contrôle de trame (FCS)

La somme de contrôle de trame de 8 bits (1 octet) nécessaire pour la distance de Hamming 4 dans une DLPDU doit toujours précéder immédiatement le délimiteur de fin. La somme de contrôle doit être structurée comme illustré en Figure 21.

Figure 21 – Codage d'octet FCS

Dans des DLPDU de longueur fixe, sans champ de données (voir 7.1.1) la somme de contrôle doit toujours être calculée à partir de la somme arithmétique en complément à deux de DA, SA et FC (sans délimiteurs de début et de fin).

Dans des DLPDU de longueur fixe avec champ de données (voir 7.2.1) et dans des DLPDU ayant une longueur variable de champ de données (voir 7.3.1) la somme de contrôle doit en outre inclure DATA_UNIT.

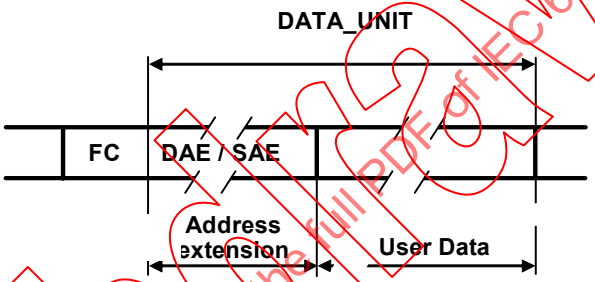
6.5.2 Transmission synchrone -Séquence de contrôle de trame (FCS)

Une séquence de contrôle de trame de 16 bits (2 octets) est exigée. Elle doit être calculée et annexée à la DLPDU, comme spécifié à l'Article 5, qui décrit également les caractéristiques de la distance de Hamming.

6.6 DATA_UNIT (UNITE DE DONNEES)

6.6.1 Généralités

DATA_UNIT doit être constituée des données utilisateur (DLSDU) de l'utilisateur DLS/DLMS et, de manière facultative, d'un ou de plusieurs octets d'extension d'adresse (voir la Figure 22). Il est admis d'utiliser jusqu'à 4 octets d'extension d'adresse (voir 6.3.1). Les données utilisateur contiennent au maximum 242 à 246 octets en fonction du nombre d'octets d'extension d'adresse utilisés.



Légende

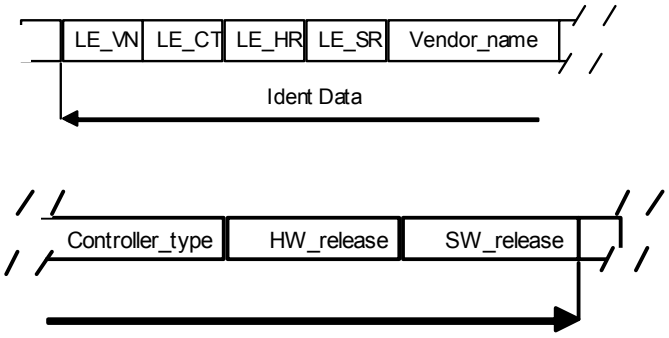
Anglais	Français
Address extension	Extension d'adresse
User data	Données utilisateur

Figure 22 – Champ de données

Les données utilisateur suivantes sont définies pour la DLPDU d'identification:

6.6.2 Données utilisateur d'identification

Comme illustré en Figure 23, les données utilisateur d'identification doivent contenir la Ident_List (liste d'identifiants) des stations avec le nom du fournisseur (Vendor_name), le type de contrôleur de bus de terrain (controller_type) ainsi que la version du matériel et du logiciel (HW/SW_release). Elles sont constituées de 200 octets au maximum:



où

LE_VN, LE_CT, LE_HR, LE_SR représentent dans chaque cas la longueur du champ de données correspondant en octets (1 octet chacun, double codage avec la signification donnée en Figure 21)

Vendor_name (VN) est le nom du fabricant, en chaîne de caractères ASCII (code ISO 7 bits, b8=0).

Controller_type (CT) est le type de contrôleur matériel, en chaîne de caractères ASCII (code ISO 7 bits, b8=0).

HW_release (HR) est la version matérielle du contrôleur, en chaîne de caractères ASCII (code ISO 7 bits, b8=0).

SW_release (SR) est la version logicielle du contrôleur, en chaîne de caractères ASCII (code ISO 7 bits, b8=0).

Figure 23 – Données utilisateur d'identification

6.7 Procédures de contrôle d'erreurs

6.7.1 Transmission asynchrone

Les erreurs de protocole de la ligne, par exemple les erreurs de structure de trame de caractère, les erreurs de dépassement et les erreurs de parité, ainsi que les erreurs de protocole de transmission, par exemple des délimiteurs de début, des octets de vérification de trame et des délimiteurs de fin dégradés, une longueur de DLPDU et des temps de réponse invalides, etc., doivent générer les réactions suivantes de la station:

Une réception incorrecte de DLPDU d'envoi/demande ou de DLPDU de jeton par une station donnée ne doit pas être traitée, acquittée ou faire l'objet d'une réponse. L'initiateur doit relancer la demande après expiration de la durée de créneau. Une nouvelle tentative de demande doit également être lancée si l'acquittement ou la réponse ont été altérés. L'initiateur doit compléter une demande uniquement après avoir reçu une réponse valide ou si la (les) nouvelle(s) tentatives) n'a pas (n'ont pas) été couronnées de succès (voir Tableau 5). Ceci signifie qu'un "envoi/demande" doit être conservé jusqu'à ce que le répondeur confirme sa réception correcte par un acquittement ou une réponse ou si la ou les nouvelle(s) tentative(s) n'a pas été (n'ont pas été) couronnées de succès. De la même manière, un répondeur doit terminer une "demande" ou un "envoi/demande" uniquement si une nouvelle demande, avec un bit de nombre de trames altéré est reçue, ou si elle est adressée à une autre station (voir 6.4, FCB).

Si une station n'acquitte pas ou ne répond pas après une ou plusieurs nouvelles tentatives, elle doit être indiquée comme "non opérationnelle". L'initiateur doit émettre la demande à cette station sans nouvelle tentative, jusqu'à ce que la station envoie un acquittement correct ou une réponse correcte lors du traitement des demandes suivantes. Après acquittement positif, l'initiateur doit de nouveau marquer la station concernée comme "opérationnelle". Lors du traitement de la réponse suivante, l'initiateur doit poursuivre le mode de fonctionnement initial avec cette station.

6.7.2 Transmission synchrone

En cas d'erreurs dans le protocole de la ligne (voir Article 9 de la CEI 61158-2) et dans le protocole d'accès au support, comme par exemple des octets de départ et des octets de FCS

erronés, des longueurs de DLPDU et des temps de réponse, etc., incorrects, la station doit réagir comme spécifié en 6.7.1.

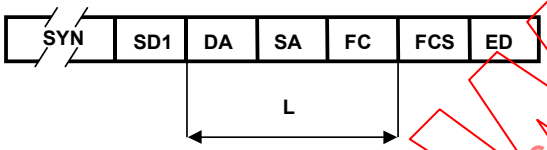
7 Structure, codage et éléments de procédure spécifiques aux DLPDU

7.1 DLPDU de longueur fixe sans aucun champ de données

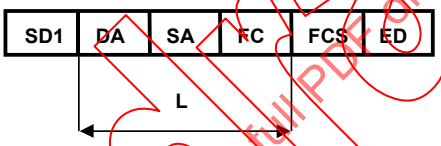
7.1.1 Transmission asynchrone

Les formats de DLPDU de longueur fixe sans champ de données doivent être tels qu'illustrés en Figure 24.

a) Format de la DLPDU de demande:



b) Format de la DLPDU d'acquittement:



c) Format de la DLPDU d'acquittement court:

où

SYN est la période de synchronisation, d'au minimum 33 bits de repos de la ligne
SD1 est le délimiteur de début, de valeur: 10 h
DA est l'adresse de destination
SA est l'adresse d'origine
FC signifie contrôle de trame
FCS est la somme de contrôle de trame
ED est le délimiteur de fin, de valeur: 16 h
L représente la longueur du champ d'information, définie par un nombre fixe d'octets: L = 3
SC est le caractère unique, de valeur: E5H.

Figure 24 – DLPDU de longueur fixe sans aucun champ de données

Règles de transmission

1) L'état au repos de la ligne doit correspondre à un niveau du signal binaire "1".
2) Chaque DLPDU de demande doit être précédée d'au moins 33 bits de repos de ligne (temps de synchronisation).
3) Aucun état au repos n'est autorisé entre des caractères UART de DLPDU.
4) Le récepteur doit vérifier:
   — pour chaque caractère UART: le bit de départ, le bit d'arrêt et le bit de parité (paire),
   — pour chaque DLPDU: le délimiteur de début, DA, SA, FCS et le délimiteur de fin,
   — et le temps de synchronisation dans le cas d'une DLPDU de demande.

En cas d'échec de la vérification, l'ensemble de la DLPDU doit être ignoré.

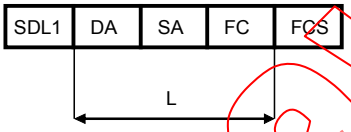
SC et SD1 (ainsi que SD2 et SD3, voir 7.2.1 et 7.3.1) ont des distances de Hamming Hd=4 et sont immunisés contre tout décalage (voir par exemple la CEI 60870-5-1), ce qui signifie que le caractère unique SC apparaît comme étant une DLPDU de Hd=4.

Pour des demandes à acquitter (émission de données avec acquittement), seul SC est un acquittement positif admissible. Pour des demandes qui nécessitent une réponse (émission et demande de données avec réponse), SC est admissible si aucune données n'est disponible (voir Tableau 3, b7=0, Code-No 9).

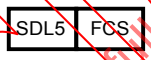
7.1.2 Transmission synchrone

Les formats de DLPDU de longueur fixe sans aucun champ de données doivent être tels qu'illustrés en Figure 25.

a) Format de la DLPDU de demande et format de la DLPDU d'acquittement:



b) Format de la DLPDU d'acquittement court:



où

SDL1 est l'octet de départ 1 (délimiteur de début 1 de la liaison de données), de code: 10H
SDL5 est l'octet de départ 5 (délimiteur de début 5 de la liaison de données), de code: E5H
DA est l'adresse de destination
SA est l'adresse d'origine
FC signifie contrôle de trame
FCS est la séquence de contrôle de trame, sur 2 octets
L représente la longueur du champ d'information, définie par un nombre fixe d'octets: L = 3.

Figure 25 – DLPDU de longueur fixe sans champ de données

Règles de transmission

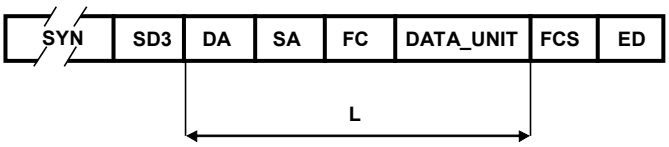
Outre les règles de transmission de la couche physique données dans l'Article 9 de la CEI 61158-2, le récepteur doit vérifier les octets SDL, DA/SA et FCS pour chaque DLPDU. En cas d'échec de la vérification, l'ensemble de la DLPDU doit être ignoré.

7.2 DLPDU de longueur fixe avec champ de données

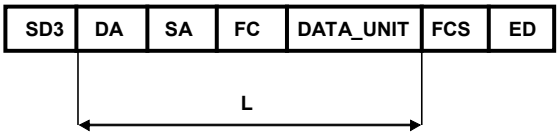
7.2.1 Transmission asynchrone

Le format de DLPDU de longueur fixe avec champ de données doit être tel qu'illustré en Figure 26.

a) Format de la DLPDU d'émission/demande:



b) Format de la DLPDU de réponse:



où

SYN est la période de synchronisation, d'au minimum 33 bits de repos de la ligne
SD3 est le délimiteur de début, de valeur: A2H
DA est l'adresse de destination
SA est l'adresse d'origine
FC signifie contrôle de trame
DATA_UNIT (unité de données) est le champ de données, de longueur fixe (L-3) = 8 octets
FCS est la somme de contrôle de trame
ED est le délimiteur de fin, de valeur: 16H

L représente la longueur du champ d'information, définie par un nombre fixe d'octets: L = 11.

Figure 26 – DLPDU de longueur fixe avec champ de données

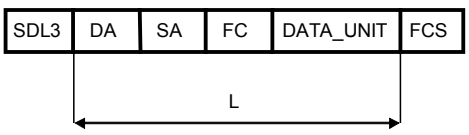
Règles de transmission

Les règles de transmission qui s'appliquent doivent être les mêmes que celles des DLPDU de longueur fixe sans champ de données (voir 7.1.1).

7.2.2 Transmission synchrone

Les formats de DLPDU de longueur fixe avec champ de données doivent être tels qu'illustrés en Figure 27.

a) Format des DLPDU d'émission/demande et de réponse :



où

SDL3 est l'octet de départ 3 de la liaison de données, de code: A2H
DA est l'adresse de destination
SA est l'adresse d'origine
FC signifie contrôle de trame
DATA_UNIT (unité de données) est le champ de données, de longueur fixe (L-3) = 8 octets
FCS est la séquence de contrôle de trame, sur 2 octets

L représente la longueur du champ d'information, définie par un nombre fixe d'octets: L = 11.

Figure 27 – DLPDU de longueur fixe avec champ de données

Règles de transmission

Les règles de transmission applicables doivent être les mêmes que celles des DLPDU de longueur fixe sans champ de données (voir 7.1.2).

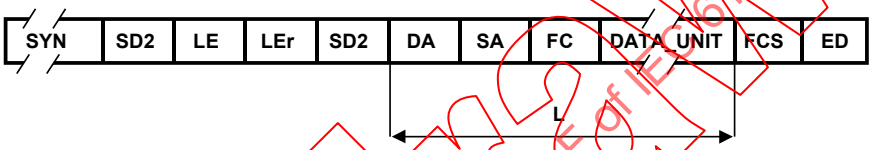
7.3 DLPDU avec longueur variable de champ de données

7.3.1 Transmission asynchrone

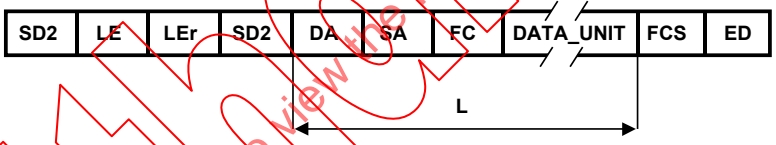
Pour un nombre variable d'octets de données, l'information de longueur doit également être transmise dans la DLPDU. Cette information de longueur doit être contenue deux fois dans un en-tête de DLPDU fixe au début de la DLPDU. Elle est ainsi protégée par une Hd = 4 et immunisée contre tout décalage.

Le format des DLPDU de longueur de champ de données variable doit être tel qu'illustré en Figure 28.

a) Format de la DLPDU d'émission/demande:



b) Format de la DLPDU de réponse:



où

SYN est la période de synchronisation, d'au minimum 33 bits de repos de la ligne
SD2 est le délimiteur de début, de valeur: 68H
LE est la longueur d'octet, de valeur autorisée: 4 à 249
LEr est la longueur de l'octet répétée
DA est l'adresse de destination
SA est l'adresse d'origine
FC signifie contrôle de trame
DATA_UNIT (unité de données) est le champ de données, de longueur variable (L-3), d'au maximum 246 octets
FCS est la somme de contrôle de trame
ED est le délimiteur de fin, de valeur: 16H

L représente la longueur du champ d'information, d'un nombre variable d'octets: L = 4 à 249.

Figure 28 – DLPDU à longueur variable de champ de données

Règles de transmission

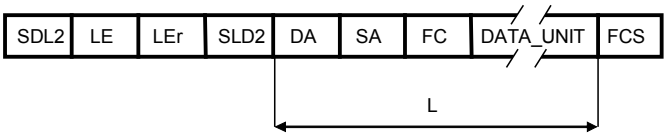
Les règles de transmission applicables doivent être les mêmes que celles des DLPDU de longueur fixe sans champ de données (voir 7.1.1).

Outre la règle d'émission 4, la LE doit être identique à LEr et les octets d'information doivent être comptés à partir de l'adresse de destination (DA) jusqu'à la somme de contrôle de trame (FCS) et le résultat doit être comparé à LE.

7.3.2 Transmission synchrone

Les formats des DLPDU de longueur de champ de données variable doivent être tels qu'illustrés en Figure 29.

a) Format des DLPDU d'émission/demande et de réponse :



où

SDL2	est l'octet de départ 2 de la liaison de données, de code: 68H
LE	est la longueur, de valeur: 4 à 249
LEr	est la longueur (répétée)
DA	est l'adresse de destination
SA	est l'adresse d'origine
FC	signifie contrôle de trame
DATA_UNIT	(unité de données) est le champ de données, de longueur variable (L-3), d'au maximum 246 octets
FCS	est la séquence de contrôle de trame, sur 2 octets

L représente la longueur du champ d'information, d'un nombre variable d'octets: L = 4 à 249.

Figure 29 – DLPDU à longueur variable de champ de données

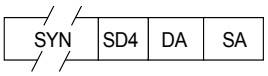
Règles de transmission

Les règles de transmission applicables doivent être les mêmes que celles des DLPDU de longueur fixe sans champ de données (voir 7.1.2). En outre, le récepteur doit vérifier que LE et LEr coïncident. Les octets d'information doivent être comptés à partir de l'adresse de destination (DA) jusqu'à la séquence de contrôle de trame (FCS) et doivent être comparés à LE.

7.4 DLPDU de jeton

7.4.1 Transmission asynchrone

Le format de la DLPDU de jeton doit être tel qu'illustré dans la Figure 30.



où

SYN	est la période de synchronisation, d'au minimum 33 bits de repos de la ligne
SD4	est le délimiteur de début, de valeur: DCH
DA	est l'adresse de destination
SA	est l'adresse d'origine.

Figure 30 – DLPDU de jeton

Règles de transmission

1) L'état au repos de la ligne doit correspondre à un niveau du signal binaire "1".
2) Chaque DLPDU de jeton doit être précédée d'au moins 33 bits de repos de ligne (temps de synchronisation).
3) Aucun état au repos n'est permis entre des caractères UART de DLPDU.

4) Le récepteur doit vérifier:

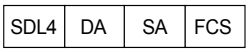
— pour chaque caractère UART: le bit de départ, le bit d'arrêt et le bit de parité (paire),

— pour chaque DLPDU: le temps de synchronisation, le délimiteur de début et les adresses DA/SA.

Si le résultat de la vérification est négatif, l'ensemble de la DLPDU doit être ignoré.

7.4.2 Transmission synchrone

Le format de la DLPDU de jeton doit être tel qu'illustré dans la Figure 31.



où

SDL4	est l'octet de départ 4 de la liaison de données, de code: DCH
DA	est l'adresse de destination
SA	est l'adresse d'origine
FCS	est la séquence de contrôle de trame, sur 2 octets.

Figure 31 – DLPDU de jeton

Règles de transmission

Les règles de transmission applicables doivent être les mêmes que celles des DLPDU de longueur fixe sans aucune donnée (voir 7.1.2).

7.5 DLPDU d'ASP

La DLPDU d'ASP est générée par la DLE en mode isochrone, pour obtenir une activité de bus au cours du temps de réserve d'un cycle isochrone et éviter ainsi l'expiration du délai (T_{TO}). Le format de la DLPDU d'ASP est similaire à celui décrit en 7.1, avec les restrictions suivantes:

— L'octet de contrôle (FC) est égal à la demande d'état DL avec réponse (Code 49H)

— DA est égale à SA

— il n'y a aucune DLPDU de réponse émise par une quelconque DLE pour cette DLPDU d'ASP.

7.6 DLPDU de SYNCH

La DLPDU de SYNCH est générée par la DLE en mode isochrone afin de marquer le début d'un nouveau cycle isochrone. Le format de la DLPDU de SYNCH est similaire à celui décrit en 7.3, avec les restrictions suivantes:

— FC, l'octet de contrôle, est égal à l'état haut d'émission de données sans acquittement (Code 46H)

— DA est égale à 127

— SAE est égale à 62

— DAE est égale à 58

— DATA_UNIT doit contenir la valeur du paramètre de fonctionnement SYNCHT

— il n'y a aucune DLPDU de réponse émise par une quelconque DLE pour cette DLPDU de SYNCH.

7.7 DLPDU d'événement temporel (TE)

La DLPDU de TE est utilisée pour synchroniser l'horloge au niveau du récepteur d'horloge. Le format de la DLPDU de TE est similaire à celui décrit en 7.1, avec les restrictions suivantes:

— FC, l'octet de contrôle, est égal à l'événement temporel (Code 40H)

— DA est égale à 127

— il n'y a aucune DLPDU de réponse émise par une quelconque DLE pour cette DLPDU de TE.

7.8 DLPDU de valeur d'horloge (CV)

La DLPDU de CV est utilisée pour transmettre la valeur de l'horloge aux récepteurs d'horloge. Le format de la DLPDU de CV est similaire à celui décrit en 7.3, avec les restrictions suivantes:

— FC, l'octet de contrôle, est égal à la valeur d'horloge (Code C0H)

— DA est égale à 127

— SAE et DAE ne sont pas utilisées

— il n'y a aucune DLPDU de réponse émise par une quelconque DLE pour cette DLPDU de CV.

7.9 Procédures de transmission

7.9.1 Transmission asynchrone

Les séquences de DLPDU admissibles (périodes de transfert de messages) en transmission asynchrone sont décrites de la Figure 32 à la Figure 34. Les séquences d'erreur et les messages de diffusion/multidiffusion sont exclus de cette description.

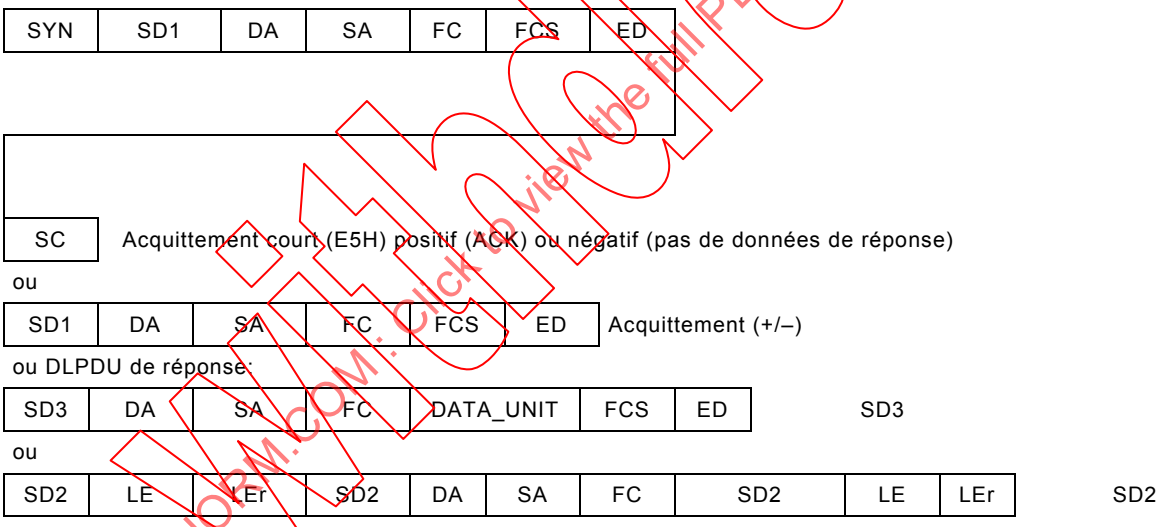


Figure 32 – DLPDU d'envoi/demande de longueur fixe sans données

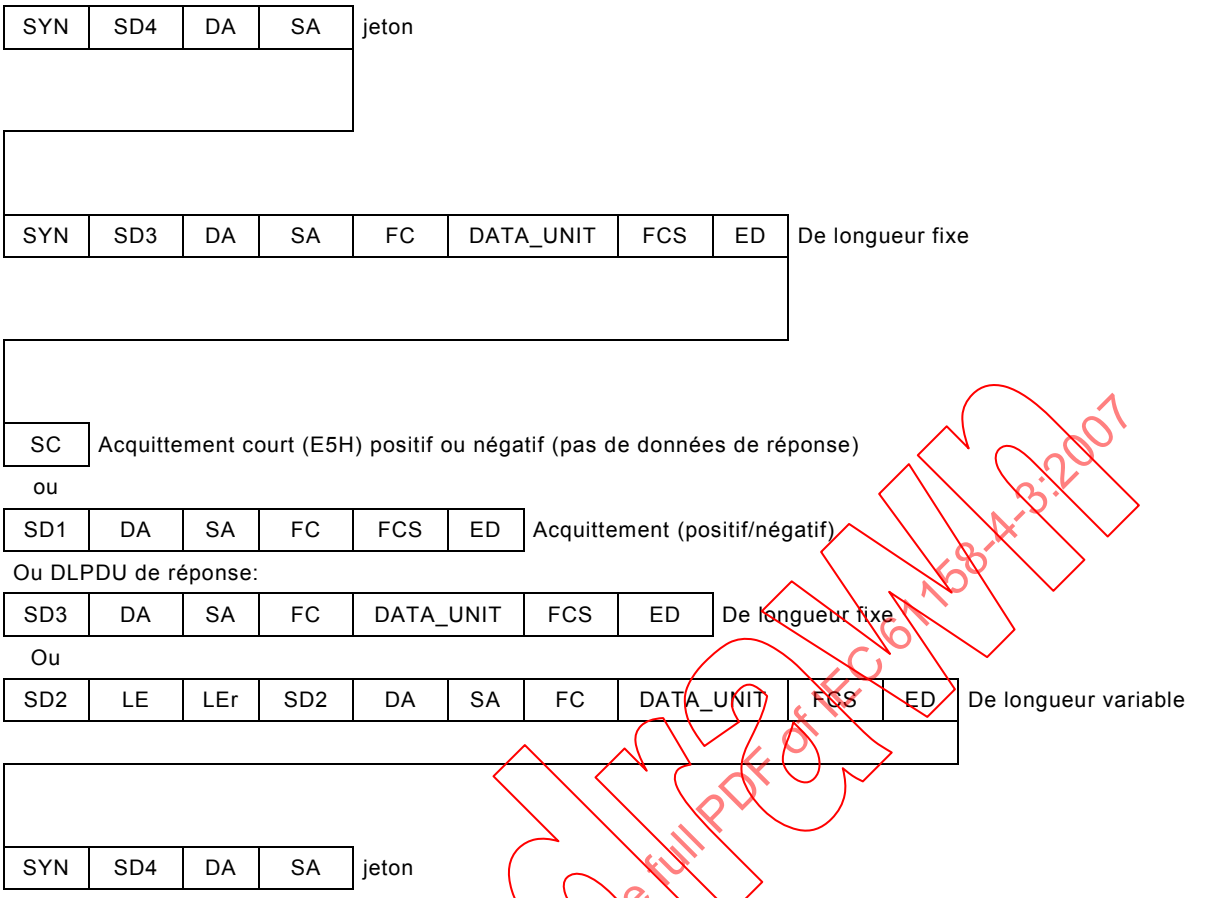


Figure 33 – DLPDU de jeton et DLPDU d'envoi/demande de longueur fixe avec données

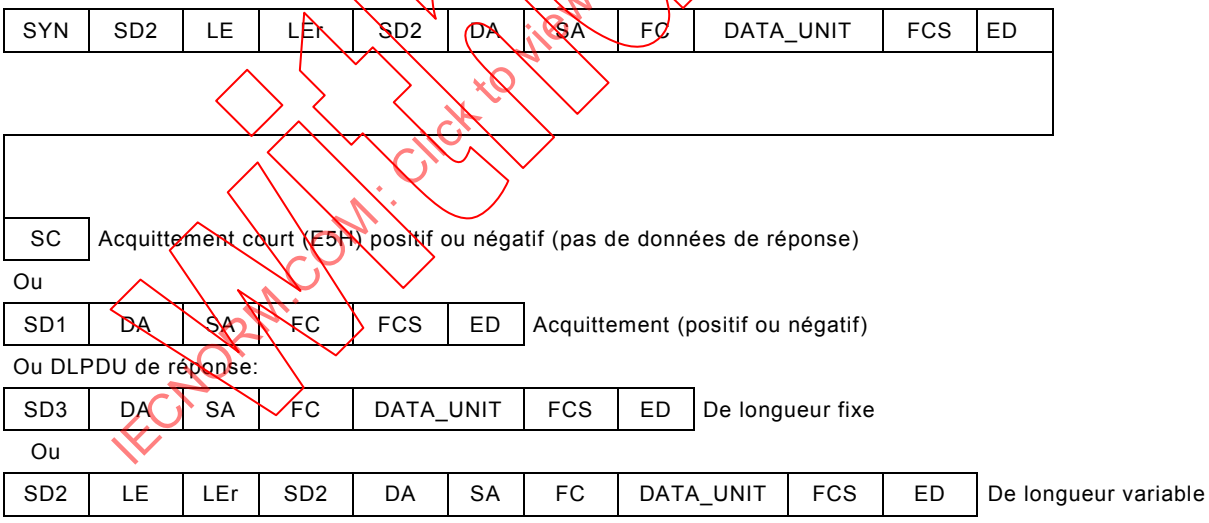


Figure 34 – DLPDU d'envoi/demande avec longueur variable du champ de données

7.9.2 Transmission synchrone

Les séquences de DLPDU admissibles (périodes de transfert de messages) en transmission synchrone sont décrites de la Figure 35 à la Figure 37. Les séquences d'erreur et les messages de diffusion/multidiffusion sont exclus de cette description.

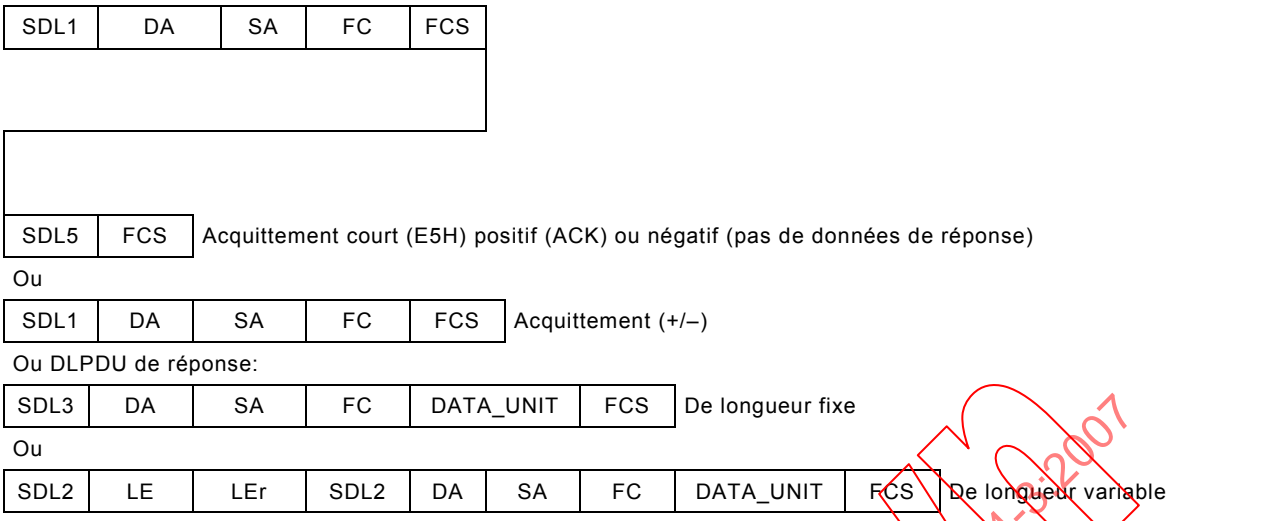


Figure 35 – DLPDU d'envoi/demande de longueur fixe sans données

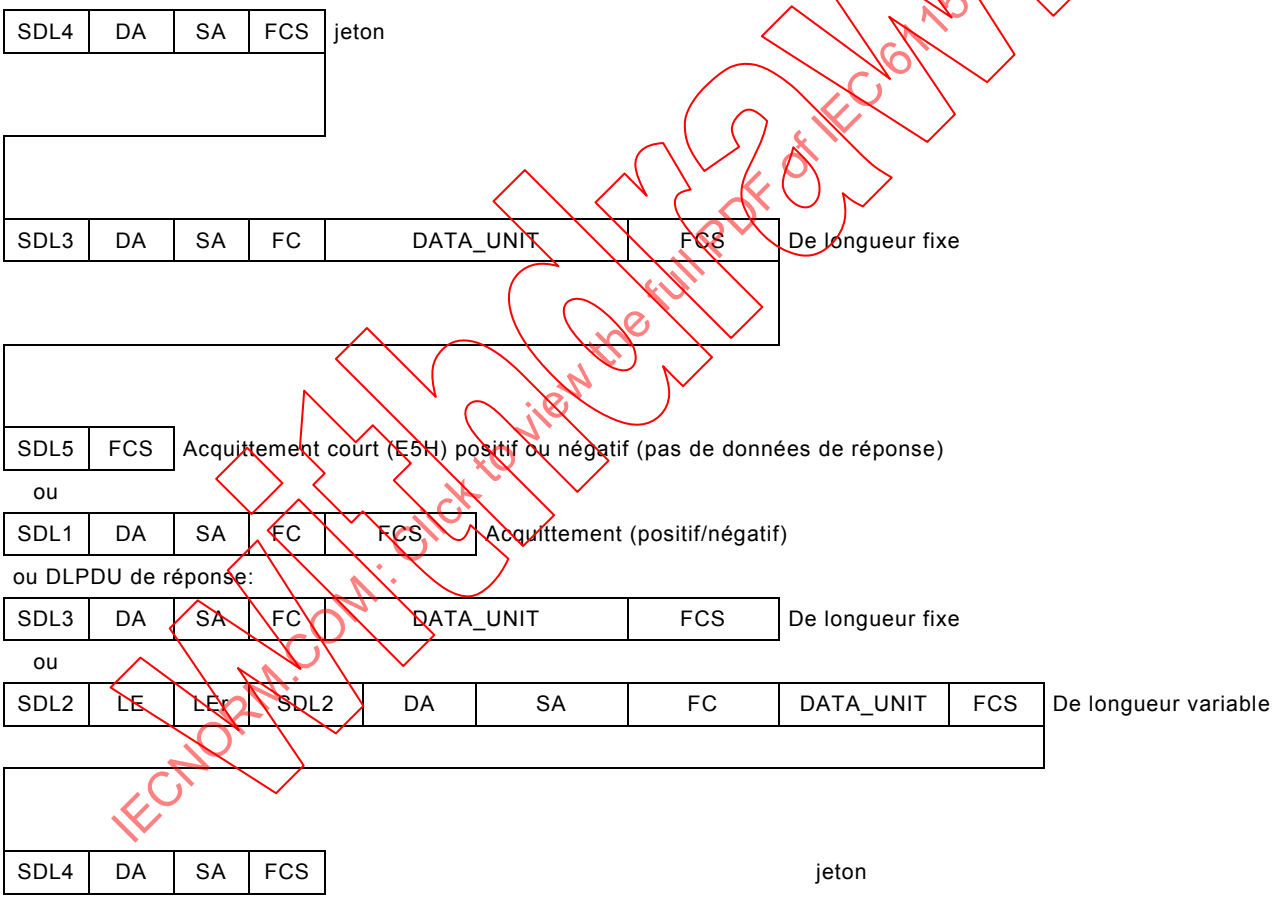


Figure 36 – DLPDU de jeton et DLPDU d'envoi/demande de longueur fixe avec données

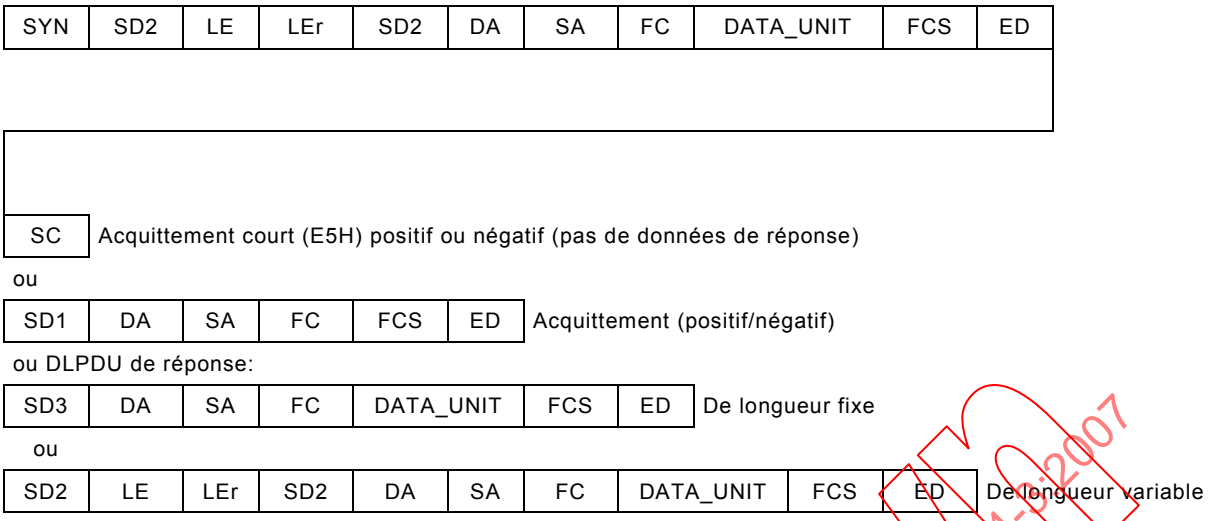


Figure 37 – DLPDU d'envoi/demande avec longueur variable du champ de données

8 Autres éléments de procédure de DLE

NOTE L'Annexe A décrit un certain nombre de diagrammes d'états finis utilisés par la DLE pour fournir ses fonctions de protocole de niveau bas et de niveau haut. La spécification de l' Annexe A complète la spécification textuelle du présent exposé et des Articles correspondants dans le corps de la présente norme. En cas de divergence, les exigences de l' Annexe A prévalent.

8.1 Initialisation d'une entité DL

Après mise sous tension (PON) l'entité DL de stations maîtres et esclaves doit passer à l'état "Offline" ("Hors ligne"). Lorsqu'elle est dans cet état, l'entité DL ne reçoit ni n'émet de signaux (DLPDU) du bus.

L'entité DL peut passer à l'état "Passive_Idle" ou "Listen_Token" uniquement à partir de l'état " Offline ", si les paramètres de fonctionnement (variables DL) ont été réglés de manière à traiter le protocole correct (voir CEI 61158-3-3). Les paramètres de fonctionnement présentés dans le Tableau 5, doivent être fournis par l'utilisateur DLMS.

Tableau 5 – Paramètres de fonctionnement

Numéro du paramètre	Nom
1	Adresse DL de station TS
2	Débit de données (kbits/s)
3	Supports individuels/redondants disponibles
4	Version de matériel
5	Version de logiciel
6	Durée de créneau T_{SL}
7	Temps de retard de station T_{SDR} min
8 (voir Note 1)	Temps de retard de station TSDR max
9 (voir Note 1)	Temps de décroissance émetteur/Temps de commutation répéteur T_{QUI}
10 (voir Note 1)	Temps de montée T_{SET}
11 (voir Note 1)	Temps de rotation cible T_{TR}
12 (voir Note 1)	Facteur de mise à jour d'intervalle G
13 (voir Note 1)	Entrée/sortie de la station maître de l'anneau logique (in_ring_desired)

Numéro du paramètre	Nom
14 (voir Note 1)	Adresse de station la plus élevée (HSA)
15 (voir Note 1)	Nombre maximal de nouvelles tentatives (max_retry_limit)
16 (voir Note 2)	Temps d'intervalle de synchronisation d'horloge T_{CSI}
17 (voir Note 3)	Contenu des données utilisateur SYNCH (SYNCHT)
18 (voir Note 3)	Durée de cycle isochrone T_{CT}
19 (voir Note 3)	Décalage temporel maximal admissible T_{TS}

NOTE 1 Ceci s'applique uniquement aux stations maîtres.

NOTE 2 Ceci s'applique uniquement aux stations capables de prendre en charge la synchronisation d'horloge.

NOTE 3 Ceci s'applique uniquement aux stations capables de fonctionner en mode isochrone.

8.2 États du contrôle d'accès au support physique de l'entité DL

8.2.1 Généralités

Un diagramme d'états de contrôle d'accès au support (MAC) d'une station maître est décrit par 11 états et par les transitions entre ces états. Le MAC d'une station esclave peut prendre 2 états.

La Figure 38 illustre un aperçu général du diagramme d'états combiné de la station maître (états 0 à 9 et 11) et de la station esclave (états 0 et 10).

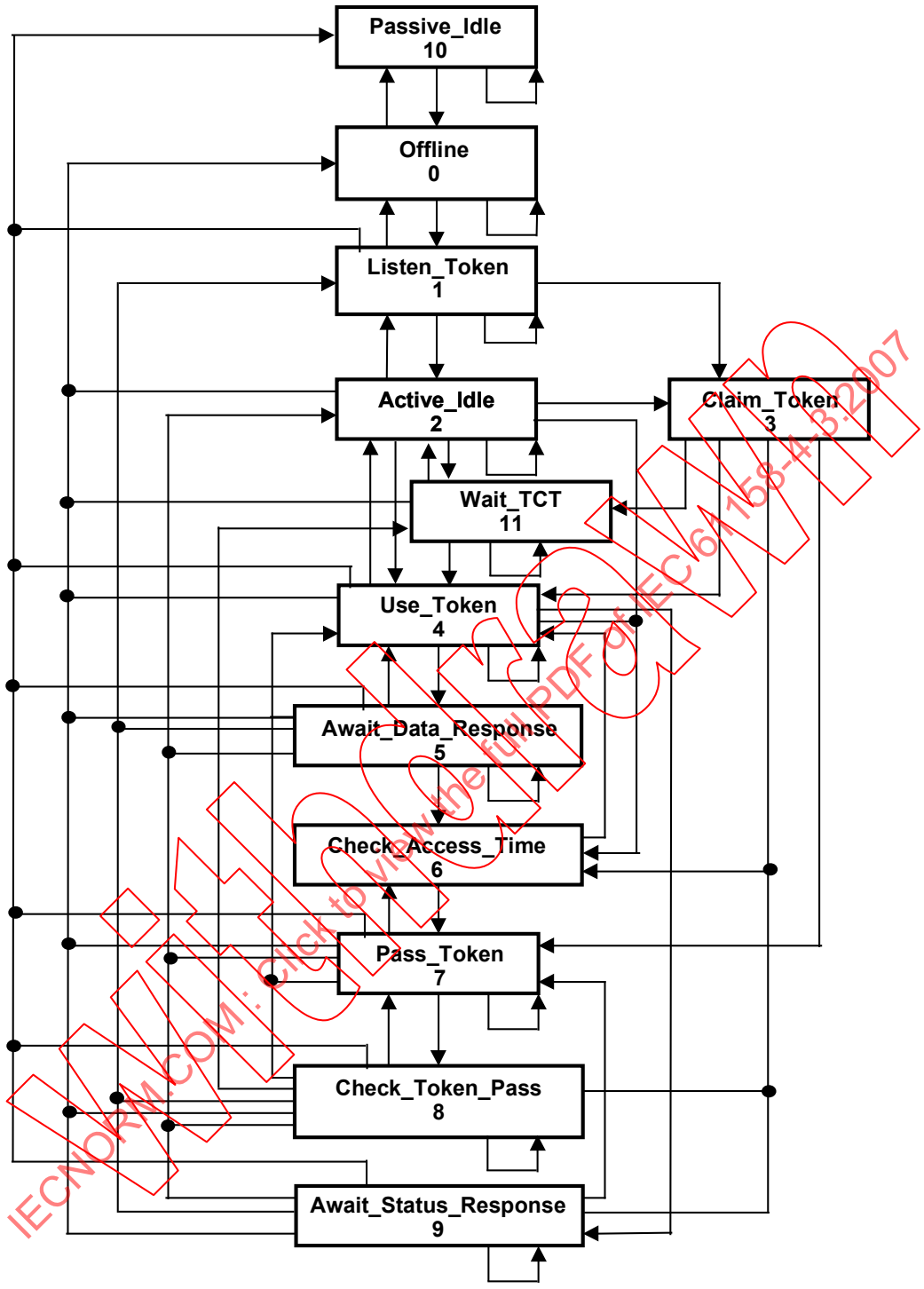


Figure 38 – Diagramme d'états de DL

8.2.2 Offline (Hors ligne)

La DLE doit passer à l'état "Offline" immédiatement après mise sous tension, après le service DLM-RESET (de réinitialisation DLM) ou après détection de certaines conditions d'erreur. Après mise sous tension, il convient que chaque station effectue un autotest. Cet autotest interne dépend de l'application et il n'a pas d'effet sur les autres stations. De ce fait, la procédure d'autotest n'est pas décrite dans la présente spécification.

Une fois achevée la séquence de mise sous tension, le MAC reste à l'état "Offline" jusqu'à ce que tous les paramètres de fonctionnement requis (voir 8.1) have aient été initialisés. Ce n'est qu'ensuite que l'entité DL peut se connecter au support de transmission sans pour autant devenir active.

8.2.3 Passive_Idle (Passif au repos)

Après initialisation de ses paramètres, le MAC de la station esclave ou de la station maître, en fonction du paramètre de fonctionnement " in_ring_desired" (in_ring_desired égal Faux) (voir Tableau 5) doit passer à l'état " Passive_Idle " et à l'écoute de la ligne. Si une DLPDU d'envoi/demande plausible, adressée à cette station, est reçue, l'entité DL doit envoyer l'acquittement ou la réponse exigée, sauf pour les DLPDU à adresse globale (message de diffusion, voir 6.3.1) et les DLPDU de jeton adressées à elles-mêmes. La DLPDU de jeton est ignorée.

Suite au service DLM-RESET, LE MAC retourne à l'état "Offline".

8.2.4 Listen_Token (Ecouter jeton)

Après initialisation de ses paramètres de fonctionnement et si la valeur du paramètre "in_ring_desired" est vraie, le MAC de la station maître doit passer à l'état " Listen_Token". Dans cet état, l'entité DL de la station maître doit surveiller la ligne afin d'identifier les stations maîtres qui sont prêtes sur l'anneau à jeton logique. A cet effet, des DLPDU de jeton sont analysées et les adresses de station qu'elles contiennent sont utilisées pour générer la liste de stations maîtres (LMS). Après avoir écouté pendant deux rotations de jeton complètes, le MAC doit passer à l'état "Active_Idle".

Au cours de la génération de la LMS, la réponse à une "Demande d'état DL avec réponse" donne "station maître non prête" après une rotation de jeton complète. Toutes les autres DLPDU ne sont pas traitées à l'état "Listen_Token"; en d'autres termes, elles ne génèrent ni acquittement ni réponse et ne sont pas notifiées à l'utilisateur DLS local.

Si le MAC détecte sa propre adresse comme adresse d'origine (SA) dans deux DLPDU de jeton lors de l'enregistrement des stations maîtres, il doit considérer qu'il existe déjà dans l'anneau une station maître ayant la même adresse. Le MAC doit ensuite repasser à l'état "Offline" et notifier cet événement à l'utilisateur DLMS local.

Si le MAC n'observe aucune activité de bus pendant la période d'expiration du délai, il doit considérer qu'une initialisation ou un rétablissement de l'anneau à jeton logique est nécessaire. Le MAC doit passer à l'état "Claim_Token".

8.2.5 Active_Idle (Actif au repos)

En quittant l'état "Listen_Token", le MAC de la station maître doit passer à l'état "Active_Idle" et attendre des messages reçus. S'il reçoit une DLPDU d'envoi/demande adressée à lui-même ou en diffusion, il doit poursuivre en acquittant ou en répondant au message, selon le cas, et doit débloquer le temporisateur de rotation du jeton s'il est affiché.

Il doit passer à l'état "Listen_Token" en cas d'erreur si deux DLPDU de jeton ayant SA = TS sont reçues l'une derrière l'autre. L'événement doit être notifié à l'utilisateur DLMS local.

Si le MAC découvre qu'il a été retiré de l'anneau à jeton logique sans qu'il en ait pris l'initiative, il doit également passer à l'état "Listen_Token" et notifier cet état (Out_of_ring) à l'utilisateur DLMS local. Il doit également passer à l'état "Listen_Token" dans le cas où la LMS du MAC n'est pas prête, c'est-à-dire lorsque SA et DA dans la DLPDU de jeton ne sont pas conformes aux informations contenues dans la LMS pendant un certain nombre de réceptions de jetons. Les DLPDU de jeton, qui indiquent l'ajout et le retrait d'un maître, ne doivent pas être considérées comme non conformes.

Si le MAC n'observe aucune activité de bus pendant la période d'expiration du délai, il doit considérer qu'un rétablissement de l'anneau à jeton logique est nécessaire. Le MAC doit tenter de revendiquer le jeton et de (ré)initialiser l'anneau logique (état "Claim_Token").

Si le MAC reçoit une "Demande d'état DL avec réponse" transmise par son prédécesseur (PS):

a) si l'adresse DL (TS) est contenue dans la LMS du MAC, le MAC doit répondre par "station maître dans l'anneau logique", ou

b) si l'adresse DL (TS) n'est pas contenue dans la LMS du MAC, le MAC doit répondre par "Station maître prête à entrer dans l'anneau logique".

Après réception d'une DLPDU de jeton adressée au MAC proprement dit et si le mode isochrone n'est pas encore établi, il doit

1) débloquer le temporisateur de rotation du jeton si ce temporisateur est figé,
2) lâcher le jeton s'il n'est pas envoyé par son prédécesseur ou si l'adresse DL (TS) n'est pas contenue dans la liste LMS du MAC. Dans les deux cas, le MAC doit mettre à jour la LMS,
3) dans le cas contraire, en traitant la transition, le T_{RR} (temps réel de rotation) doit être calculé comme étant la différence entre le temps de rotation cible et la valeur relevée du temporisateur de rotation du jeton; le temporisateur de rotation du jeton doit être relancé avec le temps de rotation cible. Le MAC doit passer:

   i) à l'état "Use_Token" si un message prioritaire est disponible, ou

   ii) à l'état "Check_Access_Time" si aucun message prioritaire n'est disponible.

Après réception d'une DLPDU de jeton adressée au MAC proprement dit et si le mode isochrone est établi, il doit

1) envoyer un message ASP, et
2) passer à l'état "Wait_TCT".

Si le MAC reçoit un message ASP et si le mode isochrone est établi, il doit geler le temporisateur de rotation du jeton.

8.2.6 Claim_Token (Revendiquer jeton)

Le MAC doit passer à l'état "Claim_Token" après l'état "Active_Idle" ou l'état "Listen_Token", une fois que son délai a expiré. Dans cet état, il doit tenter d'initialiser l'anneau logique ou de lancer l'initialisation.

Au cours de la réinitialisation, si le MAC n'est pas en mode isochrone, l'adresse de l'entité DL est contenue dans la LMS du MAC, et ainsi il passe immédiatement à l'état "Use_Token" si un message prioritaire est en attente. Dans le cas contraire, le MAC doit passer à l'état "Check_Access_Time".

Au cours de la réinitialisation, si le MAC est en mode isochrone, l'adresse de l'entité DL est contenue dans la LMS du MAC, et ainsi le MAC doit envoyer un message ASP et passer à l'état "Wait_TCT".

Au cours de la réinitialisation (l'adresse DL (TS) n'est pas contenue dans la LMS du MAC), le jeton doit tout d'abord être adressé deux fois à l'entité DL elle-même, c'est-à-dire NS = TS, notamment à l'état "Pass_Token". Ceci est nécessaire pour que l'information soit entrée dans la LMS des autres stations maîtres.

8.2.7 Wait_TCT (Attendre TCT)

Il doit passer à cet état après l'état "Claim_Token", l'état "Check_Token_Pass" ou l'état "Active_Idle" lorsqu'un message ASP a été transmis et si le maître est en mode isochrone. Dans cet état, le MAC doit transmettre autant de messages ASP que de temps de réserve disponible (T_{RCT} < (T_{CT}-T_{PSP})). Si aucun temps n'est disponible pour le passage de messages ASP, le MAC doit

a) charger le temporisateur de réserve passive avec T_{CT}-T_{RCT},

b) lancer le temporisateur de réserve passive,

c) attendre que le temporisateur de réserve passive expire, et

d) passer un message SYNCH à des fins de synchronisation.

Le MAC doit indiquer l'envoi du message SYNC à l'utilisateur DLMS local au moyen d'un événement Synch. Si le message SYNCH ne peut être envoyé au cours d'un décalage temporel maximal défini ($maxT_{SH}$), le MAC doit alors transmettre un événement Synch_Delay à l'utilisateur DLMS local.

8.2.8 Use_Token (Utiliser jeton)

Le MAC passe à l'état "Use_Token" pour traiter les périodes de transfert de messages prioritaires ou non prioritaires.

Le MAC doit passer

a) à l'état "Await_Data_Response" après chaque DLPDU d'envoi/demande émise avec acquittement ou réponse et lancer le temporisateur de créneau (voir 5.5.5), ou

b) à l'état "Check_Access_Time" dans le cas où un service SDN ou CS est émis, ou

c) à l'état "Await_Status_Response" dans le cas où une "demande d'état DL avec réponse" est émise.

8.2.9 Await_Data_Response (Attendre réponse de données)

Il doit passer à cet état en cas d'émission de demandes avec acquittement ou réponse. Le MAC attend pendant **une** durée de créneau la réception de la DLPDU d'acquittement ou de réponse.

Le MAC doit passer

a) à l'état "Check_Access_Time" afin de vérifier la durée de conservation de jeton disponible pour traiter d'autres demandes éventuelles si:

   1) une DLPDU d'acquittement valide ou une DLPDU de réponse valide est adressée à l'initiateur de la demande, ou

   2) après une ou plusieurs nouvelles tentatives, aucun acquittement ou aucune réponse valide n'est reçu(e) et aucune autre tentative n'est possible. Dans ce cas, le MAC doit aviser l'utilisateur DLS en conséquence. D'autres demandes adressées à cette station ne sont pas répétées en cas d'erreur, sauf si une période de transfert de messages correcte (émission/demande de données) a eu lieu;

b) à l'état "Use_Token", si:

   1) une DLPDU non valide est reçue (erreur de début, de fin, de longueur, erreur FCS; erreur de bit de départ, d'arrêt, de parité ou erreur de décalage) et une nouvelle tentative est possible, ou

   2) la durée de créneau a expiré (voir 5.5) et une nouvelle tentative est possible;

   et doit tenter de nouveau l'émission de la DLPDU d'envoi/demande.

La réception d'une autre DLPDU valide indique qu'une erreur a eu lieu. Le MAC doit passer à l'état "Active_Idle" et ignorer la DLPDU reçue.

8.2.10 Check_Access_Time (Vérifier temps d'accès)

Dans cet état, le temps de conservation de jeton disponible doit être calculé (voir 5.5.5). C'est seulement s'il y a encore un temps de conservation de jeton disponible que le MAC peut repasser à l'état "Use_Token". Dans le cas contraire, le MAC doit passer à l'état "Pass_Token".

Si un temps de conservation de jeton est disponible, le MAC doit transmettre:

a) si des messages prioritaires sont en attente, une DLPDU d'envoi/demande prioritaire et doit passer à l'état "Use_Token", ou

b) si des messages non prioritaires sont en attente, une DLPDU d'envoi/demande non prioritaire et doit passer à l'état "Use_Token", ou

c) lorsque le temps de mise à jour GAP expire, mais qu'un temps de conservation de jeton T_{TH} est toujours disponible, le MAC doit tenter d'enregistrer une éventuelle nouvelle station dans le GAP avant de passer le jeton, afin de l'inclure dans l'anneau logique, si nécessaire. A cet effet, le MAC transmet une "Demande d'état DL avec réponse" et passe à l'état "Use_Token".

8.2.11 Pass_Token (Passer Jeton)

Quand il est à l'état Pass_Token, le MAC doit tenter de passer le jeton à la station suivante (NS) sur l'anneau logique. Lorsqu'il transmet la DLPDU de jeton, le MAC doit simultanément vérifier que l'émetteur/récepteur fonctionne correctement. S'il ne reçoit pas sa propre DLPDU de jeton, il y a une erreur fatale sur le canal d'émission ou de réception. Le MAC doit arrêter son activité sur l'anneau logique, passer à l'état "Offline" et aviser l'utilisateur DLMS.

Si le MAC reçoit sa propre DLPDU de jeton altérée, ceci peut être dû à un émetteur ou à un récepteur provisoirement défectueux ou encore à la ligne de bus. Cette condition d'erreur n'entraîne pas l'arrêt des activités de prime abord, au lieu de cela, le MAC doit passer à l'état "Check_Token_Pass" comme après réception correcte d'une DLPDU de jeton.

Ce n'est qu'après retransmission de la DLPDU de jeton (du fait de l'absence de réaction de la part de la NS) et vérification de son état incorrect, que le MAC doit arrêter son activité sur l'anneau logique, passer à l'état "Offline" et aviser l'utilisateur DLMS.

Si le MAC a transmis le jeton avec succès, il doit ensuite passer à l'état "Check_Token_Pass".

C'est seulement si aucun successeur n'est connu, ce qui signifie qu'à ce moment-là, l'entité DL est la seule station active sur le bus, qu'il doit passer le jeton à lui-même et ensuite repasser:

a) à l'état "Use_Token" si un message prioritaire est disponible et a été envoyé, ou

b) à l'état "Check_Access_Time" si aucun message prioritaire n'est disponible;

8.2.12 Check_Token_Pass (Vérifier passage du jeton)

L'état "Check_Token_Pass" est celui au cours duquel le MAC attend pendant **une** durée de créneau qu'il y ait réaction de la part de la station à laquelle il a transmis le jeton. Ce temps d'attente permet de tenir compte du délai entre la réception de la DLPDU de jeton et la réaction, qui fait suite à l'émission, de la station à laquelle il est adressé.

Si le MAC détecte un en-tête de DLPDU valide au cours d'une durée de créneau donnée, il doit considérer que le passage de jeton est réussi. La DLPDU doit être traitée comme si elle était reçue au cours de l'état "Active_Idle".

Si une DLPDU non valide est détectée au cours de la durée de créneau, le MAC doit considérer qu'une autre station est active et par conséquent passer également à l'état "Active_Idle".

Si le MAC ne reçoit aucune DLPDU au cours d'une durée de créneau donnée, il doit de nouveau passer le jeton à son successeur, repasser à l'état "Pass_Token" et réagir comme décrit en 5.3.2.1.

Si le MAC ne reçoit aucune DLPDU au cours d'une durée de créneau donnée et qu'il a passé le jeton une deuxième fois au même successeur, le MAC doit retirer sa NS de la liste LMS, passer le jeton au nouveau successeur (NS de NS), repasser de nouveau à l'état "Pass_Token" et réagir comme décrit en 5.3.2.1.

Si le MAC reçoit un jeton avec DA égal TS et que le mode isochrone est actif, le MAC doit envoyer un message ASP et passer à l'état "Wait_TCT".

8.2.13 Await_Status_Response (Attendre Réponse d'état)

Il passe à cet état à partir de l'état "Use_Token" au cours d'une période de maintenance d'intervalle (GAP). Dans cet état, le MAC doit attendre pendant **une** durée de créneau la réception d'une DLPDU d'acquittement.

Le MAC doit passer à l'état "Check_Access_Time":

a) s'il ne reçoit rien au cours de la durée de créneau, ou

b) si une DLPDU altérée est reçue.

Si une station existante ne répond pas, elle doit être supprimée de la liste GAPL, ce qui signifie qu'elle est marquée comme adresse inutilisée. Dans ce cas, le MAC doit passer à l'état "Check_Access_Time".

Si une nouvelle station esclave ou une station maître qui ne souhaite pas être incluse dans l'anneau répond à une demande d'état, cette station doit être saisie dans la GAPL.

Si le MAC reçoit un acquittement valide indiquant "Station maître prête à entrer dans l'anneau à jeton logique" une nouvelle station maître confirme (acquittement) qu'elle souhaite être incluse dans l'anneau logique. Le MAC doit raccourcir son propre GAP pour la nouvelle NS et doit passer le jeton à cette station et se mettre à l'état "Pass_Token".

Si le MAC reçoit une autre DLPDU au lieu d'une DLPDU d'acquittement (ceci indique qu'il existe plusieurs jetons), il doit passer à l'état "Active_Idle".

8.3 Protocole de synchronisation d'horloge

8.3.1 Aperçu général

La synchronisation des horloges entre dispositifs sur un segment de bus de terrain et une application maître d'horloge sont assurées parallèlement à d'autres fonctions de communication. Un maître d'horloge est un dispositif maître de bus de terrain. Le schéma utilisé est une "correction temporelle vers l'arrière". Il y a de ce fait très peu de contraintes en temps réel imposées à un dispositif de terrain. Il n'y a aucune exigence quant à la génération des messages à des instants précis. En contrepartie, on sait à quel moment un message d'événement temporel spécial a été diffusé et ces informations sont par la suite distribuées et utilisées pour calculer des réglages d'horloge appropriés.

Il n'y a aucune confirmation concernant la réception correcte au niveau de la DLE distante; il n'y a pas non plus d'acquittements ni de nouvelles tentatives locales. Une fois les données envoyées, elles parviennent en même temps à toutes les DLE distantes (sans tenir compte du temps de propagation du signal ni des délais dus à la couche physique et à la couche liaison de données). Pour un calcul correct et en toute sécurité du temps, le transfert aura lieu en une séquence de deux messages: le premier transfert (événement temporel) et le second transfert (valeur d'horloge). En recevant l'événement temporel, chaque DLE distante lance son temporisateur de retard de réception. En recevant la valeur d'horloge, la valeur du temporisateur de retard de réception est envoyée dans l'indication. Chaque DLE distante qui a reçu la valeur d'horloge exempte d'erreur la transmet à l'utilisateur DLS au moyen de la primitive d'indication DL-CS-CLOCK-VALUE (voir la CEI 61158-3-3 service de liaison de données en mode sans connexion, service de synchronisation d'horloge). Si les erreurs ont lieu suite à une violation de la séquence (par exemple un événement temporel est reçu deux fois ou encore expiration du temps d'intervalle de synchronisation d'horloge), cette erreur est notifiée à l'utilisateur DLS.

8.3.2 Maître d'horloge de diagramme d'états

La séquence de synchronisation d'horloge est constituée de deux messages diffusés par le maître d'horloge. Lorsque le premier message, appelé événement temporel, est diffusé, toutes les DLE qui le reçoivent lancent un temporisateur de retard de réception. Le maître d'horloge envoie ensuite un second message, la valeur d'horloge, qui contient l'instant réel d'envoi de l'événement temporel. Dès réception de ce message, les DLE distantes peuvent calculer leurs valeurs de temps de retard de réception. En même temps que la valeur d'horloge reçue, chaque utilisateur DLS est capable de recalculer le moment de réception du message d'événement temporel en fonction de sa propre horloge, et de le comparer avec la valeur diffusée dans le message d'horloge. Les utilisateurs DLS peuvent alors ajuster leur propre horloge afin qu'elle corresponde à celle du maître d'horloge. Afin de prendre en charge la planification et s'assurer que les paires de messages événement temporel / valeur d'horloge correspondent, la valeur d'horloge contient également le moment où la dernière synchronisation d'horloge a eu lieu, qui est le temps envoyé lors du dernier événement temporel.

Des services de DL spéciaux sont utilisés pour prendre en charge la synchronisation d'horloge. L'horloge de station est réglée par l'utilisateur DLS, mais la mesure précise du temps d'envoi ou de réception de l'événement temporel doit être effectuée dans la DLE. Par conséquent, la DLE utilise des temporisateurs particuliers pour mesurer le délai d'émission et respectivement le délai de réception.

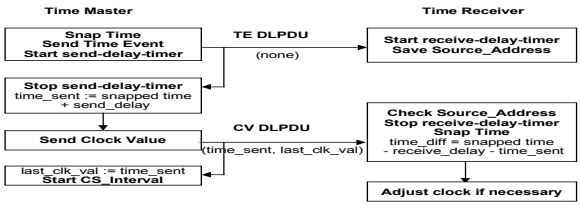
Du côté émission, la DLE dans le maître d'horloge lance son temporisateur DL – le temporisateur de retard d'émission – lorsqu'elle reçoit une demande DL-CS-TIME-EVENT de la part de l'utilisateur DLS. La DLE envoie une DLPDU de TE et arrête le temporisateur de retard d'émission une fois le dernier bit envoyé. Elle remet ensuite une confirmation à l'utilisateur DLS indiquant que le message a été envoyé, contenant un paramètre de service qui l'informe de la valeur calculée du temps de retard d'émission. L'utilisateur DLS ajoute ensuite cette valeur au temps obtenu afin de déterminer l'instant où l'événement temporel a été réellement envoyé. Une DLPDU de valeur d'horloge qui comprend ce temps est ensuite diffusée aux DLE distantes.

Du côté réception, la DLE lance son temporisateur – le temporisateur de retard de réception – lorsqu'elle reçoit la DLPDU de TE. Lorsque la valeur d'horloge est reçue (DLPDU de CV), le temporisateur de retard de réception s'arrête et une indication est transmise à l'utilisateur DLS, accompagnée du temps de retard de réception calculé afin de déterminer le moment où le message temporel a été reçu selon son horloge propre.

La séquence de synchronisation d'horloge accepte une redondance facultative des maîtres d'horloge. Même si un seul maître est en général actif à un moment donné, lorsque la commutation a lieu, il est possible que plusieurs maîtres d'horloge soient en train de diffuser les messages d'événement temporel et de valeur d'horloge. Par conséquent, lorsque

l'événement temporel est reçu, l'adresse de la station d'origine est sauvegardée par les DLE de réception. Les DLPDU de CV provenant d'autres maîtres sont ignorées.

Un aperçu général de la synchronisation d'horloge est illustré en Figure 39.



Légende

Anglais	Français
Time master	Maître d'horloge
Time receiver	Récepteur d'horloge
TE DLPDU	DLPDU de TE
CV DLPDU	DLPDU de CV
Start …	Démarrer ..
Save ..	Sauvegarder …
Check …	Vérifier …
Send clock value	Envoi de la valeur d'horloge
Stop …	Arrêter …
Adjust clock if necessary	Régler l'horloge si nécessaire
(none)	(aucune)

Figure 39 – Aperçu général de la synchronisation d'horloge

Le protocole de synchronisation d'horloge prévoit des maîtres d'horloge redondants facultatifs et utilise une simple procédure d'élection pour déterminer le maître d'horloge actif. Un seul maître d'horloge est considéré être le maître d'horloge principal sur chaque liaison. Les autres maîtres d'horloge sont les maîtres d'horloge en veille qui peuvent agir comme des sources de synchronisation d'horloge lorsque le maître d'horloge principal n'est pas actif. Les maîtres d'horloge en veille ne sont pas classés par ordre d'importance. Le protocole de synchronisation d'horloge choisit toujours le maître d'horloge principal en tant qu'origine de la synchronisation d'horloge s'il est présent. La figure ci-dessous représente la définition d'un diagramme d'états utilisé pour le fonctionnement d'un maître d'horloge (TM) principal ou d'un maître d'horloge en veille.

Les maîtres d'horloge en veille demeurent inactifs tant qu'ils reçoivent des messages de synchronisation d'horloge du maître d'horloge actif au sein d'une fenêtre de surveillance donnée. Cette fenêtre de surveillance est relancée à chaque réception de message de synchronisation d'horloge. La longueur de la fenêtre de surveillance est établie à $4 \times T_{CSI}$, c'est-à-dire un multiple du temps d'intervalle de synchronisation d'horloge.

Si aucun message de synchronisation d'horloge n'est reçu au cours de la fenêtre de surveillance, on considère qu'il y a défaillance du maître d'horloge actif. Le maître d'horloge en veille commence alors à envoyer des messages de synchronisation d'horloge.

Si un maître d'horloge en veille reçoit un message de synchronisation d'horloge d'un autre maître d'horloge en veille, il se remet à surveiller les séquences de synchronisation d'horloge. Les messages reçus du maître d'horloge principal entraînent toujours le retour du maître d'horloge en veille à sa fonction de surveillance (voir la Figure 40).

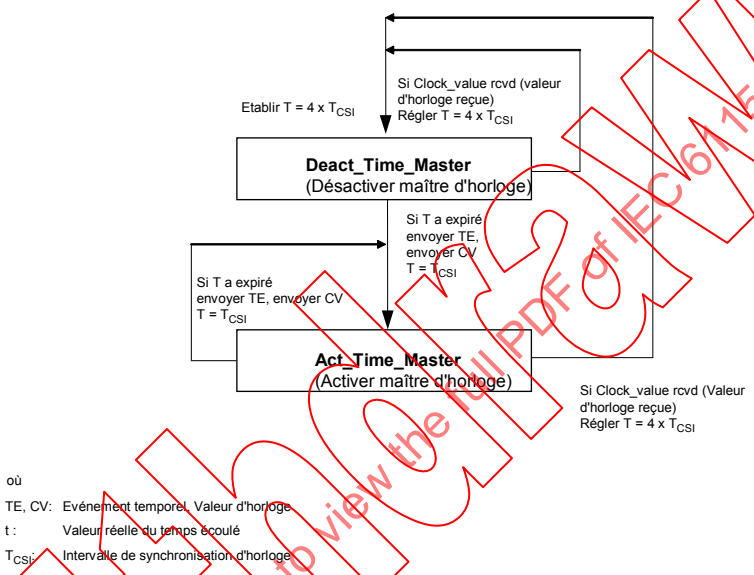


Figure 40 – Diagramme d'états de maître d'horloge

8.3.3 Récepteur d'horloge de diagramme d'états

Le diagramme d'états du récepteur est décrit en Figure 41. Le récepteur surveille les messages d'horloge dans une fenêtre temporelle qui est un multiple du temps d'intervalle de synchronisation d'horloge 4 × T_{CSI}.

Il ajuste, si nécessaire, son horloge après réception d'une séquence correcte de synchronisation d'horloge. Aucun réglage d'horloge n'est effectué et des erreurs sont indiquées dans les situations suivantes.

a) Adresses d'origine différentes dans les messages d'événements temporels et de valeurs d'horloge

b) Aucun message de synchronisation d'horloge reçu au cours d'une fenêtre temporelle

c) Absence de correspondance entre les valeurs d'horloge (last_clk_val – dernière valeur d'horloge – différente).

Le traitement de l'événement temporel, c'est-à-dire la vérification de l'adresse d'origine de la DLPDU de TE/CV et le temporisateur de retard de réception, fait partie intégrante de la DLE.

A la réception d'une valeur d'horloge valide (la SA est identique à la SA dans l'événement temporel précédemment reçu), un service DL fournit à l'utilisateur DLS le temps de

synchronisation reçu et la partie récepteur du retard de communication. La partie récepteur du retard de communication comprend le temps écoulé depuis le démarrage du temporisateur de retard de réception au moment de la réception d'une DLPDU de TE jusqu'à sa lecture à partir de la DLE après réception d'une DLPDU de CV.

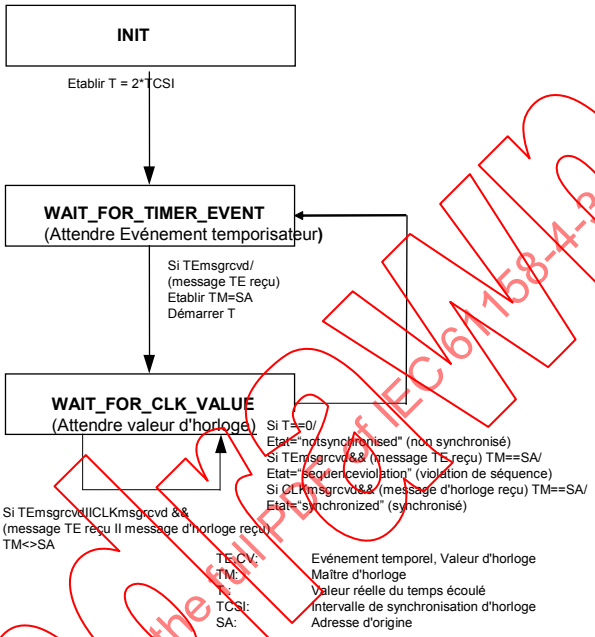


Figure 41 – Diagramme d'états de récepteur d'horloge

Les séquences de synchronisation d'horloge sont illustrées en Figure 42.

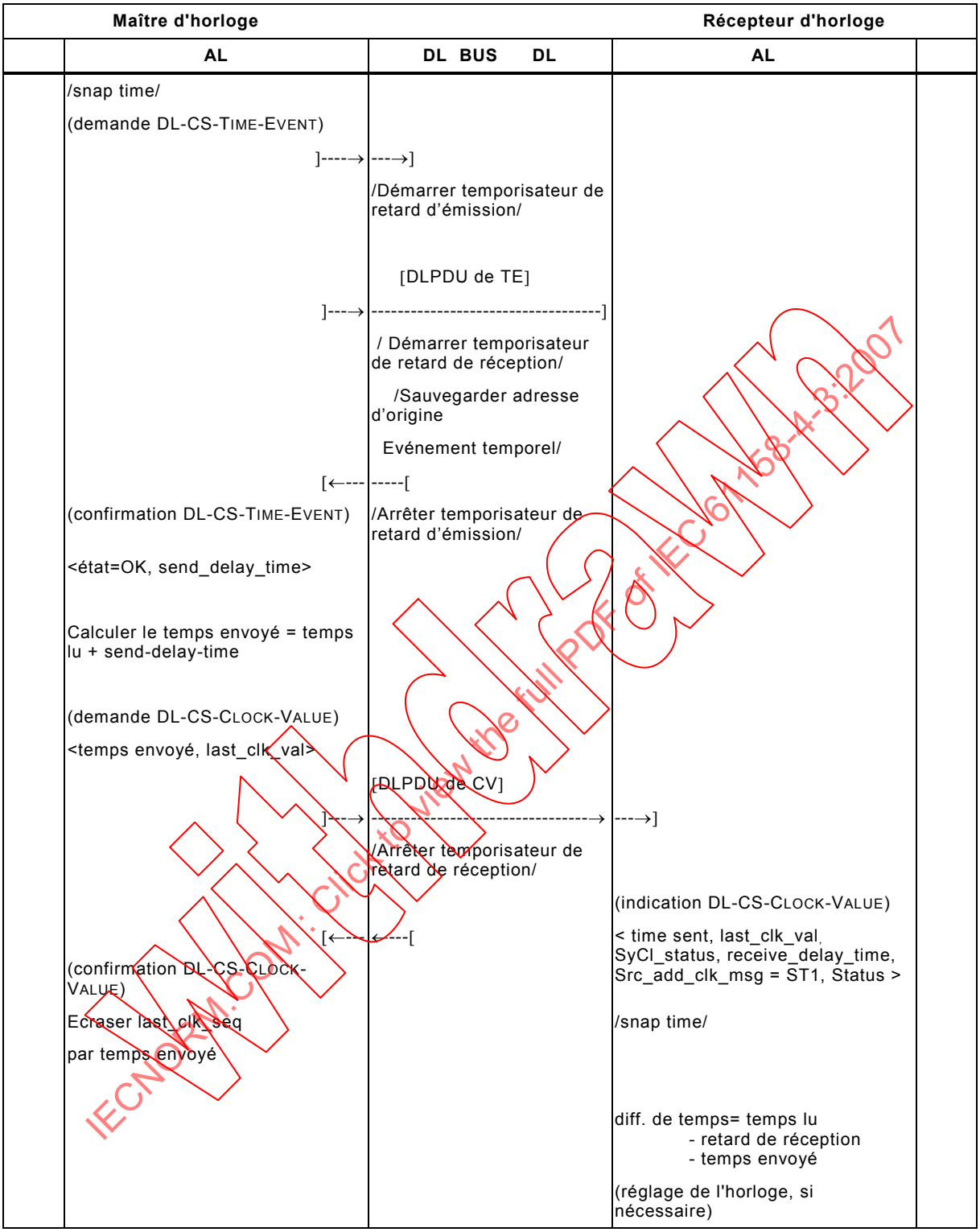


Figure 42 – Synchronisation d'horloge

Annexe A
(normative)
–
Diagrammes d'états finis de protocole DL

NOTE 1 La présente annexe décrit un certain nombre de diagrammes d'états finis utilisés par la DLE pour fournir ses fonctions de protocole de niveau bas et de niveau haut. Cette spécification complète la spécification textuelle dans le corps de la présente norme. En cas de divergence, les exigences de la présente Annexe prévalent.

NOTE 2 Les présentes descriptions de diagramme d'états finis sont nécessairement moins exhaustives qu'une description complète de l'application proprement dite. Le corps de la spécification définit des exigences et dispositions supplémentaires.

A.1 Structure globale

Le protocole DL de type 3 est constitué des trois parties suivantes:

- le traitement des services invoqués par la couche application (services DL et DLM);
- le contrôle d'accès au support physique (interactions de la gestion du jeton et des services);
- l'interface avec la couche physique y compris l'assemblage/désassemblage de PDU.

L'interface entre le gestionnaire de services (FLC/DLM) et le contrôle d'accès au support physique (MAC) comprend un ensemble de files d'attente, une liste de SAP et une ressource de données DL. (Celle ci sera également utilisée par l'unité d'émission/réception – SRU.)

L'Interface entre MAC et SRU est constituée d'un jeu de services (voir A.6.5).

Les séquences de protocole sont décrites au moyen de diagrammes d'états.

Dans les diagrammes d'état, les états sont représentés par des cases et les transitions d'états par des flèches. Les désignations des états et transitions du diagramme d'états correspondent à ceux de la liste textuelle des transitions d'états.

Cette liste de transitions d'états est structurée de la manière suivante:

La première rangée donne la dénomination de la transition.

La deuxième rangée définit l'état courant.

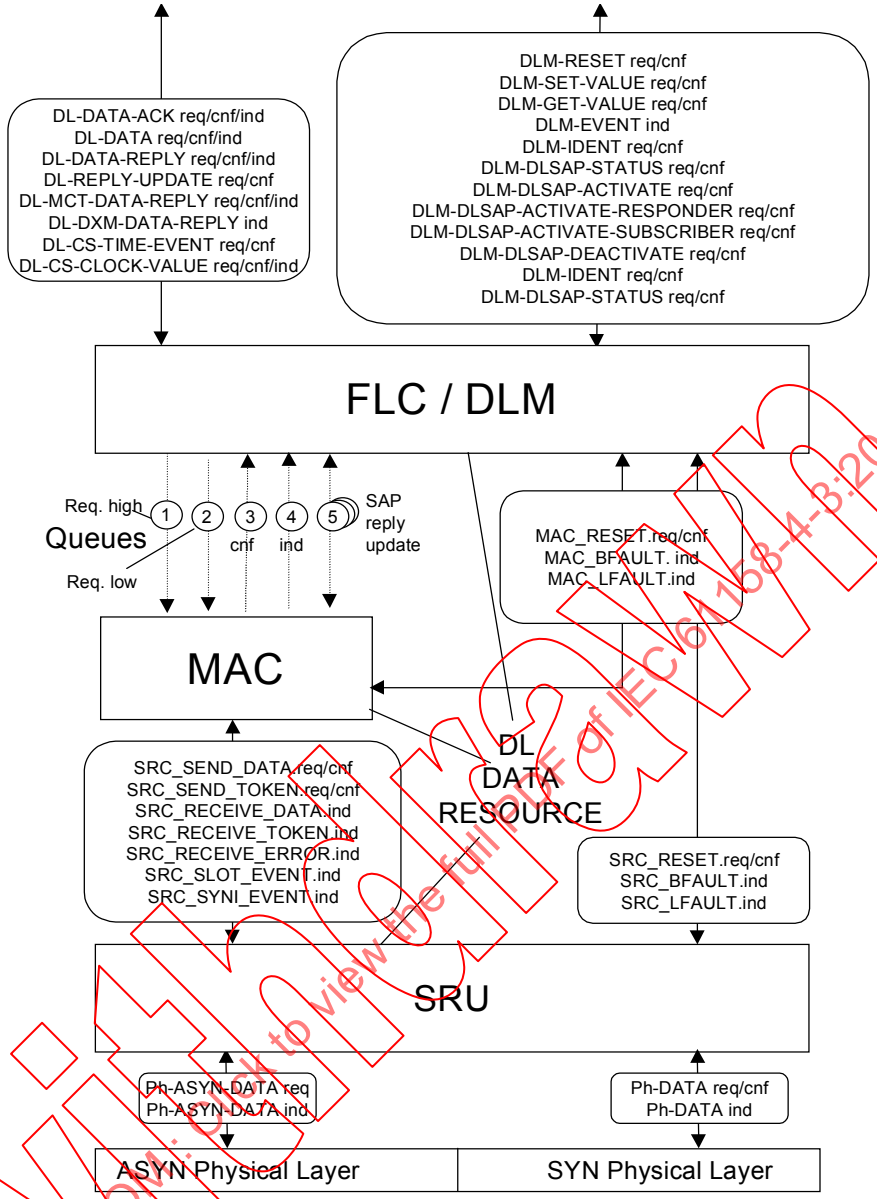
La troisième rangée fournit un événement facultatif suivi de conditions commençant par un "/" comme premier caractère de la ligne et finalement suivies par les actions commençant par un "=>" comme premier caractère de la ligne.

La dernière rangée donne l'état suivant.

Si l'événement a lieu et si les conditions sont remplies, la transition se déclenche, ce qui signifie que les actions sont exécutées et qu'il y a passage à l'état suivant.

La CEI 61158-6-3 fournit des conventions supplémentaires concernant les diagrammes d'états.

La Figure A.1 illustre la structure des machines protocolaires.



Légende

Anglais	Français
Queues	Files d'attente
SAP reply update	Mise à jour réponse SAP
DL data resource	Ressource de données DL
ASYN physical layer	Couche physique ASYN
SYN physical layer	Couche physique SYN

Figure A.1 – Structure des machines protocolaires

A.2 Variation des diagrammes d'états dans différents dispositifs

Le Tableau A.1 présente l'attribution des différentes parties des diagrammes d'états aux dispositifs. Le format des tableaux de diagrammes d'états est défini dans la CEI 61158-6-3, conventions pour les diagrammes d'états.

Tableau A.1 – Attribution des diagrammes d'états

	Diagramme d'états	Options	Remarques
Esclave	FLC	Indication SDN/SDA/SRD/CS/MSRD Demande/Confirmation de Mise à jour de réponse SRD	
		Activation/Désactivation de DLSAP	
	DLM	Jeu limité de variables FDL Pas de service d'activation de DLSAP	
	MAC	Uniquement état OFFLINE et PASSIVE-IDLE	Prise en charge de la détection du débit de données
	SRU	Pas de Jeton Pas de demande (émission) Pas de réponse (réception)	
Maître	FLC	Jeu complet de services	
	DLM	Jeu complet de services	
	MAC	Jeu complet d'états	
	SRU	Jeu complet de services/états	
Maître uniquement	MAC	Passif au Repos omis Pas de commutation actif/passif	Autre option: Passif mais sans commutation

A.3 Ressource de données DL

La ressource de données DL modélise une interface de données pour tous les diagrammes d'états de la couche liaison de données. Elle comporte des files d'attente et des tampons pour l'échange de données entre les sous-couches de la DL. Par ailleurs, elle dispose d'informations de gestion auxquelles doivent accéder les diverses sous-couches de la DL. Le Tableau A.2 illustre toutes les ressources de données de la couche liaison de données.

Tableau A.2 – Ressource de données

Nom	Élément de Structure	Type	Domaine	Remarque
Variables FDL		S0		
SAP_List		[0..63, CS, NIL] de S1		
H_List		S2		
L_List		S2		
C_List		S3		
I_List		S3		
Ident_List		S12		
S0	**Variables**			
	TS	U8	0 à 126	Adresse DL de cette station
	Data_rate		9,6; 19,2; 31,25; 45,45; 93,75; 187,5; 500; 1500, 3000; 6000; 12000 kBit/s et autres	Débit de données de ce bus de terrain
	Medium_ redundancy		simple; redondant	Disponibilité de supports redondants
	HW-Release		LE_HR; Identifiant de version du matériel	Numéro de version du matériel
	SW-Release		LE_SR; Identifiant de version du logiciel	Numéro de version du logiciel
	SYNCHT		DSAP SSAP LSDU	Contenu de la DLPDU de SYNCH
	Tct	U32	1 à 2^{24}-1 (temps binaires)	Durée de cycle isochrone
	Isochronous_ mode		0 1 2 3	non isochrone avec correction du cycle avec abandon du cycle arrêt du temporisateur TTR
	maxTsh	U8	1 à 256 (temps binaires)	décalage temporel maximal
	Tcsi	U32	1 à 232-1 (temps binaires)	Temps d'intervalle de synchronisation d'horloge
	Preamble_length	U8	8 à 64 (bits)	Longueur du préambule de trames physiques en transmission synchrone
	Tsyn	U8	4 à 32 (temps binaires)	Temps d'intervalle (gap) post-émission en transmission synchrone
	Tsl	U16	52 à 216-1 (temps binaires)	durée de créneau
	minTsdr	U16	2^0 à 2^{16}-1 (temps binaires)	Temps de retard de station le plus court
	maxTsdr	U16	2^0 à 2^{16}-1 (temps binaires)	Temps de retard de station le plus long
	Tqui	U8	0 à 28-1 (temps binaires)	Temps de décroissance de l'émetteur (durée d'état de ligne incertain) ou temps de commutation répéteur
	Tset	U8	2^0 à 28-1 (temps binaires)	Temps de montée

Nom	Élément de Structure	Type	Domaine	Remarque
	Ttr	U32	2^0 à 2^{24}-1 (temps binaires)	Temps de rotation cible
	G	U8	1 à 100	Facteur de mise à jour d'intervalle (Gap)
	in_ring_desired	Bool	Vrai; Faux	Demande d'entrée ou de sortie de l'anneau à jeton logique
	HSA	U8	0 à 126	Adresse de station la plus élevée sur ce bus de terrain
	max_retry_limit	U8	0 à 15 (de préférence 0)	Nombre maximal de nouvelles tentatives
	DLPDU_sent_ count	U32	0 à 2^{32}-1	Nombre de DLPDU envoyées
	Retry_count	U16	0 à 2^{16}-1	Nombre de DLPDU répétées
	DLPDU_sent_ count_sr	[0..126] de U32	max. 126 entrées de 0 à 2^{32}-1	Liste de nombres de DLPDU envoyées par station
	Error_count	[0..126] de U16	max. 126 entrées de 0 à 216-1	Liste de nombres de non réponses ou de réponses erronées par station
	SD_count	U32	0 à 2^{32}-1	Nombre de délimiteurs de début corrects
	SD_error_count	U16	0 à 2^{16}-1	Nombre de délimiteurs de début défectueux
	Trr	U32	2^0 à 2^{24}-1 (temps binaires)	Temps réel de rotation
	LMS	[0..126] de U8	Jusqu'à 127 adresses DL	Liste de stations maîtres sur l'anneau logique
	GAPL	[0..126] d'état de DLE	au max. 126 adresses DL (0 à 126) y compris état de DLE	Liste de toutes les stations dans son propre GAP
	Cycle_violation_ count	U32		Nombre de violations de cycle ayant eu lieu depuis le début du mode isochrone
	Tid1	U16	33 bit + 2 bit + 2 × Tset + Tqui	Variables déduites
	Tid2	U16	max (Tid1, maxTsdr)	
S1	**SAP_List**			
	Acces	U8	0..126, NIL	
	Function_List_R	Liste de		SDN_H/L,SDA_H/L,SRD_H/L, MSRD_H/L, CS(TE/CV)
	Function_List_I	Liste de		SDN_H/L, SDA_H/L,SRD_H/L, MSRD_H/L, CS(TE/CV)
	LenList	[0..16] de U8	0..246	Liste de longueurs de DLSDU autorisées selon la fonction FC
	Indication_mode		ALL/DATA/ UNCHANGED	
	Publisher_ enabled		TRUE/FALSE	
	Ibuffer	S7		Tampon d'indications
	Ubuffer	S8		Tampon de mises à jour
	Sbuffer	S7		Tampon d'abonnés

Nom	Élément de Structure	Type	Domaine	Remarque
S2	**H_List/L_List**			
	Num_entry	U16		
	First_entry	Réf. à S9		
	Last_entry	Réf. à S9		
	Insert()			
	Remove()			
S3	**C_List/I_List**			
	Num_entry	U16		
	First_entry	Réf à S10		
	Last_entry	Réf à S10		
	Insert()			
	Remove()			
S4	**REQM**			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	Serv_class		État haut/bas	
	Conf	Bool		
S5	**RESM**			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
S6	**FC**			
	Fonction		req rsp	SDN_H/L, SDA_H/L, SRD_H/L, MSRD_H/L, Ident, LSAP_Status, CS(TE/CV)
	Frame_type		req/rsp	
	FCB	U8	0,1	
	FCV	U8	0,1	
	Stn-type		Esclave, M_n_rdy, M_rdy, M_in_ring	
S7	**Ibuffer**			
	Num_entry	U16		
	First_entry	Réf à S10		
	Last_entry	Réf à S10		
	Insert()			
	Remove()			

Nom	Élément de Structure	Type	Domaine	Remarque
S8	**Ubuffer**			
	High_reference	U32		
	High_buffer	S11		
	High_transmit		SINGLE/MULTIPLE	
	Low_reference	U32		
	Low_buffer	S11		
	Low_transmit		SINGLE/MULTIPLE	
S9	**H/L_List entry**			
	Entrée suivante	Réf à S9		
	DA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
S10	**C/I_List entry**			
	Entrée suivante	Ref à S10		
	DA	U8	0..127	
	SA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
	R_Status	NO,DL,DH,OK,RS, RR,UE,NR,RDH, RDL,NA,DS		
	Référence	U32		
S11	**DLSDU**			
	Longueur	U8	0..246	
	Données	[0 à 246] de U8		
S12	**Ident_List**			
	Vendor_Name	[0..32] de U8		
	Model_Name	[0..32] de U8		
	HW_Release	[0..32] de U8		
	SW_Release	[0..32] de U8		

A.4 FLC / DLM

A.4.1 Définition des primitives

A.4.1.1 Primitives échangées entre utilisateur DL et FLC

Le Tableau A.3 présente toutes les primitives émises par l'utilisateur DL vers le FLC. Pour une description des fonctions, voir la CEI 61158-3-3.

Tableau A.3 – Primitives émises par l'utilisateur DL vers le FLC

Nom de la primitive	Paramètres associés
DL-DATA-ACK request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA-REPLY request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-MCT-DATA-REPLY request	Service_class, D_addr D_SAP_index, S_SAP_index, DLSDU,
DL-REPLY-UPDATE request	Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference
DL-CS-TIME-EVENT request	D_addr, D_SAP_index, S_SAP_index
DL-CS-CLOCK-VALUE request	D_addr, D_SAP_index, S_SAP_index, DLSDU

Le Tableau A.4 présente toutes les primitives émises par le FLC vers l'utilisateur DL. Pour une description des fonctions, voir la CEI 61158-3-3.

Tableau A.4 – Primitives émises par le FLC vers l'utilisateur DL

Nom de la primitive	Paramètres associés
DL-DATA-ACK confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status

Nom de la primitive	Paramètres associés
DL-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-REPLY-UPDATE confirm	Service_class, S_SAP_index, DL_status
DL-MCT-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-CS-TIME-EVENT confirm	D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status
DL-CS-CLOCK-VALUE confirm	D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA-ACK indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference
DL-MCT-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference
DL-DXM-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-CS-CLOCK-VALUE indication	D_addr, D_SAP_index, S-addr, S_SAP_index, DLSDU, Receive_delay_time CS_status

A.4.1.2 Primitives échangées entre utilisateur DL et DLM

Le Tableau A.5 présente toutes les primitives émises par l'utilisateur DL vers la DLM. Pour une description des fonctions, voir la CEI 61158-3-3.

Tableau A.5 – Primitives émises par l'utilisateur DL vers la DLM

Nom de la primitive	Paramètres associés
DLM-RESET request	(aucun)
DLM-SET-VALUE request	Variable_name(1 à n), Index(1 à k), Desired_value (1 à n)
DLM-GET-VALUE request	Variable_name(1 à n), Index(1 à k)
DLM-DLSAP-ACTIVATE request	S_SAP_index, Access, Service_list
DLM-DLSAP-ACTIVATE-RESPONDER request	S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled
DLM-DLSAP-ACTIVATE-SUBSCRIBER request	S_SAP_index, DLSDU_length_list
DLM-DLSAP-DEACTIVATE request	S_SAP_index
DLM-IDENT request	DL_addr
DLM-DLSAP-STATUS request	D_SAP_index, DL_addr

Le Tableau A.6 présente toutes les primitives émises par la DLM vers l'utilisateur DL. Pour une description des fonctions, voir la CEI 61158-3-3.

Tableau A.6 – Primitives émises par la DLM vers l'utilisateur DL

Nom de la primitive	Paramètres associés
DLM-RESET confirm	DLM_Status
DLM-SET-VALUE confirm	Current_value(1 à n), DLM_status(1 à n)
DLM-GET-VALUE confirm	Desired_value(1 à n), DLM_status(1 à n)
DLM-EVENT indicate	Event/Fault TSH
DLM-DLSAP-ACTIVATE confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-RESPONDER confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-SUBSCRIBER confirm	S_SAP_index, DLM_status
DLM-DLSAP-DEACTIVATE confirm	S_SAP_index, DLM_status
DLM-IDENT confirm	DL-addr, Ident_list, DLM_status
DLM-DLSAP-STATUS confirm	D_SAP_index, DL-addr, Access, Service_type(1 à n), Role_in_service_list(1 à n), DLM_status

A.4.1.3 Paramètres des primitives FLC

Le Tableau A.7 présente tous les paramètres utilisés avec des primitives échangées entre l'utilisateur DL et le FLC.

Tableau A.7 – Paramètres utilisés avec des primitives échangées entre l'utilisateur DL et le FLC

Nom du paramètre	Description
S_SAP_index	Identifiant du point d'accès de service local
D_SAP_index	Identifiant d'un point d'accès de service distant
D_addr	Adresse de station de la DLE de réception
S_addr	Adresse de station de la DLE d'émission
Service_class	Indique la priorité du service
DL_status	Etat de l'exécution du service
DLSDU	Unité de données de ou vers l'utilisateur DLS
Transmit_strategy	Mode de fonctionnement du tampon de mise à jour (simple = tampon émis une seule fois, multiple = tampon émis de manière répétée)
Reference	Référence pour identification de la DLSDU transmise à la DLE locale
Update_status	Indique la présence d'une DLSDU en provenance d'une DLE distante dans le tampon de mise à jour
Send_delay_time	Temps de retard du protocole entre l'invocation de la primitive de demande de service de synchronisation d'horloge et la transmission de la DLPDU
Receive_delay_time	Temps de retard du protocole entre la réception de la DLPDU précédente et la DLPDU courante de synchronisation d'horloge
CS_status	Etat de la séquence de synchronisation d'horloge

A.4.1.4 Paramètres des primitives DLM

Le Tableau A.8 présente tous les paramètres utilisés avec des primitives échangées entre l'utilisateur DL et la DLM.

Tableau A.8 – Paramètres utilisés avec des primitives échangées entre l'utilisateur DL et la DLM

Nom du paramètre	Description
S_SAP_index	Identifiant du point d'accès de service local
D_SAP_index	Identifiant d'un point d'accès de service distant
Variable_name(1 à n)	Liste de noms de variables DL
Desired_value (1 à n)	Liste de valeurs de la variable DL
Current_value (1 à n)	Liste de valeurs de la variable DL
Access	Permission d'utilisation d'un SAP local par une ou plusieurs DLE distantes
Service_list	Types de services acceptés au niveau d'un SAP local
DLSDU_length_list	Longueur maximale de DLSDU d'arrivée ou de départ au niveau d'un SAP local
Indication_mode	Mode d'Indication de DLPDU reçues sans données utilisateur
DLM_status	Etat de l'exécution du service
Event/Fault	Indique la cause d'un événement ou d'un défaut
DL-addr	Adresse de station de la DLE locale ou d'une DLE distante
Ident_list	Spécifie nom du fournisseur, le type de contrôleur, la version du matériel/logiciel
Service_type(1 à n)	Spécifie les services DL activés
Role_in_service_list(1 à n)	Spécifie le rôle dans le service (initiateur ou répondeur ou les deux)

A.4.2 Description du diagramme d'états

Le FLC constitue l'interface entre les machines protocolaires DL et l'utilisateur DL pour les services SDA, SDN, SRD, MSRD et CS.

La DLM constitue l'interface entre les machines protocolaires DLM et l'utilisateur DLM pour ce qui concerne les services de gestion.

Les services FLC sont traités de la même manière que les services DLM. De ce fait, le module DLM est décrit dans le même diagramme d'états que le module FLC; sa description est la suivante:

Les services sont tous traités en l'état READY (PRET).

La demande de service est d'abord validée. Une validation négative entraîne une confirmation négative. Dans le cas contraire, le service est placé dans la file d'attente de service prioritaire ou non prioritaire du paramètre Classe de Service. Les services exécutés par le MAC sont déplacés des files d'attente de demande aux files d'attente de confirmation. Tous les services valides arrivants sont placés dans la file d'attente d'indication.

Le jeu de services d'activation/désactivation SAP est utilisé pour organiser la liste SAP. Les services de mise à jour réponse sont utilisés pour placer les données de réponse dans la liste de services.

Variables locales

Le FLC / la DLM n'utilise pas de variables locales. Toutes les variables auxquelles doit accéder le FLC / la DLM sont contenues dans la Ressource de données DL.

Nomenclature des tables d'états

Les suffixes normalisés ".req", ".cnf" et ".ind" sont utilisés pour indiquer respectivement les primitives de demande, de confirmation et d'indication. Les noms des primitives de service sont entièrement en majuscules.

A.4.3 Table FLC / DLM

La table d'états FLC et DLM est donnée dans le Tableau A.9.

Tableau A.9 – Table d'états FLC/DLM

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
1	READY	DLM-RESET.req ⇒ MAC_reset := FALSE SRC_reset := FALSE MAC_RESET.req SRC_RESET.req	WAIT_RESET_CNF
2	WAIT_RESET_CNF	MAC_RESET.cnf /SRC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
3	WAIT_RESET_CNF	MAC_RESET.cnf /!SRC_reset ⇒ MAC_reset := TRUE	WAIT_RESET_CNF
4	WAIT_RESET_CNF	SRC_RESET.cnf /MAC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
5	WAIT_RESET_CNF	SRC_RESET.cnf /!MAC_reset ⇒ SRC_reset := TRUE	WAIT_RESET_CNF
6	READY	DLM-GET-VALUE.req (Variable_name((1 à n), Index(1 à k))) /!CHECK_PAR_READVALUE (Variable_name((1 à n), Index(1 à k))) ⇒ DLM_status (1 à n) := IV Current_value (1 à n) := NIL DLM-GET-VALUE.cnf (Current_value(1 à n),DLM_status(1 à n))	READY
7	READY	DLM-GET-VALUE.req (Variable_name((1 à n), Index(1 à k))) /CHECK_PAR_READVALUE (Variable_name((1 à n), Index(1 à k))) ⇒ Check_variable_names (Variable_name((1 à n), Index(1 à k))) Set_current_values (Variable_name((1 à n), Index(1 à k))) DLM-GET-VALUE.cnf (Current_value(1 à n),DLM_status(1 à n))	READY
8	READY	DLM-SET-VALUE.req (Variable_name((1 à n), Index(1 à k)), Desired_value(1 à n)) /!CHECK_PAR_SETVALUE (Variable_name((1 à n), Index(1 à k)), Desired_value(1 à n)) ⇒ DLM_status (1 à n) := IV DLM-SET-VALUE.cnf (DLM_status (1 à n))	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
9	READY	DLM-SET-VALUE.req (Variable_name((1 à n), Index(1 à k)), Desired_value(1 à n)) /CHECK_PAR_SETVALUE (Variable_name((1 à n), Index(1 à k)), Desired_value(1 à n)) ⇒ Set_variable_list (Variable_name((1 à n), Index(1 à k)), Desired_value(1 à n)) DLM-SET-VALUE.cnf (DLM_status (1 à n))	READY
10	READY	MAC_BFAULT.ind (Fault_type, Tsh) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault, Tsh)	READY
11	READY	MAC_BFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY
12	READY	MAC_LFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY
13	READY	SRC_BFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
14	READY	SRC_LFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
15	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /!CHECK_PAR_DLSAP ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
16	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
17	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && !ACTIVATED (S_SAP_index) ⇒ SET_SAP_LIST (S_SAP_index, Access, Service_list) DLM_status := OK DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
18	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /!CHECK_PAR_DLSAP_RES ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
19	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
20	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
21	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !RACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
22	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
23	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && RACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && !CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
24	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /!CHECK_PAR_DLSAP_SUB ⇒ DLM_status := IV DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
25	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && SACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
26	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && !SACTIVATED (S_SAP_index) ⇒ SET_SSAP_LIST (S_SAP_index, DLSDU_length_list) DLM_status := OK DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
27	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /!CHECK_PAR_DLSAP_DEACT ⇒ DLM_status := IV DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
28	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && !ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
29	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && ACTIVATED (S_SAP_index) ⇒ RESET_SAP_LIST (S_SAP_index) DLM_status := OK DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
30	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /!CHECK_PAR_STATUS ⇒ Access := NIL Service_list (1 à n) := NIL Role_in_service_list (1 à n) := NIL DLM_status := IV DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 à n), Role_in_service_list (1 à n), DLM_status)	READY
31	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List[D_SAP_index] = NIL ⇒ Access := NIL Service_list (1 à n) := NIL Role_in_service_list (1 à n) := NIL DLM_status := LR DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 à n), Role_in_service_list (1 à n), DLM_status)	READY
32	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List ≠ NIL[D_SAP_index] ⇒ Write_SAPstatus_list() DLM_status := OK DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 à n), Role_in_service_list (1 à n), DLM_status)	READY
33	READY	DLM-IDENT.req (D_addr) /!CHECK_PAR_IDENT ⇒ Ident_list := NIL DLM_status := IV DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
34	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List = NIL ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
35	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List ≠ NIL ⇒ Write_ident_list() DLM_status := OK DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
36	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = TRUE && D_addr ≠ Variables.TS ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
37	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = FALSE && D_addr ≠ Variables.TS ⇒ Ident_Pending := TRUE PUT_LREQ (IDENT)	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
38	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_SDN ⇒ DL_status := IV DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
39	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && !CHECK_SAP(Service_class, S_SAP_index, SDN) ⇒ DL_status := LS DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
40	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
41	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SDN_H)	READY
42	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SDN_L)	READY
43	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_SDA ⇒ DL_status := IV DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
44	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && !CHECK_SAP(Service_class, S_SAP_index, SDA) ⇒ DL_status := LS DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
45	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
46	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = high ⇒ PUT_HREQ (SDA_H)	READY
47	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = low ⇒ PUT_LREQ (SDA_L)	READY
48	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_REPLY ⇒ DL_status := IV DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
49	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
50	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
51	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SRD_H)	READY
52	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SRD_L)	READY
53	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /!CHECK_PAR_REPLY ⇒ DL_status := IV DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
54	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
55	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
56	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (MSRD_H)	READY
57	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_HREQ (MSRD_L)	READY
58	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /!CHECK_PAR_UPDATE ⇒ DL_status := IV DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY

N°	Etat courant	Evénement /condition ⇒action	Etat suivant
59	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
60	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) / CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
61	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ DL_status = OK PUT_HUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
62	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ DL_status = OK PUT_LUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
63	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = IDENT ⇒ Ident_pending := FALSE GET_IDENT_CNF() DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
64	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = DLSAP-STATUS ⇒ Status_pending := FALSE GET_DLSAPSTATUS_CNF() DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 à n), Role_in_service_list (1 à n), DLM_status)	READY
65	READY	/C_List.Num_entry≠0 && C_List.First_entry.Function = SDA_H || C_List.First_entry.Function = SDA_L ⇒ GET_SDA/SDN_CNF() DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
66	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SDN ⇒ GET_SDA/SDN_CNF() DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
67	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SRD ⇒ GET_SRD_CNF() DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
68	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = MSRD ⇒ GET_SRD_CNF() DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY