

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 4-11: Data-link layer protocol specification – Type 11 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 4-11: Spécification du protocole de la couche liaison de données –
Éléments de type 11**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2014 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
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 61158-4-11

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 4-11: Data-link layer protocol specification – Type 11 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 4-11: Spécification du protocole de la couche liaison de données –
Éléments de type 11**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XH

ICS 25.040.40; 35.100.20; 35.110

ISBN 978-2-8322-1723-8

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	7
INTRODUCTION.....	9
1 Scope.....	11
1.1 General.....	11
1.2 Specifications.....	11
1.3 Procedures.....	11
1.4 Applicability.....	12
1.5 Conformance.....	12
2 Normative references	12
3 Terms, definitions, symbols and abbreviations.....	12
3.1 Reference model terms and definitions.....	12
3.2 Service convention terms and definitions.....	14
3.3 Terms and definitions.....	15
3.4 Symbols and abbreviations.....	19
4 Overview of the DL-protocol	20
4.1 General.....	20
4.2 Overview of the medium access control.....	21
4.3 Service assumed from the PhL	22
4.4 DLL architecture.....	23
4.5 Access control machine and schedule support functions	27
4.6 Local parameters, variables, counters, timers and queues.....	28
5 General structure and encoding of PhIDU and DLPDU and related elements of procedure.....	47
5.1 Overview	47
5.2 PhIDU structure and encoding.....	47
5.3 Common MAC frame structure, encoding and elements of procedure	48
5.4 Elements of the MAC frame.....	48
5.5 Order of bit transmission	53
5.6 Invalid DLPDU.....	53
6 DLPDU-specific structure, encoding and elements of procedure.....	54
6.1 General.....	54
6.2 Synchronization DLPDU (SYN).....	54
6.3 Transmission complete DLPDU (CMP)	60
6.4 In-ring request DLPDU (REQ)	61
6.5 Claim DLPDU (CLM)	63
6.6 Command DLPDU (COM).....	64
6.7 Cyclic data and cyclic data with transmission complete DLPDU (DT) and (DT-CMP).....	65
6.8 RAS DLPDU (RAS)	67
6.9 Loop repeat request DLPDU (LRR)	68
6.10 Loop diagnosis DLPDU (LPD)	72
7 DLE elements of procedure	72
7.1 DLE elements of procedure for star-architecture.....	72
7.2 DLE elements of procedure for loop-architecture.....	96
7.3 Serializer and deserializer	150
7.4 DLL management protocol.....	150

Bibliography.....	161
Figure 1 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses	16
Figure 2 – Basic principle of medium access control	21
Figure 3 – Interaction of PhS primitives to DLE	23
Figure 4 – Data-link layer internal architecture of star-architecture	25
Figure 5 – Data-link layer internal architecture of loop-architecture	27
Figure 6 – Common MAC frame format for DLPDUs.....	48
Figure 7 – Structure of FC field	49
Figure 8 – Structure of SYN DLPDU	55
Figure 9 – Structure of CMP DLPDU	60
Figure 10 – Structure of REQ DLPDU	61
Figure 11 – Structure of CLM DLPDU	63
Figure 12 – Structure of COM DLPDU.....	65
Figure 13 – Structure of DT DLPDU	66
Figure 14 – Structure of RAS DLPDU.....	67
Figure 15 – Structure of User data of loop-architecture	67
Figure 16 – Structure of LRR DLPDU	68
Figure 17 – Open-ring under control	70
Figure 18 – Structure of LPD DLPDU	72
Figure 19 – Overall structure of DLL	73
Figure 20 – DLE state transition.....	74
Figure 21 – State transition diagram of CTRC.....	76
Figure 22 – State transition diagram of STRC	79
Figure 23 – State transition diagram of ACM.....	83
Figure 24 – State transition diagram of RMC sending and send arbitration	91
Figure 25 – State transition diagram of RMC receiving.....	94
Figure 26 – Overall structure of DLL	97
Figure 27 – DLE state transition.....	98
Figure 28 – State transition diagram of CTRC.....	100
Figure 29 – State transition diagram of STRC	104
Figure 30 – State transition diagram of ACM for 100 Mbps operation	108
Figure 31 – State transition diagram of ACM for 1 000 Mbps operation	109
Figure 32 – State transition diagram of RMC for 100 Mbps operation.....	129
Figure 33 – State transition diagram of RMC for 1 000 Mbps operation.....	130
Figure 34 – State transition diagram of DLM	153
Figure 35 – State transition diagram of DLM	157

Table 1 – Data-link layer components of star-architecture	24
Table 2 – Data-link layer components of loop-architecture	26
Table 3 – DLE-variables and permissible values of star-architecture	29
Table 4 – Observable variables and their value ranges of star-architecture	31
Table 5 – DLE variables and permissible values of loop-architecture	32
Table 6 – Observable variables and their value ranges of loop-architecture	35
Table 7 – F-type: DLPDU type	50
Table 8 – FCS length, polynomials and constants	51
Table 9 – PN-parameter: 3rd octet	55
Table 10 – Structure of CW: 4th octet	56
Table 11 – PM parameter	56
Table 12 – RMSEL parameter	56
Table 13 – Structure of CW: 4th octet	57
Table 14 – ST-parameter: 5th octet	57
Table 15 – Th-parameter: 6th, 7th and 8th octets	57
Table 16 – Tm-parameter: 9th and 10th octets	58
Table 17 – Ts-parameter: 11th and 12th octet	58
Table 18 – TI parameter: 13th and 14th octets	58
Table 19 – LL parameters: 15th to 46th octets	59
Table 20 – NM parameter	61
Table 21 – RN parameter	62
Table 22 – CLM parameter: 4th octet	63
Table 23 – DT parameter: 3rd and 4th octets	66
Table 24 – RAS parameter: 3rd and 4th octets	67
Table 25 – Format of the PS parameter: 3rd octet	69
Table 26 – The value of the PP parameter	69
Table 27 – The value of the send-enable-A/-B	69
Table 28 – The value of the receive-enable-A/-B	69
Table 29 – The value of the forward-enable-A/-B	70
Table 30 – RN parameter: 4th octet	70
Table 31 – Operational condition of the node	71
Table 32 – Primitives exchanged between DLS-user and CTRC	75
Table 33 – Primitives exchanged between CTRC and ACM	75
Table 34 – Parameters used with primitives exchanged between DLS-user and CTRC	76
Table 35 – CTRC state table	77
Table 36 – CTRC functions table	78
Table 37 – Primitives exchanged between DLS-user and STRC	78
Table 38 – Primitives exchanged between STRC and ACM	79
Table 39 – Parameters used with primitives exchanged between DLS-user and STRC	79
Table 40 – STRC state table	80
Table 41 – STRC functions table	81
Table 42 – Primitives exchanged between ACM and RMC	82
Table 43 – Parameters used with primitives exchanged between ACM and RMC	82

Table 44 – Primitives exchanged between ACM and CTRC.....	82
Table 45 – Parameters used with primitives exchanged between ACM and CTRC	82
Table 46 – Primitives exchanged between ACM and STRC.....	83
Table 47 – Parameters used with primitives exchanged between ACM and STRC.....	83
Table 48 – ACM state table.....	84
Table 49 – ACM function table	89
Table 50 – Primitives exchanged between ACM and RMC	90
Table 51 – Primitives exchanged between RMC and serializer / deserializer.....	90
Table 52 – Primitives exchanged between RMC and Ph-layer	90
Table 53 – Parameters between RMC and ACM	90
Table 54 – Parameters between RMC and Ph-layer	91
Table 55 – State table of RMC sending.....	92
Table 56 – State table of RMC send arbitration.....	93
Table 57 – State table for RMC receiving.....	94
Table 58 – RMC function table.....	96
Table 59 – Primitives exchanged between DLS-user and CTRC.....	99
Table 60 – Primitives exchanged between CTRC and ACM.....	100
Table 61 – Parameters used with primitives exchanged between DLS-user and CTRC	100
Table 62 – CTRC state table.....	101
Table 63 – CTRC functions table	102
Table 64 – Primitives exchanged between DLS-user and STRC.....	103
Table 65 – Primitives exchanged between STRC and ACM.....	103
Table 66 – Parameters used with primitives exchanged between DLS-user and STRC	103
Table 67 – STRC state table	104
Table 68 – STRC functions table	105
Table 69 – Primitives exchanged between ACM and RMC	106
Table 70 – Parameters used with primitives exchanged between ACM and RMC	106
Table 71 – Primitives exchanged between ACM and CTRC.....	106
Table 72 – Parameters used with primitives exchanged between ACM and CTRC	106
Table 73 – Primitives exchanged between ACM and STRC.....	107
Table 74 – Parameters used with primitives exchanged between ACM and STRC.....	107
Table 75 – ACM state table for 100 Mbps operation.....	110
Table 76 – ACM state table for 1 000 Mbps operation	117
Table 77 – ACM function table	126
Table 78 – Primitives exchanged between ACM and RMC	127
Table 79 – Primitives exchanged between RMC and Serializer / Deserializer.....	127
Table 80 – Primitives exchanged between RMS and Ph-layer	127
Table 81 – Parameters between RMC and ACM	128
Table 82 – Parameters between RMC and Serializer / Deserializer, Ph-layer.....	128
Table 83 – State table of RMC for 100 Mbps operation	130
Table 84 – State table of RMC for 1 000 Mbps operation	140
Table 85 – The RMC function table.....	150
Table 86 – Primitives exchanged between DLMS-user and DLM	151

Table 87 – Parameters used with primitives exchanged between DL-user and DLM.....	151
Table 88 – Event-related state change variables.....	152
Table 89 – DLM state table	153
Table 90 – DLM function table	155
Table 91 – Primitives exchanged between DLMS-user and DLM	156
Table 92 – Parameters used with primitives exchanged between DL-user and DLM.....	156
Table 93 – Event-related state change variables.....	157
Table 94 – DLM state table	158
Table 95 – DLM function table	160

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 4-11: Data-link layer protocol specification –
Type 11 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 itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 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.

Attention is drawn to the fact that the use of the associated protocol type is restricted by its 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 layer protocol type to be used with other layer protocols of the same type, or in other type combinations explicitly authorized by its intellectual-property-right holders.

NOTE Combinations of protocol types are specified in IEC 61784-1 and IEC 61784-2.

International Standard IEC 61158-4-11 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision.

The main changes with respect to the previous edition are addition in the loop (ring)-architecture. More details:

- Subclauses 4.6.1, 4.6.4 and 5.4.6, Clause 6 and 7.2 for the loop-architecture are modified to cover the additional specifications for the higher data rate in the loop-architecture;
- specifications for existing star-architecture are maintained as they are.

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

FDIS	Report on voting
65C/762/FDIS	65C/772/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.

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

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

The committee has decided that the contents of this publication will remain unchanged until the stability 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.

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 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.

Attention is drawn to the fact that 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 profile parts. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning Type 11 elements and possibly other types given in 4.2, 4.4, 4.5, 5.4, 6.2 to 6.10, 7.1 and 7.2 as follows:

AU 2007320662(B2)	[TO]	Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system
ZL 200780042584.7	[TO]	Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system
JP 4991254(B2)	[TO]	Control method for double-ring network, initializing method for double-ring network, transmission station of double-ring network, restructuring method for abnormality occurrence of double-ring network, network system, control method for network system, transmission station, and program of transmission station
KR 101149837(B1)	[TO]	Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system
RU 2420899(C2)	[TO]	Double-ring network system and method of controlling communication in said network, transmission station and programme for transmission stations
US 8411559(B2)	[TO]	Double ring network system and communication control method thereof, and transmission station, and program for transmission stations

EP 2093942(A1) [TO] Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system

US 6711131(B1) [TO] Data transmitting apparatus, network interface apparatus, and data transmitting system

JP 3461954(B2) [TO] Data transmitter

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

The holders of these patent rights have assured the IEC that they are willing to negotiate licences either free of charge or 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 is registered with IEC. Information may be obtained from:

[TO] Toshiba Corporation
1, Toshiba -cho
Fuchu-shi
Tokyo 183-8511, Japan
Attention: Intellectual Property Rights Section

Attention is drawn to the possibility that some of the elements of this document 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.

ISO (www.iso.org/patents) and IEC (<http://patents.iec.ch>) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-11: Data-link layer protocol specification – Type 11 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 all participating data-link entities

- a) in a synchronously-starting cyclic manner, according to a pre-established schedule, and
- b) in a cyclic or acyclic asynchronous manner, as requested each cycle by each of those data-link entities.

Thus this protocol can be characterized as one which provides cyclic and acyclic access asynchronously but with a synchronous restart of each cycle.

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) procedures for giving communications opportunities to all participating DL-entities, sequentially and in a cyclic manner for deterministic and synchronized transfer at cyclic intervals up to one millisecond;
- c) procedures for giving communication opportunities available for time-critical data transmission together with non-time-critical data transmission without prejudice to the time-critical data transmission;
- d) procedures for giving cyclic and acyclic communication opportunities for time-critical data transmission with prioritized access;
- e) procedures for giving communication opportunities based on standard ISO/IEC 8802-3 medium access control, with provisions for nodes to be added or removed during normal operation;
- f) 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 documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE All parts of the IEC 61158 series, as well as IEC 61784-1 and IEC 61784-2 are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-3-11:2007, *Industrial communication networks – Fieldbus specifications – Part 3-11: Data-link layer service definition – Type 11 elements*

IEC 61158-5-11:2007, *Industrial communication networks – Fieldbus specifications – Part 5-11: Application layer service definition – Type 11 elements*

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 8802-3:2000, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

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

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, 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	[ISO/IEC 7498-3]
3.1.2	calling-DL-address	[ISO/IEC 7498-3]
3.1.3	centralized multi-end-point-connection	[ISO/IEC 7498-1]
3.1.4	correspondent (N)-entities correspondent DL-entities (N=2) correspondent Ph-entities (N=1)	[ISO/IEC 7498-1]
3.1.5	demultiplexing	[ISO/IEC 7498-1]
3.1.6	DL-address	[ISO/IEC 7498-3]
3.1.7	DL-address-mapping	[ISO/IEC 7498-1]
3.1.8	DL-connection	[ISO/IEC 7498-1]
3.1.9	DL-connection-end-point	[ISO/IEC 7498-1]
3.1.10	DL-connection-end-point-identifier	[ISO/IEC 7498-1]
3.1.11	DL-connection-mode transmission	[ISO/IEC 7498-1]
3.1.12	DL-connectionless-mode transmission	[ISO/IEC 7498-1]
3.1.13	DL-data-sink	[ISO/IEC 7498-1]
3.1.14	DL-data-source	[ISO/IEC 7498-1]
3.1.15	DL-duplex-transmission	[ISO/IEC 7498-1]
3.1.16	DL-facility	[ISO/IEC 7498-1]
3.1.17	DL-local-view	[ISO/IEC 7498-3]
3.1.18	DL-name	[ISO/IEC 7498-3]
3.1.19	DL-protocol	[ISO/IEC 7498-1]
3.1.20	DL-protocol-connection-identifier	[ISO/IEC 7498-1]
3.1.21	DL-protocol-control-information	[ISO/IEC 7498-1]
3.1.22	DL-protocol-data-unit	[ISO/IEC 7498-1]
3.1.23	DL-protocol-version-identifier	[ISO/IEC 7498-1]
3.1.24	DL-relay	[ISO/IEC 7498-1]
3.1.25	DL-service-connection-identifier	[ISO/IEC 7498-1]
3.1.26	DL-service-data-unit	[ISO/IEC 7498-1]
3.1.27	DL-simplex-transmission	[ISO/IEC 7498-1]
3.1.28	DL-subsystem	[ISO/IEC 7498-1]
3.1.29	DL-user-data	[ISO/IEC 7498-1]
3.1.30	flow control	[ISO/IEC 7498-1]
3.1.31	layer-management	[ISO/IEC 7498-1]
3.1.32	multiplexing	[ISO/IEC 7498-3]

3.1.33	naming-(addressing)-authority	[ISO/IEC 7498-3]
3.1.34	naming-(addressing)-domain	[ISO/IEC 7498-3]
3.1.35	naming-(addressing)-subdomain	[ISO/IEC 7498-3]
3.1.36	(N)-entity DL-entity Ph-entity	[ISO/IEC 7498-1]
3.1.37	(N)-interface-data-unit DL-service-data-unit (N=2) Ph-interface-data-unit (N=1)	[ISO/IEC 7498-1]
3.1.38	(N)-layer DL-layer (N=2) Ph-layer (N=1)	[ISO/IEC 7498-1]
3.1.39	(N)-service DL-service (N=2) Ph-service (N=1)	[ISO/IEC 7498-1]
3.1.40	(N)-service-access-point DL-service-access-point (N=2) Ph-service-access-point (N=1)	[ISO/IEC 7498-1]
3.1.41	(N)-service-access-point-address DL-service-access-point-address (N=2) Ph-service-access-point-address (N=1)	[ISO/IEC 7498-1]
3.1.42	peer-entities	[ISO/IEC 7498-1]
3.1.43	Ph-interface-control-information	[ISO/IEC 7498-1]
3.1.44	Ph-interface-data	[ISO/IEC 7498-1]
3.1.45	primitive name	[ISO/IEC 7498-3]
3.1.46	reassembling	[ISO/IEC 7498-1]
3.1.47	recombining	[ISO/IEC 7498-1]
3.1.48	reset	[ISO/IEC 7498-1]
3.1.49	responding-DL-address	[ISO/IEC 7498-3]
3.1.50	routing	[ISO/IEC 7498-1]
3.1.51	segmenting	[ISO/IEC 7498-1]
3.1.52	sequencing	[ISO/IEC 7498-1]
3.1.53	splitting	[ISO/IEC 7498-1]
3.1.54	synonymous name	[ISO/IEC 7498-3]
3.1.55	systems-management	[ISO/IEC 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 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.3.1 common memory

virtual common memory over Type 11 fieldbus, which is shared by participating Type 11 fieldbus nodes and is primarily used for real-time communications by the time-critical cyclic data service

3.3.2 data DLPDU

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

3.3.3

DLCEP-address

DL-address which designates either

- a) one peer DL-connection-end-point, or
- b) one multi-peer publisher DL-connection-end-point and implicitly the corresponding set of subscriber DL-connection-end-points where each DL-connection-end-point exists within a distinct DLSAP and is associated with a corresponding distinct DLSAP-address

3.3.4

DL-segment, link, local link

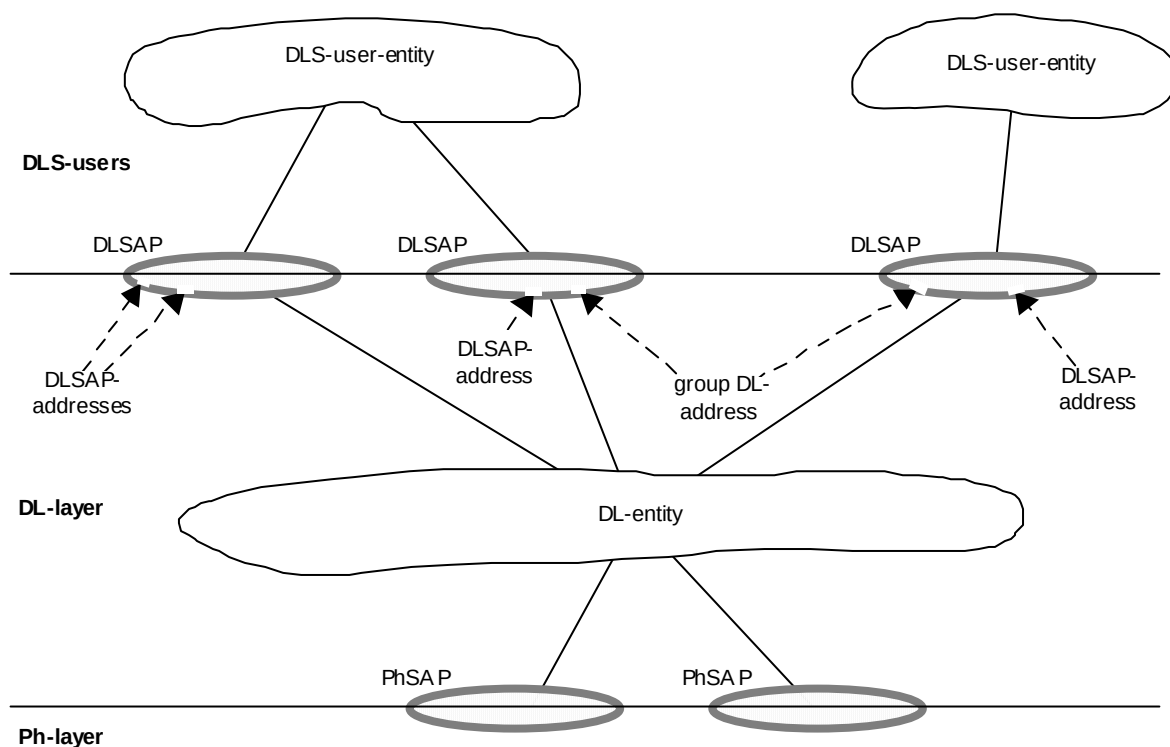
single DL-subnetwork in which any of the connected DLEs may communicate directly, without any intervening DL-relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication

3.3.5

DLSAP

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

Note 1 to entry: 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 can have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

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

3.3.6

DLSAP

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

Note 1 to entry: 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).

3.3.7

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 1 to entry: 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.

3.3.8

(individual) DLSAP-address

DL-address that designates only one DLSAP within the extended link

Note 1 to entry: A single DL-entity can have multiple DLSAP-addresses associated with a single DLSAP.

3.3.9

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 1 to entry: An extended link can be composed of just a single link.

3.3.10

FCS error

error that occurs when the computed frame check sequence value after reception of all the octets in a DLPDU does not match the expected residual

3.3.11

frame

denigrated synonym for DLPDU

3.3.12

group DL-address

DL-address that potentially designates more than one DLSAP within the extended link

Note 1 to entry: A single DL-entity may have multiple group DL-addresses associated with a single DLSAP.

Note 2 to entry: A single DL-entity also may have a single group DL-address associated with more than one DLSAP.

3.3.13

high-speed cyclic data

data conveyed by means of high-speed cyclic data transmission

3.3.14

high-speed cyclic data transmission

highest priority of time-critical cyclic data service

3.3.15

implicit token

mechanism that governs the right to transmit

Note 1 to entry: No actual token message is transmitted on the medium. Each node keeps track of the node that it believes currently holds the right to transmit. The right to transmit is passed from node to node by keeping the node that last transmitted. A slot time is used to allow a missing node to be skipped in the rotation.

3.3.16

low-speed cyclic data

data conveyed by means of low-speed cyclic data transmission

3.3.17

low-speed cyclic data transmission

lowest priority of time-critical cyclic data service

3.3.18

medium-speed cyclic data

data conveyed by means of medium-speed cyclic data transmission

3.3.19

medium-speed cyclic data transmission

second-highest priority of time-critical cyclic data service

3.3.20

multi-peer DLC

centralized multi-end-point DL-connection offering DL-duplex-transmission between a single distinguished DLS-user known as the publisher or publishing DLS-user, and a set of peer but undistinguished DLS-users known collectively as the subscribers or subscribing DLS-users

Note 1 to entry: The publishing DLS-user can send to the subscribing DLS-users as a group (but not individually), and the subscribing DLS-users can send to the publishing DLS-user (but not to each other).

3.3.21

multipoint connection

connection from one node to many nodes

Note 1 to entry: Multipoint connections allows data transfer from a single publisher to be received by many subscriber nodes.

3.3.22

node

single DL-entity as it appears on one local link

3.3.23

node DL-address

DL-address which designates the (single) DL-entity associated with a single node on a specific local link

3.3.24

node-id

two-octet primary identifier for the DLE on the local link, whose values are constrained

Note 1 to entry: A permissible value is from 1 to 255. A value 0 is specifically used for SYN node, which emits SYN frame.

3.3.25

receiving DLS-user

DL-service user that acts as a recipient of DL-user-data

Note 1 to entry: A DL-service user can be concurrently both a sending and receiving DLS-user.

3.3.26**sending DLS-user**

DL-service user that acts as a source of DL-user-data

3.3.27**slot-time**

512-bit time and 4 096-bit time of the physical signaling symbol for the data rate of 100 Mbps and 1 000 Mbps respectively specified in the ISO/IEC 8802-3:2000, Clause 29

3.3.28**sporadic message data service**

aperiodic message transfer which sporadically occurs upon DLS-user requesting one or more message to transfer, and regular ISO/IEC 8802-3 Ethernet message frame is transferred by means of this message transfer

3.3.29**SYN node**

node transmitting SYN frame

3.3.30**time-critical cyclic data service**

cyclic data transfer with three levels of data transmission at the same time, of which each data transmission level is according to the data priority and the data transmission period for real-time delivery, and of which data transmission period and total data volume for each level can be specified in designing phase and on application needs

3.3.31**token**

right to transmit on the local link

3.4 Symbols and abbreviations

ACM	Access control machine
CLM	Claim frame
CMP	Transmission complete frame
CTRC	Cyclic-transmission TX/RX control
COM	Command frame
DT	Cyclic data frame
CW	Control word (as parameter of SYN DLPDU)
DT-CMP	DT with transmission complete frame
FC	Frame control field
LL	Live list (as parameter of SYN DLPDU)
Mbps	Million bits per second
PhIDU	Ph-interface-data-unit

PM	Periodic mode (as parameter of SYN DLPDU)
PN	Transmission permits node number (as parameter of SYN DLPDU)
Pri	Priority field
RAS	RAS frame
REC	In-ring request frame
RMC	Redundancy medium control
RMSEL	Redundant medium selection
RN	Recipient node number
SN	Source node number field
ST	Slot time (as parameter of SYN DLPDU)
SYN	Synchronization frame
Th	High-speed transmission period (as transmission period)
Tm	Medium-speed transmission period (as transmission period)
Ts	Sporadic-speed transmission period (as transmission period)
TI	Low-speed transmission period (as transmission period)

4 Overview of the DL-protocol

4.1 General

This standard meets the industrial automation market objective of providing predictable time deterministic and reliable time-critical data transfer and means, which allow co-existence with non-time-critical data transfer over the ISO/IEC 8802-3 series communications media, for support of cooperation and synchronization between automation processes on field devices in a real-time application system. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty.

4.1.1 Field of applications

In industrial control systems, several kinds of field devices such as Drives, Sensors and Actuators, Programmable controllers, Distributed Control Systems and Human Machine Interface devices are required to be connected with control networks. The process control data and the state data is transferred among these field devices in the system and the communications between these field devices requires simplicity in application programming and to be executed with adequate response time. In most industrial automation systems such as food, water, sewage, paper and steel, including a rolling mill, the control network is required to provide time-critical response capability for their application, as required in ISO/TR 13283 for time-critical communications architectures.

Plant production may be compromised due to errors, which could be introduced to the control system if the network does not provide a time-critical response. Therefore, the following characteristics are required for a time-critical control network:

- deterministic response time between the control device nodes;
- ability to share process data seamlessly across the control system.

This protocol is applicable to such an industrial automation environment, in which time-critical communications is primarily required. The term “time-critical” is used to represent the presence of a time window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time-window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

4.2 Overview of the medium access control

The Type 11 fieldbus has a deterministic medium access control in order to avoid collisions that occur when a number of the nodes send data frames simultaneously, and to provide the opportunity of sending data to each node in a sequential order and within a predetermined time period. Figure 2 shows the basic principle of medium access control of the Type 11 fieldbus.

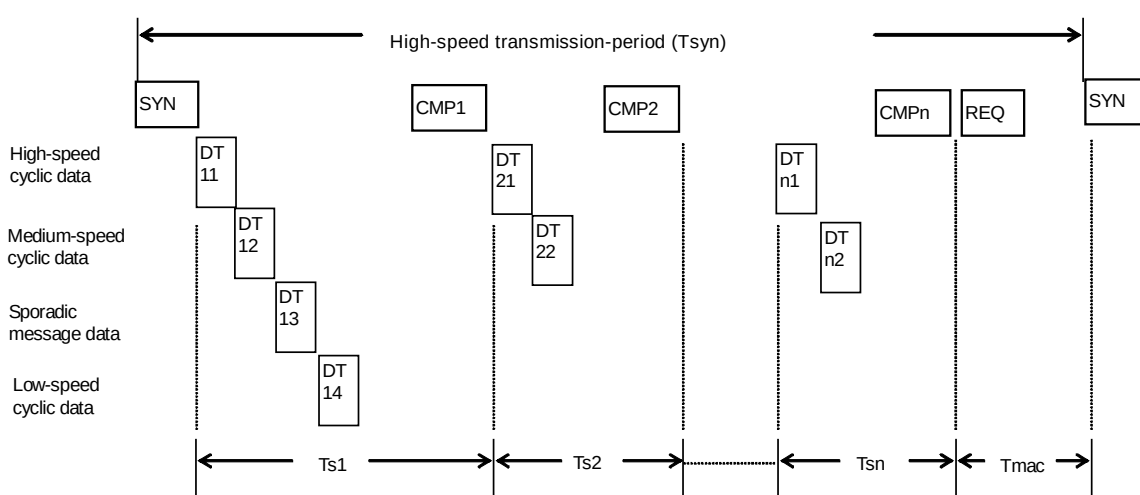


Figure 2 – Basic principle of medium access control

At the start time of every high-speed-transmission-period (T_{syn}), the SYN frame is broadcasted to all nodes. Receiving the SYN frame, the node with sequential number 1 starts sending its data frames, and after that broadcasts its CMP frame in order to indicate the completion of its data frames transmission. The Nth node can send out its data frames after receiving the CMP frame from the (N-1) node. After all the nodes send their data frames, the time period to solicit new nodes begins. The REQ frame is used for a new node requesting inclusion to the Type 11 fieldbus network. The sequential number is assigned to a new node at the time approval to join is granted.

Each node can hold the transmission right for a preset time and shall send its CMP frame to transfer the transmission right to the next node within the preset time. The data to be sent and the data to be held over are determined by priority.

Transmission includes Time-critical cyclic data and sporadic Ethernet message transmission. Cyclic data transmission is divided into High-, Medium- and Low-speed data transmission. Each node sends the High-speed cyclic data frames on each occasion when it obtains the transmission right. The data of lower priorities, that is the Medium-speed cyclic data, the sporadic Ethernet message data and the Low-speed cyclic data respectively, is sent or not

sent depending on the circumstances. The holding time of the transmission right of each node is determined by the settings of the High-speed cyclic, the Medium-speed cyclic, the sporadic Ethernet message and the Low-speed cyclic data transmission periods and by the volume of transmission data for each node. After sending all the High-speed cyclic data, the node sends the Medium-speed cyclic data. If the holding time of the transmission right ends during sending the Medium-speed cyclic data, the transmission of the Medium-speed cyclic data is interrupted, and the CMP frame is sent. The n^{th} node obtains the transmission right again during the next High-speed data transmission period, during which time all the High-speed cyclic data and the remainder of the previous Medium-speed cyclic data is sent. Tmac is the period for a new node sending out REQ frame to enter the network.

Two kinds of the architecture, that are the star- and the loop-architecture, are specified in 4.4. Particularly in the loop-architecture the order of priority for the sporadic Ethernet message data and the Low-speed cyclic data changes with the capability for sending out the Low-speed cyclic data in the 100 Mbps and the 1 000 Mbps operation. The order of priority for the sporadic Ethernet message data is of the lowest, and the Low-speed cyclic data is not capable in the 100 Mbps operation.

4.3 Service assumed from the PhL

4.3.1 General

Subclause 4.3 describes the assumed Physical Service (PhS) and the constraints on use by the Type 11 DLE. The Physical Service is assumed to provide the following service primitives specified by ISO/IEC 8802-3:2000, Clause 6.

4.3.2 Assumed primitives of PhS

The PhS is assumed to provide the following two categories of primitives to the Type 11 DL-protocol.

- service primitives for transmitting and receiving frames to / from other peer DLEs;
- service primitives that provide information needed by the local DLE to perform the media access functions.

The assumed primitives of PhS are grouped into these two categories:

- a) Transfer of Data to all other peer DLE:
 - PLS_DATA request;
 - PLS_DATA indication;
- b) Media access management by local DLE:
 - PLS_CARRIER indication;
 - PLS_SIGNAL indication;
 - PLS_DATA_VALID indication.

The interaction of the PhS primitives to the DLE is shown in Figure 3.

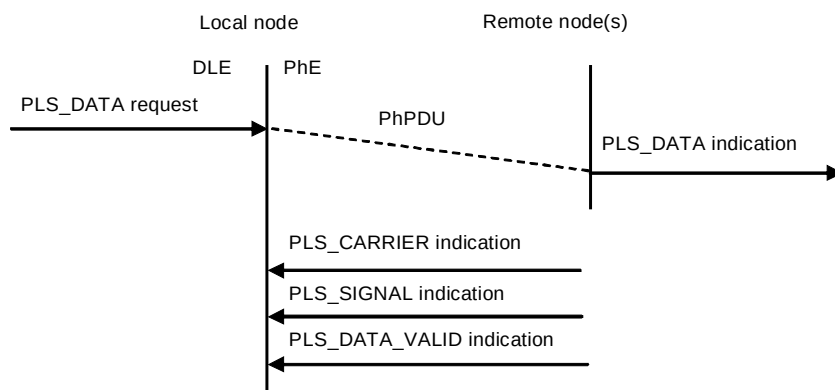


Figure 3 – Interaction of PhS primitives to DLE

4.4 DLL architecture

4.4.1 Overview

The Type 11 fieldbus DLL is modeled as a combination of control components of Access Control Machine (ACM), Cyclic transmission TX/RX Control (CTRC), Sporadic TX/RX Control (STRC), Redundancy Medium Control (RMC), Serializer, Deserializer and DLL management interface.

The Access Control Machine as the primary control component provides the function for deterministic medium access control cooperating with the Cyclic-transmission TX/RX Control, the Sporadic TX/RX Control and the Redundancy Medium Control for reliable and efficient support of higher-level connection-mode and connectionless data transfer services. Specifically the Access Control Machine has the primary responsibility for

- assuring that the local node detects and fully utilizes its assigned access time period;
- assuring that the local node does not interfere with the transmissions of other nodes, especially of the node transmitting the SYN frame;
- detecting network disruption, and initiating the SYN frame transmission for restoration of the network disruption from after prescribed time duration in which the SYN frame is not heard;
- assuring a new node adding to and removing from the network.

The DLL management interface provides DLL management functions. PhL framing and delimiters are managed by DLL functions for serializing and deserializing M_symbol requests and indications.

4.4.2 Star-architecture

The Data-link layer is comprised of the components listed in Table 1. The internal arrangement of these components, and their interfaces, are shown in Figure 4. The arrowheads illustrate the primary direction of flow of data and control.

Table 1 – Data-link layer components of star-architecture

Components	Description
Access control machine (ACM)	Deterministic medium access control and scheduling the opportunities to sending out the DLPDUs, control for adding and removing nodes, restoration from disruption. Assembles and transmits the DLPDUs to the TX framer through the RMC, receives and disassembles the DLPDUs with the control information from the RMC, and determines the timing and duration of the transmissions. Signals the RMC to sending out the DLPDUs for control and diagnosis of the Type 11 fieldbus. Buffers and dispatches in time the DLPDUs received between the DLS-user and the RMC
Cyclic transmission TX/RX control (CTRC)	Buffers and dispatches in time DLSDU received for the time-critical cyclic data transfer between the DLS-user and the ACM
Sporadic TX/RX control (STRC)	Buffers and dispatches in time DLSDU received for the Sporadic message data between DLS-user and the ACM
Redundancy medium control (RMC)	Receives the DLPDUs from the ACM and breaks them down into octet symbol requests to the Octet serializer, assembles received octets from the Octet deserializer into DLPDUs and submits them to the ACM. Selects one of two outputs of the Octet deserializers through the RX framers for medium redundancy
TX/RX framer	Receives the octet symbols from the RMC, detects and indicates the start timing of a DLPDU to the Octet serializer, passes received octets from the Octet deserializer and indicates the start timing of a DLPDU to the RMC
Octet serializer	Receives octet symbols, encodes and serializes them, and sends them as M_symbols to the PhL. It is also responsible for generating the FCS
Octet deserializer	Receives M_symbols from the PhL, converts M_symbols into octets and sends them to the RX framer. It is also responsible for checking the FCS
DLL management interface	Holds the station management variables that belong to the DLL, and manages synchronized changes of the link parameters

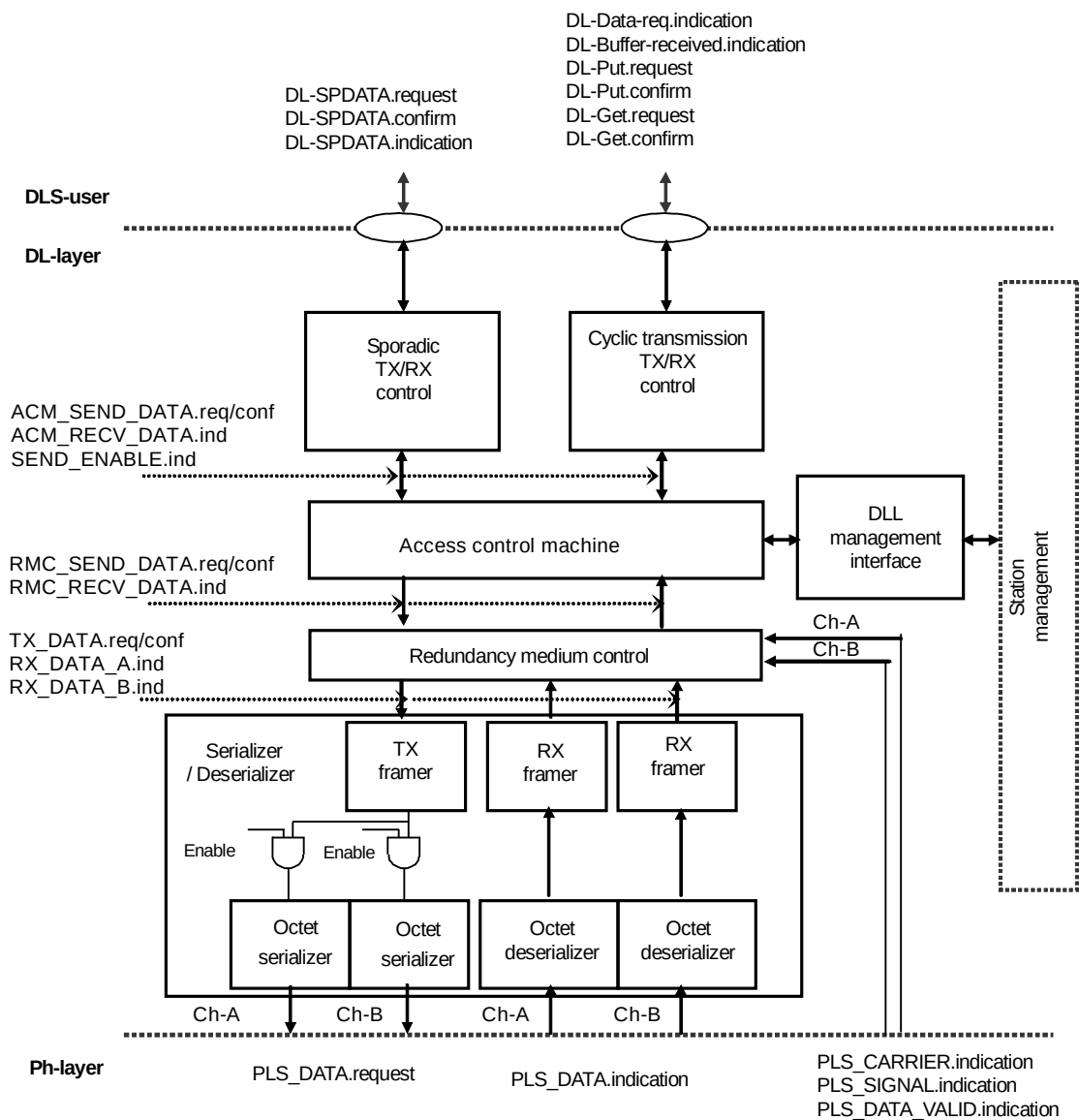


Figure 4 – Data-link layer internal architecture of star-architecture

4.4.3 Loop-architecture

The Data-link layer is comprised of the components listed in Table 2. The internal arrangement of these components, and their interfaces, are shown in Figure 5. The arrowheads illustrate the primary direction of flow of data and control.

Table 2 – Data-link layer components of loop-architecture

Components	Description
Access control machine (ACM)	Deterministic medium access control and scheduling the opportunities to sending out the DLPDUs, control for adding and removing nodes, restoration from disruption. Assembles and transmits the DLPDUs to the TX framer through the RMC, receives and disassembles the DLPDUs with the control information from the RMC, and determines the timing and duration of the transmissions. Signals the RMC to sending out the DLPDUs for control and diagnosis of the Type 11 fieldbus. Buffers and dispatches in time the DLPDUs received between the DLS-user and the RMC
Cyclic transmission TX/RX control (CTRC)	Buffers and dispatches in time DLSDU received for the time-critical cyclic data transfer between the DLS-user and the ACM
Sporadic TX/RX control (STRC)	Buffers and dispatches in time DLSDU received for the Sporadic message data between DLS-user and the ACM
Redundancy medium control (RMC)	Receives the DLPDUs from the ACM and breaks them down into the octet-symbol requests to the TX-framer, assembles received-octets from the RX-framers into the DLPDUs, selects one of two outputs from the RX-framers and submits them to the ACM. Receives the signal from the ACM and activates the transmission of the DLPDU frames for control and diagnosis of the Type 11 fieldbus
TX/RX framer	TX framer receives the octet-symbols from the RMC, detects and indicates the start timing of a DLPDU to the Octet-serializer, forwards received-octets from the Octet-deserializer connected to a Ring-port to other Octet-serializer connected to other Ring-port. RX framer passes received-octets from the Octet-deserializer and indicates the start-delimiter of a DLPDU to the RMC
Octet serializer	Receives octet symbols, encodes and serializes them, and sends them as M_symbols to the PhL. It is also responsible for generating the FCS
Octet deserializer	Receives M_symbols from the PhL, converts M_symbols into octets and sends them to the RX framer and the TX framer. It is also responsible for checking the FCS
DLL management interface	Holds the station management variables that belong to the DLL, and manages synchronized changes of the link parameters

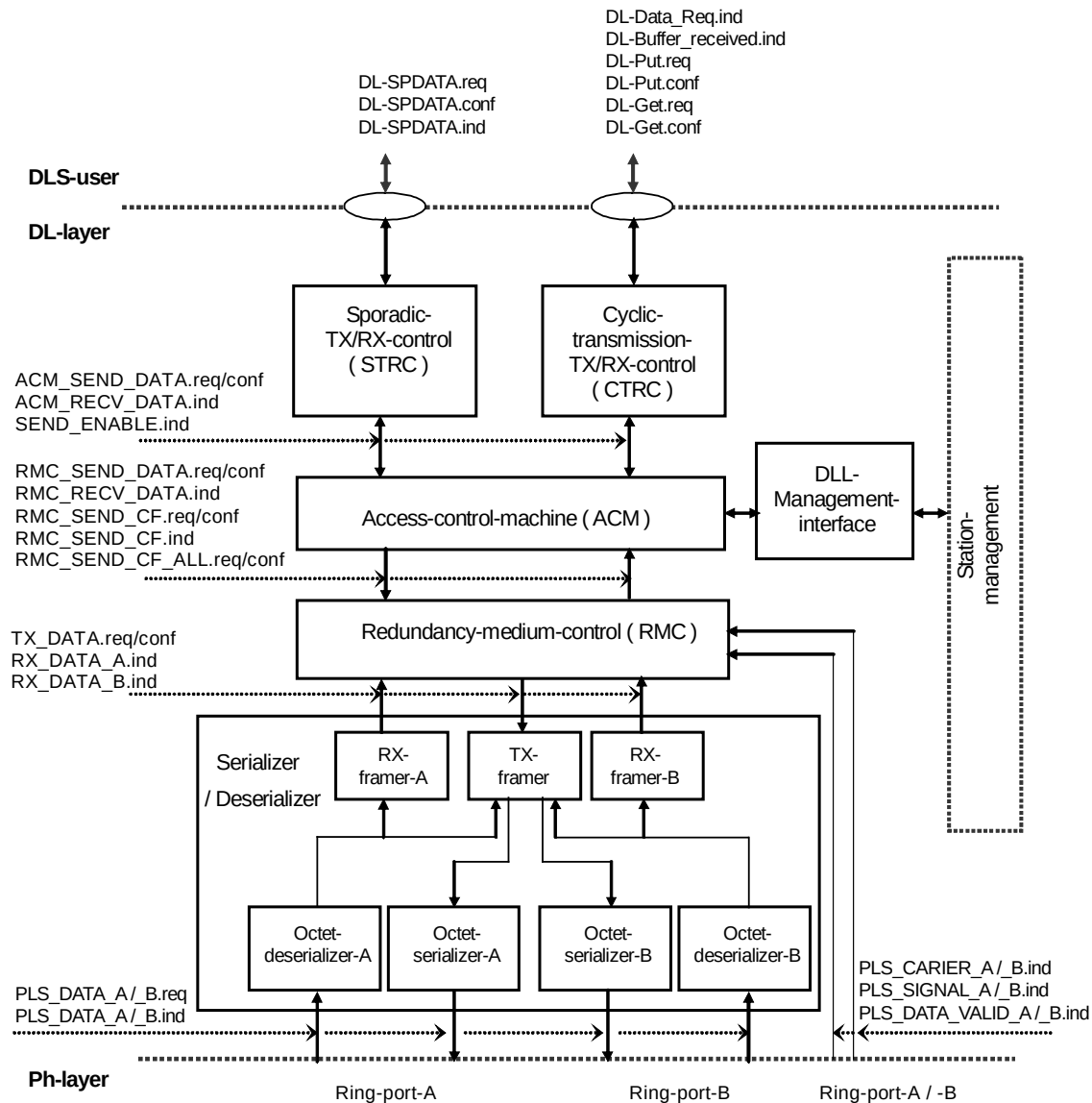


Figure 5 – Data-link layer internal architecture of loop-architecture

4.5 Access control machine and schedule support functions

The ACM functions schedule all communications between the DLEs participating in the Type 11 fieldbus, and the timing of this communications is controlled as to

- fulfill the specific medium access control to give all the DLEs the opportunities to send out two kinds of class of Time-critical cyclic data and Sporadic message data in timely, prioritized and deterministic fashion, and to detect network disruption and to initiate the restoration in appropriate time, further to add and remove nodes on line;
- provide multiple levels of Time-critical data transfer opportunities of sending data to node in sequential order and within each pre-specified time period, and that the data transfer of each level is performed within the pre-specified time duration (token holding time) and whether the data transfer of lower levels to be carried out or to be held over to later cyclic time period depends on the level and the occasion though the top level of the data transfer is always carried out at every occasion; on the other hand the whole volume of the data transfer of lower levels is transferred within each pre-specified time period;

- c) provide sporadic message data transfer opportunities of sending out to node that the request to transfer is happened sporadically by the DLS-user, and the data transfer is performed in pre-specified time period of the corresponding level of priority and is based on regular ISO/IEC 8802-3 applications.

Accurate scheduling timing is very important to support many control and data collection tasks in the applications domain of this protocol.

4.6 Local parameters, variables, counters, timers and queues

4.6.1 Overview

4.6.1.1 General

This standard uses DLS-user request parameters P(...) and local variables V(...) as a means of clarifying the effect of certain actions and the conditions under which those actions are valid, local timers T(...) as a means of monitoring actions of the distributed DLS-provider and of ensuring a local DLE response to the absence of those actions, and local counters C(...) for performing rate measurement functions. It also uses local queues Q(...) as a means of ordering certain activities, of clarifying the effects of certain actions, and of clarifying the conditions under which those activities are valid.

Unless otherwise specified, at the moment of their creation or of DLE activation:

- a) all variables shall be initialized to their default value, or to their minimum permitted value if no default is specified;
- b) all counters shall be initialized to zero;
- c) all timers shall be initialized to inactive;
- d) all queues shall be initialized to empty.

DL-management may change the values of configuration variables.

4.6.1.2 Summary of variables, parameters, counters, timers for star-architecture

Table 3 and Table 4 summarize the variables and their usage.

The data types used in the data link variables, parameters, state machines are specified in IEC 61158-5-11, Clause 5. The most significant bit of the most significant octet is always used as the most significant bit of the binary number. The bits of each octet are numbered 0 to 7, where bit 0 is the least significant bit of the octet. Each octet is transferred to the Ph layer low-bit first and multiple octets is transferred low-order octet first (little endian format).

Table 3 – DLE-variables and permissible values of star-architecture

Variable -name	Description	Permissible values	Data type
BW	Length of TCC data word in a DLPDU	The default value is 128 (octets). See 4.6.2.7	UINT
DR	Data-rate	The default value is 100 (Mbps). See 4.6.2.11	UINT
HTh	Maximum observational time period for detecting the continuous high-speed transmission cycle disrupted	2 to $2^{16}-1$ (in units of 1 ms). The default value is $3 \times V(Th)$. See 4.6.2.12	UDINT
HTI	Maximum observational time period for detecting the continuous low-speed transmission cycle disrupted	2 to $2^{16}-1$ (in units of 1 ms). The default value is $3 \times V(Th)$. See 4.6.3.5	UDINT
HTm	Maximum observational time period for detecting the continuous medium-speed transmission cycle disrupted	2 to $2^{16}-1$ (in units of 1 ms). The default value is $3 \times V(Th)$. See 4.6.2.14	UDINT
IA	Individual address of this node	Individual 48-bit address identical to ISO/IEC 8802-3. See 4.6.2.16	USINT[6]
IGP	Time interval from the end of a previous frame to the start of the consecutive frame sent by a node	The default value is 24 (0,96 μ s in units of 0,04 μ s). See 4.6.2.17	UDINT
IP	IP address of this node	32-bit address identical to IPv4. See 4.6.2.18	USINT[4]
MD	Maximum distance on the connection path between any 2 nodes	1 to 100 (km). The default value is 8. See 4.6.2.21	UINT
MDD	Maximum difference of the distance of two redundant physical mediums on the connection path between any 2 nodes	0 to 2 000 (in m). The default value is 500. See 4.6.3.7	UINT
MGA	Multicast group address	Multicast 48-bit address identical to ISO/IEC 8802-3, for logically associated Type 11 fieldbus nodes. The default value is 0x01-0x00-0x5e-0x50-0x00-0x01. See 4.6.2.22	USINT[6]
MN	Maximum node number	254 (default value). See 4.6.2.23	USINT
MPD	Maximum difference of the signal delay time propagating over two redundant physical mediums on the connection path between any 2 nodes	1 to 0x7c (in units of 0,04 μ s). The default value is 0x7c ($\approx 5 \mu$ s). See 4.6.3.8	UDINT
MRT	Maximum number of the repeater units on the connection path between any 2 nodes	0 to 7. The default value is 3. See 4.6.2.24	UINT
MTHT	Maximum-token-hold-time for high-speed cyclic data transmission	1 to $2^{16}-1$ (octet times). The default value is 0x30B4 (= 900 μ s). See 4.6.2.25	UDINT
PBh	List of up to 2 048 data buffers using for sending out high-speed TCC data	1 to 2 048. See 4.6.2.28	UINT
PBI	List of up to 2 048 data buffers using for sending out low-speed TCC data	1 to 2 048. See 4.6.3.10	UINT
PBm	List of up to 2 048 data buffers using for sending out medium-speed time-critical cycle data	1 to 2 048. See 4.6.2.29	UINT
RCS	Receive-channel switching control for receiving frames. Designates the switching control for receiving frames out of one of two receive-channel A and B corresponding to each of the redundant media A and B	“Automatic”, “Force A”, “Force B” “Automatic”: Automatically switch to the proper receive-channel. “Force A”, “Force B”: Force to switch Receive-channel A or B respectively. The default is “Automatic”. See Table 10, Table 12, 4.6.3.12	BOOL
RMGP	Maximum time-interval for one receive-channel, which has already completed one frame received and has waited for IGP time, to wait for the completion of a frame to be received on the other receive-channel in order to detect the other receive-channel disrupted	250 (in units of 0,04 μ s). The value 250 represents 10 μ s. See 4.6.3.14	UDINT

Variable -name	Description	Permissible values	Data type
ST	The slot-time is the fundamentally observational time unit, used in the DLME for observing to initiate action such as re-initialization in sending out the CLM frame, and also used in the DLME for observing the CMP frame sent by the corresponding node, and in case that the node failed to send out, used in the DLME for initiating action just as the CMP frame is received	1 to 255 (in units of physical symbol times for 512 bits, identical to ISO/IEC 8802-3). The default value is 20 (= 100 µs). See Table 14, 4.6.2.38	USINT
SCMP	Medium-access-control-time for substitute CMP	1 to 255 (in units of physical symbol times for 512 bits, identical to ISO/IEC 8802-3). The default value is 20 (= 100 µs). See 4.6.2.31	USINT
SCMPL	Permissible repetitive count within which the node can behave like that the node received the CMP frame even though the corresponding node had failed to send out the CMP frame. The node determines the corresponding node is out of service if the count is over	1 to 16. The default value is 3. See 4.6.2.33	USINT
SEN _A SEN _B	Send enable control for sending frames out of the transmitter-A or –B	"True" / "Enable" or "False" / "Disable" The default value is "True" / "Enable". See 4.6.3.16	BOOL
Th	High-speed transmission period	1 to 160 (in units of 0,1 ms). See Table 15, 4.6.2.39	USINT[3]
Tl	Low-speed transmission period	1 to 10 000 (in units of 1 ms). See Table 18, 4.6.3.17	UINT
TISPD	Time-interval cyclically processed for Sporadic message data service	1 to 1 000 (in units of 1 ms). The default value is 100. See Table 17, 4.6.2.40, 4.6.2.41	UINT
Tm	Medium-speed transmission period	1 to 1 000 (in units of 1 ms). The default value is 100. See Table 16, 4.6.2.42	UINT
TMAC	Maximum observation time period for the SYN node to solicit new nodes into the Type 11 fieldbus	1 to 100 (in units of 0.1 ms). The default value is 100. See 4.6.2.43, 4.6.2.44	USINT
Tsl	DLE number able to send out next slot	1 to 255. See 4.6.2.46	USINT
TSYN	Target-periodic-time from SYN frame arrival to SYN frame arrival	1 to 160 (in units of 0,1 ms). See 4.6.2.47	UDINT
TTRT0	Target-token-rotation-time for access class 0	1 to 2 ¹⁶ -1 (octet time). The default value is P(TSYN). See 4.6.3.18	UDINT
TTRT1	Target-token-rotation-time for access class 1	1 to 2 ¹⁶ -1 (octet time). The default value is P(TSYN). See 4.6.2.48	UDINT
TTRT2	Target-token-rotation-time for access class 2	1 to 2 ¹⁶ -1 (octet time). The default value is P(TSYN). See 4.6.2.49	UDINT
TN	Node identifier number of the node. The initial value is 0 on power-up or reset	1 to 255. See 4.6.2.45	USINT

Table 4 – Observable variables and their value ranges of star-architecture

Variable -name	Description	Range of value	Data type
AIGP _A AIGP _B	IGP monitor over the receive-channel A and B	0 to 24, continuously decrementing. The value 24 represents 0,96 µs. See 4.6.3.1	UDINT
ARMGP	The observed time period for one receive-channel, which has already completed one frame received and has waited for IGP time (0,96 µs), to wait the completion of a frame received on the other receive-channel in order to detect the other receive-channel disrupted	0 to 250, continuously decrementing. The value 250 represents 10 µs. See 4.6.3.2	UDINT
ASCMP	Collection of counters of total number of nodes. Each counter corresponding to a node indicates observed repetitive count that the corresponding node failed to send out the CMP frame and the other node can behave like that the CMP frame received without error. The node determines the corresponding node is out of service when the count reaches the pre-specified number. The number counted is indicated and is incremented coincidentally at each DLME	1 to 16. The default value is equal to the value of SCMPL, which is 3. See 4.6.2.1	USINT
ATHT	The observed time period of the MTHT	0 to MTHT, continuously decrementing. See 4.6.2.2	UDINT
ATSYN	The observed time period from SYN frame arrival to SYN frame arrival	1 to 2 ¹⁶ (octet times). See 4.6.2.3	UDINT
ATTRT0	The observed time period of the TTRT0	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.3.3	UDINT
ATTRT1	The observed time period of the TTRT1	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.2.4	UDINT
ATTRT2	The observed time period of the TTRT2	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.2.5	UDINT
CDh	Cumulative count of transmitted high-speed-cyclic data frames	0 to 2 ³² -1. See 4.6.2.8	UDINT
CDH _{blk}	Information data status of the data buffer with the identifier number "blk", indicating the corresponding TCC data being active (healthy) or inactive (unhealthy)	True: active. False: inactive. See 4.6.2.9	BOOL
CDl	Cumulative count of transmitted low-speed-cyclic data frames	0 to 2 ³² -1. See 4.6.3.4	UDINT
CDm	Cumulative count of transmitted medium-speed-cyclic data frames	0 to 2 ³² -1. See 4.6.2.10	UDINT
LL	Live list indicating whether the corresponding node, at this moment, is connected to and running in the Type 11 fieldbus domain in the received SYN frame. The Live list is a set of 256 Booleans, represented in 32 octets, in which each bit corresponds to a node in the Type 11 fieldbus domain and indicates the current status of that node	256-bit set of Booleans True: node is connected and working. False: node is not connected or not working. See 4.6.2.19	USINT[16]
LN	Extracted number from LL at each node and used by each node to decide whether the node is able to send the data frame out over the medium	1 to 255. See 4.6.2.20	USINT
NCD _A , NCD _B	Cumulative count of No-Carrier detected on the receive-channel A or B	0 to 2 ³² -1. See 4.6.3.9	UDINT
NONC	Permissible repetitive count of no CMP frame received by the SYN node within the corresponding consecutive Tsyn cycles, that is 256 times by SCMPL, in order to detect no other node except the SYN node in the Type 11 fieldbus domain	1 to 16. The default value is 3. See 4.6.2.26, 4.6.2.27	USINT

Variable -name	Description	Range of value	Data type
RE _A , RE _B	Cumulative count of DLPDUs received in error on the receive-channel A or B. RE _A is incremented when an error is detected on receive-channel A while receive-channel B is error-free. RE _B corresponds, but with interchanged channel roles.	0 to 2 ³² -1. See 4.6.3.13	UDINT
ROK _A , ROK _B	Cumulative count of DLPDU-received without error on the receive-channel A or B	0 to 2 ³² -1. See 4.6.3.15	UDINT
SL	The observed time period of the SL	0 to V(SL). See 4.6.2.35, 4.6.2.36	USINT
SD	Cumulative count of transmitted sporadic data frames	0 to 2 ³² -1. See 4.6.2.34	UDINT

4.6.1.3 Summary of variables, parameters, counters, timers for loop-architecture

Table 5 and Table 6 summarise the variables and their usage. The data types used in the data link variables, parameters, state machines are specified in IEC 61158-5-11, Clause 5.

The data types used in the data link variables, parameters, state machines are specified in IEC 61158-5-11, Clause 5. The most significant bit of the most significant octet is always used as the most significant bit of the binary number. The bits of each octet are numbered 0 to 7, where bit 0 is the least significant bit of the octet. Each octet is transferred to the Ph layer low-bit first and multiple octets is transferred low-order octet first (little endian format).

Table 5 – DLE variables and permissible values of loop-architecture

Variable -name	Description	Permissible values	Data type
BW	Length of TCC data word in a DLPDU	16 to 128. The default value is 64. See 4.6.2.7	UINT
DR	Data-rate	The value is 100 or 1 000 (Mbps). See 4.6.2.11	UINT
ESYN	Enable SYN	“Enable” or “Disable”. “Enable” is capable of SYN node. “Disable” is not capable of SYN node. The default value is “Enable”. See 6.4.3, 4.6.4.5	
FWE _{AB} , FWE _{BA}	Forwarding control for receiving frame enable to each direction to/from Ring-port-A and -B. FWE _{AB} controls forwarding frames receiving from Ring-port-A to Ring-port-B. On the other hand, FWE _{BA} controls forwarding frames receiving from Ring-port-B to Ring-port-A	“Enable” or “Disable”. “Enable”: permit forwarding frames receiving from one Ring-port to other Ring-port. “Disable”: prohibit forwarding frames between Ring-ports. The default value is “Disable”. See 6.9.3, 4.6.4.6	BOOL
HTh	Maximum observational time period for detecting the continuous high-speed transmission cycle disrupted	2 to 2 ¹⁶ -1 (in units of: – 1 ms for 100 Mbps operation; – 0,1 ms for 1 000 Mbps operation). The default value is 3 x V(Th). See 4.6.2.12	UDINT
HTI	Maximum observational time period for detecting the continuous low-speed transmission cycle disrupted	2 to 2 ¹⁶ -1 (in units of 1 ms). The default value is 3 x V(Th). See 4.6.4.7	UDINT
HTm	Maximum observational time period for detecting the continuous medium-speed transmission cycle disrupted	2 to 2 ¹⁶ -1 (in units of 1 ms). The default value is 3 x V(Th). See 4.6.2.14	UDINT
IA	Individual address of this node	Individual 48-bit address identical to ISO/IEC 8802-3. See 4.6.2.16	SWORD[6]

Variable -name	Description	Permissible values	Data type
IGP	Time interval from the end of a previous frame to the start of the consecutive frame sent by a node	The default value is: – 24 for 100 Mbps operation; – 12 for 1 000 Mbps operation. See 4.6.2.17	UDINT
IP	IP address of this node	32-bit address identical to IPv4. See 4.6.2.18	USINT[4]
MD	Maximum distance on the connection path between any 2 nodes	1 to 20 (km). The default value is 4. See 4.6.2.21	UINT
MDAD	Maximum distance between adjacent node	The value is: – 1 to 2 (km) for 100 Mbps operation; – .1 to 5 (km) for 1 000 Mbps operation. The default value is 1. See 4.6.4.11	UINT
MGA	Multicast group address	Multicast 48-bit address identical to ISO/IEC 8802-3, for logically associated Type 11 fieldbus nodes. The default value is 0x01-0x00-0x5e-0x50-0x00-0x01. See 4.6.2.22	USINT[6]
MN	Maximum node number	254 (default value). See 4.6.2.23	USINT
MRT	Maximum number of the repeater units on the connection path between any 2 nodes	0 to 252. The default value is 30. See 4.6.2.24	USINT
MTHT	Maximum-token-hold-time for high-speed cyclic data transmission	1 to $2^{16}-1$ (octet time). The default value is 0x30B4 (= 900 µs for 100 Mbps and 90 µs for 1 000 Mbps operation). See 4.6.2.25	UDINT
PBh	List of data buffers using for sending out high-speed TCC data	The value is: – 1 to 2 048 for 100 Mbps operation; – 1 to 4 096 for 1 000 Mbps operation. See 4.6.2.28	UINT
PBl	List of data buffers using for sending out low-speed TCC data	The value is: – 1 to 2 048 for 100 Mbps operation; – 1 to 4 096 for 1 000 Mbps operation. See 4.6.4.16	UINT
PBm	List of data buffers using for sending out medium-speed TCC data	The value is: – 1 to 2 048 for 100 Mbps operation; – 1 to 4 096 for 1 000 Mbps operation. See 4.6.2.29	UINT
PS _A PS _B	Port state control for blocking or non-blocking frame receiving from the Ring-port-A or –B	“Blocked” or “Non-blocked”. The default value is “Blocked”. See Table 27, 4.6.4.18	BOOL
REN _A REN _B	Receive enable control for receiving frames from the Ring-port-A or –B.	“Enable” or “Disable”. The default value is “Enable”. See Table 28, 4.6.4.20	BOOL
ST	The slot-time is the fundamentally observational time unit, used in the DLME for observing to initiate action such as re-initialization in sending out the CLM frame, and also used in the DLME for observing the CMP frame sent by the corresponding node, and in case that the node failed to send out, used in the DLME for initiating action just as the CMP frame is received	1 to 255 (in units of physical symbol times for 512 bits in 100 Mbps operation and for 4 096 bits in 1 000 Mbps operation, identical to ISO/IEC 8802-3). The default value is 20 (= 100 µs for 100 Mbps, and 8 µs for 1 000 Mbps operation). See Table 14, 4.6.2.38	USINT
SCMP	Medium-access-control-time for substitute CMP	1 to 255 (in units of physical symbol times for 512 bits in 100 Mbps operation and for 4 096 bits in 1 000 Mbps operation, identical to ISO/IEC 8802-3). The default value is 20 (=100 µs for 100 Mbps, and 8 µs for 1 000 Mbps operation). See 4.6.2.31	USINT

Variable -name	Description	Permissible values	Data type
SCMPL	Permissible repetitive count within which the node can behave like that the node received the CMP frame even though the corresponding node had failed to send out the CMP frame. The node determines the corresponding node is out of service if the count is over	1 to 16. The default value is 3. See 4.6.2.33	USINT
SEN _A SEN _B	Send enable control for sending frames out of the Ring-port-A or –B	“True” / “Enable” or “False” / “Disable”. The default value is “True” / “Enable”. See Table 25, 4.6.4.22	BOOL
Th	High-speed transmission period	1 to 160 (in units of 0,1 ms). See Table 15, 4.6.2.39	USINT[3]
Tl	Low-speed transmission period	1 to 10 000 (in units of 1 ms). The default value is 100. See Table 18, 4.6.4.23	UINT
TISPD	Time-interval cyclically processed for Sporadic message data service	1 to 1 000 (in units of 1 ms). The default value is 100. See Table 17, 4.6.2.40, 4.6.2.41	UINT
Tm	Medium-speed transmission period	10 to 1 000 (in units of 1 ms). The default value is 100. See Table 16, 4.6.2.42	UINT
TMAC	Maximum observation time period for the SYN node to solicit new nodes into the Type 11 fieldbus	1 to 100 (in units of 0.1 ms for 100 Mbps and of 0,01 ms for 1 000 Mbps operation). The default value is 100. See 4.6.2.43, 4.6.2.44	USINT
Tsl	DLE number able to send out next slot	1 to 255. See 4.6.2.46	USINT
TSYN	Target-periodic-time from SYN frame arrival to SYN frame arrival	1 to 160 (in units of 0,1 ms). See 4.6.2.47	UDINT
TTRT0	Target-token-rotation-time for access class 0	1 to $2^{16}-1$ (octet time). The default value is P(TSYN). See 4.6.4.24	UDINT
TTRT1	Target-token-rotation-time for access class 1	1 to $2^{16}-1$ (octet time). The default value is P(TSYN). See 4.6.2.48	UDINT
TTRT2	Target-token-rotation-time for access class 2	1 to $2^{16}-1$ (octet time). The default value is P(TSYN). See 4.6.2.49	UDINT
TN	Node identifier number of the node. The initial value is 0 on power-up or reset	1 to 255. See 4.6.2.45	USINT

Table 6 – Observable variables and their value ranges of loop-architecture

Variable -name	Description	Range of value	Data type
AN _A AN _B	Node number of adjacent node neighboring Ring-port-A or -B	1 to 254. See 4.6.4.1	USINT
ASCMP	Collection of counters of total number of nodes. Each counter corresponding to a node indicates observed repetitive count that the corresponding node failed to send out the CMP frame and the other node can behave like that the CMP frame received without error. The node determines the corresponding node is out of service when the count reaches the pre-specified number. The number counted is indicated and is incremented coincidentally at each DLME	1 to 16. The default value is equal to the value of SCMPL, which is 3. See 4.6.2.1	USINT
ATHT	The observed time period of the MHTT	0 to MHTT, continuously decrementing. See 4.6.2.2	UDINT
ATSYN	The observed time period from SYN frame arrival to SYN frame arrival	1 to 2 ¹⁶ (octet times). See 4.6.2.3	UDINT
ATTRT0	The observed time period of the TTRT0	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.4.2	UDINT
ATTRT1	The observed time period of the TTRT1	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.2.4	UDINT
ATTRT2	The observed time period of the TTRT2	1 to 2 ¹⁶ -1, continuously decrementing. See 4.6.2.5	UDINT
CDh	Cumulative count of transmitted high-speed-cyclic data frames	0 to 2 ³² -1. See 4.6.2.8	UDINT
CDH _{blk}	Information data status of the data buffer with the identifier number “blk”, indicating the corresponding TCC data being active (healthy) or inactive (unhealthy)	True: active. False: inactive. See 4.6.2.9	BOOL
CDm	Cumulative count of transmitted medium-speed-cyclic data frames	0 to 2 ³² -1. See 4.6.2.10	UDINT
CRE _A CRE _B	Observed repetitive count for the abnormal reception error detected during receiving incoming frame from the Ring-port-A or -B respectively. The count is decreased by each abnormal reception detection repetitively but reset to the initial value when the abnormal reception is cleared. When reached to 0, abnormal reception error is detected	0 to 15. The default value is 2. See 4.6.4.4	USINT
LINK _A LINK _B	Link state indicating the condition of physical link connection at Ring-port-A or -B	“Abnormal” or “Normal”. See 4.6.4.9	BOOL
LSYN	Node number of the node in current master state	1 to 254. See 4.6.4.10	USINT
LL	Live list indicating whether the corresponding node, at this moment, is connected to and running in the Type 11 fieldbus domain in the received SYN frame. The Live list is a set of 256 Booleans, represented in 32 octets, in which each bit corresponds to a node in the Type 11 fieldbus domain and indicates the current status of that node	256-bit set of Booleans. True: node is connected and working. False: node is not connected or not working. See 4.6.2.19	USINT[16]
LN	Extracted number from LL at each node and used by each node to decide whether the node is able to send the data frame out over the medium	1 to 255. See 4.6.2.20	USINT
NCD _A , NCD _B	Cumulative count of No-Carrier detected on the Ring-port-A or B	0 to 2 ³² -1. See 4.6.4.12	UDINT

Variable -name	Description	Range of value	Data type
NONC	Permissible repetitive count of no CMP frame received by the SYN node within the corresponding consecutive Tsyn cycles, that is 256 times by SCMP, in order to detect no other node except the SYN node in the Type 11 fieldbus domain	1 to 16. The default value is 3. See 4.6.2.26, 4.6.2.27	USINT
NOS	Node operation state indicating the operation condition of a node in normal or open state	“Normal” or “Open”. The default value is “Open”. See 4.6.4.13	BOOL
NRF _A NRF _B	Observed time period in which no frame received from Ring-port-A or –B after from other Ring-port-B or –A respectively, a frame has been received	0 to 15 (in units of Th time). The default value is 2. See 4.6.4.14	USINT
NS	Node state indicating the node condition operating as master or slave	“Master” or “Slave”. The default value is “Slave”. See 4.6.4.15	BOOL
PP	Primary port indicating the Ring-port-A or –B through which SYN or CLM received first since this node is power-up, or in the system start-up, or in the system initialization	“Ring-port-A” or “Ring-port-B”. The default value is “Ring-port-A”. See Table 26, 4.6.4.17	BOOL
RE _A , RE _B	Cumulative count of DLPDUs received in error on the Ring-port-A or –B. RE _A is incremented when an error is detected on the Ring-port-A while the Ring-port-B respectively is error-free. RE _B corresponds, but with interchanged channel roles	0 to 2 ³² -1. See 4.6.4.20	UDINT
ROK _A , ROK _B	Cumulative count of DLPDU-received without error on the Ring-port-A or –B	0 to 2 ³² -1. See 4.6.4.21	UDINT
SL	The observed time period of the SL	0 to V(SL). See 4.6.2.35, 4.6.2.36	USINT
SD	Cumulative count of transmitted sporadic data frames	0 to 2 ³² -1. See 4.6.2.34	UDINT

4.6.2 Type 11 common variables, parameters, counters, timers and queues

4.6.2.1 C(ASCMP) : ASCMP count

A collection of counters, each of which indicates the number of observed repetitive count that the corresponding node failed to send out the CMP frame and the other node can behave like that the CMP frame received without error. The number of C(ASCMP) is incremented coincidentally at each node in Type 11 fieldbus. The value is in the range of 1 to V(SCMPL) and the default value is 3.

4.6.2.2 T(ATHT) : MTHT monitor

T(ATHT) is used by the DLE to measure the time elapsed of MTHT. The value is decremented in the range of P(MTHT) to 0.

4.6.2.3 T(ATSYN) : TSYN monitor

T(ATSYN) is used by the DLE to monitor the time period from SYN frame arrival to SYN frame arrival. The value is indicated in octet time by the data signaling rate.

4.6.2.4 T(ATTRT1) : TTRT1 monitor

T(ATTRT1) is used by the DLE to monitor the time period of TTRT1. The value is decremented in the range of 2¹⁶-1 to 1, of which unit is in octet time by the data-signaling rate.

4.6.2.5 T(ATTRT2) : TTRT2 monitor

T(ATTRT2) is used by the DLE to monitor the time period of TTRT2. The value is decremented in the range of $2^{16}-1$ to 1, of which unit is in octet time by the data-signaling rate.

4.6.2.6 Q(BF_{blk}) : Data buffer used for sending and receiving a DLPDU for time-critical cyclic data DLPDU

The total number of buffers is equivalent of the total number of DLCEP. Each buffer with the identifier “_{blk}” number corresponds to a DLCEP with the identifier number “identifier”.

4.6.2.7 V(BW), P(BW) : Length of time-critical cyclic data word in a DLPDU

This variable holds and designates the length of the Time-critical cyclic data word in a DLPDU.

The possible value of the length in octet is as follows:

- a) star-architecture: The default value is 128;
- b) loop-architecture: 16 to 128. The default value is 64.

4.6.2.8 C(CDh) : Cumulative count of high-speed-cyclic data frame sent

C(CDh) indicates the number of cumulative count of high-speed-cyclic data frame sent on the medium. C(CDh) is a roll-over binary counter of 32-bit length.

4.6.2.9 V(CDH_{blk}) Information data status of the data buffer with identifier number “_{blk}”

This variable indicates the status of the corresponding time-critical cyclic data being active (healthy) or inactive (unhealthy). The value “True” indicates active, and the value “False” indicates inactive.

4.6.2.10 C(CDm) : Cumulative count of medium-speed-cyclic data frame sent

C(CDm) indicates the number of cumulative count of medium-speed-cyclic data frame sent on the medium. C(CDm) is a roll-over binary counter of 32 bits length.

4.6.2.11 V(DR) : Data-rate

The value of this variable indicates the data signaling rate in Mbps. The value is 100 for the star-architecture and is either 100 or 1 000 for the loop-architecture.

4.6.2.12 V(HTh), P(HTh) : Maximum observational time period for detecting the continuous high-speed transmission cycle disrupted

This variable holds and designates the value of maximum observational time period for detecting the continuous High-speed transmission cycle disrupted. The value of this variable is in the range of 2 to $2^{16}-1$, of which the unit is 1 ms in 100 Mbps and 0,1 ms in 1 000 Mbps operation. The default value is $3 \times V(Th)$.

4.6.2.13 T(HTh) : HTh monitor

T(HTh) is used by the DLE to measure the HTh time elapsed to detect the continuous High-speed transmission cycle failed. The value is decremented in the range of V(HTh) to 1.

4.6.2.14 V(HTm), P(HTm) : Maximum observational time period for detecting the continuous medium-speed transmission cycle disrupted

This variable holds and designates the value of maximum observational time period for detecting the continuous Medium-speed transmission cycle disrupted. The value of this variable is in the range of 2 to $2^{16}-1$, of which the unit is 1ms. The default value is $3 \times V(Th)$.

4.6.2.15 T(HTm) : HTm monitor

T(HTm) is used by the DLE to measure the HTm time elapsed to detect the continuous Medium-speed transmission cycle disrupted. The value is decremented in the range of V(HTm) to 1.

4.6.2.16 V(IA) : Individual-address-of-this-node

This variable holds the individual address of this node in the 48-bit length specified by ISO/IEC 8802-3. The value is set by DLMS.

4.6.2.17 V(IGP) : Inter-frame-time-gap

This variable indicates the value of the time interval from the end of a previous frame to the start of the consecutive frame sent by a node. The value is equivalent to 0,96 μ s for 100 Mbps and 0,096 μ s for 1 000 Mbps operation and is identical to that specified by ISO/IEC 8802-3.

4.6.2.18 V(IP), P(IP) : IP address of this node

This variable holds and designates DLSAP value assigned for Type 11 fieldbus Sporadic message data service.

4.6.2.19 V(LL) : Live-list

This variable indicates the current operational status, whether a corresponding node is connecting to and is running normally in the Type 11 fieldbus. The possible value is "True" or "False", "True" means the node is connecting to and working normally and "False" is not. V(LL) is used by the DLE and is generated from the information conveyed by SYN frame. Live-list is a collection of 8 words of 32-bit length, each bit of which corresponds to the node in the Type 11 fieldbus and indicates the current operational status. Each bit corresponds to the node number V(TN) in a sequential order from 0 to 255 in little endian format.

4.6.2.20 V(LN) : Live-node-number

This variable indicates the TN number of node, extracted from V(LL), which connects to and is running normally in the Type 11 fieldbus at this point. V(LN) is used by the DLE to decide whether the node is able to send the data frame out over the medium. The range of this value is 1 to 255.

4.6.2.21 V(MD), P(MD) : Maximum-distance

The value of this variable holds, and is set by DLMS, the maximum distance in kilometer on the connection path between any 2 nodes.

The range of this value is as follows:

- a) star-architecture: 1 to 100, and the default value is 8;
- b) loop-architecture: 1 to 20, and the default value is 4.

4.6.2.22 V(MGA), P(MGA) : Multicast group address

This variable holds and designates the multicast group address, used by the DLE to build logically associated group of the Type 11 fieldbus nodes, in 48-bit length specified by and identical to ISO/IEC 8802-3. The default value is 0x01-0x00-0x5e-0x50-0x00-0x01.

4.6.2.23 V(MN), P(MN) : Maximum-node-number

This variable holds the maximum node number, and is set by DLMS. The default value of this variable is 254.

4.6.2.24 V(MRT), P(MRT) : Maximum-number-of-the-repeater-unit

This variable holds, and is set by the DLMS, maximum number of the repeater units on the connection path between any 2 nodes.

The possible value is as follows:

- a) star-architecture: 0 to 7, and the default value is 3;
- b) loop-architecture: 0 to 252, and the default value is 30.

4.6.2.25 V(MTHT), P(MTHT) : Maximum-token-hold-time

This variable holds and designates the value of maximum-token-hold-time for high-speed cyclic data transmission. The range of this value is 1 to $2^{16}-1$ of which unit is in octet time by the data signaling rate. The default value is 0x30B4, that is approx. 900 μ s for 100 Mbps and 90 μ s for 1 000 Mbps operation.

4.6.2.26 V(NONC), P(NONC) : Permissible repetitive count of no CMP

This variable holds and designates the value of permissible repetitive count of no CMP frame received by the SYN node within the corresponding consecutive T_{syn} cycles, that is 256 times by V(SCMPL), in order to detect no other node except the SYN node in the Type 11 fieldbus. The range of this value is 1 to 16, and the default value is 3.

4.6.2.27 C(NONC) : NONC count

C(NONC) indicates the number of repetitive count of no CMP frame received by the SYN node within $V(MN) \times V(SCMPL)$ of the consecutive T_{syn} cycles. C(NONC) reached to the number of V(NONC), the SYN node detects and recognizes no other node hooking up to the Type 11 fieldbus.

4.6.2.28 V(PBh), P(PBh) Maximum number of the data buffers used for sending high-speed time-critical cyclic data

This variable holds and designates the maximum number of the data buffers used for sending high-speed time-critical cyclic data.

4.6.2.29 V(PBm), P(PBm) Maximum number of the data buffers used for sending medium-speed time-critical cyclic data

This variable holds and designates the maximum number of the data buffers used for sending medium-speed time-critical cyclic data.

4.6.2.30 Q(QBF_{SPD}) : Queue buffer for sporadic data

The queue buffer for the Sporadic message data service holds the DLPDU by DL-user to be sent. The queue buffer size to transfer as well as to receive the Sporadic message data is implementation matter.

4.6.2.31 V(SCMP), P(SCMP) : Medium-access-control-time for substitute CMP

This variable holds and designates the value of maximum observational time period for the node to initiate the action like that the node received the CPM frame even though the corresponding node failed to send out the CMP frame.

4.6.2.32 T(SCMP) : SCMP monitor

T(SCMP) is used by the DLE of the node to measure the SCMP time elapsed to initiate the action like that the node received the CMP frame even though the corresponding node failed to send out the CMP frame. The value is decremented in the range of V(SCMP) to 0.

4.6.2.33 V(SCMPL), P(SCMPL) : Permissible repetitive count of substitute CMP

This variable holds and designates the value of permissible repetitive count within which the node can behave like that the node received the CMP frame without error even though the corresponding node failed to send out the CMP frame. The range of this value is 1 to 16 and the default value is 3.

4.6.2.34 C(SD) : Cumulative count of sporadic data frame sent

C(SD) indicates the number of cumulative count of sporadic data frames sent on the medium. C(SD) is a roll-over binary counter of 32-bit length.

4.6.2.35 V(SL), P(SL) : Silence time

This variable holds and designates the value of the silence time or inactivity time period for detecting the current SYN node out of order, following to initiate sending the CLM frame out from this node in order to claim new SYN node if this node is designated and permitted to be a SYN node. The value is set by the DLE to the T(SL) timer as follows:

$$T(SL) = V(TSYN) + V(ST) \times V(TN) \times 2$$

4.6.2.36 T(SL) : SL monitor

T(SL) is used by the DLE to measure the time elapsed of SL. The value is decremented in the range of V(SL) to 0.

4.6.2.37 V(SM), P(SM) : Subnet address mask

This variable holds and designates the value of subnet address mask for V(IP) for Type 11 fieldbus Sporadic message data service.

4.6.2.38 V(ST), P(ST), T(ST) : Type 11 fieldbus slot-time

Slot-time is the fundamentally observational time unit, used in the DLME for observation of initiation of the action such as re-initialization in sending out the CLM frame, for observation of initiation of the action that the node behaves like the CMP frame received even though the corresponding node failed to send out the CMP frame.

This variable holds and designates the fundamentally observational time value of slot time. The range of the value is 1 to 255, of which time is in 512-bit physical symbol time for 100 Mbps operation and in 4 096 bit physical symbol time for 1 000 Mbps operation specified by and identical to ISO/IEC 8802-3. The default value is 20, which is equivalent to Approx.100 μs for 100 Mbps operation and 8 μs for 1 000 Mbps operation, i.e. 20 x 512 x 1/100 μs in case of 100 Mbps operation.

The slot time value is calculated from the following formula:

$$V(ST) = \text{round up } [2 \times V(MD) + V(MRT)/2 + 2]$$

4.6.2.39 V(Th), P(Th) : High-speed transmission period

This variable holds and designates the cyclic time period of High-speed time-critical data transmission. P(Th) is used by the DLE and the value is set to V(SYN). The range of this value is 1 to 160, of which the unit is 0,1ms. The default value is 100.

4.6.2.40 V(TISPD), P(TISPD) : Time-interval for sporadic-message-data-service

This variable holds and designates the value of time interval cyclically processed for Sporadic-message-data service. The range of this value is 1 to 1 000, of which unit is in 1 ms. The default value is 100.

4.6.2.41 T(TISPD) : TISPD monitor

T(TISPD) is used by the DLE to measure the time interval of TISPD.

4.6.2.42 V(Tm), P(Tm) : Medium-speed transmission period

This variable holds and designates the cyclic time period of Medium-speed time-critical data transmission. P(Tm) is used by the DLE. The range of this value is 1 to 1 000, of which the unit is 1ms. The default value is 100.

4.6.2.43 V(TMAC), P(TMAC) : Medium-access-control-time to solicit new nodes

This variable holds and designates the value of maximum observational time period for the SYN node to solicit new nodes into the Type 11 fieldbus. During the time period new nodes attempt to send out the REQ frame to the SYN node. The default value is 100, the unit of which is in 0,1 ms for 100 Mbps operation and in 0,01 ms for 1 000 Mbps operation.

4.6.2.44 T(TMAC) : TMAC monitor

T(TMAC) is used by the DLE of the SYN node to measure the time elapsed of TMAC in order to wait for REQ frames sent from node which attempts to enter the Type 11 fieldbus. The value is decremented in the range of V(TMAC) to 0.

4.6.2.45 V(TN), P(TN) : This-node

This variable holds and designates the node identifier number of this node. The initial value is 0 on power-up or reset of this node. The range of this value is 1 to 255.

4.6.2.46 V(Tsl) : DLE number able to send out next slot

This variable is used by the DLE to hold and indicate the node identifier number of V(TN) of which the node can send out its data next in the sequential order in succession to a node currently having completed sending out all its data and CMP or DT-CMP DLPDU. The value of this variable is updated and set to the value of V(TN) of the node next to send out, every time on the reception of CMP or DT-CMP DLPDU. The range of this variable is 1 to 255.

4.6.2.47 V(TSYN) : Target-periodic-time of synchronization

This variable indicates the target time period from SYN frame arrival to SYN frame arrival. The value is equal to V(Th), and the value of P(Th) is used by the DLMS and is set to V(Th). The possible range of this time period is 0,1 ms to 160 ms. V(SYN) and V(Th) indicate the cyclic time period in which all the node with high-speed cyclic data can obtain the transmission right to send out all of the high-speed cyclic data.

4.6.2.48 V(TTTR1), P(TTTR1) : Target-token-rotation-time for access class 1

This variable holds and designates the value of target-token-rotation-time for access class 1, especially for sporadic message data transmission. The range of this value is 1 to $2^{16}-1$ of which unit is in octet time by the data signaling rate. The default value is equal to P(TSYN). This value is used by the DLE and should be equal and common to all DLEs of the Type 11 fieldbus.

P(TTTR1) is used by the DLME and the value is the time period in which a node with sporadic message data can expect to obtain the transmission right to send out.

4.6.2.49 V(TTTR2), P(TTTR2) : Target-token-rotation-time for access class 2

This variable holds and designates the value of target-token-rotation-time for access class 2, especially for medium-speed cyclic data transmission. The range of this value is 1 to $2^{16}-1$ of which unit is in octet time by the data signaling rate. The default value is equal to P(TSYN). This value is used by the DLE and should be equal and common to all DLEs of the Type 11 fieldbus.

P(TTTR2) is used by the DLME, and the value is the expected time period in which all nodes with medium-speed cyclic data can obtain the transmission right and send out all of the medium-speed cyclic data.

4.6.3 Star-architecture specific variables, parameters, counters, timers and queues

4.6.3.1 T(AIGP_A), T(AIGP_B) : IGP monitor over the receive-channel A and B

T(AIGP_A) and T(AIGP_B) are used by the DLE to measure the time elapsed of IGP over the receive-channel A and B respectively. The value is decremented in the range of 24 to 0, and value 24 is equivalent to 0,96 μ s.

4.6.3.2 T(ARMGP) : RMGP monitor

T(ARMGP) is used by the DLE to monitor the time elapsed of RMGP. The value is decremented in the range of 250 to 0, and value 250 is equivalent to 10 μ s.

4.6.3.3 T(ATTRT0) : TTRT0 monitor

T(ATTRT0) is used by the DLE to monitor the time period of TTRT0. The value is decremented in the range of $2^{16}-1$ to 1, of which unit is in octet time by the data-signaling rate.

4.6.3.4 C(CDI) : Cumulative count of low-speed-cyclic data frame sent

C(CDI) indicates the number of cumulative count of low-speed-cyclic data frames sent on the medium. C(CDI) is a roll-over binary counter of 32-bit length.

4.6.3.5 V(HTI), P(HTI) : Maximum observational time period for detecting the continuous low-speed transmission cycle

This variable holds and designates the value of maximum observational time period for detecting the continuous Low-speed transmission cycle disrupted. The value is in the range of 2 to $2^{16}-1$, of which the unit is 1ms. The default value is $3 \times V(Th)$.

4.6.3.6 T(HTI) : HTI monitor

T(HTI) is used by the DLE to measure the HTI time elapsed to detect the continuous Low-speed transmission cycle disrupted. The value is decremented in the range of V(HTI) to 1.

4.6.3.7 V(MDD), P(MDD) : Maximum-difference-of-the-distance

This variable holds, and is set by DLMS, the maximum difference in meter of the distance of two redundant physical mediums on the connection path between any 2 nodes. The range of this value is 0 to 2 000, and the default value is 500.

4.6.3.8 V(MPD), P(MPD) : Maximum-distance-of-the-signal-propagate-delay

This variable holds, and is set by DLMS, the maximum difference in microsecond of the signal delay time propagating over two redundant physical mediums on the connection path between any 2 nodes. The range of this value is 1 to 0x7c, and the default value is 0x7c which is equivalent to approx. 5 µs, i.e. 0x7c (=124) x 0,04 µs.

4.6.3.9 C(NCD_A), C(NCD_B) : Cumulative count of non-carrier detected on the receive-channel A or B

C(NCD_A) or C(NCD_B) indicates the number of cumulative count of non-carrier detected on one receive-channel while the carrier occurs on the other side of receive-channel. C(NCD_A) and C(NCD_B) are roll-over binary counters of 32-bit length.

4.6.3.10 V(PBI), P(PBI) : Maximum number of the data buffers used for sending low-speed time-critical cyclic data

This variable holds and designates the maximum number of the data buffers used for sending low-speed time-critical cyclic data. The value is 0 to 2 048.

4.6.3.11 V(RCS), P(RCS) : Receive-channel switching control for receiving frames

This variable holds and designates the switching control for receiving frames out of one of two receive-channel A and B corresponding to each of the redundant medium A and B.

The possible value is “Automatic”, “Force A” or “Force B”. “Automatic” indicates that the switching control for receiving frames is automatically to switch to the proper receive-channel. “Force A” or “Force B” indicates that the switching control is forced to switch to Receive-channel A or B respectively. The default value is “Automatic”.

4.6.3.12 V(RCS_A), V(RCS_B) : Receive-channel A or B selected for receiving frames

This variable indicates the corresponding receive-channel is selected for receiving frames from the other node. The value “True” indicates that the corresponding receive-channel is selected, and “False” is not.

4.6.3.13 C(RE_A), C(RE_B) : Cumulative count of DLPDU received with error on the receive-channel A or B

C(RE_A) or C(RE_B) indicates the number of cumulative count of DLPDU received with error while on the other side of the receive-channel no error occurs. C(RE_A) and C(RE_B) are roll-over binary counters of 32-bit length.

4.6.3.14 V(RMGP), P(RMGP) : IGP over other redundant media

This variable holds and designates the value of maximum time-interval for one receive-channel, which has already completed one frame received and has waited for IGP time, to wait the completion of a frame to be received on the other receive-channel in order to detect the other receive-channel disrupted. The default value is 250, of which unit is in 0,04 µs, and is equivalent to 10 µs.

4.6.3.15 C(ROK_A), C(ROK_B) : Cumulative count of DLPDU received without error on the receive-channel A or B

C(ROK_A) or C(ROK_B) indicates the number of the cumulative count of DLPDU received without error on the receive-channel A or B respectively. C(ROK_A) and C(ROK_B) are roll-over binary counters of 32-bit length.

4.6.3.16 V(SEN_A), P(SEN_A); V(SEN_B), P(SEN_B) : Transmitter A or B enabled to send out frames

This variable designates the transmitter A or B enable to send out frames from, and indicates the state of the transmitter A or B being enabled or disabled. The value “True” is to be enabled and “False” is not for the corresponding Transmitter A or B. The default value is “True”.

4.6.3.17 V(TI), P(TI) : Low-speed transmission period

This variable holds and designates the cyclic time period of Low-speed time-critical data transmission. P(TI) is used by the DLE. The range of this value is 100 to 1 000, of which the unit is 1 ms, and the default value is 100.

4.6.3.18 V(TTROT), P(TTROT) : Target-token-rotation-time for access class 0

This variable holds and designates the value of target-token-rotation-time for access class 0, especially for low-speed cyclic data transmission. The range of this value is 1 to $2^{16}-1$ of which unit is in octet time by the data signaling rate. The default value is equal to P(TSYN). This value is used by the DLE and should be equal and common to all DLEs of the Type 11 fieldbus.

P(TTROT) is used by the DLME, and the value is the expected time period in which all nodes with low-speed cyclic data can obtain the transmission right and send out all of the low-speed cyclic data.

4.6.4 Loop-architecture specific variables, parameters, counters, timers and queues

4.6.4.1 V(AN_A), V(AN_B) : Node-number of adjacent node-A, -B

This variable holds the node-number of adjacent node. The V(AN_A) and V(AN_B) indicate the node-number of the adjacent node in the neighborhood of the Ring-port-A and -B of this node respectively. The initial value on power-up or reset of this node is 0.

4.6.4.2 T(ATTRT0) : TTRT0 monitor

T(ATTRT0) is used by the DLE to monitor the time period of TTRT0 in 1 000 Mbps operation. The value is decremented in the range of $2^{16}-1$ to 1, of which unit is in octet time by the data-signaling rate.

4.6.4.3 C(CDI) : Cumulative count of low-speed-cyclic data frame sent

C(CDI) indicates the number of cumulative count of low-speed-cyclic data frames sent on the medium. C(CDI) is a roll-over binary counter of 32-bit length.

4.6.4.4 V(CRE_A), P(CRE_A), C(CRE_A); V(CRE_B), P(CRE_B), C(CRE_B) : Maximum permissible repetitive count for abnormal-reception error-A, -B

This variable holds, is set by the DLE and indicates the number of observed repetitive count of the abnormal-reception error detected during receiving the incoming frames from the Ring-port-A and -B respectively. The abnormal-reception is happened under the condition of the reception-abnormal and the long-silence of reception. The range of this variable is 15 to 0. The default value is 2. The count is decremented by each abnormal-reception detected

repetitively, and is set to the default value when the abnormal-reception is cleared. When reached to 0, the RMC recognizes the failure happened.

4.6.4.5 V(ESYN), P(ESYN): Enable SYN node

This variable holds, is set by the DLE and indicates that the DLE is capable of SYN node. The value “Enable” is to be enabled and “Disable” is not. The default value is “Enable”.

4.6.4.6 V(FWE_{AB}), P(FWE_{AB}), V(FWE_{BA}), P(FWE_{BA}) : Forwarding state-AB, -BA

This variable holds and designates the Ring-port-A or –B enable to forward frames received from the Ring-port-A to –B or from the Ring-port-B to –A respectively.

The value “Enable” is to be enabled and “Disable” is not for corresponding Ring-port-A or –B. the default value is “Disable”.

4.6.4.7 V(HTI), P(HTI) : Maximum observational time period for detecting the continuous low-speed transmission cycle

This variable holds and designates the value of maximum observational time period for detecting the continuous Low-speed transmission cycle disrupted in 1 000 Mbps operation. The value is in the range of 2 to $2^{16}-1$, of which the unit is 1ms. The default value is $3 \times V(Th)$.

4.6.4.8 T(HTI) : HTI monitor

T(HTI) is used by the DLE to measure the HTI time elapsed to detect the continuous Low-speed transmission cycle disrupted in 1 000 Mbps operation. The value is decremented in the range of V(HTI) to 1.

4.6.4.9 V(LINK_A), V(LINK_B) : Link state-A, -B

This variable indicates the link state of the Ring-port-A and -B of this node. The value of this variable is “normal” or “abnormal” according to the physical link connection to the trunk specified by ISO/IEC 8802-3.

4.6.4.10 V(LSYN) : Node number in the master-state

This variable indicates the node number of the node in the current master-state.

4.6.4.11 V(MDAD), P(MDAD) : Maximum-distance between adjacent node

The value of this variable holds, and is set by DLE, the maximum-distance in kilometer between the adjacent nodes. The range of this value is 1 to 2 for 100 Mbps operation and 1 to 5 for 1 000 Mbps operation. The default value is 1.

4.6.4.12 C(NCD_A), C(NCD_B) : Cumulative count of no-carrier detected -A, -B

C(NCD_A) or C(NCD_B) indicates the number of cumulative count of no-carrier detected on the Ring-port-A or –B respectively while the carrier occurs on the other port. The C(NCD_A) and the C(NCD_B) are roll-over binary counters of 32-bit length.

4.6.4.13 V(NOS) : Node-operation state

This variable holds and indicates the operation state of the node which is in the normal- or the open-state. The value of this variable is either “Normal” or “Open” according the normal-state or the open-state respectively. The initial value on power-up or reset of this node is “Open”.

4.6.4.14 $V(NRF_A)$, $P(NRF_A)$, $T(NRF_A)$; $V(NRF_B)$, $P(NRF_B)$, $T(NRF_B)$: Maximum observational time period for no frame received -A, -B

This variable holds, is set, and is used by the DLE to monitor the time period in which no frame received from the Ring-port -A or -B after from the other port, that is the Ring-port-B or -A respectively, a frame has been received. The range of this value is 15 to 0. The default value is 2, of which the unit is equal to $V(TSYN)$. The timer is set to 2 and starts to clock every time when any frame has been received from the other port, and is decremented during no frame received from the port.

4.6.4.15 $V(NS)$: Node state

This variable holds and indicates the node state which is the master- or slave-state. The value of this variable is "Master" if in the master-state, or "Slave" in the slave-state. The initial value on power-up or reset of this node is "Slave".

4.6.4.16 $V(PBI)$, $P(PBI)$: Maximum number of the data buffers used for sending low-speed time-critical cyclic data

This variable holds and designates the maximum number of the data buffers used for sending low-speed time-critical cyclic data in 1 000 Mbps operation. The range of this value is 1 to 4 096.

4.6.4.17 $V(PP)$: Primary-port

This variable holds and indicates the port through which the SYN-frame first received since this node is power-up, or in the system start-up, or in the system initialization. The value of this variable is "Ring-port-A" or "Ring-port-B".

4.6.4.18 $V(PS_A)$, $P(PS_A)$; $V(PS_B)$, $P(PS_B)$: Ring-port state-A, -B

These variables hold and designate the port state of this node. The $V(PS_A)$, $P(PS_A)$ and the $V(PS_B)$, $P(PS_B)$ holds, and is set by the DLE, the port state of the Ring-port-A and -B respectively. The value is either "Blocked" or "Non-blocked". Each initial value on power-up or reset of this node is "Blocked".

4.6.4.19 $C(RE_A)$, $C(RE_B)$: Cumulative count of frame received with error -A, -B

$C(RE_A)$ or $C(RE_B)$ indicates the number of cumulative count of the frame received with error from the Ring-port-A or -B respectively while no error occurs on the other port. The $C(RE_A)$ and the $C(RE_B)$ are roll-over binary counters of 32-bit length.

4.6.4.20 $V(REN_A)$, $P(REN_A)$; $V(REN_B)$, $P(REN_B)$: Receive enable -A, -B

These variables hold and designate the receive enable from the Ring-port-A or -B of this node. The $V(REN_A)$, $P(REN_A)$ and the $V(REN_B)$, $P(REN_B)$ holds, and is set by the DLME, the state of receive frame enable from the Ring-port-A and -B respectively.

The value of this variable is "Enable" or "Disable". The default value is "Enable".

4.6.4.21 $C(ROK_A)$, $C(ROK_B)$: Cumulative count of frame received without error-A, -B

$C(ROK_A)$ or $C(ROK_B)$ indicates the number of the cumulative count of the frame received without error from the Ring-port-A or -B respectively. The $C(ROK_A)$ and the $C(ROK_B)$ are roll-over binary counters of 32-bit length.

4.6.4.22 $V(\text{SEN}_A)$, $P(\text{SEN}_A)$; $V(\text{SEN}_B)$, $P(\text{SEN}_B)$: Send enable -A, -B

These variables hold and designate the send enable out of the Ring-port-A or -B of this node. The $V(\text{SEN}_A)$, $P(\text{SEN}_A)$ and the $V(\text{SEN}_B)$, $P(\text{SEN}_B)$ holds, and is set by the DLME, the state of send-frame-enable out of the Ring-port-A and -B respectively.

The value of this variable is "Enable" or "Disable". The default value is "Enable".

4.6.4.23 $V(\text{TI})$, $P(\text{TI})$: Low-speed transmission period

This variable holds and designates the cyclic time period of Low-speed time-critical data transmission in 1 000 Mbps operation. $P(\text{TI})$ is used by the DLE. The range of this value is 1 to 10 000, of which the unit is 1 ms, and the default value is 100.

4.6.4.24 $V(\text{TTRT}_0)$, $P(\text{TTRT}_0)$: Target-token-rotation-time for access class 0

This variable holds and designates the value of target-token-rotation-time for access class 0, especially for low-speed cyclic data transmission. The range of this value is 1 to $2^{16}-1$ of which unit is in octet time by the data signaling rate. The default value is equal to $P(\text{TSYN})$. This value is used by the DLE and should be equal and common to all DLEs of the Type 11 fieldbus.

$P(\text{TTRT}_0)$ is used by the DLME, and the value is the expected time period in which all nodes with low-speed cyclic data can obtain the transmission right and send out all of the low-speed cyclic data.

5 General structure and encoding of PhIDU and DLPDU and related elements of procedure

5.1 Overview

The DLL and its procedures are those necessary to provide the services offered to the DLS user by using the services available from the PhL. Clause 5 describes the structure and semantics of PhIDU, MA_PDU, DLPDU and the procedure, commonly used in this standard nevertheless this portion is identical to and fully compliant with ISO/IEC 8802-3.

Within Clause 5, any reference to bit k of an octet is a reference to the bit whose weight in a one-octet unsigned is 2^k , and this is sometimes referred to as "little endian" bit numbering.

5.2 PhIDU structure and encoding

The local MAC sublayer uses the service primitives provided by the PLS sublayer specified by ISO/IEC 8802-3:2000, Clause 6. All of the service primitives provided by the PLS sublayer are as follows and are considered mandatory:

- a) PLS_DATA request;
- b) PLS_DATA indication;
- c) PLS_CARRIER indication;
- d) PLS_SIGNAL indication;
- e) PLS_DATA_VALID indication.

In the case where specifications of 100 Mbps and above are applied, the reconciliation sublayer maps the signals provided at the MII for 100 Mbps and at the GMII for 1 000 Mbps to the PLS service primitives as specified by ISO/IEC 8802-3:2000, Clause 22 and Clause 35 respectively. The PLS service primitives provided by the reconciliation sublayer behave in exactly same manner as defined in Clause 6 of ISO/IEC 8802-3:2000.

5.3 Common MAC frame structure, encoding and elements of procedure

5.3.1 MAC frame structure

5.3.1.1 Common MAC frame format for the Type 11 fieldbus DLPDU

The structure of the MAC frame for the Type 11 fieldbus DLPDU is encapsulated in the frame format of Ethernet V2.0 specified by ISO/IEC 8802-3:2000, Clause 3. The value of the Length/Type field is designated to 0x888B, which is authorized and registered as the protocol identification number by the IEEE Registration Authority, to be identified as the Type 11 fieldbus frame. Figure 6 shows the MAC frame format used common to the Type 11 fieldbus DLPDU except for sporadic data transmission of the Type 11 fieldbus.

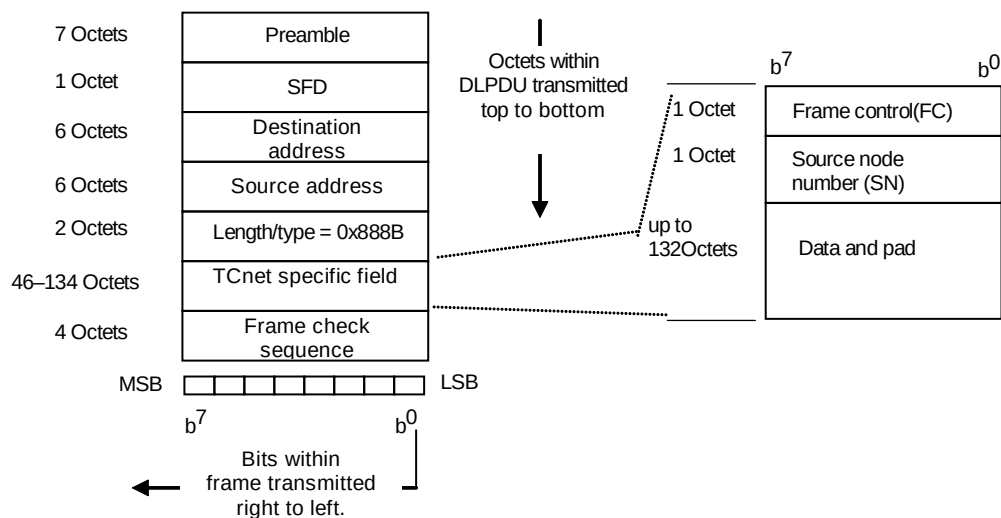


Figure 6 – Common MAC frame format for DLPDUs

5.3.1.2 MAC frame format for Type 11 fieldbus sporadic DLPDU

The MAC frame format used for sporadic data transmission of the Type 11 fieldbus shall be according to ISO/IEC 8802-3:2000, Figure 3-1.

5.4 Elements of the MAC frame

5.4.1 General

The elements of the MAC frame are the preamble, the start frame delimiter, the destination address, the source address, the length/type code, and the frame check sequence, all as specified by ISO/IEC 8802-3:2000, Clause 3.

5.4.2 Preamble field

The preamble of the MAC frame is identical to ISO/IEC 8802-3:2000, Clause 3. The preamble field is a 7-octet field that is used to allow the physical signaling part circuitry to reach its steady-state synchronization with the receiving frame timing.

The preamble pattern is:

“10101010 10101010 10101010 10101010 10101010 10101010 10101010”

5.4.3 Start frame delimiter (SFD)

The Start Frame Delimiter (SFD) is identical to ISO/IEC 8802-3:2000, Clause 3. The SFD field is the sequence of bit pattern “10101011”. It immediately follows the preamble pattern and indicates the start of a frame.

5.4.4 Address field

5.4.4.1 General

The address fields (both Destination Address and Source Address) are identical in structure and semantics to the address field of the basic MAC frame, described in ISO/IEC 8802-3:2000, Clause 3. Each address field shall be 48-bit in length.

5.4.4.2 Destination address field (DA)

The Destination Address (DA) field is identical to ISO/IEC 8802-3:2000, Clause 3. The Destination Address field specifies the station(s) for which the frame is intended. It may be an individual or multicast (including broadcast) address. The value of DA for the Type 11 fieldbus DLPDU except for the Type 11 fieldbus sporadic data transmission is always set to V(MGA).

5.4.4.3 Source address field (SA)

The Source Address (SA) field is identical to ISO/IEC 8802-3:2000, Clause 3. The Source Address field specifies the station sending the frame. The Source Address field is not interpreted by the DLE as well as the CSMA/CD MAC sublayer. The value of SA is always set to V(IA).

5.4.5 Length/type field

The Length/type field is identical to ISO/IEC 8802-3:2000, Clause 3. In order to be identified as a Type 11 fieldbus frame, the value of the length/type field is set to 0x888B, which is authorized and registered as protocol identification number for Type 11 fieldbus by the IEEE Registration Authority. Every frame with the value all but 0x888B is identical to the frame according to ISO/IEC 8802-3:2000, Clause 3 and is processed as a Type 11 fieldbus sporadic data frame.

5.4.6 Frame control field (FC)

5.4.6.1 Structure of FC field

The structure of the Frame control field is shown in Figure 7.

The F-type field is used by the DLE to identify and designate the frame type of the Type 11 fieldbus for medium access control. The Pri field is used by the DLE to designate and identify the service class of time-critical cyclic data frame.

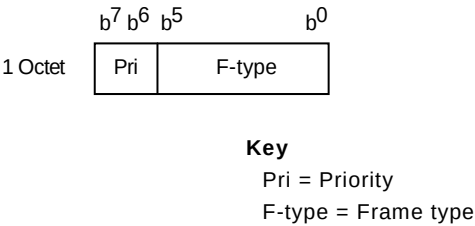


Figure 7 – Structure of FC field

5.4.6.2 Frame type (F-type) field

Table 7 shows the list of F-types, the values and the corresponding Type 11 fieldbus frames. All F-types are reserved for future use except that listed in Table 7.

Table 7 – F-type: DLPDU type

F-type value	Frame type	Frame name
0x00	CLM	Claim frame
0x01	SYN	Synchronization frame
0x02, 0x22	REQ	In-ring request frame -Star-architecture: the value is 0x02 -Loop-architecture: the value is 0x22
0x04	COM	Command frame
0x05	RAS	RAS frame
0x07	DT	Cyclic data frame
0x08	CMP	Transmission complete frame
0x0F	DT-CMP	DT with transmission complete frame
0x23	LPD	Loop diagnosis frame
0x26	LRR	Loop repeat request frame

5.4.6.3 Priority field (Pri)

The priority field is used by the DLE to designate and identify the service class of time-critical cyclic data frame, which is applied to both DT and DT-CMP. 3 levels of service class are provided for the star-architecture and the loop-architecture in 1 000 Mbs operation. The loop-architecture in 100 Mbps operation provides 2 levels of service class. The top to lowest level is level 3 to 1. High-speed time-critical data transmission is assigned to the level 3 of the service class, Medium-speed time-critical data transmission to the level 2 and Low-speed time-critical data transmission to the level 1. The Low-speed time-critical data transmission is not provided for the loop-architecture in 100 Mbps operation. Other type of frame except for DT and DT-CMP is assigned to the level 3 of the service class.

5.4.7 Source node number field (SN)

The value of SN is the node identifier number and is equal to the value of V(TN) of the node which has sent the Type 11 fieldbus DLPDU.

5.4.8 Data and pad field

The data field contains a sequence of N octets which provides full data transparency in the sense that any arbitrary sequence of octet values may appear in the data field up to a maximum number specified by ISO/IEC 8802-3. A minimum frame size is required, that is *mimFrameSize* by ISO/IEC 8802-3, and if a frame size is less than *mimFrameSize*, then the data field is extended by appending extra bits in units of octets. The frame, from the DA field through the FCS field inclusive, is at least *mimFrameSize* bits.

The structure of this field for the Type 11 fieldbus DLPDU is described in Clause 6.

5.4.9 Frame check sequence (FCS)

5.4.9.1 General

The frame check sequence (FCS) construction, polynomial and expected residual are identical to ISO/IEC 8802-3:2000, Clause 3.

Within Clause 5, 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 .

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

For most of the protocol Types in this standard, as in other International Standards (for example, ISO/IEC 3309:1993, ISO/IEC 8802 and ISO/IEC 9314-2:1989), 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 $n - k$ (equal 32) 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:

The generic form of the generator polynomial for this FCS construction is specified in formula (4) and the polynomial for the receiver’s expected residue is specified in formula (8). The specific polynomials for each DL-protocol type are specified in Table 8.

Table 8 – FCS length, polynomials and constants

Item	Value
$n-k$	32
$G(X)$	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$ a, b, c
$R(X)$	$X^{31} + X^{30} + X^{26} + X^{25} + X^{24} + X^{18} + X^{15} + X^{14} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$ c, d
a	Code words $D(X)$ constructed from this $G(X)$ polynomial have Hamming distance 4 for lengths $\leq 22\,901$ octets, Hamming distance 5 for lengths ≤ 371 octets, Hamming distance 6 for lengths ≤ 33 octets, Hamming distance 7 for lengths ≤ 21 octets, and Hamming distance 8 for lengths ≤ 11 octets.
b	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 $1 + X^{-j} + X^{-k}$.
c	These are the same polynomials and method as specified in ISO/IEC 8802-3 (Ethernet).
d	The remainder $R(x)$ should be 1100 0111 0000 0100 1101 1101 0111 1011 (X^{31} to X^0 , respectively) in the absence of errors.

5.4.9.2 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) are regarded as vectors $M(X)$, $F(X)$, and $D(X)$, of dimension k , $n - k$, and n , respectively, in an extension field over $GF(2)$. If the message bits are $m_1 \dots m_k$ and the FCS bits are $f_{n-k-1} \dots f_0$, where

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

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

$f_7 \dots f_0$ 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),

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

This “as transmitted” ordering is critical to the error detection properties of the FCS.

then the message vector $M(X)$ is

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

and the FCS vector $F(X)$ is

$$\begin{aligned} F(X) &= f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= f_{31}X^{31} + \dots + f_0 \end{aligned} \quad (\text{for the case of } k = 32) \quad (2)$$

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

$$\begin{aligned} D(X) &= M(X) X^{n-k} + F(X) \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{32} + f_{31}X^{31} + \dots + f_0 \end{aligned} \quad (\text{for the case of } k = 32) \quad (3)$$

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

The redundant check bits $f_{n-k-1} \dots f_0$ of the FCS are 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

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \dots + 1 \quad (4)$$

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

$$\begin{aligned} L(X) &= \frac{X^{n-k} + 1}{X + 1} = X^{n-k-1} + X^{n-k-2} + \dots + X + 1 \\ &= X^{31} + X^{14} + X^{13} + X^{12} + \dots + X^2 + X + 1 \end{aligned} \quad (\text{for the case of } k = 32) \quad (5)$$

That is,

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

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.

As a typical implementation when $n-k = 32$, 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 formula (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.

5.4.9.3 At the receiving DLE

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

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

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

A remainder $R(X)$ is computed for $V(X)$, the received DLPDU and FCS, by a method similar to that used by the sending DLE (see 5.4.9.2) in computing $F(X)$

$$\begin{aligned} R(X) &= L(X) X^u + V(X) X^{n-k} \text{ (modulo } G(X)) \\ &= r_{n-k-1}X^{n-k-1} + \dots + r_0 \end{aligned} \quad (8)$$

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) 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 can be discarded without further analysis.

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 formula (4).

5.5 Order of bit transmission

The order of bit transmission is identical to ISO/IEC 8802-3:2000, Clause 3. Each octet of the DLPDU, with the exception of the FCS, shall be transmitted low-order bit first; the FCS shall be transmitted in the order specified in 5.4.9.2.

5.6 Invalid DLPDU

An invalid DLPDU shall be defined as one that meets at least one of the following conditions, and is almost identical to ISO/IEC 8802-3:2000, Clause 3:

- The frame length is inconsistent with a length value specified in the Length/Type field. If the Length/Type field contains a type value as defined by ISO/IEC 8802-3:2000, 3.2.6, then the frame length is assumed consistent with this field and should not be considered an invalid DLPDU on this basis.
- It is not an integral number of octets in length.
- The bits of the incoming DLPDU (exclusive of the FCS field itself) do not generate a CRC value identical to the one received.
- It is inconsistent with a F-type value of Type 11 fieldbus DLPDU.

The contents of invalid DLPDU shall not be passed to the DL-user or DLE. The occurrence of invalid DLPDU may be communicated to network management.

Invalid DLPDU may be ignored, discard, or used in private manner by DL-user other than RTE DL-user. The use of such DLPDUs is beyond the scope of this standard.

6 DLPDU-specific structure, encoding and elements of procedure

6.1 General

Clause 6 defines the structure, contents and encoding of each type and format of the DLPDU except for the Sporadic DLPDU of the Type 11 fieldbus, and specifies elements of procedure for the DLPDU.

Within each subclause of Clause 6, the structure, contents, parameters and encoding of the DLPDU are described, and the Type 11 fieldbus specific part of the DLPDU structure, which is shown in 5.3.1.1, is specified. The aspects relating to the sending and receiving of DLS-users and their DLEs are further described. All data format and encoding is described in little endian format throughout Clause 6.

Within Clause 6, any reference to bit K of an octet is a reference to the bit whose weight in a one-octet unsigned is 2^K , and this is sometimes referred to as "little endian" bit numbering.

Whenever a conditional action is specified and the specific enabling condition does not occur, then the corresponding action also does not occur. The effect eventually is that events which do not meet any of the enabling conditions specified in a service procedure will have no consequential actions with respect to that specific service procedure.

6.2 Synchronization DLPDU (SYN)

6.2.1 General

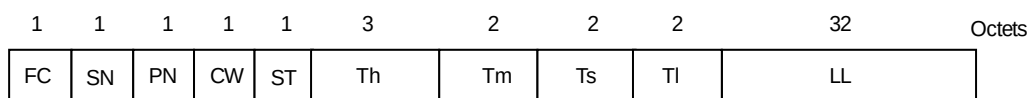
A DLE which is as the SYN node or operating currently in master state sends the SYN DLPDU periodically at the specific time intervals of $V(Th)$ for the synchronization of time-critical cyclic data transmission by all related DLEs in the Type 11 fieldbus.

A DLE which attempts to be on line, sends REQ DLPDU for claiming to the SYN node to be added in the Type 11 fieldbus under the condition when the DLE has received a SYN DLPDU with the value of PN filed equal to the $V(TN)$. The time frame, which REQ DLPDU is sent, is during the MAC control period of T_{mac} . The MAC control period of T_{mac} starts immediately after the DLE has completely received a CMP DLPDU or DT-CMP DLPDU by the node with the biggest number of the value of $V(TN)$.

A DLE which is on line and is functioning as a SYN node, controls the time-critical cyclic data transmission of the Type 11 fieldbus DLPDUs, solicits new node to join and manages node drop-out, controls and manages all of the parameters related to keep cyclic and stable data transmission. A DLE not operating as a SYN node shall attempt to become a new SYN node to keep cyclic and stable data transmission in succession to the current SYN node when the current SYN node happens to malfunction. A DLE not operating as a SYN node decides the order and the timing to send out its time-critical cyclic data transmission using the value of $V(LL)$ in the SYN DLPDU received.

6.2.2 Structure of SYN DLPDU

The structure of SYN DLPDU is shown in Figure 8.



Key

FC = Frame control (11000001)	SN = Source node number
PN = Transmission permits node number	CW = Control word
ST = Slot time	Th = High-speed periodic time
Tm = Medium-speed periodic time	Ts = Sporadic message rotation time
TI = Low-speed periodic time2	LL = Live list

Figure 8 – Structure of SYN DLPDU

6.2.3 Parameters of SYN DLPDU

6.2.3.1 Transmission permits node number (PN)

100 Mbps operation both of star- and loop-architecture related to the PN is as follows:

- A DLE attempting to be on line is permitted to send REQ DLPDU out for claiming to enter into the Type 11 fieldbus to the SYN node when the value of PN in the SYN DLPDU becomes equal to the value of V(TN). The DLE can send REQ DLPDU out to the SYN node in the time frame of the MAC control period during the time slot of V(TSYN) in which the DLE has received the SYN DLPDU with equal value of PN to that of the V(TN).

1 000 Mbps operation of loop-architecture related to the PN is as follows:

- A DLE is permitted to send out RAS DLPDU every time and COM DLPDU upon occurrence of events, to the SYN node when the value of PN in the SYN DLPDU becomes equal to the value of V(TN). The DLE can send out RAS DLPDU and COM DLPDU in the time frame of V(LN) equal to the value of V(TN) during the time slot of V(TSYN) in which the DLE has received the SYN DLPDU with equal value of PN to that of the V(TN), and the DLE sends out RAS DLPDU and COM DLPDU before sending out DT DLPDU.

The range of this value is 0x1 to 0xFF. The value is incremented by each SYN DLPDU and rolled over from 0xFF to 0x1.

Table 9 shows the structure of the PN parameter.

Table 9 – PN-parameter: 3rd octet

Transmission permits node number							
PN (1 to 255)							
7	6	5	4	3	2	1	0

6.2.3.2 Control word (CW)

6.2.3.2.1 CW for star-architecture

6.2.3.2.1.1 Structure of CW

The structure of Control word (CW) is shown in Table 10.

Table 10 – Structure of CW: 4th octet

Control word							
Periodic mode	Reserved					Redundant medium selection	
PM	0					RMSEL	
7	6	5	4	3	2	1	0

6.2.3.2.1.2 Periodic mode (PM) parameter

The “PM” parameter indicates the mode of synchronization for the Time-critical cyclic data transmission.

“Free-run” mode is based on simple periodic data transmission, that is, the SYN node sends out the SYN frame immediately after the SYN node detects that all nodes have sent their time-critical cyclic data within the time period of V(TSYN). On the other hand, “Constant period” mode is based on time-synchronized data transmission with a signal from a clock source of constant time-period.

Table 11 shows the mode of PM parameter.

Table 11 – PM parameter

PM mode	Value	Description
Free-run	0	Cyclic data transmission based on free run
Constant period	1	Cyclic data transmission based on time-synchronization

6.2.3.2.1.3 Redundant medium selection (RMSEL)

This parameter indicates and designates the mode of redundant medium selection to all nodes. The value of RMSEL is used and set to V (RCS) by the DLE for switching control to the receive frames out of one of two receive-channel A and B corresponding to each of the redundant medium A and B. The possible value is “Automatic”, “Force A” and “Force B”.

“Automatic” designates that the switching control for receiving frames is automatically to switch to the proper receive-channel. “Force A” or “Force B” designates that the switching control is forced to switch to Receive-channel A or B respectively. The initial value is “Automatic”.

The mode of RMSEL parameter is summarized in Table 12.

Table 12 – RMSEL parameter

RMSEL mode	Value	Description
Automatic	00	Automatically switch to the proper receive-channel
–	01	Reserved for future use
Force A	10	Force to switch receive-channel A
Force B	11	Force to switch receive-channel B

6.2.3.2.2 CW for loop-architecture

6.2.3.2.2.1 Structure of CW

The structure of CW is shown in Table 13.

Table 13 – Structure of CW: 4th octet

Control word							
Reserved (0x0)				PM	Reserved (0x0)		
7	6	5	4	3	2	1	0

6.2.3.2.2.2 Periodic mode (PM) parameter

The “PM” parameter indicates the mode of synchronization for the Time-critical cyclic data transmission. The PM parameter is equal to that of star-architecture specified in 6.2.3.2.1.2.

6.2.3.3 Slot time (ST)

This parameter is the fundamentally observational time unit, using in the DLE for the observation of initiating action such as re-initialization in sending out CLM frame, for sending out the CMP frame specifically by the current SYN node in substitute for the node failed to send out the CMP frame.

V(ST) holds the value of ST. The range of this value is 1 to 255, of which time is in 512-bit physical symbol time in 100 Mbps operation of the star- and the loop-architecture and is in 4 096-bit physical symbol time in 1 000 Mbps operation of the loop-architecture specified by and identical to ISO/IEC 8802-3. The default value is 20, which is equivalent to approximately 100 μ s for 100 Mbps operation and 8 μ s for 1 000 Mbps operation, that means $20 \times 512 \times 1/100 \mu$ s in case of 100 Mbps operation.

Table 14 shows the structure of ST parameter.

Table 14 – ST-parameter: 5th octet

V(ST) : Slot time							
7	6	5	4	3	2	1	0

6.2.3.4 High-speed transmission period (Th)

This parameter designates the cyclic time period of High-speed time-critical data transmission. The value of Th is used by the DLE, and ultimately set to V(Th), V(TSYN) equally for each node on-line. The range of this value is 1 250 to 2 000 000, of which the unit is in 80 ns.

Table 15 shows the structure of Th parameter.

Table 15 – Th-parameter: 6th, 7th and 8th octets

High-speed transmission period																							
Th ($2^7 - 2^0$)								Th ($2^8 - 2^{15}$)								Th ($2^{16} - 2^{23}$)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.5 Medium-speed transmission period (T_m)

This parameter designates the cyclic time period of Medium-speed time-critical data transmission. The value of T_m is used by the DLE, and is set to V(T_m) equally for each node on-line. The range of this value is 1 to 1 000, of which the unit is in 1 ms.

Table 16 shows the structure of T_m parameter.

Table 16 – T_m-parameter: 9th and 10th octets

Medium-speed transmission period															
Tm (2 ⁷ - 2 ⁰)								Tm (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.6 Sporadic message transmission target-token-rotation-time period (T_s)

This parameter designates the value of target-token-rotation-time for sporadic message data transmission. This value is used by the DLE, and is set to V(TT_{RTn}) equally for each node on-line. The range of this value is 1 to 1 000, of which the unit is in 1 ms. V(TT_{RTn}) is V(TT_{RT1}) for 100 Mbps operation both of the star- and the loop-architecture, and is V(TT_{RT0}) for 1 000 Mbps operation of the loop-architecture.

Table 17 shows the structure of T_s parameter.

Table 17 – T_s-parameter: 11th and 12th octet

Sporadic message transmission target-rotation-time period															
Ts (2 ⁷ - 2 ⁰)								Ts (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.7 Low-speed transmission period (T_l)

This parameter designates the cyclic time period of Low-speed time-critical data transmission for the star-architecture and for 1 000 Mbps operation of the Loop-architecture, and is not used for 100 Mbps operation of the loop-architecture. The value of T_l is used by the DLE for star-architecture, and is set to V(T_l) equally for each node on-line. The range of this value is 1 to 10 000, of which the unit is in 1 ms.

Table 18 shows the structure of T_l parameter. The T_l field is reserved and the value is set to 0x0 for 100 Mbps operation of the loop-architecture.

Table 18 – T_l parameter: 13th and 14th octets

Low-speed transmission period															
TI (2 ⁷ - 2 ⁰)								TI (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.8 Live list (LL)

This parameter indicates the current operational status, whether a corresponding node is on line and is running normal in the Type 11 fieldbus. Each bit of value “1” indicates the corresponding node on-line and normal, and a node of value “0” is off-line.

V(LL) is used by the DLE and is generated from the information conveyed by the SYN frame. Live list is a collection of 8 words of 32-bit length, each of which corresponds to each node in the Type 11 fieldbus and indicates the current operational status. Each bit in the LL word corresponds to the node number V(TN) in a sequential order from 0 to 255 in little endian format.

Table 19 shows the structure of LL parameter.

Table 19 – LL parameters: 15th to 46th octets

Live-list																	
LL ($2^7 - 2^0$)									LL ($2^8 - 2^{15}$)								
15 th	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	16 th
17 th	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24	18 th
19 th	39	38	37	36	35	34	33	32	47	46	45	44	43	42	41	40	20 th
21 th	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56	22 th
23 th	71	70	69	68	67	66	65	64	79	78	77	76	75	74	73	72	24 th
⋮																	⋮
⋮																	⋮
41 th	215	214	213	212	211	210	209	208	223	222	221	220	219	218	217	216	42 th
43 th	231	230	229	228	227	226	225	224	239	238	237	236	235	234	233	232	44 th
45 th	247	246	245	244	243	242	241	240	255	254	253	252	251	250	249	248	46 th

6.2.4 User data

No user data is conveyed by the SYN DLPDU.

6.2.5 Sending SYN DLPDU

A DLE of current SYN node sends out SYN DLPDU with a set of parameters, as stated above, which are managed, maintained and set to SYN DLPDU by the DLE of current SYN node.

6.2.6 Receiving SYN DLPDU

Each DLE of the node, which is on line or is to be on line, except SYN node takes the following actions on reception of SYN DLPDU.

- As for the star-architecture, the value of RMSEL is used to determine the switching control for receiving frames out of one of two receive-channel A and B corresponding to each of the redundant medium A and B, and is reflected in V(RCS).
- Each value of ST, Th, Ts and TI, however the TI is not used in 100 Mbps operation of the loop-architecture, received in SYN DLPDU and the corresponding value of V(ST), V(Th), V(Tm) and V(TI) in a node is compared respectively. If the value is different, then each value of ST, Th, Ts and TI is to be a new value of each variable V(ST), V(Th), V(Tm) and V(TI) respectively.
- The value of LL is to be a new value of V(LL).
- The DLE, which has sent REQ DLPDU out, shall confirm that the corresponding bit to the V(TN) in LL becomes “true”. If the value of the corresponding bit of LL is “true”, then the node of the V(TN) is on line. The DLE of the V(TN) shall start to send out its Time-critical Cyclic data every time immediately after the DLE has received CMP or DT-CMP DLPDU from a node that is on line and of which the node identifier number can be obtained from the LL and is lower in sequential order in the LL.

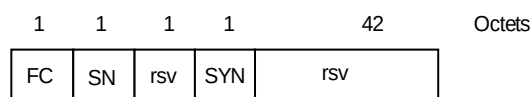
6.3 Transmission complete DLPDU (CMP)

6.3.1 General

CMP DLPDU is sent to indicate all of the time-critical cyclic data transmission completed at the end of data transmission. Two types of DLPDU for indicating the data transmission completed is specified, one is CMP DLPDU and the other is DT DLPDU.

6.3.2 Structure of CMP DLPDU

Figure 9 shows the structure of the CMP DLPDU.



Key

FC = Frame control (11001000)

SN = Source node number

rsv = Reserved (0x0)

SYN = SYN node number

Figure 9 – Structure of CMP DLPDU

6.3.3 SYN node number (SYN) parameter

This parameter indicates the number of the V(TN) of the current SYN node.

6.3.4 User data of CMP DLPDU

No user data is conveyed by the CMP DLPDU.

6.3.5 Sending CMP DLPDU

CMP DLPDU is sent by the DLE at the timing and condition specified as follows.

- a) The DLE which is in the assigned time frame to send out its data and is just sending out its data in the V(TSYN) time period, shall send out CMP DLPDU at the time when the DLE has completed sending out all of the data both of Time-critical Cyclic data and Sporadic message data. The CMP DLPDU is sent by the last frame out of the node to notify its data transmission completed to all Type 11 fieldbus nodes.

When a DLE sends out time-critical cyclic data as the last DLPDU, the DLE shall send out DT-CMP DLPDU in substitution of CMP DLPDU.

- b) A DLE of SYN node shall wait for CMP or DT-CMP DLPDU from the DLE, which is sending its data out, in the duration of V(SCMP). When the DLE of SYN node has failed to receive CMP or DT-CMP DLPDU, the DLE of SYN node shall initiate sending out CMP DLPDU in substitution for the DLE which has not send out its CMP or DT-CMP DLPDU.

6.3.6 Receiving CMP DLPDU

Each DLE of the node, which is on line, shall activate the followings on reception of CMP or DT-CMP DLPDU:

- a) The value of V(Tsl) is updated to the number of V(TN) of which node is to send out its date, after searched next node out of V(LL), translated into and get actual node identifier number of V(TN). In searching the next node, the DLE checks and extracts the first bit of V(LL) being “true” in the greater order than the corresponding bit position of the node which has finished sending out. When the first bit of being “true” is found, and according to the bit position found in the V(LL), the DLE shall translate the bit position into the actual node identifier number of V(TN),

- b) On condition that the value of the node identifier number, which is translated from the bit position in the V(LL), is equal to the number of V(TN) of this node, the DLE shall begin to send out data of every high-speed time-critical cyclic data, Medium-speed time-critical cyclic data, Sporadic message data and Low-speed time-critical cyclic data in this order in case of the star-architecture. On the other hand in case of the loop-architecture, the order is every high-speed time-critical cyclic data, Medium-speed time-critical cyclic data, Low-speed time-critical cyclic data and Sporadic message data. However the Low-speed time-critical cyclic data is not used in 100 Mbps operation of the loop-architecture.

The Medium-speed Time-critical cyclic data, the Sporadic message data and the low-speed time-critical cyclic data is able to send out from the node if the condition is met respectively, and each data transmission shall terminate when one of the following conditions met:

- All the data with each priority level has finished sending out,
- The corresponding observational timer has expired,
- The T(MTHT) for High-speed time-critical cyclic data transmission has been expired.

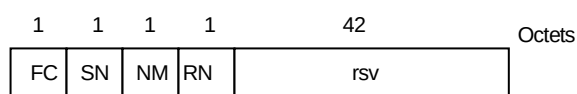
6.4 In-ring request DLPDU (REQ)

6.4.1 General

REQ DLPDU is sent by the DLE to the SYN node to claim to be added in Type 11 fieldbus.

6.4.2 Structure of REQ DLPDU

Figure 10 shows the structure of the REQ DLPDU.



Key

FC = Frame control (11000010)

SN = Source node number

NM = Node mode

RN = Recipient node number

rsv = Reserved (0x0)

Figure 10 – Structure of REQ DLPDU

6.4.3 Node mode (NM) – ESYN parameter

This parameter indicates the mode of operation of the DLE of a node which is claiming to be added in Type 11 fieldbus. The NM is used for 1 000 Mbps operation of the loop-architecture, otherwise the value is set to 0x0.

Table 20 shows the format of the Node mode (NM).

The “ESYN” parameter indicates that the DLE of a node is capable of SYN node. The value “Enable” / “true” is capable and the value “Disable” / “false” is not capable of SYN node. The V(ESYN) holds the value of ESYN and is used by the DLE of the node.

Table 20 – NM parameter

NM: Node mode							
Reserved (0x0)			ESYN	Reserved (0x0)			
7	6	5	4	3	2	1	0

6.4.4 Recipient node number (RN) parameter

This parameter designates the recipient node number of which the node is to receive the REQ DLPDU. The RN is used for loop-architecture, otherwise the value is set to 0x0.

Table 21 shows the format of the RN parameter.

Table 21 – RN parameter

RN: Recipient node number							
V(RN) (1 to 254)							
7	6	5	4	3	2	1	0

6.4.5 User data of REQ DLPDU

No user data is conveyed by the REQ DLPDU.

6.4.6 Sending REQ DLPDU

In case of the 100 Mbps operation both of the star- and the loop-architecture, the REQ DLPDU is sent by the DLE at the timing and condition specified as follows:

- REQ DLPDU is sent by the DLE of a node which attempts to be a member in the Type 11 fieldbus, claiming to be added and to be on-line, on condition that the DLE has received a SYN DLPDU with the value of PN field equal to the V(TN) and further in the time frame of the MAC control period of Tmac;
- The start of the MAC control period of Tmac is immediately after the DLE has completely received a CMP DLPDU or a DT-CMP DLPDU out of a node with the biggest number of the value of V(TN) in Type 11 fieldbus.

On the other hand in case of the 1 000 Mbps operation of the loop-architecture, the REQ DLPDU is sent by the DLE at the timing and condition specified as follows:

- REQ DLPDU is sent by the DLE of a node which attempts to be a member in the Type 11 fieldbus, claiming to be added and to be on-line, on condition that the DLE has received a SYN DLPDU on the V(PP) and further in the time frame of the MAC control period during the time slot of V(TSYN) in which the DLE has received the SYN DLPDU;
- The start of the MAC control period is immediately after the DLE has completely received a CMP DLPDU or a DT-CMP DLPDU out of a node with the biggest number of the value of V(TN) in type 11 fieldbus. The DLE sends REQ DLPDU out of the V(PP) of the node;
- The original REQ DLPDU is once blocked at the neighboring node to the attempting node in direction of V(PP). The neighboring node sends out the REQ DLPDU to the SYN node in the time frame of V(LN) equal to the value of V(TN) during the time slot of V(TSYN) before sending out DT DLPDU.

6.4.7 Receiving REQ DLPDU

REQ DLPDU is received by the DLE of the current SYN node in operation. Received REQ DLPDU, the DLE extracts the number of V(TN) of sending DLE out of the value of SN field in the REQ DLPDU, and sets to “1” (being on-line or live) the corresponding bit of the V(TN) in V(LL). Immediately after the MAC control period of Tmac during which the DLE received a REQ DLPDU, the DLE sends out SYN DLPDU with the value of extracted number of V(TN) in the LL field.

6.5 Claim DLPDU (CLM)

6.5.1 General

CLM DLPDU is sent by the DLE claiming for re-synchronization of the Type 11 fieldbus on condition that the current SYN node malfunctions and the T(SL) timer to monitor SYN node inactive has expired. CLM DLPDU can be sent from the node which is designated and permitted to be the SYN node. The value of T(SL) of each node is different and depends on the value of the V(ST) and the V(TN).

6.5.2 Structure of CLM DLPDU

Figure 11 shows the structure of the CLM DLPDU.

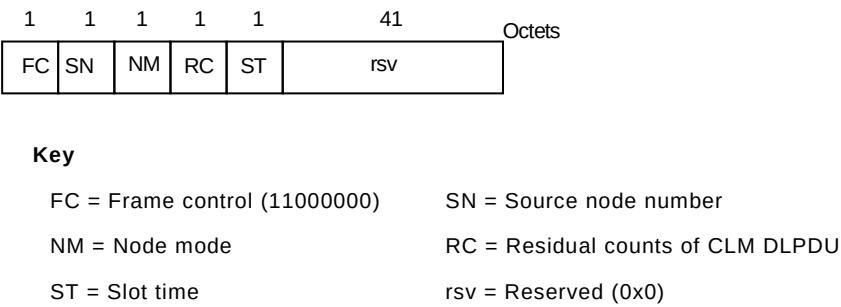


Figure 11 – Structure of CLM DLPDU

6.5.3 Parameter of CLM DLPDU

6.5.3.1 Residual counts of CLM DLPDU parameter (RC)

This parameter indicates the residual counts for the DLE to send out CLM DLPDU. The value is decremented in the range of V(ST) to 0. The default value is 20. Reached to count “0”, the DLE becomes a SYN DLE.

Table 22 shows the structure of the CLM parameter.

Table 22 – CLM parameter: 4th octet

Residual Counts of CLM DLPDU							
V(RC) (2 ⁷ - 2 ⁰)							
7	6	5	4	3	2	1	0

6.5.3.2 Slot time (ST) parameter

The value of this parameter is set to that defined in 6.2.3.3.

6.5.3.3 Node mode (NM) parameter

The value of this parameter is set to that defined in 6.4.3.

6.5.4 User data of CLM DLPDU

No user data is conveyed by the CLM DLPDU.

6.5.5 Sending and receiving CLM DLPDU

CLM DLPDU is sent by the DLE with V(ESYN) enable, which is provided with the SYN node function, claiming for re-synchronization of the Type 11 fieldbus.

After the DLE has been powered and initialized, once the T(SL) timer has expired, the T(ST) timer is successively triggered. When the T(ST) timer has expired, CLM DLPDU is sent by the DLE.

The DLE shall send out CLM DLPDU in a predefined number of times, which depends on the value of V(ST). The number of times is calculated by the following formula, in which “N” is the number of times.

$$N = \text{round up } [2 \times V(\text{MD}) + V(\text{MRT}) / 2 + 2]$$

When a CLM DLPDU is being sent, and when the collision has happened, the DLE immediately stops sending out CLM DLPDU and starts to monitor and check the condition over the common medium.

When the DLE has successfully received a CLM DLPDU from other DLE, the DLE shall extract the value of SN within the CLM DLPDU and compare the value of SN with its V(TN). When the value of SN is lower than the V(TN), the DLE shall not persist in sending out another CLM DLPDU. On the contrary when the V(TN) is lower, the DLE shall again attempt to send out CLM DLPDU after both of T(SL) and consecutive T(ST) expired.

When the DLE has successfully sent CLM DLPDU in total counts of V(ST), the DLE becomes SYN DLE which functions as new SYN node. On the other hand SYN DLPDU received from other DLE, the DLE shall cease to send out CLM DLPDU for re-synchronization.

The DLE that is not provided with the SYN node function or is not designated to be a SYN node, shall not send out CLM DLPDU, and furthermore shall not respond to CLM DLPDU received from another node.

6.6 Command DLPDU (COM)

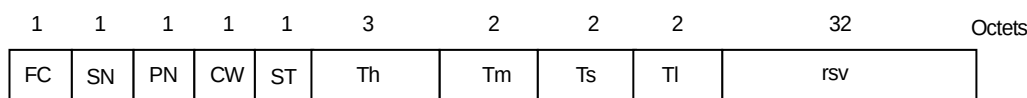
6.6.1 General

COM DLPDU is sent by the DLE to ask the DLE of the SYN node to equalize and change the value of the Type 11 fieldbus parameters of all member nodes, and the change shall be reflected in the SYN DLPDU to be sent in the next V(Th) time period. The Type 11 fieldbus parameters to be equalized are as follows:

- a) V(ST): Slot time(ST),
- b) V(Th), V(Tm), V(Ts), V(TI): The cyclic time periods, the target-token-rotation time period,
- c) CW: Node control word.

6.6.2 Structure of COM DLPDU

Figure 12 shows the structure of the COM DLPDU.



Key

FC = Frame control (11000100)	SN = Source node number
PN = Transmission permits node number	CW = Control word
ST = Slot time	Th = High-speed periodic time
Tm = Medium-speed periodic time	Ts = Sporadic message rotation time
TI = Low-speed periodic time	rsv = Reserved (0x0)

Figure 12 – Structure of COM DLPDU

6.6.3 Parameters of COM DLPDU

The parameters of COM DLPDU are equal to that of the SYN DLPDU specified in 6.2.3.

6.6.4 User data of COM DLPDU

No user data is conveyed by the COM DLPDU.

6.6.5 Sending and receiving COM DLPDU

The DLE is activated to send COM DLPDU by receiving DLM_SET_Value primitive, especially requiring change of each value of V(ST), V(Th), V(Tm), V(Ts), V(TI) and CW.

In case of 100 Mbps operation both of the star- and the loop-architecture, the COM DLPDU is sent by the DLE at the timing and condition specified as follows:

- COM DLPDU is sent out to the SYN node by the DLE in the time frame of the MAC control period during the time slot of V(TSYN) in which the DLE has received the SYN DLPDU with equal value of PN to that of the V(TN) after receiving the DLM_SET_Value primitive.

On the other hand in case of 1 000 Mbps operation of the loop-architecture, the COM DLPDU is sent by the DLE at the timing and condition specified as follows:

- COM DLPDU is sent out to the SYN node by the DLE in the time frame of V(LN) equal to the value of V(TN) during the time slot of V(TSYN) in which the DLE has received the SYN DLPDU with equal value of PN to that of the V(TN) after receiving the DLM_SET_Value primitive;
- The DLE sends COM DLPDU before sending out either or both DT DLPDU and RAS DLPDU.

When the DLE of the SYN node has successfully received COM DLPDU from the DLE of another node, the DLE reflects the requirement by COM DLPDU to SYN DLPDU to be sent in the next V(Th) time period.

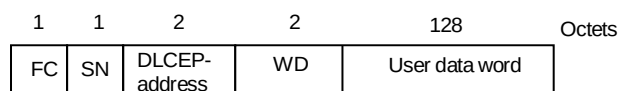
6.7 Cyclic data and cyclic data with transmission complete DLPDU (DT) and (DT-CMP)

6.7.1 General

Either DT DLPDU or DT-CMP DLPDU is used to convey the Time-critical cyclic data to all Type 11 fieldbus member nodes. The difference of DT DLPDU and DT-CMP DLPDU is indication of transmit completion or not.

6.7.2 Structure of DT DLPDU

Figure 13 shows the structure of the DT DLPDU.



Key

FC = Frame control (DT: pp000111, DT-CMP: pp001111
"pp" corresponds to priority level.)

SN = Source node number WD = Word length

Figure 13 – Structure of DT DLPDU

6.7.3 Parameters of DT DLPDU

6.7.3.1 DLCEP-address parameter

This parameter indicates DLCEP for predefined multi-point publisher DL-connection.

Table 23 shows the structure of the DT parameter.

Table 23 – DT parameter: 3rd and 4th octets

DLCEP-address															
DLCEP-address ($2^7 - 2^0$)								DLCEP-address ($2^8 - 2^{15}$)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.7.3.2 Word length parameter (WD)

6.7.3.2.1 WD for star-architecture

This parameter indicates the length of the User-data-word. The value is 0x40, and the value is held into V(BW).

6.7.3.2.2 WD for loop-architecture

This parameter indicates the length of the User-data-word. The value is one of 0x8, 0x0c, 0x10, 0x14, 0x18, 0x24, 0x30 and 0x40 selectable for 100 Mbps operation, 0x14 and 0x40 selectable for 1 000 Mbps operation. The value is held into V(BW).

6.7.4 Sending DT or DT-CMP DLPDU

The CTRC(Cyclic-transmission TX/RC control) manages to send out DT DLPDU or DT-CMP DLPDU on receipt of DL-PUT primitive from the DL-user.

The type of High-speed cyclic data transmission or Medium-speed cyclic data transmission or Low-speed cyclic data transmission shall be designated to the pp subfield of FC of the DT DLPDU and the DT-CMP DLPDU. The Low-speed cyclic data transmission is not used in 100 Mbps operation of the loop-architecture.

6.7.5 Receiving DT or DT-CMP DLPDU

The DLE that has received DT DLPDU or DT-CMP DLPDU shall update the corresponding Receive_buffer associated with DLCEP address field in the DT DLPDU or the DT-CMP DLPDU, and notify the DLS-user using DL-Buffer-Received indication primitive that the data in the specified receive-buffer is updated and is available to read out.

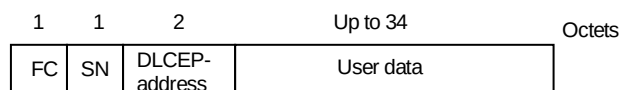
6.8 RAS DLPDU (RAS)

6.8.1 General

RAS DLPDU is used for transfer the RAS(Reliability, Availability, Serviceability) related information to all member node.

6.8.2 Structure of RAS DLPDU

Figure 14 shows the structure of the RAS DLPDU.



Key

FC = Frame control (11000101)

SN = Source node number

Figure 14 – Structure of RAS DLPDU

6.8.3 DLCEP-address parameters

This parameter indicates the DLCEP for predefined multi-point publisher DL-connection.

Table 24 shows the structure of the RAS parameter.

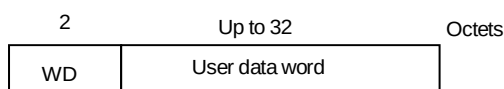
Table 24 – RAS parameter: 3rd and 4th octets

DLCEP-address															
DLCEP-address ($2^7 - 2^0$)								DLCEP-address ($2^8 - 2^{15}$)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.8.4 User data

RAS DLPDU conveys the user data, which is specifically associated with the RAS related information of the local DLE, up to 34 octets one time, however in which WD parameter of 2 octet length is included and used for loop-architecture so that user data is conveyed up to 32 octets. Figure 15 shows the structure of user data of loop-architecture.

The DLE of each local node may handle the RAS related information over 34 octets, and then the total number of the RAS related information of each node is broken down into several fragments and each fragment is conveyed on the RAS DLPDU.



Key

WD = Word length of User data word

Figure 15 – Structure of User data of loop-architecture

6.8.5 Sending and receiving RAS DLPDU

The DLE of local node can broadcast the RAS related information to all Type 11 fieldbus member nodes using the RAS DLPDU.

In case of 100 Mbps operation both of the star- and the loop-architecture, the RAS DLPDU is sent by the DLE at the timing and condition specified as follows:

- The RAS DLPDU is sent during TMAC time duration by the DLE of each local node each by each in order of V(TN), therefore the transmission of the RAS DLPDU or broadcasting of the RAS related information of local node is carried out on background basis.

On the other hand in case of 1 000 Mbps operation of the loop-architecture, the RAS DLPDU is sent by the DLE at the timing and condition specified as follows:

- The RAS DLPDU is sent out in the time frame of V(LN) equal to the value of V(TN) during the time slot of V(TSYN) in which the DLE has received the SYN DLPU with equal value of PN to that of the V(TN), and the DLE sends out RAS DLPDU before sending out DT DLPDU.

When the RAS DLPDU has been received, the DLE shall update the corresponding Receive_buffer associated with the DLCEP address field in the RAS DLPDU and notify the DLS-user using the DL-Buffer-Receive indication primitive that the data in specific receive-buffer is updated and is available to read out.

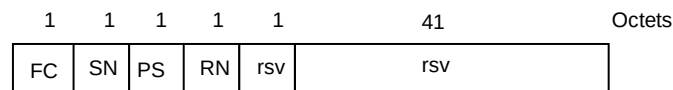
6.9 Loop repeat request DLPDU (LRR)

6.9.1 General

LRR DLPDU is specific use for loop-architecture and is sent for recovering the open-ring upon failure.

6.9.2 Structure of LRR DLPDU

Figure 16 shows the structure of the LRR DLPDU.



Key

FC = Frame control (11100110)

SN = Source node number

PS = Port status

RN = Recipient node number

rsv = Reserved (0x0)

Figure 16 – Structure of LRR DLPDU

6.9.3 Parameters of LRR DLPDU

6.9.3.1 Structure of Port status (PS)

6.9.3.1.1 General

This parameter indicates the status of the node. Table 25 shows the structure of the port status.

Table 25 – Format of the PS parameter: 3rd octet

PP: Port status							
Reserved	Primary Port (PP)	Send-enable-A/-B		Receive-enable-A/B		Forward enable-A/-B	
7	6	5	4	3	2	1	0

6.9.3.1.2 Primary port (PP)

This parameter indicates the ring-port through which the SYN-frame first received since this node is power-up, or in the system start-up, or in the system initialization.

Table 26 shows the value of the primary port parameter.

Table 26 – The value of the PP parameter

Primary port (PP)	Value	Description
Ring-port-A	0	The Ring-port-A is as the primary port.
Ring-port-B	1	The Ring-port-B is as the primary port.

6.9.3.1.3 Send- Receive- and Forward-enable parameters

These parameters indicate the operational status of the ring-port whether the ring-port is in the send-, the receive- and the forward-enabled or not.

Table 27, Table 28 and Table 29 show the value of the send-, the receive-, the forward-enable-A / -B parameters.

Table 27 – The value of the send-enable-A/-B

Send enable-A/-B	Value	Description
Send-disable	00	Both the Ring-port-A and -B are in disabled
Send-enable-A	01	The Ring-port-A is in enabled
Send-enable-B	10	The Ring-port-B is in enabled
Send-enable-A and -B	11	Both the Ring-port-A and -B is in enabled

Table 28 – The value of the receive-enable-A/-B

Receive enable-A/-B	Value	Description
Receive-disable	00	Both the port-A and -B is in disabled
Receive-enable-A	01	The port-A is in enabled
Receive-enable-B	10	The port-B is in enabled
Receive-enable- A and -B	11	Both the port-A and -B is in enabled

Table 29 – The value of the forward-enable-A/-B

Forward enable-A/-B	Value	Description
Forward-disable	00	The forwarding between the port-A and B is disabled. Both Ring-port –A and –B are in Blocked
Forward-enable-A	01	The forwarding from the port-A to -B is enabled. Ring-port-A is in Non-blocked and Ring-port-B is in Blocked
Forward-enable-B	10	The forwarding from the port-B to -A is enabled. Ring-port-B is in Non-blocked and Ring-port-A is in Blocked
Forward-enable-A to -B	11	The forwarding between the port-A and B is enabled. Both Ring-port–A and –B are in Non-blocked

6.9.3.2 Recipient node number (RN)

This parameter designates the recipient node number of which the node is to receive the LRR DLPDU. The format is shown in Table 30.

Table 30 – RN parameter: 4th octet

RN: Recipient node number							
V(RN) (1 to 254)							
7	6	5	4	3	2	1	0

6.9.4 Sending and receiving LRR DLPDU

The type 11 fieldbus of loop-architecture is in a ring structure, in which each node is connected in series to two other nodes. The DLPDU is sent in both directions out of the node at a time so that the DLPDU is passed through sequentially between the nodes. In order to circumvent the DLPDU being circulated for ever in the ring, there exist open-points purposely under control over the type 11 fieldbus so that a DLPDU is not returned any more to the source node along through the circular path.

Figure 17 shows an open-ring structure under control as an example, and Table 31 summarizes the operation condition of each node.

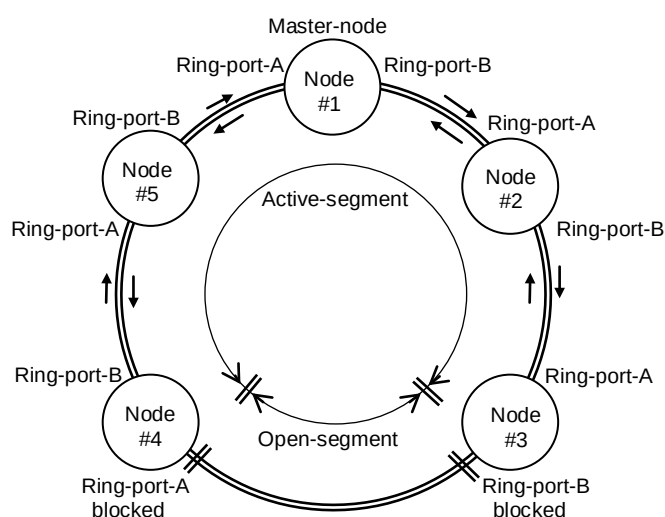


Figure 17 – Open-ring under control

Table 31 – Operational condition of the node

Node #	Node condition			
	Master/Slave -node state	Normal/Open state	Ring-port-A/-B state	
			Ring-port-A	Ring-port-B
1	Master	Normal	Non-blocked	Non-blocked
2	Slave	Normal	Non-blocked	Non-blocked
3	Slave	Open	Non-blocked	Blocked
4	Slave	Open	Blocked	Non-blocked
5	Slave	Normal	Non-blocked	Non-blocked

The DLE provides function to manage the remedy action in cooperation with the DLE in other nodes upon a single point of failure in order to maintain the communications between the nodes in healthy operation.

The sending and receiving LRR DLPDU to compensate a single point of failure are as follows:

- a) A DLE in the open-state (hereafter DLE1) which is at one end of current active-segment and suffers a single point of failure over the path to the node in the master-state, shall detect long-silence of reception;
- b) The DLE1 in the open-state, detecting the long-silence of reception, shall send out LRR DLPDU with the RN equal to the node-identifier-number of the adjacent node, that is another node (or DLE, hereafter DLE2) in open-state at the edge of the current active-segment, in both direction at a time at the timing given by the ACM;
- c) The DLE1:
 - shall wait to receive LRR DLPDU with the RN equal to V(TN);
 - shall enter into normal-state if receiving within a specified time period a LRR DLPDU with the RN equal to V(TN) out of the DLE2;
 - shall change the port status “Blocked” of the Ring-port to the current active-segment and further the port state “Non-blocked” of the Ring-port on the current open-segment if not receiving the LRR DLPDU with the RN equal to V(TN) within the specified time period out of the DLE which is the adjacent node on the current active-segment;
- d) The DLE2 which is receiving LRR DLPDU with the RN equal to V(TN), shall enter into normal-state;
- e) The DLE (hereafter DLE3) which is in normal-state and is over the path between the single point of failure and the node in open-state (DLE1), shall send out LRR DLPDU with the RN equal to the node-identifier-number of the adjacent node at the side of the Ring-port through which a LRR DLPDU is receiving, in both direction at a time;
- f) The DLE3:
 - shall wait during a specified time period, to receive LRR DLPDU with the RN equal to V(TN);
 - shall maintain in the normal-state if receiving within a specified time period a LRR DLPDU with the RN equal to V(TN) from the opposite side of the Ring-port through which the DLE3 first received a LRR DLPDU;
 - shall change the port status “Blocked” of a Ring-port at the opposite side of a Ring-port through which the DLE3 first received a LRR DLPDU if not receiving a LRR DLPDU with the RN equal to V(TN) from the Ring-port within the specific time period, and shall enter into open-state.

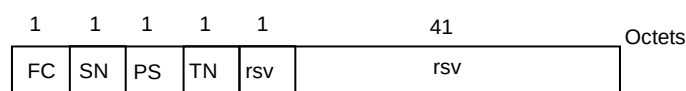
6.10 Loop diagnosis DLPDU (LPD)

6.10.1 General

The LPD DLPDU is sent to diagnose the condition of the links between the adjacent nodes.

6.10.2 Structure of LPD DLPDU

Figure 18 shows the structure of the LPD DLPDU.



Key

FC = Frame control (11100011)

SN = Source node number

PS = Port status

TN = This node number

rsv = Reserved (0x0)

Figure 18 – Structure of LPD DLPDU

6.10.3 Parameters of LPD DLPDU

6.10.3.1 Port status (PS)

This parameter is equivalent to the PS in 6.9.3.1.

6.10.3.2 This node number (TN)

This parameter indicates the node-identifier-number of V(TN), of which the node sent out the LPD DLPDU.

6.10.4 Sending and receiving LPD DLPDU

Each DLE periodically sends out LPD DLPDU with the node identifier number V(TN) of the node and the port status in operation so that the neighboring nodes are able to collect these information.

- Each DLE shall send out the LPD DLPDU in both direction at a time at the specific instant which is given periodically by the ACM of loop-architecture;
- The LPD DLPDU shall be sent out of each node all together at the same instant;
- Each DLE on receiving LPD DLPDU shall not forward the LPD DLPDU receiving from one Ring-port to the other Ring-port.

7 DLE elements of procedure

7.1 DLE elements of procedure for star-architecture

7.1.1 Overall structure

The DLL is composed of control elements of Cyclic transmission TX/RX Control (CTRC), Sporadic TX/RX Control (STRC), Access Control Machine(ACM), Redundancy Medium Control(RMC), TX/RX Framer, Octet Serializer, Octet Deserializer and DLL Management Interface.

The ACM as the primary control element provides the function for deterministic medium access control cooperating with the CTRC, the STRC and the RMC for reliable and efficient support both of higher-level connection-mode and connection-less mode data transfer services.

The DLL management interface provides DLL management functions.

Figure 19 depicts the overall structure of the DLL.

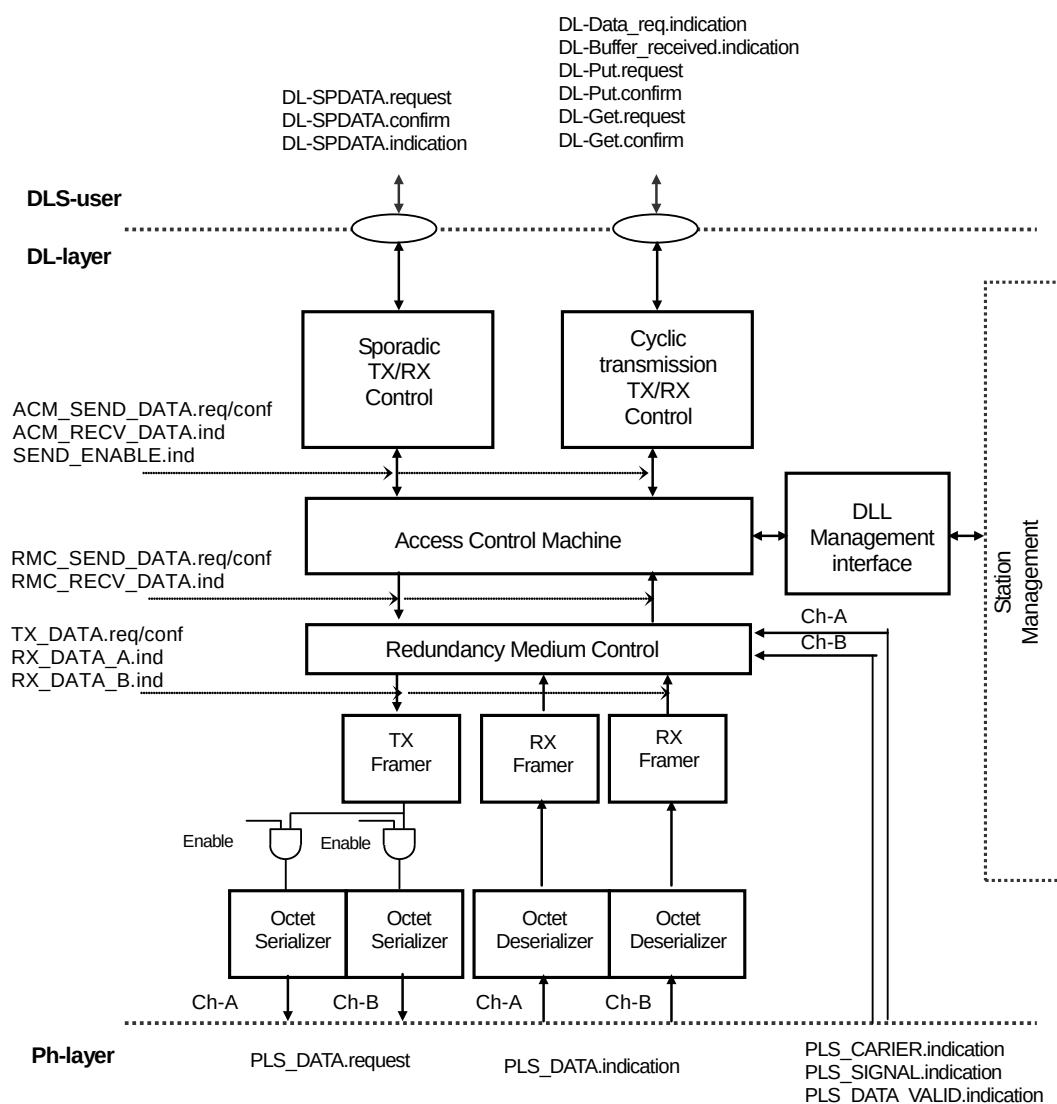


Figure 19 – Overall structure of DLL

7.1.2 Initialization

Upon power-up or after the reception of the DLM-RESET request primitive, the DLE shall go into the OFF-LINE state. When in OFF-LINE, the DLE shall not transmit and shall ignore any DLPDU received.

When all the variables for normal DLE operation are set up by the DLM-Set-Value request primitive, the state is changed from OFF-LINE to STATION MODE CONTROL in which the DLE shall start normal DL operation.

Figure 20 depicts the DLE state transition.

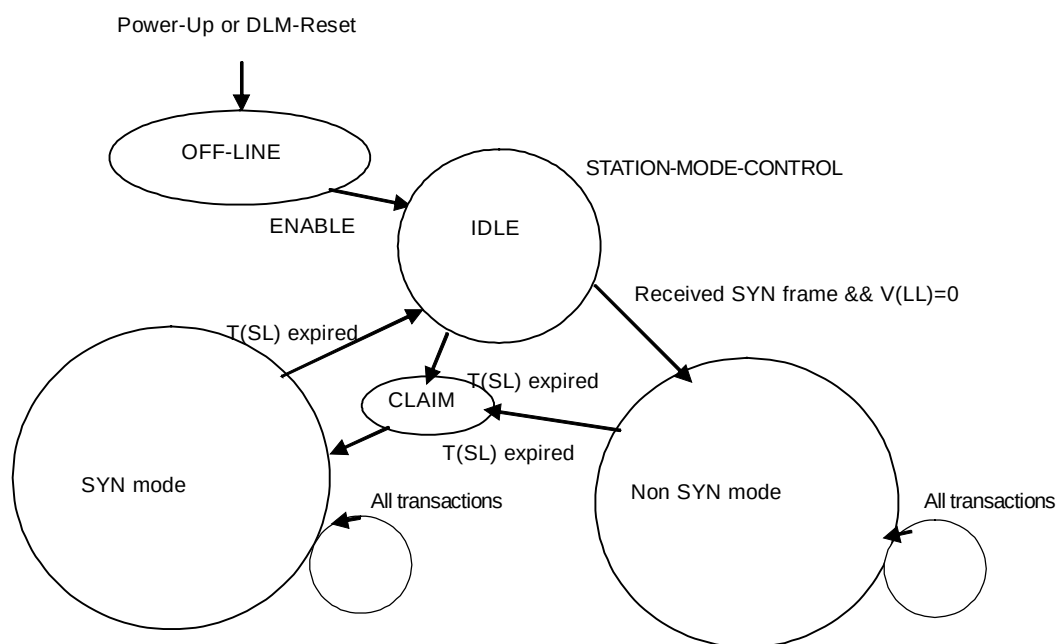


Figure 20 – DLE state transition

7.1.3 Cyclic transmission TX/RX control (CTRC)

7.1.3.1 Overview

The Cyclic transmission TX/RX control (CTRC) is responsible for buffering and dispatching the DLSDU received for the Time-critical cyclic data transfer between the DLS-user and the ACM.

DL-Data-Req indication primitive by the CTRC informs the DLS-user to initiate the data transfer from each DLS-user buffer to the send buffer of the DLE using DL-Put request primitive. The DLS-user buffer is addressed with the DLCEP-identifier parameter in the DL-Data-req indication primitive from the CTRC.

Each DLS-user buffer contains data for Time-critical cyclic data transmission. After the DLS-user data received, the CTRC issues ACM_SEND_DATA request primitive to initiate ACM sending out the DLS-user data as DT or DT-CMP DLPDU.

DL-Data-Req indication primitive is issued by the CTRC in response to SEND_ENABLE indication primitive from the ACM. The SPEED Class parameter in SEND_ENABLE indication primitive allows the CTRC to activate transfer of corresponding class of data-buffer by the DLS-user. The classes designated by the speed-class parameter correspond to the classes of High-speed cyclic data, Medium-speed cyclic data and Low-speed cyclic data transfer.

The data transfer of each Speed-class designated is dependent on the level of transfer priority:

- As for Priority class 3, which corresponds to the High-speed cyclic data transfer, when received SEND_ENABLE indication primitive, the CTRC shall handle all of the data of every DLS-user buffer corresponding to the High-speed Cyclic data transfer or the High-speed class is to be delivered to the ACM and to be sent on the transmission medium all at once on each occasion;

- b) As for Priority class 2 and 0, which corresponds to the Medium-speed cyclic data transfer or the Medium-speed class and the Low-speed cyclic data transfer or the Low-speed class respectively, the CTRC shall handle on each occasion the data of every DLS-user buffer corresponding to the Medium-speed and the Low-speed class is sent or not depending on the condition. The token-holding time of each class 2 and 0 governs the condition. After sending out all the High-speed cyclic data, the Medium-speed cyclic data is sent. If the token-holding time ends during sending the Medium-speed cyclic data, the data transfer is interrupted and the CMP DLPDU or the DT-CMP DLPDU is sent at this occasion. On the next occasion, the remainder of the previous Medium-speed cyclic data is to be sent depending on the same condition. As for the Low-speed cyclic data, the same control as for the Medium-speed cyclic data transfer is to be done. The occasion happens every V(TSYN) time period.

When the ACM_RECV_DATA indication primitive has been received from the ACM, the CTRC stores the DLPDU by the ACM into each receive-buffer, which is addressed with the DLCEP-identifier parameter in ACM_RECV_DATA indication primitive. Furthermore, the CTRC issues DL-Buffer-received indication primitive to inform the DLS-user the receive-buffer of the DLE is updated and is available to read out. The DLS-user can read out the data in the receive-buffer using DL-Get request primitive in response to DL-Buffer-received indication primitive from the CTRC.

7.1.3.2 Primitive definitions

7.1.3.2.1 Primitives exchanged between DLS-user and CTRC

The primitives exchanged between DLS-user and CTRC is shown in Table 32 and the primitives exchanged between the CTRC and the ACM is summarized in Table 33.

Table 32 – Primitives exchanged between DLS-user and CTRC

Primitive name	Source	Associated parameters
DL-Data-req.indication	CTRC	DLCEP-identifier
DL-Put.request	DL-user	DLCEP-identifier, DLSDU-length, DLSDU
DL-Put.confirm	CTRC	DLCEP-identifier, Status
DL-Buffer-received.indication	CTRC	DLCEP-identifier
DL-Get.request	DL-user	DLCEP-identifier
DL-Get.confirm	CTRC	DLCEP-identifier, DLSDU-length, DLSDU, Status

Table 33 – Primitives exchanged between CTRC and ACM

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_RECV_DATA.ind	ACM	DLPDU

7.1.3.2.2 Parameters used with primitives exchanged between DLS-user and CTRC

The parameters used for interaction between the DLS-user and the CTRC are summarized in Table 34.

Table 34 – Parameters used with primitives exchanged between DLS-user and CTRC

Parameter name	Description
DLCEP-identifier	Identifier to designate Send-Buffer or Receive-Buffer
DLSDU	The contents of Send-Buffer or Receive-Buffer, which is Time-critical Cyclic data processed by CTRC
DLSDU-length	The length of DLSDU
Speed-class	Speed-class to designate the class of Time-critical Cyclic data transfer, that is for High-speed Cyclic data, Medium-speed Cyclic data and Low-speed Cyclic data
Status	Status report whether the requested service is successfully provided or failed for the reason specified

7.1.3.3 CTRC state table

The state transition diagram of the CTRC is depicted in Figure 21, and the CTRC state table is shown in Table 35.

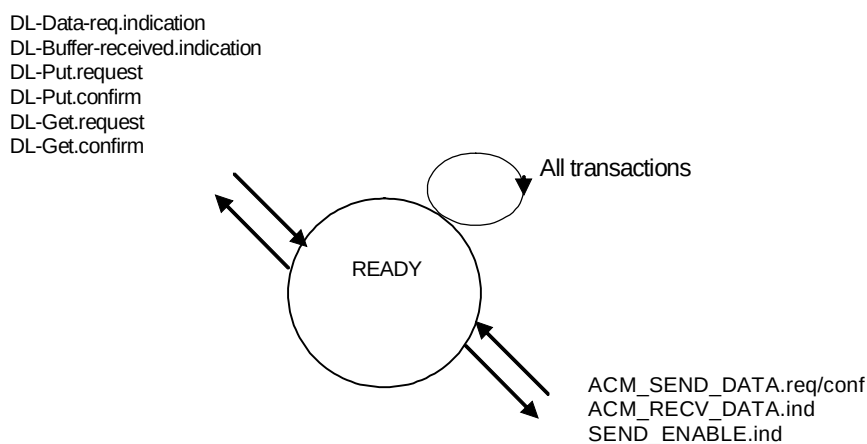


Figure 21 – State transition diagram of CTRC

Table 35 – CTRC state table

#	Current state	Event /condition ⇒actions	Next state
1	READY	DL-Put.request {DLCEP-identifier, DLSDU-length, DLSDU} / CHECK_PAR_PUT (DLCEP-identifier, DLSDU) = "True" ⇒ PUT_BUFFER (DLCEP-identifier, DLSDU) DL-Put.confirm { DLCEP-identifier, DLSDU-length, Status := "success" }	READY
2	READY	DL-Put.request {DLCEP-identifier, DLSDU-length, DLSDU} / CHECK_PAR_PUT (DLCEP-identifier, DLSDU) = "False" ⇒ DL-Put.confirm { DLCEP-identifier, DLSDU-length, Status := "failure" }	READY
3	READY	DL-Get.request {DLCEP-identifier } / CHECK_PAR_GET (DLCEP-identifier) = "True" ⇒ DL-Get.confirm { DLCEP-identifier, DLSDU-length, DLSDU := GET_BUFFER(DLCEP-identifier), Status := "success" }	READY
4	READY	DL-Get.request {DLCEP-identifier} / CHECK_PAR_GET (DLCEP-identifier) = "False" ⇒ DL-Get.confirm { DLCEP-identifier, DLSDU-length, DLSDU := null, Status := "failure" }	READY
5	READY	SEND_ENABLE.ind {Speed-class} / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "True" ⇒ DL-Data-req indication { DLCEP-identifier := GET_NEXT_ID(Speed-class) } DLSDU := GET_BUFFER (DLCEP-identifier) DLPDU := BUILD_DT (Speed-class, DLCEP-identifier, DLSDU) ACM_DATA.req { DLPDU } ACM_DATA.conf { } -- immediate ⇒ NEXT(Speed-class)	READY
6	READY	SEND_ENABLE.ind {Speed-class} / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "False" ⇒ ACM_DATA.req { null -- no data } ACM_DATA.conf { } -- immediate ⇒ NEXT(Speed-class)	READY
7	READY	ACM_DATA.ind {DLPDU} / Class <> "SCOPRADIC" && CHECK_PAR_DT (DLPDU) ⇒ DL-Buffer-received.indication { DLPDU.DLCEP-address }	READY

7.1.3.4 Functions of CTRC

All functions of the CTRC are summarized in Table 36.

Table 36 – CTRC functions table

Function Name	Input	Output	Operation
CHECK_PAR_PUT	DLCEP-identifier, DLSDU	True/False	Check that all parameters of DLCEP-identifier and DLSDU of DL-put.request primitive are valid. If valid, "True" is returned, otherwise "False" is returned
PUT_BUFFER	DLCEP-identifier, DLSDU	(none)	Store DLSDU into the Send-Buffer associated with the DLCEP-identifier
CHECK_PAR_GET	DLCEP-identifier	True/False	Check that that DLCEP-identifier of DL-get.request primitive is valid. If valid, "True" is returned
GET_BUFFER	DLCEP-identifier	DLSDU	Get DLSDU in the Receive-Buffer associated with DLCEP-identifier
CHECK_NEXT_SEND	Speed-class	True/False	Check that the DLSDU specified by Speed-class exists. If it exists, "True" is returned, otherwise "False" is returned
GET_NEXT_ID	Speed-class	DLCEP-identifier	Get next DLCEP-identifier of the DLS User-Buffer specified by Speed-class
BUILD_DT	Speed-class, DLCEP-identifier, DLSDU	DLPDU	Build into DLPDU of Time-critical Cyclic data out of the data in the Send-Buffer. DLPDU is assembled as follows. DLPDU.DA:= V(MGA) DLPDU.SA:= V(IA) DLPDU.Len/Type := 0x888b DLPDU.FC := Speed-class + "DT" DLPDU.SN := V(TN) DLPDU.DLCEP.address := DLCEP-identifier DLPDU.DLSDU := DLSDU
CHECK_PAR_DT	DLPDU	True/False	Check that the specified DLPDU is valid. If valid, "True" is returned
NEXT	Speed-class		Check that additional DLSDU specified by the Speed-class is remained. If remained, the control is returned to the top of current state

7.1.4 Sporadic TX/RX control (STRC)

7.1.4.1 Overview

The Sporadic TX/RX control (STRC) is responsible for buffering and dispatching in time DLSDU received for the Sporadic message data between DLS-user and the ACM.

7.1.4.2 Primitive definitions

7.1.4.2.1 Primitives exchanged between DLS-user and STRC

Table 37 shows all primitives exchanged between DLS-user and STRC and Table 38 shows all primitives exchanged between the STRC and the ACM.

Table 37 – Primitives exchanged between DLS-user and STRC

Primitive name	Source	Associated parameters
DL-SPDATA.request	DLS-user	DA, MSDU
DL-SPDATA.confirm	STRC	DA, Status
DL-SPDATA.indication	STRC	DA, SA, MSDU, Rec-Status

Table 38 – Primitives exchanged between STRC and ACM

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_SEND_DATA.ind	ACM	DLPDU

7.1.4.2.2 Parameters used with primitives exchanged between DLS-user and STRC

All parameters used with primitives exchanged between the DLS-user and the STRC is shown in Table 39.

Table 39 – Parameters used with primitives exchanged between DLS-user and STRC

Parameter name	Description
DA	Destination Address
SA	Source Address
MSDU	MAC service data unit
Status	Indicate whether the requested service of DL-SPDATA.request is successfully provided or failed for the reason specified
Rec-Status	Indicates whether the DLPDU had received without error or not
DLPDU	DLPDU for Sporadic message transfer
Speed-class	Speed-class requested by DLS-user. In Sporadic message transfer service, Speed-class is specified as "Sporadic" or Class 1.

7.1.4.3 STRC state table

The state transition diagram of the STRC is depicted in Figure 22, and the STC state table is shown in Table 40.

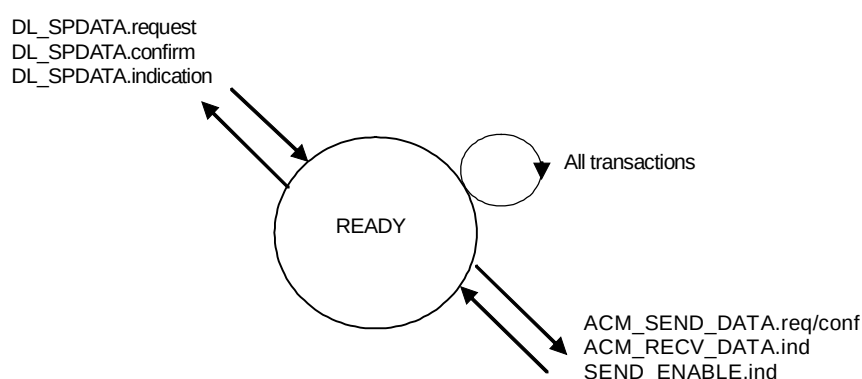
**Figure 22 – State transition diagram of STRC**

Table 40 – STRC state table

#	Current state	Event /condition ⇒actions	Next state
1	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () <> "Full" ⇒ QUEUE_SPDATA (DA, MSDU) DL-SPDATA.confirm { DA, MSDU, Status := "success" }	READY
2	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () = "Full" ⇒ DL-SPDATA.confirm { DA, MSDU, Status := "failure - The SPDATA Queue is full" }	READY
3	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "False" ⇒ DL-SPDATA.confirm { DA, MSDU, Status := "Failure - Invalid parameter" }	READY
4	READY	SEND_ENABLE.ind {Speed-class} / Speed-class = "SPORADIC" && CHECK_SPDATAQ () <> "Empty" ⇒ DLPDU := BUILD_SPDATA (DA, MSDU := DEQUEUE_SPDATA ()) ACM_SEND_DATA.req { DLPDU } ACM_SEND_DATA.conf { } -- immediate response ⇒ (no action taken)	READY
5	READY	ACM_SEND_DATA.ind {DLPDU} / Speed-class = "SPORADIC" && CHECK_SPDATA (DLPDU) ⇒ DL-Buffer-received.indication { DLPDU.DLCEP-address }	READY

7.1.4.4 Functions of STRC

All functions of the STRC are summarized in Table 41.

Table 41 – STRC functions table

Function name	Input	Output	Operation
CHECK_PAR_SPDATA	DA, MSDU	True/False	Check that all of parameters, DA and DSDU provided with DL-SPDATA.request are valid. If valid, "True" is returned
CHECK_SPDATAQ	(none)	status	Check that the Queue condition for SPDATA is fully queued. The returned status is any one of "Full", "Empty" and "Queued"
QUEUE_SPDATA	DA, MSDU	(none)	Queues the input data into the SPDATA Queue on a FIFO basis
DEQUEUE_SPDATA	(none)	DA, MSDU	Dequeue from the SPDATA queue on a FIFO basis
BUILD_SPDATA	DA, MSDU	DLPDU	Build into DLPDU of Sporadic message data. DLPDU is assembled as follows. DLPDU.DA := MSDU.DA DLPDU.SA := V(IA) DLPDU.Len/Type := MSDU.LEN/Type DLPDU.MSDU := MSDU.DLSDU
CHECK_RCV_SPDATA	DLPDU	True/False	Check that the specified DLPDU is valid. If valid, "True" is returned

7.1.5 Access control machine (ACM)

7.1.5.1 Overview

The access control machine (ACM) is responsible for deterministic medium access control and scheduling opportunities to send out the DLPDUs, for control to add and remove nodes and for restoration from network disruption to be system down.

The ACM has the primary responsibility for

- assuring that the local node detects and fully utilizes its assigned access time period;
- assuring that the local node does not interfere with the transmissions of other nodes, especially of the node transmitting the SYN DLPDU;
- detecting network disruption, and initiating the SYN DLPDU transmit for restoration of the network disruption from after prescribed time duration in which the SYN DLPDU is not heard;
- assuring a new node adding to and removing from the network.

7.1.5.2 ACM and the schedule support functions

The ACM functions schedule all communications between the DLEs participating in the Type 11 fieldbus, and the timing of this communications is controlled as to:

- fulfill the specific medium access control to give all the DLEs the opportunities to send out 2 kinds of class of the Time-critical cyclic data and Sporadic message data in timely, prioritized and deterministic fashion, and to detect the network disruption and to initiate the restoration in appropriate time, further to add and remove the nodes on line;
- provide 3 levels of the Time-critical data transfer opportunities of sending data to the node in sequential order and within each pre-specified time period, and that the data transfer of each level is performed within the pre-specified time duration (token holding time) and whether the data transfer of lower levels to be carried out or to be held over to the later cyclic time period depends on the level and the occasion though the top level of the data transfer is always carried out at every occasion, on the other hand, the whole volume of the data transfer of lower levels is transferred within each pre-specified time period;
- provide the sporadic message data transfer opportunities of sending out to the node that the request to transfer is happened sporadically by the DLS-user, and the data transfer is

performed within a pre-specified time period of the corresponding level of priority and is based on the regular ISO/IEC 8802-3 applications.

7.1.5.3 Primitive definitions

7.1.5.3.1 Primitives exchanged between ACM and RMC

Table 42 summarizes all primitives exchanged between the ACM and the RMC.

Table 42 – Primitives exchanged between ACM and RMC

Primitive name	Source	Associated parameters
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU

The parameters used with the primitives exchange between the ACM and the RMC are described in Table 43.

Table 43 – Parameters used with primitives exchanged between ACM and RMC

Parameter name	Description
RMSDU	SDU of RMC
RMC-status	Status indicating the results of the request to RMC

7.1.5.3.2 Primitives exchanged between ACM and CTRC

Table 44 summarizes all primitives exchanged between the ACM and the CTRC.

Table 44 – Primitives exchanged between ACM and CTRC

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	ACM_status
ACM_RECV_DATA.ind	ACM	DLPDU

The parameters used with the primitives exchange between the ACM and the CTRC are described in Table 45.

Table 45 – Parameters used with primitives exchanged between ACM and CTRC

Parameter name	Description
ACM_Status	Status indicating the results
DLPDU	SDU of ACM
Speed-class	Speed class indicating Time-critical Cyclic data transmission

7.1.5.3.3 Primitives exchanged between ACM and STRC

Table 46 lists all primitives exchanged between the ACM and the STRC.

Table 46 – Primitives exchanged between ACM and STRC

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_SEND_DATA.ind	ACM	DLPDU

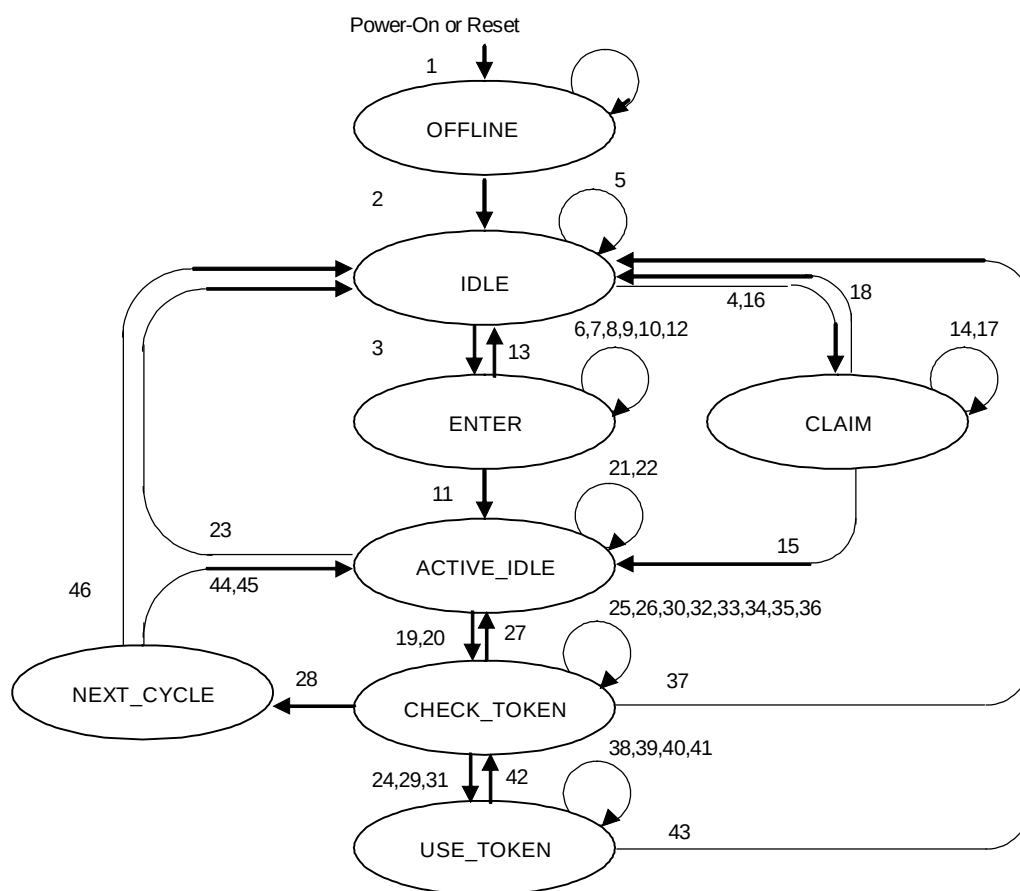
The parameters used with the primitives exchange between the ACM and the STRC are described in Table 47.

Table 47 – Parameters used with primitives exchanged between ACM and STRC

Parameter name	Description
ACM_Status	Status indicating the results
DLPDU	SDU of ACM
Speed-class	Speed class indicating Sporadic message data transmission

7.1.5.4 ACM state table

The state transition diagram is shown in Figure 23.

**Figure 23 – State transition diagram of ACM**

The state "OFFLINE" is entered on power-up or when DLM-Reset.request primitive is issued. In this state, all of the variables, parameters and configuration information are set up by the DLE. When "In_Ring_Desire" becomes true, the state changes to "IDLE".

The state "IDLE" is to wait and be ready to join the Type 11 fieldbus membership. When a SYN DLPDU has been received from another SYN node, the state change to "ENTER" in which the REQ DLPDU is sent to claim adding into the Type 11 fieldbus membership to the current SYN node. Moreover, in "IDLE" state, when the signals over the transmission media continues to be inactive during a specified time period, the state becomes "CLAIM", in which the node attempts to be a new SYN node.

The state "ACTIVE_IDLE" is in the Type 11 fieldbus membership, and the node manages to obtain the transmission right by the Type 11 fieldbus medium access control, either when the node is operating as the SYN mode by transmitting the SYN DLPDU or as the non-SYN mode receiving the SYN DLPDU by other SYN node, the state is in "CHECK_TOKEN".

The state "CHECK_TOKEN" waits until $V(Tsl)$ becomes equal to $V(TN)$ in order to send out the data over the transmission media. When $V(Tsl)$ becomes equal to $V(TN)$, the state changes to "USE_TOKEN".

The state "USE_TOKEN" is that the node is able to send out the Time-critical cyclic transmission data and the Sporadic transmission data. When all data has been sent or the token holding time is expired, the state changes to "NEXT_CYCLE".

The state "NEXT_CYCLE" is for solicit new node as a primary function.

The ACM state table is shown in Table 48.

Table 48 – ACM state table

#	Current state	Event /condition ⇒actions	Next State
1	Any states	POWER-ON or RESET =>	OFFLINE
2	OFFLINE	In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && RMSDU.PN = V(TN) && $V(LL) < V(TN) > = 0$ => -- receiving SYN frame and LL this node not live -- SYN_frame := RMSDU $V(LN) := 1$ START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	EXPIRED_TIMER (T(SL)) = "True" => RMSDU := BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / => -- receiving a frame except the above -- START_TIMER (T(SL), V(SL))	IDLE
6	ENTER	$V(LL) < V(LN) > = 1$ && $V(LN) < > V(TN)$ => START_TIMER (T(SCMP), V(Tsl))	ENTER

#	Current state	Event /condition ⇒actions	Next State
7	ENTER	V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER
8	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN) = V(MN) => -- Live Node = Maximum Node No. -- RMSDU := BUILD_PDU (REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
9	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++ START_TIMER (T(SL), V(SL))	ENTER
10	ENTER	RMC_SEND_DATA.conf { } => (none)	ENTER
11	ENTER	RMC_RECV_DATA.ind { RMSDU } && FC = SYN / V (LL)<V(TN)> = 1 => SYN_frame := RMSDU V(LN) := 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
12	ENTER	EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
13	ENTER	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
14	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time <= V(MN) => All_Slot_Time++ START_TIMER (T(SL), V(SL))	CLAIM
15	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = V(MN) => V(LL) = 0 V(LL)<V(TN)> := 1 RMSDU := BUILD_PDU (SYN) SYN_frame := RMSDU RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
16	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => START_TIMER (T(SL), V(SL))	IDLE
17	CLAIM	RMC_RECV_DATA.ind { RMSDU } => -- receiving a frame except the above -- START_TIMER (T(SL), V(SL))	CLAIM
18	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE

#	Current state	Event /condition ⇒actions	Next State
19	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN => -- receiving SYN frame-- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1	CHECK_TOKEN
20	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = SYN => -- sent SYN frame-- SYN_mode := "True" V(LN) := 1	CHECK_TOKEN
21	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = DT => -- sending DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
22	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC <> Type 11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
23	ACTIVE_IDLE	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
24	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) => -- Get token -- Speed-class := 3 – High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(MAC), V(MAC))	USE_TOKEN
25	CHECK_TOKEN	V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN
26	CHECK_TOKEN	V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	CHECK_TOKEN
27	CHECK_TOKEN	/V(LN) = V(MN) && SYN_frame .PN = V(TN) && SYN_mode = "False" => -- Non-SYN mode, on MAC control time -- IF Required COM frame THEN RMSDU := BUILD_PDU (COM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) ENDIF	ACTIVE_IDLE
28	CHECK_TOKEN	/V(LN) = V(MN) && SYN_frame .PN <> V(TN) && SYN_mode = "True" => -- in SYN mode, on MAC control time -- START_TIMER (T(MAC), V(MAC))	NEXT_CYCLE

#	Current state	Event /condition ⇒actions	Next State
29	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN)++ = V(TN) => -- Get token -- -- matching V(LN) and on receiving CMP frame -- STOP_TIMER (T(SCMP)) Speed-class := 3 -- High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
30	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN)++ <> V(TN) => -- not matching V(LN) and on receiving CMP frame -- START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN
31	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT-CMP && V(LN)++ = V(TN) => -- matching V(LN) and on receiving DT-CMP frame -- Speed-class := 3 -- High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
32	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT-CMP && V(LN)++ <> V(TN) => -- not matching V(LN) and on receiving DT-CMP frame START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN
33	CHECK_TOKEN	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = DT => -- sending DT frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
34	CHECK_TOKEN	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
35	CHECK_TOKEN	EXPIRED_TIMER (T(SCMP)) / => V(LN)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif START_TIMER (T(SL), V(SL))	CHECK_TOKEN
36	CHECK_TOKEN	RMC_SEND_DATA.conf { RMSDU } / RMSDU.FC = CMP => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
37	CHECK_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
38	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send DT or Sporadic frame -- RMPDU := BUILD_PDU (DLPDU) RMC_SEND_DATA.req { DLPDU } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	Current state	Event /condition ⇒actions	Next State
39	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (DLPDU.FC = DT) (DLPDU.FC <> Type-11 fieldbus && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
40	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (DLPDU.FC = DT) (DLPDU.FC <> Type-11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> 0 => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class := Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
41	USE_TOKEN	ACM_SEND_DATA.req { NIL } / Speed-class <> 0 => -- Next Speed-class -- Speed-class := Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
42	USE_TOKEN	ACM_SEND_DATA.req { NIL } / Speed-class = 0 => -- All sent -- RMSDU := BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
43	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
44	NEXT_CYCLE	/EXPIRED_TIMER (T(MAC)) = "True" =>	ACTIVE_IDLE
45	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ => -- receiving REQ from non-SYN node -- V(LL)<RMSDU.SN> := 1 RMSDU := BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } SYN_frame := RMSDU START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
46	NEXT_CYCLE	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE

7.1.5.5 Functions used by ACM

All the functions used by the ACM are summarized in Table 49.

Table 49 – ACM function table

Function name	Input	Output	Operation
BUILD_PDU	FC	RMSDU	According to requested FC(Frame Control), assemble the Type-11 fieldbus frame
EXPIRED_TIMER	T(x)	True/False	When requested timer x has expired, "True" is returned, otherwise False is returned
MAINT_LL	V(LN)	(none)	The node specified V(LN) is read when T(SCMP) expired. When T(SCMP) expires all at number-of-times continuation of C(ASCMP) in the past, the corresponding bit of V(LL) is cleared
START_TIMER	T(x), V(y)	(none)	Timer x is set by value of y, and is activated
START_THT	Speed-class	(none)	T(ATHT) is activated according to the V(TTRT) remaining value of Access-class specified by Speed-class
START_TTRT	Speed-class	(none)	T(ATTRT) is activated according to the V(TTRT) value of Access-class specified by Speed-class
STOP_TIMER	T(x)	(none)	Timer x is deactivated

7.1.6 Redundancy medium control (RMC)

7.1.6.1 Overview

The redundancy medium control (RMC) manages the selection of two inputs from two RX framers, and moreover of two outputs to two Octet serializers for reliable support, without interruption of higher-level data transfer services. The function of RMC is as follows:

- When the DLPDUs are received from the ACM, the RMC breaks them down into octet symbol requests to both Octet serializers so that the DLPDUs are send out over both of the transmission media A and B simultaneously;
- If a collision has happened during the sending-out of the DLPDUs over one of two transmission media, the data transmission over which the collision happened is stopped immediately and the JAM signal specified by ISO/IEC 8802-3 is sent for the next DLPDU ready to be sent. Nevertheless, if on another transmission medium, no collision happens, the data transmission over the transmission medium is continued without interruption;
- The RMC selects one of two outputs from the Octet deserializers. The selection is on first-come basis to a frame without error, and the RMC submits octets of frame to the ACM;
- The RMC provides a function purposely to select and designate one of two inputs from two RX framers. This function is managed according to the value of the RMSEL parameter in the CW field of the received SYN DLPDU. The relationship between the redundant medium selection and the value of the RMSEL parameter is specified in 6.2.3.2.1.3. In the case of a single medium system the control command by RMSEL is ignored.

7.1.6.2 Primitive definitions

7.1.6.2.1 Primitives exchanged between ACM and RMC

Table 50 shows all of the primitives exchanged between the ACM and the RMC. Table 51 shows the primitives between the RMC and the Serializer /Deserializer; furthermore, the primitives between the RMC and the Ph-layer are shown in Table 52.

Table 50 – Primitives exchanged between ACM and RMC

Primitive name	Source	Associated parameters
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU

Table 51 – Primitives exchanged between RMC and serializer / deserializer

Primitive name	Source	Associated parameters
TX_DATA_A.req	RMC	RMPDU
TX_DATA_B.req	RMC	RMPDU
TX_DATA_A.conf	Serializer_A	Status
TX_DATA_B.conf	Serializer_B	Status
RX_DATA_A.ind	Deserializer_A	RMPDU, Status
RX_DATA_B.ind	Deserializer_B	RMPDU, Status

Table 52 – Primitives exchanged between RMC and Ph-layer

Primitive name	Source	Associated parameters
PLS_CARRIER_A.indication	Ph-layer_A	CARRIER_STATUS(ON/ OFF)
PLS_SIGNAL_A.indication	Ph-layer_A	SIGNAL_STATUS(ERROR/NO_ERROR)
PLS_DATA_VALID_A.indication	Ph-layer_A	DATA_VALID_STATUS(DATA_VALID/DATA_NOT_VALID)
PLS_CARRIER_B.indication	Ph-layer_B	CARRIER_STATUS(CARRIER_ON/CARRIER_OFF)
PLS_SIGNAL_B.indication	Ph-layer_B	SIGNAL_STATUS(SIGNAL_ERROR/NO_SIGNAL_ERROR)
PLS_DATA_VALID_B.indication	Ph-layer_B	DATA_VALID_STATUS(DATA_VALID/DATA_NOT_VALID)

7.1.6.2.2 Parameters used between RMC and ACM, Ph-layer

Table 53 and Table 54 show the parameters used for interaction between the RMC and the ACM or the Ph-layer respectively.

Table 53 – Parameters between RMC and ACM

Parameter name	Description
RMSDU	Service data unit used between RMC and ACM
RMC-status	Status for result of request, condition of receiving etc
TX_ENABLE	Enable/Disable control for data transmission

Table 54 – Parameters between RMC and Ph-layer

Parameter name	Description
CARRIER_STATUS	Status for condition of carrier signal on or not over medium
SIGNAL_STATUS	Status for collision happened
DATA_VALID_STATUS	Status for received data available

7.1.6.3 RMC state table

7.1.6.3.1 State table for sending and send arbitration

Figure 24 shows the state transition diagram of the RMC sending and send arbitration. The state table of the RMC sending is shown in Table 55, and furthermore the state table of the RMC send arbitration is shown in Table 56.

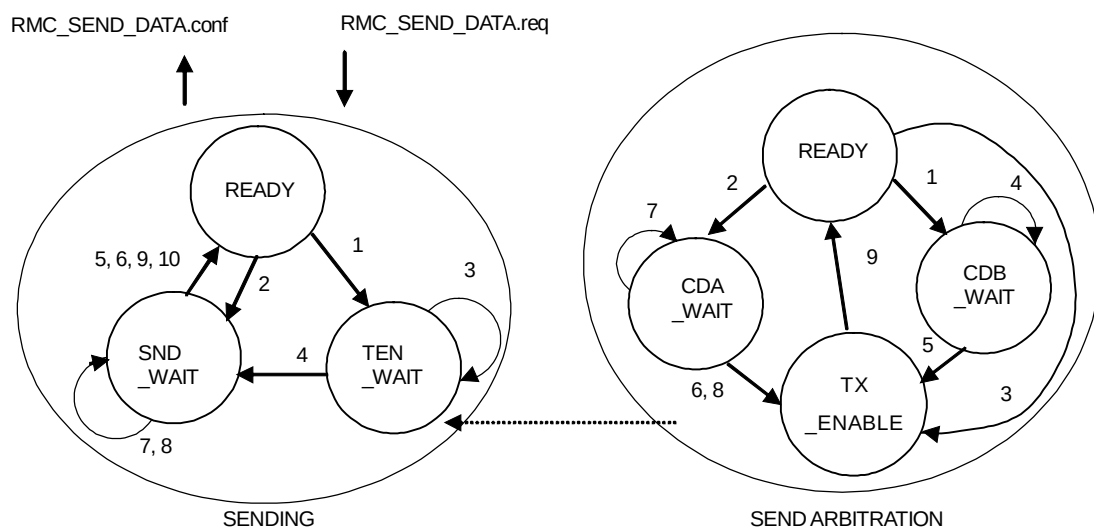
**Figure 24 – State transition diagram of RMC sending and send arbitration**

Table 55 – State table of RMC sending

#	Current state	Event /condition ⇒actions	Next state
1	READY	RMC_SEND_DATA.req { RMSDU } / RMC_STATE <> "TX_ENABLE" ⇒ (no action)	TEN_WAIT
2	READY	RMC_SEND_DATA.req { RMSDU } / RMC_STATE = "TX_ENABLE" ⇒ ----- Under transmission enable SNDC = TX_DATA_AB (RMSDU)	SND_WAIT
3	TEN_WAIT	/ RMC-STATE <> "TX_ENABLE" ⇒ (no action)	TEN_WAIT
4	TEN_WAIT	/ RMC_STATE = "TX_ENABLE" ⇒ ----- Channel A into transmission enable SNDC = TX_DATA_AB (RMSDU)	SND_WAIT
5	SND_WAIT	TX_DATA_A.conf { } / SNDC = 1 ⇒ -----Transmission completed through Channel A RMC_SEND_DATA.conf { RMC-status := "success"}	READY
6	SND_WAIT	TX_DATA_B.conf { } / SNDC = 1 ⇒ -----Transmission completed through Channel B RMC_SEND_DATA.conf { RMC-status := "success"}	READY
7	SND_WAIT	TX_DATA_A.conf { } / SNDC = 2 ⇒ ---- Transmission completed on Channel A but B incomplete SNDC := SNDC - 1	SND_WAIT
8	SND_WAIT	TX_DATA_B.conf { } / SNDC = 2 ⇒ ---- Transmission completed on Channel B but A incomplete SNDC := SNDC - 1	SND_WAIT
9	SND_WAIT	TX_DATA_A.conf { } / SNDC = 0 ⇒ -----Transmission completed through Channel A RMC_SEND_DATA.conf { RMC-status := "fail"}	READY
10	SND_WAIT	TX_DATA_B.conf { } / SNDC = 0 ⇒ -----Transmission completed through Channel B RMC_SEND_DATA.conf { RMC-status := "fail"}	READY

Table 56 – State table of RMC send arbitration

#	Current state	Event /condition ⇒actions	Next state
1	READY	PLS_CARRIER_A.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over medium A START_TIMER (T(AIGP _A)) START_TIMER (T(ARMGP))	CDB_WAIT
2	READY	PLS_CARRIER_B.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over medium B START_TIMER (T(AIGP _B)) START_TIMER (T(ARMGP))	CDA_WAIT
3	CDB_WAIT	EXPIRED_TIMER (T(AIGPT)) = "True" / ⇒ ----- No transmission from Channel A	TX_ENABLE
4	CDB_WAIT	PLS_CARRIER_B.indication { CARRIER_STATUS (OFF) } / ⇒ ----- No transmission from both Channel A and B START_TIMER (T(AIGP _B)) STOP_TIMER (T(ARMGP))	CDB_WAIT
5	CDB_WAIT	EXPIRED_TIMER (T(AIGP _B)) = "True" / ⇒ ----- Inter-packet-gap time elapsed over channel B	TX_ENABLE
6	CDA_WAIT	EXPIRED_TIMER (T(ARMGP)) = "True" / ⇒ ----- No transmission from Channel A	TX_ENABLE
7	CDA_WAIT	PLS_CARRIER_A.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over both medium A and B START_TIMER (T(AIGP _A)) STOP_TIMER (T(ARMGP))	CDB_WAIT
8	CDB_WAIT	EXPIRED_TIMER (T(AIGP _A)) = "True" / ⇒ ----- Inter-packet-gap time elapsed over channel A	TX_ENABLE
9	TX_ENABLE	/ ⇒ ----- Signal TX_ENABLE to Sending state machine	READY

In the state transitions the RMC performs as follows:

- When DLPDUs are received from the ACM, the RMC breaks them down into octet symbol requests to both Octet Serializers so that the DLPDUs are send-out over both of the transmission media A and B simultaneously;
- If a collision has happened while sending-the DLPDUs over one of two transmission media, the data transmission over which the collision happened is stopped immediately and the JAM signal specified by ISO/IEC 8802-3:2000, Clause 4 is sent for the next DLPDU ready to be sent. Nevertheless if on another transmission medium, no collision happens, the data transmission over the transmission medium is continued without interruption;
- If a collision has happened over both transmission media, the JAM signal is sent over both transmission media. In that case, there happens no re-transmission;
- Both cases of a collision happening over one of and both of two transmission media are carried out on the basis of DLPDU by the DLE;
- TX_ENABLE signal is actually activated by the later of the expired timing of T(GPA) and T(GPB);

- f) After one of T(AIGP*) expires and a further wait for T(ARMGP) expires, if not received when the other T(AIGP*) is expired, the RMC decides the corresponding channel enters into abnormal condition. The reestablishment from abnormal state is by the expiration signal of the T(AIGP*) received.

7.1.6.3.2 State table for RMC receiving

Figure 25 depicts the state transition diagram of the RMC receiving, and the state table of the RMC receiving is shown in Table 57.

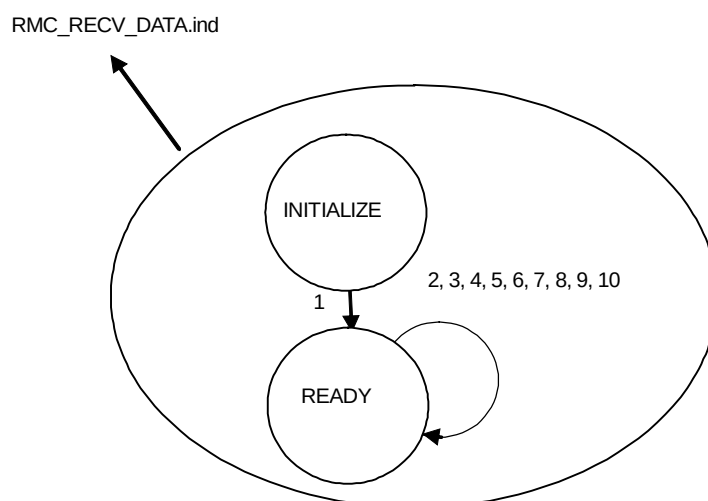


Figure 25 – State transition diagram of RMC receiving

Table 57 – State table for RMC receiving

#	Current state	Event /condition ⇒actions	Next state
1	INITIALIZE	POWER-ON or RESET / ⇒ RCVM := DECIDE_RCVM () ----- Determine Receive channel	READY
2	READY	FORCE_RCVM(RCS) / ⇒ RCVM := DECIDE_RCVM ()	READY
3	READY	RX_DATA_A.ind { RMPDU } / RCVM = MEDIA-A && Status := NORMAL && RMPDU = "SYN" frame ⇒ LAST_RMA := NORMAL RMSDU := RMPDU RMC_RECV_DATA.ind { RMSDU } RCVM := DECIDE_RCVM ()	READY
4	READY	RX_DATA_A.ind { RMPDU } / RCVM = MEDIA-A && Status := "NORMAL" ⇒ LAST_RMA := "NORMAL" RMSDU := RMPDU RMC_RECV_DATA.ind { RMSDU }	READY
5	READY	RX_DATA_A.ind { RMPDU } / RCVM <> MEDIA-A && Status := "NORMAL" ⇒ LAST_RMA := "NORMAL"	READY

#	Current state	Event /condition ⇒actions	Next state
6	READY	RX_DATA_A.ind { RMPDU } / Status := "ERROR" ⇒ LAST_RMA := ERROR RCVM := DECIDE_RCVM ()	READY
7	READY	RX_DATA_B.ind { RMPDU } / RCVM = MEDIA-B && Status := "NORMAL" && RMPDU = "SYN" frame ⇒ LAST_RMB := "NORMAL" RMSDU := RMPDU RMC_RECV_DATA.ind { RMSDU } RCVM := DECIDE_RCVM ()	READY
8	READY	RX_DATA_B.ind { RMPDU } / RCVM = MEDIA-B && Status := "NORMAL" ⇒ LAST_RMB := "NORMAL" RMSDU := RMPDU RMC_RECV_DATA.ind { RMSDU }	READY
9	READY	RX_DATA_B.ind { RMPDU } / RCVM <> MEDIA-B && Status := "NORMAL" ⇒ LAST_RMB := "NORMAL"	READY
10	READY	RX_DATA_A.ind { RMPDU } / Status := "ERROR" ⇒ LAST_RMB := "ERROR" RCVM := DECIDE_RCVM ()	READY

In the state transitions the RMC performs as follows:

- a) At initial stage the Receive-Media A is selected;
- b) Switching control by "FORCE_RCV()" is executed immediately after receiving the command;
- c) Switching policy is as follows:
 - Receive-Media A or B is selected according to "Manual-select";
 - Receive-Media A or B is selected according to the mode specified by the RMSEL in the SYN DLPDU received;
 - Once selected and without any error happening, the selected Receive-Media is maintained;
 - If any error happens, the other Receive-Media is selected.

7.1.6.4 Functions used by RMC

The functions used by the RMC are summarized in Table 58.

Table 58 – RMC function table

Function name	Description and operation
EXPIRED_TIMER (T(X))	When the requested timer x has expired, "True" is returned; otherwise, "False" is returned
FORCE_RCV (RCS)	<pre> if RCS:= Force_A then RCV := MEDIA-A endif if RCS := Force_B then RCV = MEDIA-B if RCS := Automatic then RCV := DECIDE_RCV() </pre>
START_TIMER (v(t))	Activate the timer specified . STOP_TIMER() forces to stop the timer specified and running. "Expired Timer" event happens at the time that a designated timer expired
STOPT_TIMER(v(t))	Forcibly stop the timer specified
DECIDE_RCV ()	Receive-media is selected by "FORCE_RECV", "Manual-select", "RMSEL in SYN DLPDU" and is switched on a condition basis of previous DLPDU received under "Automatic" mode Switching control is executed in the order of descending priority of "FORCE_RCV", "Manual-select", "RMSEL in SYN DLPDU" and a condition basis under "Automatic" mode. <pre> If FORCE_RCV(RCS := MEDIA-A or := MEDIA-B) then return MEDIA-A or MEDIA-B f else if "Manual-select" := MEDIA-A then return MEDIA-A endif else if "Manual-select" := MEDIA-B then return MEDIA-B endif else if RMSEL of SYNDLPDU := MEDIA-A then return MEDIA-A endif else if RMSEL of SYN DLPDU := MEDIA-B then return MEDIA-B endif else if RCV := MEDIA-A && LAST_RMA <> NORMAL then return MEDIA-B endif else if RCV := MEDIA-B && LAST_RMB <> NORMAL then return MEDIA-A endif endif </pre>
TX_DATA_AB (PDU)	Send the PDU out over both Media-A and B at the same time, however it is prohibited to send out over the Media banned. The requested counts of TX_DATA_A or B. request is returned. <pre> sndc:= 0; if Media-A is permitted to send out then TX_DATA_A.req { PDU } sndc := sndc + 1 endif if Media-B is permitted to send out then TX_DATA_B.req { PDU } sndc := sndc + 1 endif return sndc </pre>

7.2 DLE elements of procedure for loop-architecture

7.2.1 Overall structure

Figure 26 depicts the overall structure of the DLL. The DLL is composed of control elements of Cyclic-transmission-TX/RX-control (CTRC), Sporadic-TX/RX-control (STRC), Access-control-machine (ACM), Redundancy-medium-control (RMC), TX/RX-framer, Octet-serializer /-deserializer and DLL-management-interface.

The ACM as the primary control element provides the function for deterministic medium access control cooperating with the CTRC, the STRC and the RMC for reliable and efficient support both of higher-level connection-mode and connection-less mode data transfer services.

The DLL management interface provides DLL management functions.

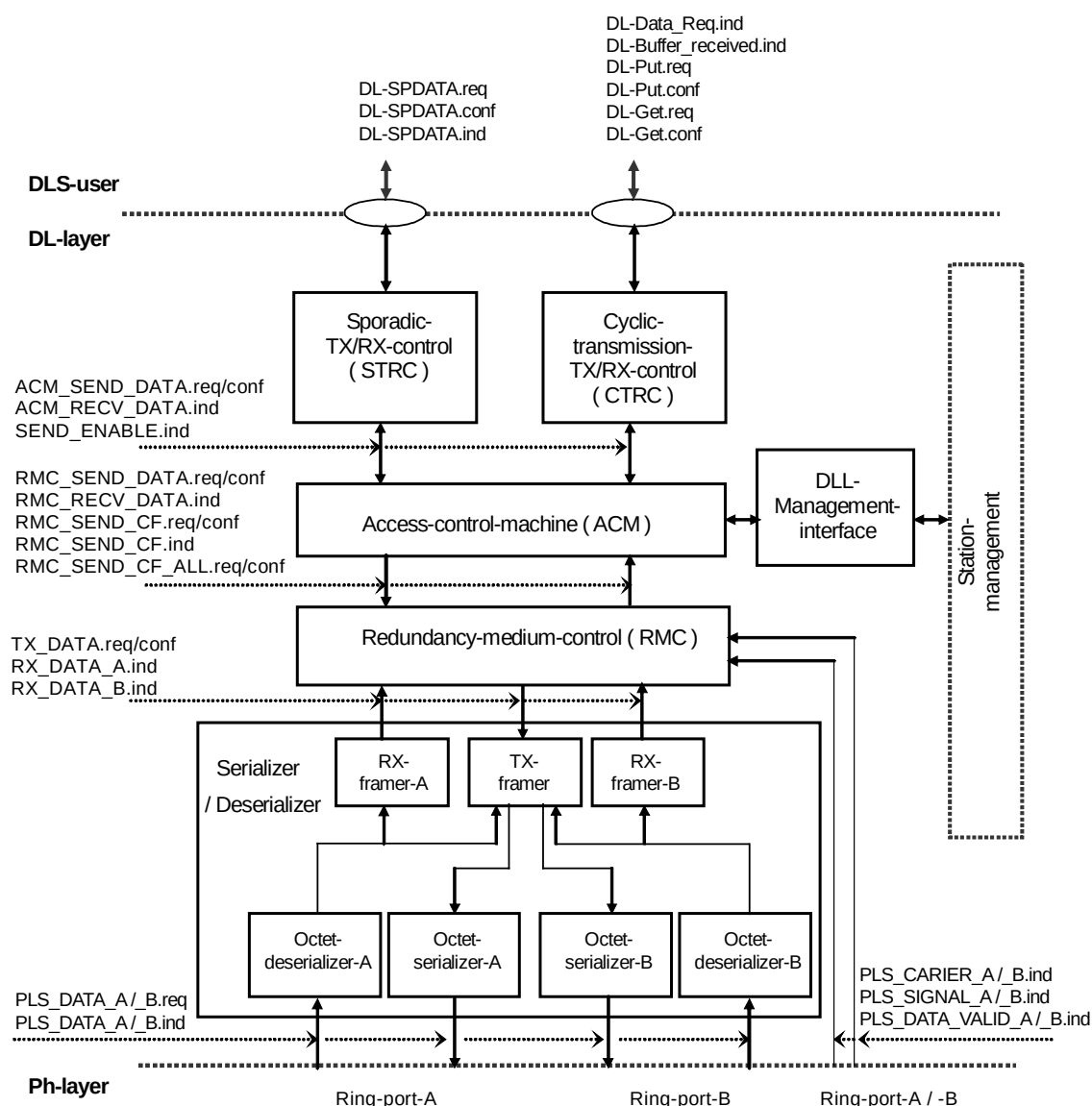


Figure 26 – Overall structure of DLL

7.2.2 Initialization

Upon power-up or after the reception of the DLM-RESET request primitive, the DLE shall go into the OFF-LINE state. When in OFF-LINE, the DLE shall not transmit and shall ignore any DLPDU received.

When all the variables for normal DLE operation are set up by the DLM-Set-Value request primitive, the state is changed from OFF-LINE to STATION MODE CONTROL in which the DLE shall start normal DL operation.

Figure 27 depicts the DLE state transition.

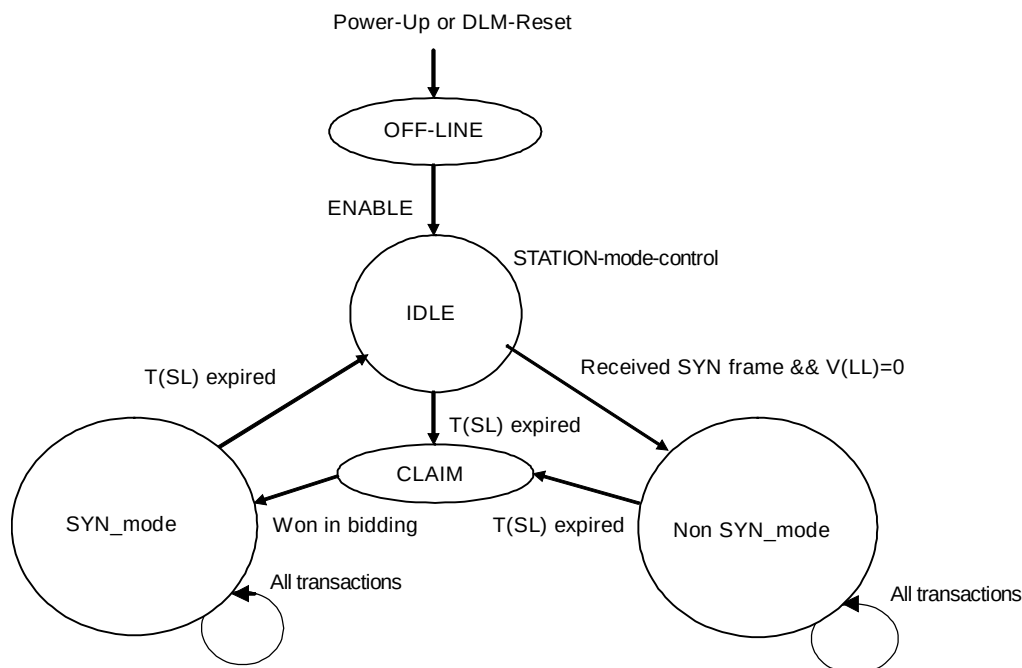


Figure 27 – DLE state transition

7.2.3 Cyclic transmission TX/RX control (CTRC)

7.2.3.1 Overview

The Cyclic-transmission-TX/RX-control(CTRC) is responsible for buffering and dispatching the DLSDU received for the Time-critical-cyclic-data-transfer between the DLS-user and the ACM.

DL-Data-Req.indication primitive by the CTRC informs the DLS-user to initiate the data transfer from each DLS-user buffer to the send buffer of the DLE using DL-Put request primitive. The DLS-user buffer is addressed with the DLCEP-identifier parameter in the DL-Data-Req indication primitive from the CTRC.

Each DLS-user buffer contains data for Time-critical-cyclic-data-transmission. After the DLS-user data received, the CTRC issues ACM_SEND_DATA request primitive to initiate ACM sending out the DLS-user data as DT or DT-CMP DLPDU.

DL-Data-Req indication primitive is issued by the CTRC in response to SEND_ENABLE indication primitive from the ACM. The Speed-class parameter in SEND_ENABLE indication primitive allows the CTRC to activate transfer of corresponding class of data-buffer by the DLS-user. The classes designated by the Speed-class parameter correspond to the classes of High-speed-cyclic-data, Medium-speed-cyclic-data and low-speed-cyclic-data transfer .

The data transfer of each Speed-class designated is dependent on the level of transfer priority:

- a) As for priority-class 3, which corresponds to the High-speed-cyclic-data transfer, when received SEND_ENABLE indication primitive, the CTRC shall handle all of the data of every DLS-user buffer corresponding to the High-speed-cyclic-data transfer or the High-speed-class is to be delivered to the ACM and to be sent on the transmission medium all at once on each occasion;

- b) As for the priority-class 2, which corresponds to the Medium-speed-cyclic-data transfer or the Medium-speed-class respectively, the CTRC shall handle on each occasion the data of every DLS-user buffer corresponding to the Medium-speed-class is sent or not depending on the condition. The token-holding-time of priority-class 2 governs the condition. After sending out all the High-speed-cyclic-data, the Medium-speed-cyclic-data is sent. If the token-holding-time ends during sending the Medium-speed-cyclic-data, the data transfer is interrupted and the CMP DLPDU or the DT-CMP DLPDU is sent at the occasion. On the next occasion, the remainder of the previous Medium-speed-cyclic-data is to be sent depending on the same condition. As for the Lower priority-class n, the same control as for the Medium-speed-cyclic-data transfer is to be done. The occasion happens every V(TSYN) time period.

In case of 100 Mbps operation, 3 classes of the Speed-class shall be supported as follows.

- The Speed-class 3 is for the High-speed-cyclic-data transfer, the Speed-class 2 is for the Medium-speed-cyclic-data transfer and the Speed-class 1 is for the Sporadic-message-data transfer.

On the other hand in case of 1 000 Mbps operation, 4 classes of the Speed-class shall be supported as follows.

- The Speed-class 3 is for the High-speed-cyclic-data transfer, the Speed-class 2 is for the Medium-speed-cyclic-data transfer, the Speed-class 1 is for the Low-speed-cyclic-data transfer and the Speed-class 0 is for the Sporadic-message-data transfer.

When the ACM_RECV_DATA indication primitive has been received from the ACM, the CTRC stores the DLPDU by the ACM into each receive-buffer, which is addressed with the DLCEP-identifier parameter in ACM_RECV_DATA indication primitive. Furthermore, the CTRC issues DL-Buffer-received indication primitive to inform the DLS-user the receive-buffer of the DLE is updated and is available to read out. The DLS-user can read out the data in the receive-buffer using DL-Get request primitive in response to DL-Buffer-received indication primitive from the CTRC.

7.2.3.2 Primitive definitions

7.2.3.2.1 Primitives exchanged between DLS-user and CTRC

The primitives exchanged between the DLS-user and the CTRC, and the primitives exchanged between the CTRC and the ACM are same as of star-architecture, and are summarized in Table 59 and Table 60 respectively.

Table 59 – Primitives exchanged between DLS-user and CTRC

Primitive name	Source	Associated parameters
DL-Data-Req.ind	CTRC	DLCEP-identifier
DL-Put.req	DL-user	DLCEP-identifier, DLSDU-length, DLSDU
DL-Put.conf	CTRC	DLCEP-identifier, Status
DL-Buffer-received.ind	CTRC	DLCEP-identifier
DL-Get.req	DL-user	DLCEP-identifier
DL-Get.conf	CTRC	DLCEP-identifier, DLSDU-length, DLSDU, Status

Table 60 – Primitives exchanged between CTRC and ACM

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_RECV_DATA.ind	ACM	DLPDU

7.2.3.2.2 Parameters used with primitives exchanged between DLS-user and CTRC

The parameters used for interaction between the DLS-user and the CTRC are summarized in Table 61.

Table 61 – Parameters used with primitives exchanged between DLS-user and CTRC

Parameter name	Description
DLCEP-identifier	Identifier to designate the Send-Buffer or the Receive-Buffer
DLSDU	The contents of the Send-Buffer or the Receive-Buffer, which is the Time-critical- cyclic-data processed by the CTRC
DLSDU-length	The length of DLSDU
Speed-class	Speed-class to designate the class of the Time-critical-cyclic-data transfer, that is for the High-speed-cyclic-data, the Medium-speed-cyclic-data and the Low-speed-cyclic-data. The value of the Speed-class for the High-speed-cyclic-data is 3, and is 2 for the Medium-speed-cyclic-data. For the Low-speed-cyclic-data in 1 000 Mbps operation the value of the Speed-class is 1
Status	Status report whether the requested service is successfully provided or failed for the reason specified

7.2.3.3 CTRC state table

The state transition diagram of the CTRC is depicted in Figure 28, and the CTRC state table is shown in Table 62.

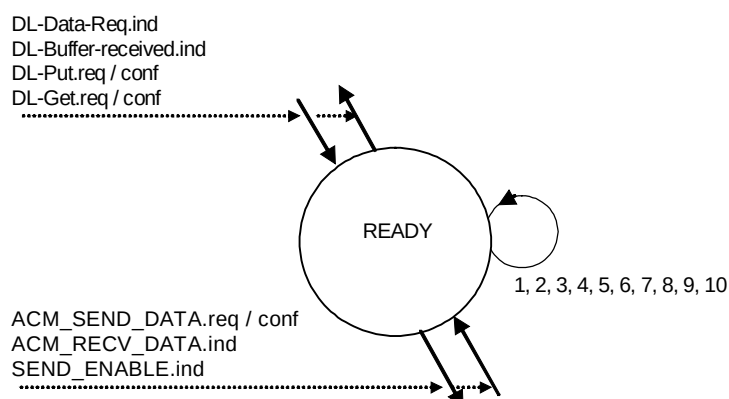


Figure 28 – State transition diagram of CTRC

Table 62 – CTRC state table

#	Current state	Event /condition =>actions	Next state
1	READY	DL-Put.req { DLCEP-identifier, DLSDU-length, DLSDU } / CHECK_PAR_PUT (DLCEP-identifier, DLSDU) = "True" => Status := "success" PUT_BUFFER (DLCEP-identifier, DLSDU) DL-Put.conf { DLCEP-identifier, DLSDU-length, Status }	READY
2	READY	DL-Put.req { DLCEP-identifier, DLSDU-length, DLSDU } / CHECK_PAR_PUT (DLCEP-identifier, DLSDU) = "False" => Status := "failure" DL-Put.conf { DLCEP-identifier, DLSDU-length, Status }	READY
3	READY	DL-Get.req { DLCEP-identifier } / CHECK_PAR_GET (DLCEP-identifier) = "True" => Status := "success" DLSDU := GET_BUFFER (DLCEP-identifier) DL-Get.conf { DLCEP-identifier, DLSDU-length, DLSDU, Status }	READY
4	READY	DL-Get.req { DLCEP-identifier } / CHECK_PAR_GET (DLCEP-identifier) = "False" => Status := "failure" DLSDU := "null" DL-Get.conf { DLCEP-identifier, DLSDU-length, DLSDU, Status }	READY
5	READY	SEND_ENABLE.ind { Speed-class } / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND (Speed-class) = "True" => DLCEP-identifier := GET_NEXT_ID (Speed-class) DL-Data-req ind { DLCEP-identifier } DLSDU := GET_BUFFER (DLCEP-identifier) DLPDU := BUILD_DT (Speed-class, DLCEP-identifier, DLSDU) ACM_SEND_DATA.req { DLPDU }	READY
6	READY	SEND_ENABLE.ind { Speed-class } / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "False" => ACM_SEND_DATA.req { null } ----- no data remained to send out	READY
7	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class <> "SPORADIC" && CHECK_PAR_DT (DLPDU) = "True" => DLCEP-identifier := DLPDU.DLCEP-address DL-Buffer-received.ind { DLCEP-identifier }	READY
8	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class <> "SPORADIC" && CHECK_PAR_DT (DLPDU) = "False" => none	READY
9	READY	ACM_SEND_DATA.conf { } / DLPDU <> "null" => none	READY
10	READY	ACM_SEND_DATA.conf { } / DLPDU = "null" => NEXT (Speed-class)	READY

7.2.3.4 Functions used by CTRC

All functions of the CTRC are summarized in Table 63.

Table 63 – CTRC functions table

Function Name	Input	Output	Operation
CHECK_PAR_PUT	DLCEP-identifier, DLSDU	True/False	Check that all the parameters of DL-Put.req primitive are valid. If valid, "True" is returned, otherwise "False" is returned
PUT_BUFFER	DLCEP-identifier, DLSDU	(none)	Store DLSDU into the Send-buffer associated with the DLCEP-identifier
CHECK_PAR_GET	DLCEP-identifier	True/False	Check that DLCEP-identifier of DL-Get.req primitive is valid. If valid, "True" is returned
GET_BUFFER	DLCEP-identifier	DLSDU	Get DLSDU in the buffer associated with DLCEP-identifier
CHECK_NEXT_SEND	Speed-class	True/False	Check that the DLSDU specified by Speed-class exists. If exists, "True" is returned, otherwise "False" is returned
GET_NEXT_ID	Speed-class	DLCEP-identifier	Get next DLCEP-identifier of the DLS User-buffer specified by the Speed-class
BUILD_DT	Speed-class, DLCEP-identifier, DLSDU	DLPDU	Build into a DLPDU out of the data in the Send-buffer. DLPDU is assembled as follows: DLPDU.DA := V(MGA) DLPDU.SA := V(IA) DLPDU.Len/Type := 0x888b DLPDU.FC := Speed-class + "DT" DLPDU.SN := V(TN) DLPDU.DLCEP.address := DLCEP-identifier DLPDU.DLSDU := DLSDU
CHECK_PAR_DT	DLPDU	True/False	Check that the DLPDU is valid. If valid, "True" is returned, otherwise "False" is returned
NEXT	Speed-class		Check that DLSDUs specified by the Speed-class exist in the Send-buffer. If exist, get the DLCEP-identifier of a DLSDU on the Send-buffer, issue DL_Data_Req.ind primitive with the obtained DLCEP-identifier to the DLS-user, and after that, repeat same sequence with new DLCEP-identifier of a DLSDU next to the DLSDU designated in the previous sequence. The process shall go in cycles to all the DLSDUs specified by the Speed-class on the Send-buffer

7.2.4 Sporadic TX/RX control (STRC)

7.2.4.1 Overview

The Sporadic-TX/RX-control (STRC) is responsible for buffering and dispatching in time DLSDU received for the Sporadic-message-data between the DLS-user and the ACM.

7.2.4.2 Primitive definitions

7.2.4.2.1 Primitives exchanged between DLS-user and STRC

Table 64 shows all primitives exchanged between the DLS-user and the STRC, and Table 65 shows all primitives exchanged between the STRC and the ACM. These primitives are same as of star-architecture.

Table 64 – Primitives exchanged between DLS-user and STRC

Primitive name	Source	Associated parameters
DL-SPDATA.req	DLS-user	DA, MSDU
DL-SPDATA.conf	STRC	DA, Status
DL-SPDATA.ind	STRC	DA, SA, MSDU, Rec-status

Table 65 – Primitives exchanged between STRC and ACM

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_RECV_DATA.ind	ACM	DLPDU

7.2.4.2.2 Parameters used with primitives exchanged between DLS-user and STRC

All parameters used with primitives exchanged between the DLS-user and the STRC is shown in Table 66. The parameters are same as of star-architecture.

Table 66 – Parameters used with primitives exchanged between DLS-user and STRC

Parameter name	Description
DA	Destination-address
SA	Source-address
MSDU	MAC-service-data-unit
Status	Indicate whether the requested service of DL-SPDATA.req is successfully provided or failed for the reason specified
Rec-status	Indicates whether the DLPDU had received without error or not
DLPDU	DLPDU for Sporadic-message-transfer
Speed-class	Speed-class requested by DLS-user. In Sporadic-message-transfer service, Speed-class is specified as "Sporadic" or "Class 1" for 100 Mbps operation and is specified as "Sporadic" or "Class 0" for 1 000 Mbps operation

7.2.4.3 STRC state table

The state transition diagram of the STRC is depicted in Figure 29, and the STC state table is shown in Table 67.

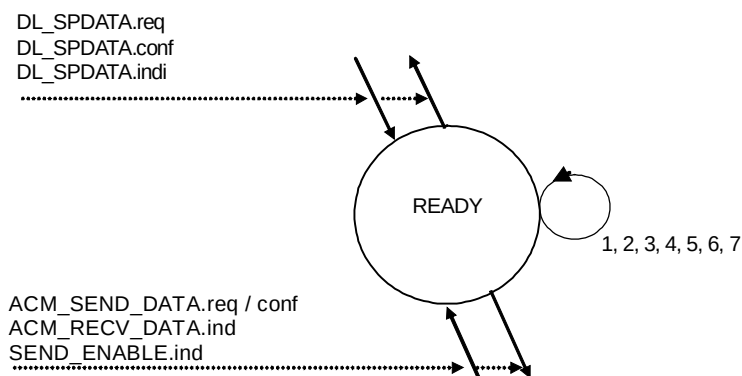


Figure 29 – State transition diagram of STRC

Table 67 – STRC state table

#	Current state	Event /condition =>actions	Next state
1	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () <> "Full" => QUEUE_SPDATA (DA, MSDU) DL-SPDATA.confirm { DA, MSDU, Status := "success" }	READY
2	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () = "Full" => Status := "Failure – The SPDATA-queue is full" DL-SPDATA.conf { DA, MSDU, Status }	READY
3	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "False" => Status := "Failure – Invalid parameter" DL-SPDATA.confirm { DA, MSDU, Status }	READY
4	READY	SEND_ENABLE.ind {Speed-class} / Speed-class = "SPORADIC" && CHECK_SPDATAQ () <> "Empty" => MSDU := DEQUEUE_SPDATA () DLPDU := BUILD_SPDATA (DA, MSDU) ACM_SEND_DATA.req { DLPDU }	READY
5	READY	ACM_SEND_DATA.conf { } / => none	READY
6	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class = "SPORADIC" && CHECK_RECV_SPDATA (DLPDU) = "True" => DL-SPDATA.ind { DLPDU }	READY
7	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class = "SPORADIC" && CHECK_RECV_SPDATA (DLPDU) = "False" => none	READY

7.2.4.4 Functions used by STRC

All functions of the STRC are summarized in Table 68.

Table 68 – STRC functions table

Function name	Input	Output	Operation
CHECK_PAR_SPDATA	DA, MSDU	True/False	Check that all the parameters of DL-SPDATA.req are valid. If valid, "True" is returned, otherwise "False" is returned
CHECK_SPDATAQ	(none)	Status	Check that the queue condition for SPDATA is full. The returned status is any one of "Full", "Empty" and "Queued"
QUEUE_SPDATA	DA, MSDU	(none)	Queues the input data into the SPDATA-queue on First-In-First-Out basis
DEQUEUE_SPDATA	(none)	DA, MSDU	Dequeue from the SPDATA-queue on First-In-First-Out basis
BUILD_SPDATA	DA, MSDU	DLPDU	Build into a DLPDU of Sporadic-message-data. The DLPDU is assembled as follows: DLPDU.DA := MSDU.DA DLPDU.SA := V(IA) DLPDU.Len/Type := MSDU.LEN/Type DLPDU.MSDU := MSDU.DLSDU
CHECK_RCV_SPDATA	DLPDU	True/False	Check that the specified DLPDU is valid. If valid, "True" is returned, otherwise "False" is returned

7.2.5 Access control machine (ACM)

7.2.5.1 Overview

The access-control-machine (ACM) is responsible for deterministic medium-access-control and scheduling opportunities to send out the DLPDUs, for control to add and remove nodes and for restoration from network disruption to be system down.

The ACM has the primary responsibility for

- assuring that the local node detects and fully utilizes its assigned access time period;
- assuring that the local node does not interfere with the transmissions of other nodes, especially of the node transmitting the SYN DLPDU;
- detecting network disruption, and initiating the SYN DLPDU transmit for restoration of the network disruption from after prescribed time duration in which the SYN DLPDU is not heard;
- assuring a new node adding to and removing from the network.

7.2.5.2 ACM and the schedule support functions

The ACM functions schedule all communications between the DLEs participating in the Type 11 fieldbus, and the timing of this communications is controlled as to:

- fulfill the specific medium-access-control to give all the DLEs the opportunities to send out two kinds of class of the Time-critical-cyclic data and Sporadic-message data in timely, prioritized and deterministic fashion, and to detect the network disruption, and to initiate the restoration in appropriate time, and further to add and remove the nodes on line;
- provide multiple levels of the Time-critical-data-transfer opportunities of sending data to the node in sequential order and within each pre-specified time period, and that the data transfer of each level is performed within the pre-specified time duration (token holding time) and whether the data transfer of lower levels to be carried out or to be held over to the later cyclic time period depends on the level and the occasion though the top level of the data transfer is always carried out at every occasion, on the other hand, the whole volume of the data transfer of lower levels is transferred within each pre-specified time period;

- c) provide the sporadic-message-data-transfer opportunities of sending out to the node that the request to transfer is happened sporadically by the DLS-user, and the data transfer is performed within a pre-specified time period of the corresponding level of priority and is based on the regular ISO/IEC 8802-3 applications.

7.2.5.3 Primitive definitions

7.2.5.3.1 Primitives exchanged between ACM and RMC

Table 69 lists all primitives and Table 70 shows the parameters used with the primitives exchanged between the ACM and the RMC.

Table 69 – Primitives exchanged between ACM and RMC

Primitive name	Source	Associated parameters
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU
RMC_SEND_CF.req	ACM	RMSDU
RMC_SEND_CF.conf	RMC	RMC-status
RMC_SEND_CF.ind	RMC	RMC-status
RMC_SEND_CF_ALL.req	ACM	RMSDU
RMC_SEND_CF_ALL.conf	RMC	RMC-status

Table 70 – Parameters used with primitives exchanged between ACM and RMC

Parameter name	Description
RMSDU	SDU of RMC
RMC-status	Status indicating the results of the request to RMC

7.2.5.3.2 Primitives exchanged between ACM and CTRC

Table 71 lists all primitives and Table 72 shows the parameters used with the primitives exchanged between the ACM and the CTRC.

Table 71 – Primitives exchanged between ACM and CTRC

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	ACM-status
ACM_RECV_DATA.ind	ACM	DLPDU

Table 72 – Parameters used with primitives exchanged between ACM and CTRC

Parameter name	Description
ACM-status	Status indicating the results of the request to ACM
DLPDU	SDU of ACM
Speed-class	Speed-class indicating Time-critical Cyclic-data-transmission

7.2.5.3.3 Primitives exchanged between ACM and STRC

Table 73 lists all primitives and Table 74 shows the parameters used with the primitives exchanged between the ACM and the STRC.

Table 73 – Primitives exchanged between ACM and STRC

Primitive name	Source	Associated parameters
SEND_ENABLE.ind	ACM	Speed-class
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Status
ACM_SEND_DATA.ind	ACM	DLPDU

Table 74 – Parameters used with primitives exchanged between ACM and STRC

Parameter name	Description
ACM-Status	Status indicating the results of the request to ACM
DLPDU	SDU of ACM
Speed-class	Speed class indicating Time-critical Cyclic data transmission

7.2.5.4 ACM state table

The state transition diagrams for 100 Mbps and 1 000 Mbps operation are shown in Figure 30 and Figure 31 respectively.

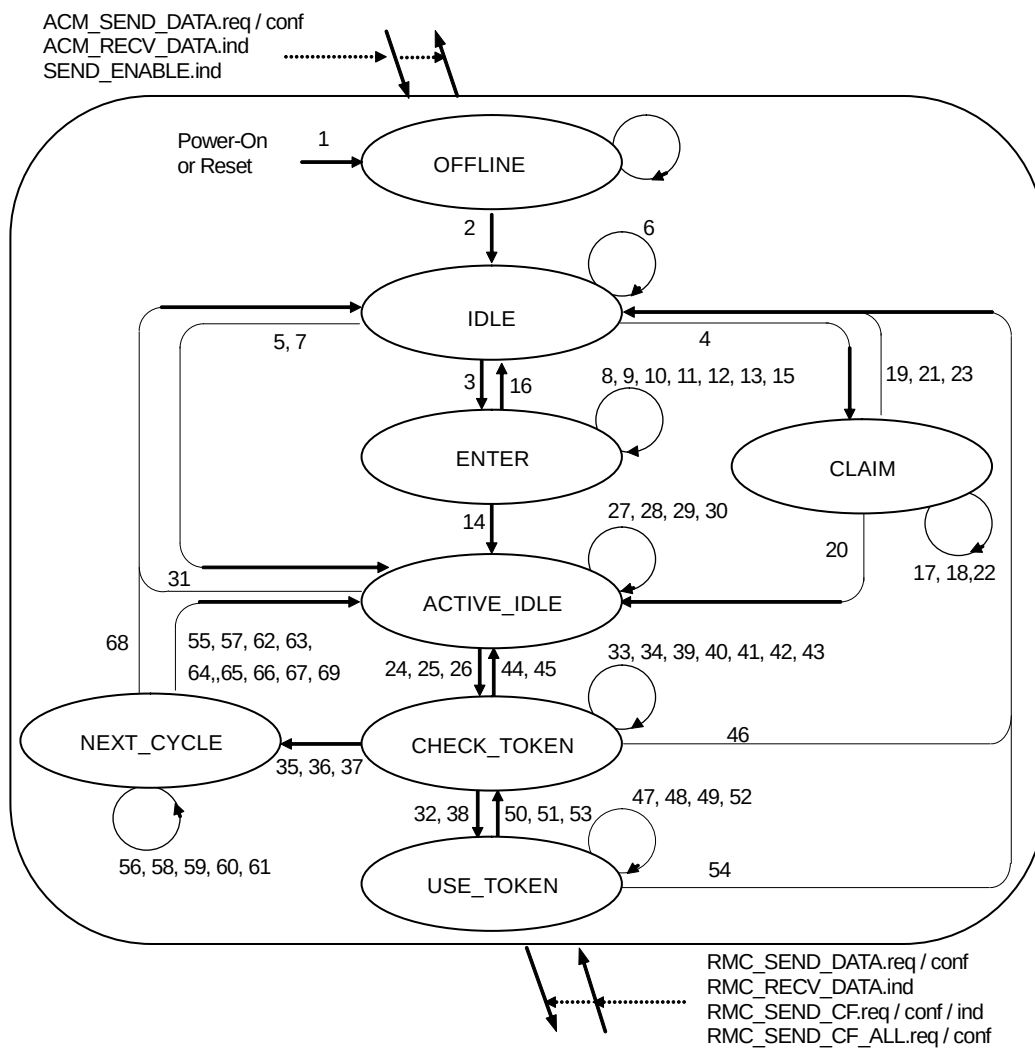


Figure 30 – State transition diagram of ACM for 100 Mbps operation

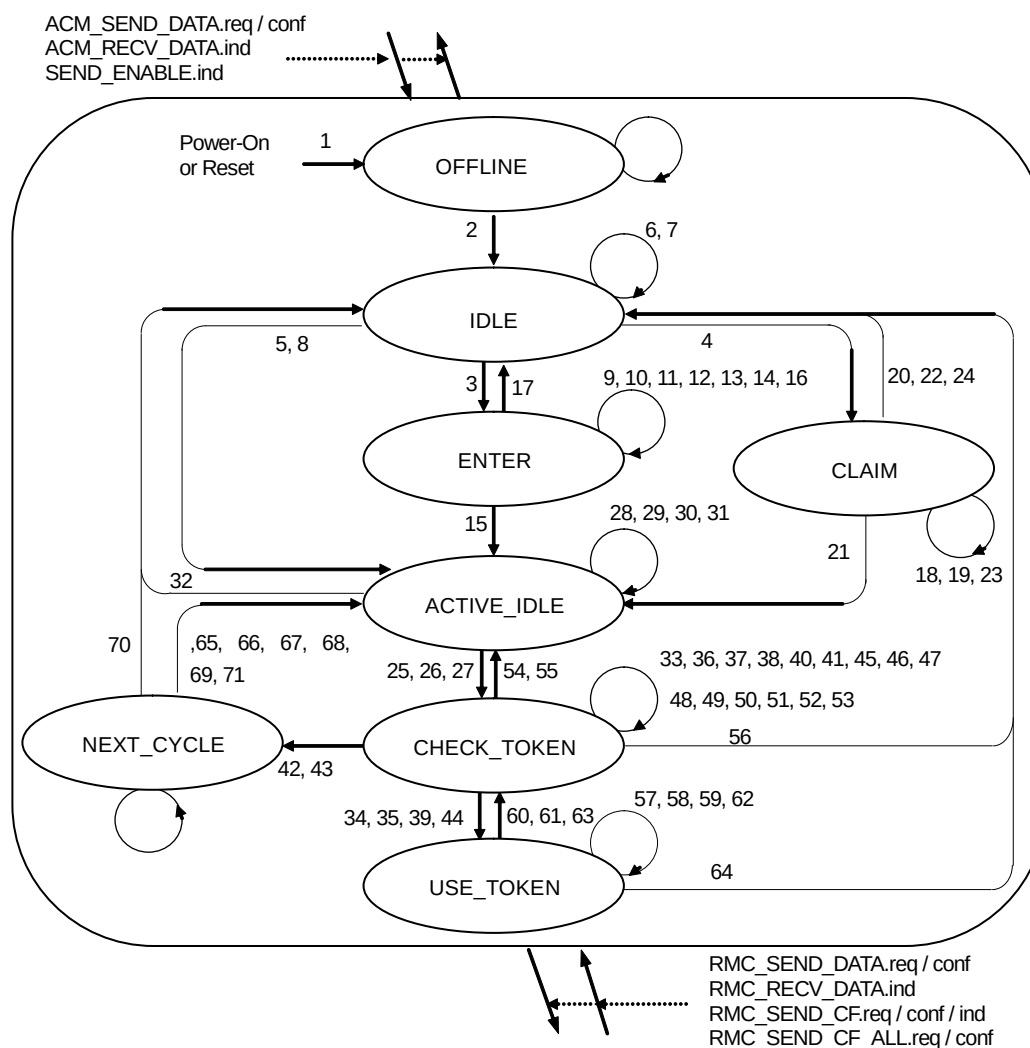


Figure 31 – State transition diagram of ACM for 1 000 Mbps operation

The state "OFFLINE" is entered on power-up or when DLM-Reset.request primitive is issued. In this state, all of the variables, parameters and configuration information are set up by the DLE. When "In_Ring_Desire" becomes true, the state changes to "IDLE".

The state "IDLE" is to wait and be ready to join the type 11 fieldbus membership. When a SYN DLPDU has been received from another SYN-node, the state change to "ENTER" in which the REQ DLPDU is sent to claim adding into the type 11 fieldbus membership to the current SYN-node. Moreover, in "IDLE" state, when the signals over the transmission-media continues to be inactive during a specified time period, the state becomes "CLAIM", in which the node attempts to be a new SYN-node.

The state "ACTIVE_IDLE" is in the type 11 fieldbus membership, and the node manages to obtain the transmission-right by the type 11 fieldbus medium-access-control, either when the node is operating as the SYN-mode by transmitting the SYN DLPDU or as the non-SYN-mode receiving the SYN DLPDU by other SYN-node, the state is in "CHECK_TOKEN".

The state "CHECK_TOKEN" waits until $V(LN)$ becomes equal to $V(TN)$ in order to send out the data over the transmission media. When $V(LN)$ becomes equal to $V(TN)$, the state changes to "USE_TOKEN".

The state "USE_TOKEN" is that the node is able to send out the Time-critical cyclic-transmission data and the Sporadic-transmission data. When all data has been sent or the token-holding-time is expired, the state changes to "NEXT_CYCLE".

The state "NEXT_CYCLE" is for solicit new node as a primary function.

The ACM state tables for 100 Mbps and 1 000 Mbps operation are shown in Table 75 and Table 76.

Table 75 – ACM state table for 100 Mbps operation

#	Current state	Event /condition =>actions	Next State
1	Any states	/ POWER-ON or RESET =>	OFFLINE
2	OFFLINE	/ In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) = 0 => -- receiving SYN and LL is not alive -- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	/ EXPIRED_TIMER (T(SL)) = "True" && V(LL) = 0 => ----sending out CLM All_Slot_Time := V(ST) RMSDU := BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "True" => -- receiving SYN and this node is alive -- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
6	IDLE	RMC_SEND_CF.ind { } / V(LL) <> 0 && V(LL)<V(TN)> = "True" && V(LN) = V(TN) => -----activate sending out LRR RMSDU := BUILD_PDU (LRR) RMS_SEND_CF.req { RMSDU } START_TIMER (T(SL), V(SL))	IDLE
7	IDLE	RMS_SEND_CF.conf { } / RMSDU.FC = LRR => none	ACTIVE_IDLE
8	ENTER	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	ENTER
9	ENTER	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER

#	Current state	Event /condition =>actions	Next State
10	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) = V(MN) && V(LL)<V(TN)> = "False" && SYN_frame.PN = V(TN) => RMSDU := BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
11	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++	ENTER
12	ENTER	/ V(LN) = V(MN) && V(LL)<V(TN)> = "False" && SYN_frame.PN = V(TN) => RMSDU := BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
13	ENTER	RMC_SEND_DATA.conf { } => none	ENTER
14	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V (LL)<V(TN)> = "True" => SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
15	ENTER	/ EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
16	ENTER	/ EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
17	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time > 0 => All_Slot_Time-- START_TIMER (T(SL), V(SL)) START_TIMER (T(IGP), V(IGP))	CLAIM
18	CLAIM	/ EXPIRED_TIMER (T(IGP))="True" => RMSDU:=BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSD } START_TIMER (T(SL), V(SL))	CLAIM
19	CLAIM	RMC_SEND_DATA.ind {RMSDU} / RMSDU.FC = (SYN CLM) => START_TIMER (T(SL), V(SL)) STOP_TIMER (T(IGP))	IDLE
20	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = 0 => All_Slot_Time := V(ST) V(LL) = 0 V(LL)<V(TN)> := 1 RMSDU := BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE

#	Current state	Event /condition =>actions	Next State
21	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => All_Slot_Time := V(ST) START_TIMER (T(SL), V(SL))	IDLE
22	CLAIM	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> (SYN CLM) => -- receiving a frame except SYN, CLM -- START_TIMER (T(SL), V(SL))	CLAIM
23	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- V(LL) := 0 START_TIMER (T(SL), V(SL))	IDLE
24	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "False" => --- receiving SYN in non-SYN_mode---- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1	CHECK_TOKEN
25	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "True" => ---Receiving SYN in SYN_mode----- SYN_Frame := RMSDU if SYN_frame.SN < V(TN) then SYN_mode := 0 V(LL) := SYN_frame.LL V(LN) := 1 endif	CHECK_TOKEN
26	ACTIVE_IDLE	RMC_SEND_DATA.conf { } / RMSDU.FC = SYN && CHECK_COL () = "False" => -- sent SYN ---- SYN_mode := 1 V(LN) := 1	CHECK_TOKEN
27	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
28	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
29	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (DT DT-CMP) => -- receiving DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
30	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
31	ACTIVE_IDLE	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE

#	Current state	Event /condition =>actions	Next State
32	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) Speed-class := 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
33	CHECK_TOKEN	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	CHECK_TOKEN
34	CHECK_TOKEN	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	CHECK_TOKEN
35	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN <> V(TN) && SYN_mode = "True" => ---- in SYN-mode, on MAC-control-time-period -- START_TIMER (T(TMAC), V(TMAC))	NEXT_CYCLE
36	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN = 255 && V(LL)<V(TN)> = "True" => RMSDU := BUILD_PDU (LPD) -----send LPD RMC_SEND_CF_ALL.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL))	NEXT_CYCLE
37	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN = V(TN) && V(LL)<V(TN)> = "True" => RMSDU := BUILD_PDU (RAS) -----send RAS RMC_SEND_DATA.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL))	NEXT_CYCLE
38	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) => -- Get token -- ----- matching V(LN), on receiving CMP or DT-CMP STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN> := 0 Speed-class := 3 -- Highest Priority START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif	USE_TOKEN
39	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ <> V(TN) =>--not matching V(LN).on receiving CMP or DT-CMP START_TIMER (T(SCMP), V(SCMP)) C(ASCMP)<RMSDU.SN>:=0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif	CHECK_TOKEN
40	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT => ---- receiving DT ---- C(ASCMP)<RMSDU.SN> := 0 ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN

#	Current state	Event /condition =>actions	Next State
41	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
42	CHECK_TOKEN	/ EXPIRED_TIMER (T(SCMP)) => C(ASCMP)<V(LN)>++ V(ANONC)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif V(LN)++	CHECK_TOKEN
43	CHECK_TOKEN	RMC_SEND_DATA.conf { } / RMSDU.FC = (CMP DT-CMP) => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
44	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
45	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
46	CHECK_TOKEN	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
47	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send out DT or Sporadic frame -- RMSDU := BUILD_PDU (DLPDU) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	USE_TOKEN
48	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
49	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> 1 => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class := Speed-class – 1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	Current state	Event /condition =>actions	Next State
50	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class = 1 => ---Expired THT and Speed-class =1 ---- ACM_SEND_DATA.conf { RMC-status } RMSDU := BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
51	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / RMSDU.FC = DT-CMP => ACM_SEND_DATA.conf { RMC-status } STOP_TIMER (T(ATHT)) Speed-class := Speed-class – 1 if Speed-class = 2 then START_TTRT (Speed-class) START_TTRT (Speed-class – 1) else if Speed-class = 1 then START_TTRT (Speed-class) endif	CHECK_TOKEN
52	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class <> 1 => -- Next Speed-class -- Speed-class := Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
53	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class = 1 => -- All sent -- RMSDU := BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
54	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
55	NEXT_CYCLE	RMC_SEND_CF_ALL.conf { } / RMSDU.FC = LPD => none	ACTIVE_IDLE
56	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && SYN_frame.PN = V(TN) && SYN_mode = "True" => START_TIMER (T(IGP), V(IGP))	NEXT_CYCLE
57	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && SYN_frame.PN = V(TN) && SYN_mode = "False" => if ACM_SEND_DATA.req { COM } then START_TIMER (T(IGP), V(IGP)) else none endif	ACTIVE_IDLE
58	NEXT_CYCLE	/EXPIRED_TIMER (T(IGP)) = "True" && SYN_mode = "False" => RMSDU := BUILD_PDU (COM) ----send COM RMC_SEND_DATA.req { RMSDU }	NEXT_CYCLE

#	Current state	Event /condition =>actions	Next State
59	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = COM => none	NEXT_CYCLE
60	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "True" => -- receiving REQ at SYN-node-- V(LL)<RMSDU.SN> := 1 C(NONC) := 0 RMSDU := BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
61	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "False" => --receiving REQ at Non-SYN-node-- none	ACTIVE_IDLE
62	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = COM && SYN_mode = "True" => -- receiving COM -- V(CW) := RMSDU.CW V(ST) := RMSDU.ST V(TSYN) := RMSDU.Th V(TTRT2) := RMSDU.Tm DLM_Set_value.req { Variable-name, Desired-value } C(NONC) := 0 RMSDU := BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
63	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
64	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
65	NEXT_CYCLE	/ EXIPERED_TIMER (T(TMAC)) = "True" && SYN.mode = "True" && V(ANONC) < (V(MN) – 1) => C(NONC) := 0 V(ANONC) := 0 RMSDU := BUILD_PDU (SYN) ----send SYN RMS_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
66	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "True" && (V(ANONC) >= (V(MN) – 1)) && (C(NONC)++ >= V(NONC)) => C(NONC) := 0 V(ANONC) := 0 SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
67	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "False" => none	ACTIVE_IDLE

Table 76 – ACM state table for 1 000 Mbps operation

#	Current state	Event /condition =>actions	Next State
1	Any states	/ POWER-ON or RESET =>	OFFLINE
2	OFFLINE	/ In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) = 0 => -- receiving SYN and LL is not alive -- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	/ EXPIRED_TIMER (T(SL)) = "True" && V(LL) = 0 => ----sending out CLM All_Slot_Time := V(ST) if V(ESYN) = "True" then RMSDU := BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "True" => -- receiving SYN and this node is alive -- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
6	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "False" => -- receiving SYN and this node is not alive -- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	IDLE
7	IDLE	RMC_SEND_CF.ind { } / V(LL) <> 0 && V(LL)<V(TN)> = "True" && V(LN) = V(TN) => -----activate sending out LRR RMSDU := BUILD_PDU (LRR) RMS_SEND_CF.req { RMSDU } START_TIMER (T(SL), V(SL))	IDLE
8	IDLE	RMS_SEND_CF.conf { } / RMSDU.FC = LRR => none	ACTIVE_IDLE
9	ENTER	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	ENTER
10	ENTER	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER

#	Current state	Event /condition =>actions	Next State
11	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) = V(MN) && V(LL)<V(TN)> = "False" => RMSDU := BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
12	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++	ENTER
13	ENTER	/ V(LN) = V(MN) && V(LL)<V(TN)> = "False" => RMSDU := BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
14	ENTER	RMC_SEND_DATA.conf { } => none	ENTER
15	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V (LL)<V(TN)> = "True" => SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
16	ENTER	/ EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
17	ENTER	/ EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
18	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time > 0 => All_Slot_Time-- START_TIMER (T(SL), V(SL)) START_TIMER (T(IGP), V(IGP))	CLAIM
19	CLAIM	/ EXPIRED_TIMER (T(IGP))="True" => if V(ESYN) = "True" then RMSDU:=BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSD } START_TIMER (T(SL), V(SL)) endif	CLAIM
20	CLAIM	RMC_SEND_DATA.ind {RMSDU} / RMSDU.FC = (SYN CLM) => START_TIMER (T(SL), V(SL)) STOP_TIMER (T(IGP))	IDLE

#	Current state	Event /condition =>actions	Next State
21	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = 0 => All_Slot_Time := V(ST) V(LL) = 0 V(LL)<V(TN)> := 1 RMSDU := BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
22	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => All_Slot_Time := V(ST) START_TIMER (T(SL), V(SL))	IDLE
23	CLAIM	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> (SYN CLM) => -- receiving a frame except SYN, CLM -- START_TIMER (T(SL), V(SL))	CLAIM
24	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- V(LL) := 0 START_TIMER (T(SL), V(SL))	IDLE
25	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "False" => --- receiving SYN in non-SYN_mode---- SYN_frame := RMSDU V(LL) := SYN_frame.LL V(LN) := 1	CHECK_TOKEN
26	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "True" => ---Receiving SYN in SYN_mode----- SYN_Frame := RMSDU if SYN_frame.SN < V(TN) then SYN_mode := 0 V(LL) := SYN_frame.LL V(LN) := 1 endif	CHECK_TOKEN
27	ACTIVE_IDLE	RMC_SEND_DATA.conf { } / RMSDU.FC = SYN && CHECK_COL () = "False" => -- sent SYN ---- SYN_mode := 1 V(LN) := 1	CHECK_TOKEN
28	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
29	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
30	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (DT DT-CMP) => -- receiving DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE

#	Current state	Event /condition =>actions	Next State
31	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
32	ACTIVE_IDLE	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
33	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) SYN_frame.PN <> V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) if CHECK_SDU_BF (FC := REQ) = "False" then Speed-class := 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) endif	USE_TOKEN
34	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) SYN_frame.PN <> V(TN) => ----- Get token ----- if CHECK_SDU_BF (FC := REQ) = "True" then RMSDU := GET_SDU_BF (FC := REQ) --send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif	CHECK_TOKEN
35	CHECK_TOKEN	RMC_SEND_DATA.conf { } / RMSDU.FC = REQ && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN <> V(TN) => Speed-class := 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
36	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) if CHECK_SDU_BF (FC := REQ) = "True" then RMSDU := GET_SDU_BF (FC := REQ) RMC_SEND_DATA.req { RMSDU } --- send REQ START_TIMER (T(SL), V(SL)) else if CHECK_SDU_BF (FC := COM) = "True" then RMSDU := GET_SDU_BF (FC := COM) RMC_SEND_DATA.req { RMSDU } ---send COM START_TIMER (T(SL), V(SL)) else RMSDU := BUILD_PDU (RAS) RMC_SEND_DATA.req { RMSDU } ---send RAS START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) endif	CHECK_TOKEN

#	Current state	Event /condition =>actions	Next State
37	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = REQ && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => if CHECK_SDU_BF (FC := COM) = "True" then RMSDU := GET_SDU_BF (FC := COM) RMC_SEND_DATA.req { RMSDU } ---send COM START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
38	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = COM && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => RMSDU := BUILD_PDU (RAS) RMC_SEND_DATA.req { RMSDU } ---send RAS START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) </pre>	CHECK_TOKEN
39	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => Speed-class := 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) </pre>	USE_TOKEN
40	CHECK_TOKEN	<pre> / V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP)) </pre>	CHECK_TOKEN
41	CHECK_TOKEN	<pre> / V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++ </pre>	CHECK_TOKEN
42	CHECK_TOKEN	<pre> / V(LN) = V(MN) && V(LL)<V(TN)> = "False" => ---- MAC-control-time-period -- START_TIMER (T(TMAC), V(TMAC)) </pre>	NEXT_CYCLE
43	CHECK_TOKEN	<pre> / V(LN) = V(MN) && V(LL)<V(TN)> = "True" => RMSDU := BUILD_PDU (LPD) -----send LPD RMC_SEND_CF_ALL.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) </pre>	NEXT_CYCLE

#	Current state	Event /condition =>actions	Next State
44	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN <> V(TN) => -- Get token -- ----- matching V(LN), on receiving CMP or DT-CMP STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN> := 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC := REQ) = "False" then Speed-class := 3 -- Highest Priority START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) endif </pre>	USE_TOKEN
45	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN <> V(TN) => -- Get token -- STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN> := 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC := REQ) = "True" then RMSDU := GET_SDU_BF (FC := REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
46	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN = V(TN) => -- Get token -- STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN> := 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC := REQ) = "True" then RMSDU := GET_SDU_BF (FC := REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) else if CHECK_SDU_BF (FC := COM) = "True" then RMSDU := GET_SDU_BF (FC := COM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) else RMSDU := BUILD_PDU (RAS) -----send RAS RMC_SEND_DATA.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
47	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ <> V(TN) => --not matching V(LN).on receiving CMP or DT-CMP START_TIMER (T(SCMP), V(SCMP)) C(ASCMP)<RMSDU.SN>:=0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif </pre>	CHECK_TOKEN

#	Current state	Event /condition =>actions	Next State
48	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT => ---- receiving DT ---- C(ASCMP)<RMSDU.SN> := 0 ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
49	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
50	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "True" => -- receiving REQ at SYN-node-- V(LL)<RMSDU.SN> := 1 C(NONC) := 0	CHECK_TOKEN
51	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = COM && SYN_mode = "True" => -- receiving COM -- V(CW) := RMSDU.CW V(ST) := RMSDU.ST V(TSYN) := RMSDU.Th V(TTRT2) := RMSDU.Tm DLM_Set_value.req { Variable-name, Desired-value } C(NONC) := 0	CHECK_TOKEN
52	CHECK_TOKEN	/ EXPIRED_TIMER (T(SCMP)) => C(ASCMP)<V(LN)>++ V(ANONC)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif V(LN)++	CHECK_TOKEN
53	CHECK_TOKEN	RMC_SEND_DATA.conf { } / RMSDU.FC = (CMP DT-CMP) => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
54	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
55	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
56	CHECK_TOKEN	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
57	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send out DT or Sporadic frame -- RMSDU := BUILD_PDU (DLPDU) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	Current state	Event /condition =>actions	Next State
58	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
59	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> "Sporadic" => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class := Speed-class – 1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
60	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class = "Sporadic" => ---Expired THT and Speed-class ="Sporadic"----- ACM_SEND_DATA.conf { RMC-status } RMSDU := BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
61	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / RMSDU.FC = DT-CMP => ACM_SEND_DATA.conf { RMC-status } STOP_TIMER (T(ATHT)) Speed-class := Speed-class – 1 if Speed-class = 2 then START_TTRT (Speed-class) START_TTRT (Speed-class – 1) else if Speed-class = 1 then START_TTRT (Speed-class) endif	CHECK_TOKEN
62	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class <> "Sporadic" => -- Next Speed-class -- Speed-class := Speed-class – 1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
63	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class = "Sporadic" => -- All sent -- RMSDU := BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
64	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
65	NEXT_CYCLE	RMC_SEND_CF_ALL.conf { } / RMSDU.FC = LPD => none	ACTIVE_IDLE

#	Current state	Event /condition =>actions	Next State
66	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "False" => --receiving REQ at end node of active-segment-- PUT_SDU_BF (RMSDU.FC, RMSDU)	ACTIVE_IDLE
67	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode := 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
68	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
69	NEXT_CYCLE	/ EXIPERED_TIMER (T(TMAC)) = "True" && SYN.mode = "True" && V(ANONC) < (V(MN) – 1) => C(NONC) := 0 V(ANONC) := 0 RMSDU := BUILD_PDU (SYN) ----send SYN RMS_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
70	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "True" && (V(ANONC) >= (V(MN) – 1)) && (C(NONC)++ >= V(NONC)) => C(NONC) := 0 V(ANONC) := 0 SYN_mode := "False" START_TIMER (T(SL), V(SL))	IDLE
71	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "False" => none	ACTIVE_IDLE

7.2.5.5 Function used by ACM

All the functions used by the ACM are summarized in Table 77.

Table 77 – ACM function table

Function name	Input	Output	Operation
BUILD_PDU	FC	RMSDU	According to requested FC(Frame Control), assemble the Type-11 fieldbus frame
EXPIRED_TIMER	T(x)	True/False	When requested timer x has expired, "True" is returned, otherwise False is returned
MAINT_LL	V(LN)	(none)	The current operational status of the corresponding node specified V(LN) is read out of the V(LL) when T(SCMP) expired. When T(SCMP) expires all at number-of-times continuation of C(ASCMP) in the past, the corresponding bit of V(LL) is cleared
START_TIMER	T(x), V(y)	(none)	Timer x is set by value of y, and is activated
START_THT	Speed-class	(none)	T(ATHT) is activated according to the V(TTTR) remaining value of Access-class specified by Speed-class
START_TTRT	Speed-class	(none)	T(ATTTRT) is activated according to the V(TTTRT) value of Access-class specified by Speed-class
STOP_TIMER	T(x)	(none)	Timer x is deactivated
CHECK_COL ()	(none)	True/False	When the collision occurs, "True" is returned, otherwise "False" is returned.
PUT_SDU_BF	FC, RMSDU	(none)	Store RMSDU into the RMSDU-buffer associated with FC (Frame Control)
GET_SDU_BF	FC	RMSDU	Get RMSDU in the RMSDU-buffer associated with FC (Frame Control)
CHECK_SDU_BF	FC	True/False	Check that the RMSDU specified by FC (Frame Control) exists in the RMSDU-buffer. If it exists, "True" is returned, otherwise "False" is returned

7.2.6 Redundancy medium control (RMC)

7.2.6.1 Overview

The Redundancy-medium-control (RMC) provides the function for time-deterministic media-redundancy of ring-shaped Type 11 fieldbus, in cooperation with the ACM, by making use of the function of the Serializer / Deserializer which is composed of TX-framer, RX-framer-A/-B, Octet-serializer-A/-B and Octet-deserializer-A/-B, and further the services available from the physical layer. The function of RMC is as follows:

- The RMC manages the connection between the neighboring nodes which are connected in series to two other nodes in the ring, and the data sent out in both directions of the ring at a time is managed to pass through sequentially between nodes;
- The RMC purposely forms and manages the location of open-points in order to circumvent the data being circulated forever in the ring so that the data sent out of a node is not returned any more to the source node along through the circular path;
- The RMC manages the location of open-points at which the ring is split into two parts of the active and the open segment, and a node over the active segment is managed to be a member participating and communicating in the Type 11 fieldbus, on the other hand, a node over the open segment is not a member and is removed from or is to be joined into the Type 11 fieldbus;
- The RMC manages the location of the open-points in the ring, by initializing network for start-up, by changing the location for the remedy action as media-redundancy upon failure in order to prevent from total system stop and to maintain the communications capability among the nodes in healthy operation, by changing the location for new node insertion and for a member node removal;

- e) When the DLPDUs are received from the ACM, the RMC breaks them down into octet symbol requests to both Octet-serializers passed through TX-framer so that the DLPDUs are sent out both of the Ring-ports simultaneously;
- f) When the signals for control and diagnosis are received from the ACM, the RMC activates the transmissions of the MAC PDU out of the Ring-ports;
- g) When the octet symbol indications are received from the Octet-deserializer, the RX-framers assemble them into the MAC PDUs, and the RMC selects one of two outputs from the RX-framers so that the MAC PDUs are submitted to the ACM.

7.2.6.2 Primitive definitions

7.2.6.2.1 Primitives exchanged between ACM and RMC

Table 78 shows all of the primitives exchanged between the ACM and the RMC. Table 79 shows the primitives between the RMC and the Serializer / Deserializer which are composed of the Octet-serializer / -deserializer and the TX/RX-framer; furthermore, the primitives between the Serializer / Deserialize and the Ph-layer are shown in Table 80.

Table 78 – Primitives exchanged between ACM and RMC

Primitive name	Source	Associated parameters
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU
RMC_SEND_CF.req	ACM	RMSDU -----LRR specific
RMC_SEND_CF.conf	RMC	RMC-status
RMC_SEND_CF.ind	RMC	RMC-status
RMC_SEND_CF_All.req	ACM	RMSDU -----LPD specific
RMC_SEND_CF_All.conf	RMC	RMC-status

Table 79 – Primitives exchanged between RMC and Serializer / Deserializer

Primitive name	Source	Associated parameters
TX_DATA.req	RMC	RMPDU
TX_DATA.conf	Serializer-A / -B	Status
RX_DATA_A.ind	Deserializer-A	RMPDU, Status
RX_DATA_B.ind	Deserializer-B	RMPDU, Status

Table 80 – Primitives exchanged between RMS and Ph-layer

Primitive name	Source	Associated parameters
PLS_CARRIER_A.ind	Ph-layer: Ring-port-A	CARRIER_STATUS (CARRIER_ON / CARRIER_OFF)
PLS_SIGNAL_A.ind	Ph-layer: Ring-port-A	SIGNAL_STATUS (SIGNAL_ERROR / NO-SIGNAL_ERROR)
PLS_DATA_VALID_A.ind	Ph-layer: Ring-port-A	DATA_VALID_STATUS (DATA_VALID / DATA_NOT_VALID)
PLS_CARRIER_B.ind	Ph-layer: Ring-port-B	CARRIER_STATUS (CARRIER_ON / CARRIER_OFF)
PLS_SIGNAL_B.ind	Ph-layer: Ring-port-B	SIGNAL_STATUS (SIGNAL_ERROR / NO-SIGNAL_ERROR)
PLS_DATA_VALID_B.ind	Ph-layer: Ring-port-B	DATA_VALID_STATUS (DATA_VALID / DATA_NOT_VALID)

7.2.6.2.2 Parameters used between RMC and ACM, RMC and Serializer / Deserializer

Table 81 and Table 82 show the parameters used for interaction between the RMC and the ACM or the Ph-layer respectively.

Table 81 – Parameters between RMC and ACM

Parameter name	Description
RMSDU	Service-data-unit used between RMC and ACM
RMC-status	Status for result of request, condition of receiving etc

Table 82 – Parameters between RMC and Serializer / Deserializer, Ph-layer

Parameter name	Description
RMPDU	Protocol-data-unit used between RMC and Serializer / Deserialiser
CARRIER_STATUS	Status for condition of carrier-signal on or not over the medium
SIGNAL_STATUS	Status for condition of collision happened or not
DATA_VALID_STATUS	Status for received data valid and available

7.2.6.3 RMC state table

Figure 32 and Figure 33 show the state transition diagrams of the RMC for 100 Mbps and 1 000 Mbps operation respectively. The state tables of the RMC for 100 Mbps and 1 000 Mbps operation are shown in Table 83 and Table 84 respectively.

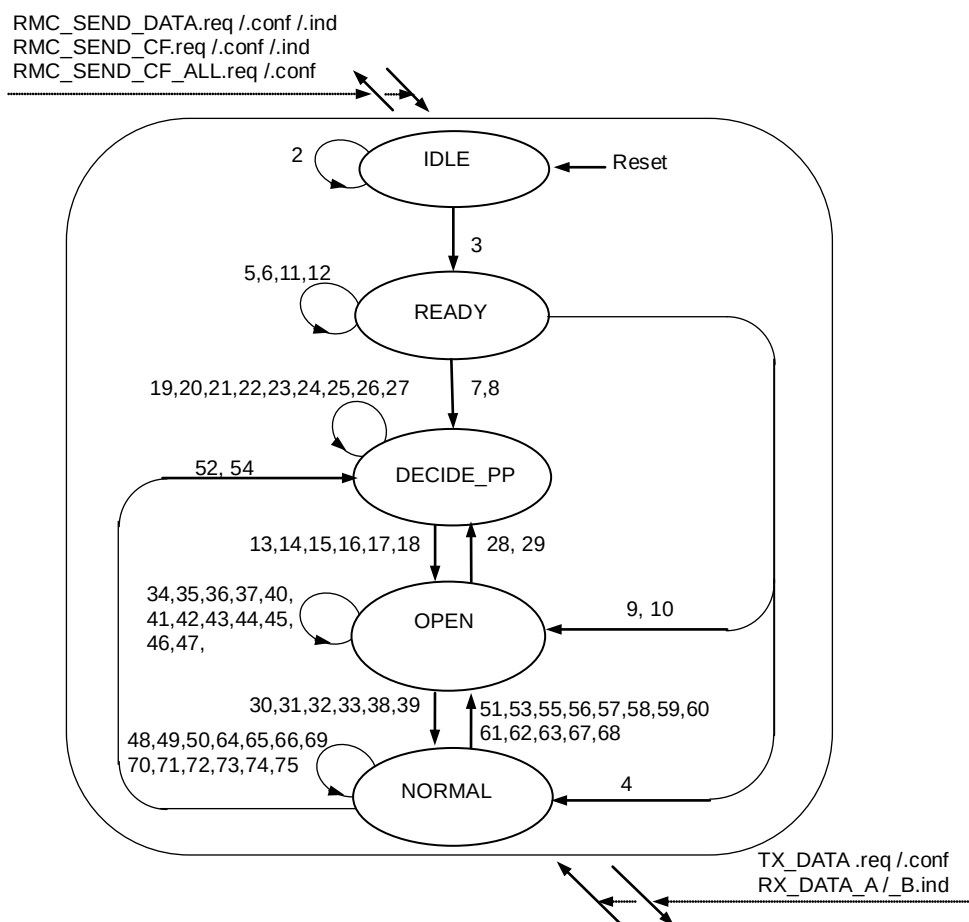


Figure 32 – State transition diagram of RMC for 100 Mbps operation

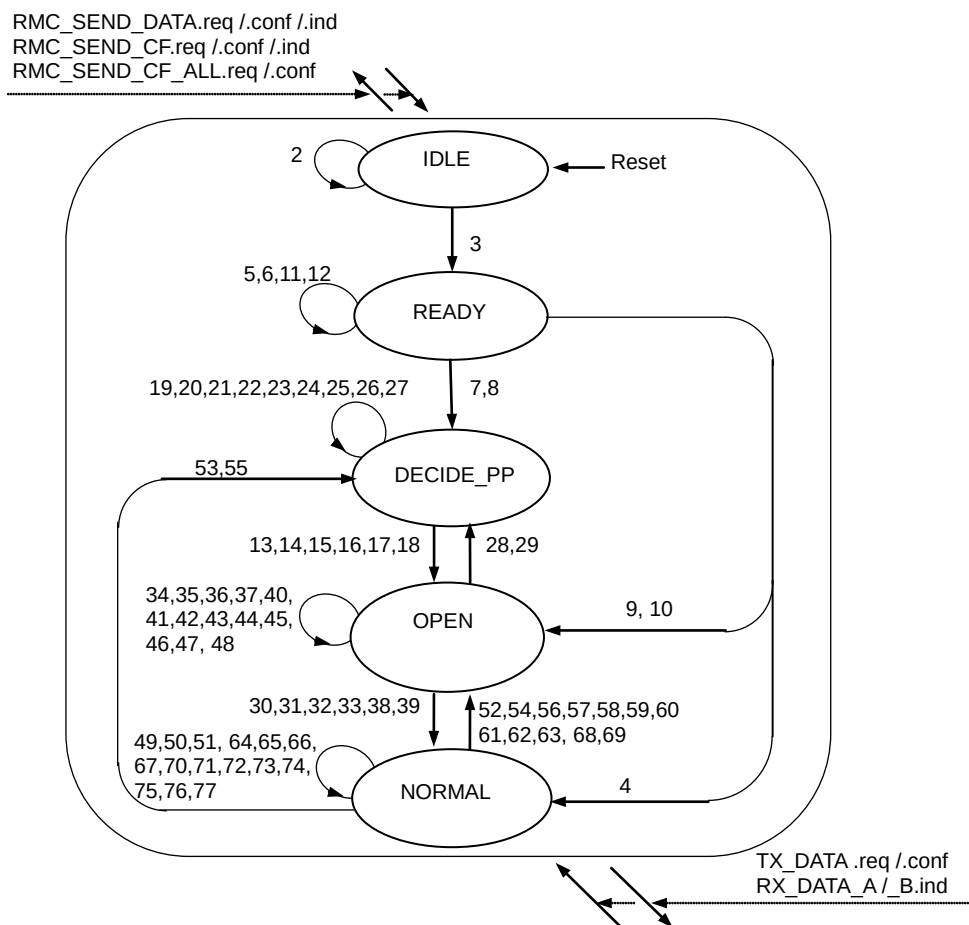


Figure 33 – State transition diagram of RMC for 1 000 Mbps operation

Table 83 – State table of RMC for 100 Mbps operation

#	Current state	Event /condition =>actions	Next state
1	Any states	/ RESET = "True" => RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable")	IDLE
2	IDLE	/ ACTIVE_ACM () = "False" => none ----- Wait for ACM being activated	IDLE
3	IDLE	/ ACTIVE_ACM () = "True" => -----ACM being activated RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Disable", "Disable")	READY

#	Current state	Event /condition =>actions	Next state
4	READY	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN && V(LINK _A) := "Normal" && V(LINK _B) := "Normal" => -----send out CLM or SYN SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B (" Enable ", "Disable") V(LSYN) := V(TN) RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
5	READY	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMC-status := "Failure" RMC_SEND_DATA.conf { RMC-Status }	READY
6	READY	RMC_SEND_DATA.req { RMSDU } / V(LINK _A) := "Abnormal" && V(LINK _B) := "Abnormal" => RMC-status := "LINK _{AB} -abnormal" RMC_SEND_DATA.conf { RMC-status }	READY
7	READY	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => ----- receiving CLM or SYN from Ring-port-A SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-Port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP
8	READY	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => -----receiving CLM/SYN-frame from Ring-port-B SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
9	READY	/ V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" => SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) := null RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN

#	Current state	Event /condition =>actions	Next state
10	READY	/ V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN) := null RMC-status := "LINK _A -abnormal" RMC_REC_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
11	READY	/ V(LINK _A) := "Abnormal" && V(LINK _B) := "Abnormal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable") V(LSYN) := null RMC-status := "LINK _{AB} -abnormal" RMC_REC_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	READY
12	READY	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	READY
13	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----again receiving from Ring-port-A => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
14	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----again receiving from Ring-port-B => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B))	OPEN
15	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----receiving from different Ring-port => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN
16	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----receiving from different Ring-port => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN

#	Current state	Event /condition =>actions	Next state
17	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" -----LINK _B into abnormal => V(LSYN) := RMPDU.SN RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU := RMPDU, RMC-status } START_TIMER (T(NRF _A))	OPEN
18	DECIDE_PP	RX_DATA_B.ind { RMPDU } /(RMPDU.FC = CLM SYM) && V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" -----LINK _A into abnormal => V(LSYN) := RMPDU.SN RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU := RMPDU, RMC-status } START_TIMER (T(NRF _B))	OPEN
19	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A) := RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	DECIDE_PP
20	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B) := RMPDU.SN ----- Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	DECIDE_PP
21	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => IND_PROC (RMPDU, REN _A) -----receiving on Ring-port-A START_TIMER (T(NRF _A))	DECIDE_PP
22	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => IND_PROC (RMPDU, REN _B) -----receiving on Ring-port-B START_TIMER (T(NRF _B))	DECIDE_PP
23	DECIDE_PP	RMC_SEND_CF.req { RMSDU } / RMSDU = LRR => -----wrong request for LRR RMC_SEND_DATA.conf (RMC-status := "Failure")	DECIDE_PP
24	DECIDE_PP	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => -----request for LPD SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") RMPDU := BUILD_RMPDU (FC := RMSDU.FC) TX_DATA.req { RMPDU } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	DECIDE_PP
25	DECIDE_PP	RMC_SEND_DATA.req { RMPDU } / => -----request for data-frame RMC_SEND_DATA.conf (RMC-status := "Failure")	DECIDE_PP
26	DECIDE_PP	TX_DATA.conf { RMPDU } / => -----conformation of LPD sent out RESTORE_FWE () SEND_COMP (RMPDU)	DECIDE_PP
27	DECIDE_PP	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	DECIDE_PP

#	Current state	Event /condition =>actions	Next state
28	OPEN	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") -----open-point at Ring-port-B V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A)) STOP_TIMER (T(NRF _B))	DECIDE_PP
29	OPEN	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") -----open-point at Ring-port-A V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B)) STOP_TIMER (T(NRF _A))	DECIDE_PP
30	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-B" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
31	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-A" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
32	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-B" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU }	NORMAL
33	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-A" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU }	NORMAL
34	OPEN	/ EXPIRED_TIMER (T(NRF _A)) && V(PP) = "Ring-port-A" => RMC-status := "No frame received from Ring-port-A" RMC_SEND_CF.ind { RMC-status } -----activate new ring configuration	OPEN
35	OPEN	/ EXPIRED_TIMER (T(NRF _B)) && V(PP) = "Ring-port-B" => RMC-status := "No frame received from Ring-port-B" RMC_SEND_CF.ind { RMC-status } ----- activate new ring configuration	OPEN

#	Current state	Event /condition =>actions	Next state
36	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _A)) => RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
37	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _B)) => RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
38	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _A)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _A))	NORMAL
39	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _B)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _B))	NORMAL
40	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A) := RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	OPEN
41	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B) := RMPDU.SN -----Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	OPEN
42	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	OPEN
43	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	OPEN
44	OPEN	RMC_SEND_DATA.req { RMSDU } / => if RMSDU.FC = (SYN CLM) then SET_NODE_COND ("Master", "Open", V(PS _A), V(PS _B)) V(LSYN) := V(TN) endif RMPDU := BUILD_RMPDU (RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but LPD and LRR	OPEN

#	Current state	Event /condition =>actions	Next state
45	OPEN	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => RMPDU := BUILD_RMPDU (RMPDU.FC) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TX_DATA.req { RMPDU } ----- sending LPD	OPEN
46	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = REQ => RMPDU := BUILD_RMPDU (RMSDU.FC) if V(PP) = "Ring-port-A" then SET_SEN ("Enable", "Disable") else SET_SEN ("Disable", "Enable") endif SET_NODE_COND ("Slave", "Open", V(PS _A), V(PS _B)) SAVE_FWE () TX_DATA.req { RMPDU } -----sending REQ	OPEN
47	OPEN	TX_DATA.conf {Status } / => RESTORE_FWE () SEND_COMP (RMPDU) --issuing RMC_SEND_DATA /_CF /_CF_ALL.conf	OPEN
48	NORMAL	TX_DATA.conf { RMPDU } / RMPDU.FC = CLM SYN => RMC_SEND_DATA.conf { RMSDU := RMPDU }	NORMAL
49	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN => SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") V(LSYN) := V(TN) RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending CLM or SYN	NORMAL
50	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMPDU := BUILD_RMPDU { RMSDU.FC } TX_DATA.req { RMPDU } -----sending all but CLM and SYN	NORMAL
51	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
52	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP

#	Current state	Event /condition =>actions	Next state
53	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN
54	NORMAL	RX_DATA_B.ind {RMPDU-B} / (RMPDU-B.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
55	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = Ring-port-B" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
56	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = Ring-port-A" && V(NS) := "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN
57	NORMAL	/ V(LINK _A) = "Abnormal" => -----Link fault on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
58	NORMAL	/ V(LINK _B) = "Abnormal" => -----Link fault on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
59	NORMAL	/ C(CRE _A) >= P(CRE _A) => -----abnormal-reception on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN

#	Current state	Event /condition =>actions	Next state
60	NORMAL	/ C(CRE _B) >= P(CRE _B) => -----abnormal-reception on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
61	NORMAL	/ EXPIRED_TIMER (T(NRF _A)) = "True" => -----long-silence on Ring-port-A V(NOS) := "Open"	OPEN
62	NORMAL	/ EXPIRED_TIMER (T(NRF _B)) = "True" => -----long-silence on Ring-port-B V(NOS) := "Open"	OPEN
63	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU := BUILD_PDU(FC := LRR) RMPDU.RN := V(AN _A) SAVE_FWE () SET_FWE_AB_BA ("Enable", "Disable") SET_SEN ("Enable", "Disable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _A))	NORMAL
64	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU := BUILD_PDU(FC := LRR) RMPDU.RN := V(AN _B) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Enable") SET_SEN ("Disable", "Enable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _B))	NORMAL
65	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _A))	NORMAL
66	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _B))	NORMAL
67	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF _A)) = "True" => -----No receipt from neighboring node to Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _A))	OPEN

#	Current state	Event /condition =>actions	Next state
68	NORMAL	/ EXPIRED_TIMER (T(ST) = "True" && ACTIVE_TIMER (T(NRF _B)) = "True" => -----No receipt from neighboring node to Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _B))	OPEN
69	NORMAL	RMC_SEND_CF_ALL.req { RMSDU } / => RMPDU := BUILD_RMPDU (FC := LPD) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TIME_DELAY (V(IGP)) TX_DATA.req { RMPDU } -----sending LPD	NORMAL
70	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-A V(AN _A) := RMPDU.SN V(FWE _{AB}) := "Enable" START_TIMER (T(NRF _A))	NORMAL
71	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-B V(AN _B) := RMPDU.SN V(FWE _{BA}) := "Enable" START_TIMER (T(NRF _B))	NORMAL
72	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	NORMAL
73	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU-B, REN _B) START_TIMER (T(NRF _B))	NORMAL
74	NORMAL	RMC_SEND_DATA.req { RMSDU } / => -----sending all but LPD or LRR if RMSDU.FC = SYN CLM then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(LSYN) := V(TN) endif SET_FWE_AB_BA ("Enable", "Enable") RMPDU := BUILD_RMPDU (FC := RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
75	NORMAL	TX_DATA.conf { RMPDU } / RMPDU <> CLM SYN => RESTORE_FWE () SEND_COMP (RMPDU)	NORMAL

#	Current state	Event /condition =>actions	Next state
8	READY	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => -----receiving CLM/SYN-frame from Ring-port-B SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
9	READY	/ V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" => SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) := null RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
10	READY	/ V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN) := null RMC-status := "LINK _A -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
11	READY	/ V(LINK _A) := "Abnormal" && V(LINK _B) := "Abnormal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable") V(LSYN) := null RMC-status := "LINK _{AB} -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	READY
12	READY	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	READY
13	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----again receiving from Ring-port-A => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN

#	Current state	Event /condition =>actions	Next state
14	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----again receiving from Ring-port-B => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B))	OPEN
15	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----receiving from different Ring-port => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN
16	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----receiving from different Ring-port => V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
17	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" -----LINK _B into abnormal => V(LSYN) := RMPDU.SN RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU := RMPDU, RMC-status } START_TIMER (T(NRF _A))	OPEN
18	DECIDE_PP	RX_DATA_B.ind { RMPDU } /(RMPDU.FC = CLM SYM) && V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" -----LINK _A into abnormal => V(LSYN) := RMPDU.SN RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU := RMPDU, RMC-status } START_TIMER (T(NRF _B))	OPEN
19	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A) := RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	DECIDE_PP
20	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B) := RMPDU.SN ----- Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	DECIDE_PP
21	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A)) -----receiving on Ring-port-A	DECIDE_PP
22	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B)) -----receiving on Ring-port-B	DECIDE_PP

#	Current state	Event /condition =>actions	Next state
23	DECIDE_PP	RMC_SEND_CF.req { RMSDU } / RMSDU = LRR => -----wrong request for LRR RMC_SEND_DATA.conf (RMC-status := "Failure")	DECIDE_PP
24	DECIDE_PP	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => -----request for LPD SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") RMPDU := BUILD_RMPDU (FC := RMSDU.FC) TX_DATA.req { RMPDU } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	DECIDE_PP
25	DECIDE_PP	RMC_SEND_DATA.req { RMPDU } / => -----request for data-frame RMC_SEND_DATA.conf (RMC-status := "Failure")	DECIDE_PP
26	DECIDE_PP	TX_DATA.conf { RMPDU } / => -----conformation of LPD sent out RESTORE_FWE () SEND_COMP (RMPDU)	DECIDE_PP
27	DECIDE_PP	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	DECIDE_PP
28	OPEN	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") -----open-point at Ring-port-B V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A)) STOP_TIMER (T(NRF _B))	DECIDE_PP
29	OPEN	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") -----open-point at Ring-port-A V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B)) STOP_TIMER (T(NRF _A))	DECIDE_PP
30	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-B" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
31	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-A" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL

#	Current state	Event /condition =>actions	Next state
32	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-B" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU }	NORMAL
33	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-A" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU }	NORMAL
34	OPEN	/ EXPIRED_TIMER (T(NRF _A)) && V(PP) = "Ring-port-A" => RMC-status := "No frame received from Ring-port-A" RMC_SEND_CF.ind { RMC-status } -----activate new ring configuration	OPEN
35	OPEN	/ EXPIRED_TIMER (T(NRF _B)) && V(PP) = "Ring-port-B" => RMC-status := "No frame received from Ring-port-B" RMC_SEND_CF.ind { RMC-status } ----- activate new ring configuration	OPEN
36	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _A)) => RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
37	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _B)) => RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req (RMPDU) -----sending LRR out START_TIMER (T(ST))	OPEN
38	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _A)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _A))	NORMAL
39	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _B)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _B))	NORMAL
40	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A) := RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	OPEN

#	Current state	Event /condition =>actions	Next state
41	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B) := RMPDU.SN -----Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	OPEN
42	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	OPEN
43	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	OPEN
44	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = (SYN CLM) => if V(ESYN) = "True" then SET_NODE_COND ("Master", "Open", V(PS _A), V(PS _B)) V(LSYN) := V(TN) RMPDU := BUILD_RMPDU (FC := RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but LPD and LRR else RMC-status := "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	OPEN
45	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> (SYN CLM REQ) => RMPDU := BUILD_RMPDU (FC := RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but SYN, CLM, LPD, LRR, REQ	OPEN
46	OPEN	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => RMPDU := BUILD_RMPDU (RMPDU.FC) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TX_DATA.req { RMPDU } ----- sending LPD	OPEN
47	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = REQ => RMPDU := BUILD_RMPDU (RMSDU.FC) if V(PP) = "Ring-port-A" then SET_SEN ("Enable", "Disable") else SET_SEN ("Disable", "Enable") endif SET_NODE_COND ("Slave", "Open", V(PS _A), V(PS _B)) SAVE_FWE () TX_DATA.req { RMPDU } -----sending REQ	OPEN
48	OPEN	TX_DATA.conf {Status } / => RESTORE_FWE () SEND_COMP (RMPDU) --issuing RMC_SEND_DATA /_CF /_CF_ALL.conf	OPEN
49	NORMAL	TX_DATA.conf { RMPDU } / RMPDU.FC = CLM SYN => RMC_SEND_DATA.conf { RMSDU := RMPDU }	NORMAL

#	Current state	Event /condition =>actions	Next state
50	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN => if V(ESYN) = "True" then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") V(LSYN) := V(TN) RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending CLM or SYN else RMC-status := "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	NORMAL
51	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMPDU := BUILD_RMPDU { RMSDU.FC } TX_DATA.req { RMPDU } -----sending all but CLM and SYN	NORMAL
52	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
53	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP
54	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN

#	Current state	Event /condition =>actions	Next state
55	NORMAL	RX_DATA_B.ind {RMPDU-B} / (RMPDU-B.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
56	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = Ring-port-B" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	OPEN
57	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = Ring-port-A" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" V(LSYN) := RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	OPEN
58	NORMAL	/ V(LINK _A) = "Abnormal" => -----Link fault on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
59	NORMAL	/ V(LINK _B) = "Abnormal" => -----Link fault on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
60	NORMAL	/ C(CRE _A) >= P(CRE _A) => -----abnormal-reception on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
61	NORMAL	/ C(CRE _B) >= P(CRE _B) => -----abnormal-reception on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
62	NORMAL	/ EXPIRED_TIMER (T(NRF _A)) = "True" => -----long-silence on Ring-port-A V(NOS) := "Open"	OPEN
63	NORMAL	/ EXPIRED_TIMER (T(NRF _B)) = "True" => -----long-silence on Ring-port-B V(NOS) := "Open"	OPEN

#	Current state	Event /condition =>actions	Next state
64	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU := BUILD_PDU(FC := LRR) RMPDU.RN := V(AN _A) SAVE_FWE () SET_FWE_AB_BA ("Enable", "Disable") SET_SEN ("Enable", "Disable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _A))	NORMAL
65	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU := BUILD_PDU(FC := LRR) RMPDU.RN = V(AN _B) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Enable") SET_SEN ("Disable", "Enable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _B))	NORMAL
66	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _A))	NORMAL
67	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _B))	NORMAL
68	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF _A)) = "True" => -----No receipt from neighboring node to Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _A))	OPEN
69	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF _B)) = "True" => -----No receipt from neighboring node to Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _B))	OPEN
70	NORMAL	RMC_SEND_CF_ALL.req { RMSDU } / => RMPDU := BUILD_RMPDU (FC := LPD) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TIME_DELAY (V(IGP)) TX_DATA.req { RMPDU } -----sending LPD	NORMAL

#	Current state	Event /condition =>actions	Next state
71	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-A V(AN _A) := RMPDU.SN V(FWE _{AB}) := "Enable" START_TIMER (T(NRF _A))	NORMAL
72	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-B V(AN _B) := RMPDU.SN V(FWE _{BA}) := "Enable" START_TIMER (T(NRF _B))	NORMAL
73	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	NORMAL
74	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU-B, REN _B) START_TIMER (T(NRF _B))	NORMAL
75	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = (SYN CLM) => -----sending SYN or CLM if V(ESYN) = "True" then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(LSYN) := V(TN) SET_FWE_AB_BA ("Enable", "Enable") RMPDU := BUILD_RMPDU (FC := RMSDU.FC) TX_DATA.req { RMPDU } else RMC-status := "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	NORMAL
76	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> (SYN CLM) => -----sending all but LPD or LRR SET_FWE_AB_BA ("Enable", "Enable") RMPDU := BUILD_RMPDU (FC := RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
77	NORMAL	TX_DATA.conf { RMPDU } / RMPDU <> CLM SYN => RESTORE_FWE () SEND_COMP (RMPDU)	NORMAL

7.2.6.4 Functions used by RMC

The functions used in the RMC state table are summarized in Table 85.

Table 85 – The RMC function table

Function name	Description and operation
ACTIVE_ACM ()	When the ACM is in activity, "True" is returned, otherwise, "False" is returned
ACTIVE_TIMER (v(t))	When the requested timer v(t) is in activity, "True" is returned, otherwise, "False" is returned
BUILD_RMPDU (FC)	According to the requested frame-control-information(FC), a ring-control-frame is correspondingly assembled
EXPIRED_TIMER (v(t))	When the requested timer v(t) has expired, "True" is returned, otherwise, "False" is returned
IND_PROC (PDU, ren)	if ren = "Enabled" && (PDU <> (LPD LRR)) then RMC_RECV_DATA.ind { PDU } endif
RESTORE_FWE ()	Restore V(FWE _{AB}) and V(FWE _{BA})
RESET_VAR ()	Reset the variables to the default value
SAVE_FWE ()	Save V(FWE _{AB}) and V(FWE _{BA}) temporarily
SEND_COMP (RMPDU)	if RMPDU.FC = LRR then RMC_SEND_CF.conf { RMC-status := "Success" } elseif RMPDU.FC = LPD then RMC_SEND_CF_ALL.conf { RMC-status := "Success" } else RMC_SEND_DATA.conf { RMC-status := "Success" } endif
SET_FWE_AB_BA (fweab, fweba)	V(FWE _{AB}) := fweab -----value is either "Enable" or "Disable" V(FWE _{BA}) := fweba -----value is either "Enable" or "Disable"
SET_NODE_COND (Node-state, Operation-state, Ring-port-A-state, Ring-port-B-state)	V(NS) := Node-state -----value is either "Master" or "Slave" V(NOS) := Node-operation-state -----value is either "Normal" or "Open" V(PS _A) := Ring-port-A-state -----value is either "Non-blocked" or "Blocked" V(PS _B) := Ring-port-B-state -----value is either "Non-blocked" or "Blocked"
SET_REN_A_B (rena, renb)	V(REN _A) := rena -----value is either "Enable" or "Disable" V(REN _B) := renb -----value is either "Enable" or "Disable"
SET_SEN (sena, senb)	V(SEN _A) := sena -----value is either "Enable" or "Disable" V(SEN _B) := senb -----value is either "Enable" or "Disable"
START_TIMER (v(t))	Activate the requested timer V(t); "Expired Timer" event happens upon the timer expired
STOP_TIMER (v(t))	Stop the requested timer V(T) forcibly
TIME_DELAY (v(x))	Wait by the time period specified by v(x)

7.3 Serializer and deserializer

The serializer and deserializer are identical to ISO/IEC 8802-3.

7.4 DLL management protocol

7.4.1 DLL management protocol for star-architecture

7.4.1.1 Overview

The interface protocol between the DLM and the DLMS-user is described. The DLL management protocol provides the DLL management services specified in IEC 61158-3-11, Clause 5 by making use of the services available by the DLMS-user.

In 7.4.1, full implementation matters and local matters are intentionally excluded.

7.4.1.2 Primitive definitions

7.4.1.2.1 Primitives exchanged between DLMS-user and DLM

Table 86 shows all primitives exchanged by the DLMS-user and the DLM.

Table 86 – Primitives exchanged between DLMS-user and DLM

Primitive name	Source	Associated parameters
DLM-Reset request	DLMS-user	(none)
DLM-Reset confirm	DLM	Status
DLM-Set-value request	DLMS-user	Variable-name, Desired-value
DLM-Set-value confirm	DLM	Status
DLM-Get-variable request	DLMS-user	Variable-name
DLM-Get-variable confirm	DLM	Status, Current_value
DLM-Event indication	DLM	Event-identifier, Additional-information
DLM-Set-publisher-configuration request	DLMS-user	Desired-speed-class, Desired-configuration
DLM-Set-publisher-configuration confirmation	DLM	Status
DLM-Get-publisher-configuration request	DLMS-user	Desired-speed-class
DLM-Get-publisher-configuration confirmation	DLM	Status, Current-configuration
DLM-Activate-TCC request	DLMS-user	Desired-speed-class
DLM-Activate-TCC confirmation	DLM	Status
DLM-Deactivate-TCC request	DLMS-user	Desired-speed-class
DLM-Deactivate-TCC confirmation	DLM	Status

7.4.1.2.2 Parameters used with primitives exchanged between DLMS-user and DLM

All parameters used with primitives exchanged between the DLMS-user and the DLM is summarized in Table 87. Table 88 summarizes the variables associated with state change events.

Table 87 – Parameters used with primitives exchanged between DL-user and DLM

Parameter name	Description
Status	Status of the service execution
Variable-name	DL variable name to be addressed
Desired-value	DL variable value to be set
Current-value	Current DL variable value to be read out
Event-identifier / Additional-information	Event whose occurrence is being announced with related DL variable and the state changed
Desired-speed	Speed-class for the Time-critical Cyclic data transmission
Desired-configuration	A set of DLCEP-identifiers to be set as a publisher for the Time-critical Cyclic data transmission with desired-speed class
Current-configuration	A current set of DLCEP-identifiers set as a publisher for the Time-critical Cyclic data transmission with desired-speed class

The data types are specified in IEC 61158-5-11, Clause 5. The most significant bit of the most significant octet is always used as the most significant bit of the binary number.

Table 88 – Event-related state change variables

Operating parameters			
Variable name	Description	Additional-information	Data type
RCS_A , RCS_B	Receive-channel A, receive-channel B selected for receiving packets	True: the receive-channel A or B selected for receiving packets respectively. False: not selected for receiving packets on channel A or B respectively	BOOL
SEN_A , SEN_B	Transmitter A, transmitter B enabled to send out packets	True: enabled to send out packets from transmitter A or B respectively. False: disabled to send out packets from transmitter A or B respectively. Default value is "True"	BOOL
IS	IN or out of service of total node operation	True : in service False : out of service	BOOL
FE	Fatal error occurred	True: fatal error occurred and system entered into out of service. False : under normal operation	BOOL
TCIS	In or out of service of TCC data transmission	True : In service of TCC data transmission False : out of TCC data transmission service	BOOL
CRCC	Receive-channel changed	True: Receive-channel changed. Receiving channel can be checked out using RCS_A and RCS_B	BOOL
$LINK_A$, $LINK_B$	Ethernet link channel A or B is running or not	True: Ethernet link is running. False: Ethernet link is out of service	BOOL
CTCC	Transmit-channel changed	True: Transmit-channel changed. Transmitting channel can be checked out using SEN_A and SEN_B	BOOL
CLLC	Live-list content changed	True: LL content changed	BOOL
DDLE	Duplicate DLEs detected with same node identifier number of TN	True: duplicate DLEs detected with same node identifier number of TN. False : only one DLE with this number of TN	BOOL
CCDHS	State, indicating active (healthy) or inactive (unhealthy) of the corresponding TCC data, is changed. State is reflected on the corresponding information data block "CDH _{blk} "	True: state changed active from/to inactive. False: no state changed	BOOL

7.4.1.3 DLM state table

The DLM state transition diagrams are shown in Figure 34 and Table 89.

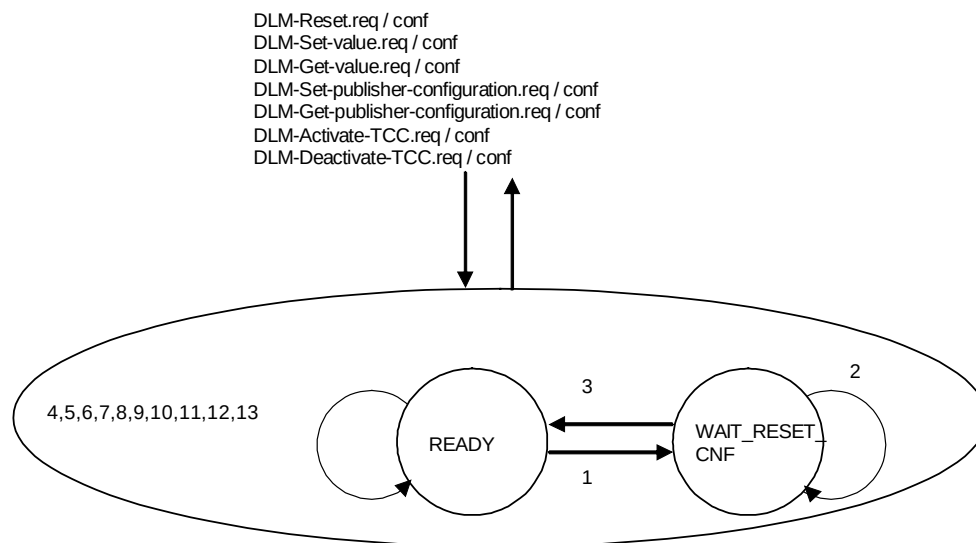


Figure 34 – State transition diagram of DLM

Table 89 – DLM state table

#	Current state	Event /condition ⇒actions	Next state
1	READY	DLM-Reset.req { } ⇒ MAC_reset := MAC_RESET() RMC_reset := "FALSE"	WAIT_RESET_CNF
2	WAIT_RESET_CNF	/ MAC_reset := "TRUE" && RMC_reset := "FALSE" ⇒ RMC_reset := RMC_RESET	WAIT_RESET_CNF
3	WAIT_RESET_CNF	/ ⇒ RESET_VAR () -- MAC and RMC reset completed DLM_RESET.cnf { Status := "Success" }	READY
4	READY	DLM-Set-value.req {Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) = "Valid" ⇒ SET_VALUE (Variable-name, Desired-value) Status := "Success" DLM-Set-value.conf {Status}	READY
5	READY	DLM-Set-value.req {Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) <> "Valid" ⇒ Status := "Failure" DLM-Set-value.conf {Status}	READY
6	READY	DLM-Get-value.req {Variable-name } / CHECK_VAR (Variable-name) = "Valid" ⇒ Current-value := GET_CURRENT_VAL (Variable-name) Status := "Success" DLM-Get-value.conf { Current-value, Status }	READY
7	READY	DLM-Get-value.req {Variable-name } / CHECK_VAR (Variable-name) <> "Valid" ⇒ Current-value := NIL Status := "Failure" DLM-Get-value.conf { Current-value, Status }	READY

#	Current state	Event /condition ⇒actions	Next state
8	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) = "Valid" ⇒ SET_PC (Desired-speed-class, Desired-configuration) Status := "Success" DLM-Set-publisher-configuration.conf { Status }	READY
9	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) <> "Valid" ⇒ Status := "Failure" DLM-Set-publisher-configuration.conf { Status }	READY
10	READY	DLM-Get-publisher-configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) = "Valid" ⇒ Current-configuration := GET_PC (Desired-speed-class) Status := "Success" DLM-Get-Publisher-Configuration.conf { Status, Current- configuration }	READY
11	READY	DLM-Get-Publisher-Configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) <> "Valid" ⇒ Current-configuration := NIL Status := "Failure" DLM-Get-Publisher-Configuration.conf { Status, Current- configuration }	READY
12	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) = "Valid" ⇒ STOP_TCC (Desired-speed-class) Status := "Success" DLM-Activate-TCC.conf { Status }	READY
13	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) = "Valid" ⇒ START_TCC (Desired-speed-class) Status := "Success" DLM-Deactivate-TCC.conf { Status }	READY
14	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) <> "Valid" ⇒ Status := "Failure" DLM-Deactivate-TCC.conf { Status }	READY

7.4.1.4 Function

The DLM Functions are summarized in Table 90.

Table 90 – DLM function table

Function name	Description and operation
CHECK_PC (Desired-speed-class, Desired-configuration)	Check that the requested Desired-speed-class and Desired-configuration are valid. Possible Desired-speed-class values are - High-speed - Medium-speed or Low-speed Check that DLCEP-address value and total amount of Desired-configuration. Furthermore, when TCC data transmission of the requested Desired-speed-class is inactive, the value "valid" is returned
CHECK_SPEED (Desired-speed-class)	Check that the requested Desired-speed-classes are valid. Possible Desired-speed-classes are - High-speed - Medium-speed - Low-speed
CHECK_VALUE (Variable-name, Desired-value)	Check that the requested Variables with Desired value are valid. Possible variables with the value are defined in subclause 4.6
CHECK_VAR (Variable-name)	Check that the requested Variables are valid. Possible variables are defined in subclause 4.6
CHECK_ATCC (Desired-speed-class)	Check that the requested Desired-speed-classes are valid. Possible Desired-speed-classes are - High-speed - Medium-speed - Low-speed Furthermore, when TCC data transmission of the requested Desired-speed-class is inactivated, the value "valid" is returned
CHECK_DTCC (Desired-speed-class)	Check that the requested Desired-speed-classes are valid. Possible Desired-speed-classes are - High-speed - Medium-speed - Low-speed Furthermore, when TCC data transmission of the requested Desired-speed-class is inactivated, the value "valid" is returned
GET_CURRENT_VAL (Variable-name)	Get the value of requested Variable
RESET_VAR	Initialize all valuables and the set of the Publisher-configuration for each Speed-class. TCC data transfer service is deactivated
START_TCC (Desired-speed)	Activate the TCC data transmission of requested Desired-speed-class
STOP_TCC (Desired-speed)	Deactivate the TCC data transmission of the requested Desired-speed-class

7.4.2 DLL management protocol for loop-architecture

7.4.2.1 Overview

The interface protocol between the DLM and the DLMS-user is described. The DLL management protocol provides the DLL management services specified in IEC 61158-3-11, Clause 5 by making use of the services available by the DLMS-user.

In 7.4.2, full implementation matters and local matters are intentionally excluded.

7.4.2.2 Primitive definitions

7.4.2.2.1 Primitives exchanged between DLMS-user and DLM

Table 91 shows all primitives exchanged by the DLMS-user and the DLM.

Table 91 – Primitives exchanged between DLMS-user and DLM

Primitive name	Source	Associated parameters
DLM-Reset request	DLMS-user	none
DLM-Reset confirm	DLM	Status
DLM-Set-value request	DLMS-user	Variable-name, Desired-value
DLM-Set-value confirm	DLM	Status
DLM-Get-variable request	DLMS-user	Variable-name
DLM-Get-variable confirm	DLM	Status, Current_value
DLM-Event indication	DLM	Event-identifier, Additional-information
DLM-Set-publisher-configuration request	DLMS-user	Desired-speed-class, Desired-configuration
DLM-Set-publisher-configuration confirmation	DLM	Status
DLM-Get-publisher-configuration request	DLMS-user	Desired-speed-class
DLM-Get-publisher-configuration confirmation	DLM	Status, Current-configuration
DLM-Activate-TCC request	DLMS-user	Desired-speed-class
DLM-Activate-TCC confirmation	DLM	Status
DLM-Deactivate-TCC request	DLMS-user	Desired-speed-class
DLM-Deactivate-TCC confirmation	DLM	Status

7.4.2.2.2 Parameters used with primitives exchanged between DLMS-user and DLM

All parameters used with primitives exchanged between the DLMS-user and the DLM is summarized in Table 92.

Table 92 – Parameters used with primitives exchanged between DL-user and DLM

Parameter name	Description
Status	Status of the service execution
Variable-name	DL variable name to be addressed
Desired-value	DL variable value to be set
Current-value	Current DL variable value to be read out
Event-identifier / Additional-information	Event whose occurrence is being announced with related DL variable and the state changed
Desired-speed	Speed-class for the Time-critical-cyclic-data transmission
Desired-configuration	A set of DLCEP-identifiers to be set as a publisher for the Time-critical-cyclic-data transmission with desired-speed class
Current-configuration	A current set of DLCEP-identifiers set as a publisher for the Time-critical-cyclic-data transmission with desired-speed class

Table 93 summarizes the variables associated with state change events.

The data types are specified in IEC 61158-5-11, Clause 5. The most significant bit of the most significant octet is always used as the most significant bit of the binary number.

Table 93 – Event-related state change variables

Operating parameters			
Variable name	Description	Additional-information	Data type
NS	Node state changed into “Master” or “Slave”	True : “Master” False : “Slave”	BOOL
NOS	Node operation state changed into “Normal” or “Open”	True : “Normal” False : “Open”	BOOL
PS _A , PS _B	Ring-port state A or B changed into “Blocked” or “Non-blocked”	True : “Non-blocked” False : “Blocked”	BOOL
LINK _A , LINK _B	Ethernet link over Ring-port A or B changed into running or not	True: Ethernet link is running False: Ethernet link is out of service	BOOL
AN _A , AN _B	Adjacent node changed	Value is in the range of 254 to 0	USINT
NRF _A , NRF _B	No frame received occurred over LINK _A or B	True : no frame received occurred over observational time period False : frames received periodically	BOOL
CRE _A , CRE _B	Abnormal-reception-error occurred over LINK _A or B	True : abnormal-reception-error occurred over permissible repetitive counts False : no abnormal-reception-error	BOOL
LSYN	Node in “Master” state changed	Value is in the range of 254 to 0	USINT
DDLE	Duplicate DLEs detected with same node identifier number of TN	True: duplicate DLEs detected with same node identifier number of TN. False : only one DLE with this number of TN	BOOL
TCIS	In or out of service of TCC data transmission	True : In service of TCC data transmission False : out of TCC data transmission service	BOOL
IS	IN or out of service of total node operation	True : in service False : out of service	BOOL
FE	Fatal error occurred	True: fatal error occurred and system entered into out of service. False : under normal operation	BOOL

7.4.2.3 DLM state table

The DLM state transition diagram is shown in Figure 35 and the DLM state table is shown in Table 94 respectively.

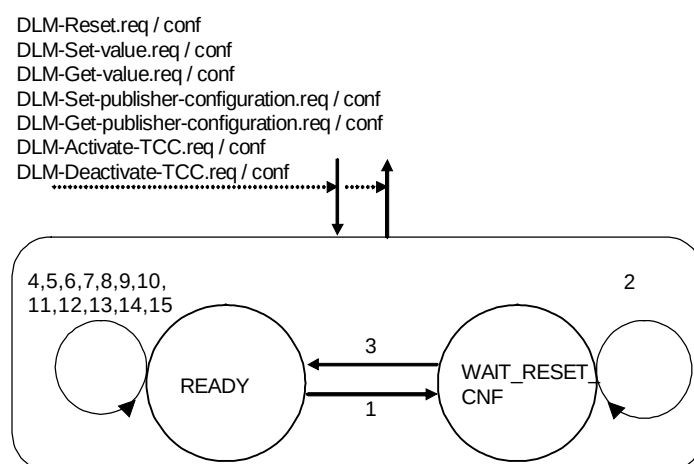
**Figure 35 – State transition diagram of DLM**

Table 94 – DLM state table

#	Current state	Event /condition =>actions	Next state
1	READY	DLM-Reset.req { } / => ACM_reset := ACM_RESET() RMC_reset := "FALSE"	WAIT_RESET_CNF
2	WAIT_RESET_CNF	/ ACM_reset = "True" && RMC_reset = "False" => RMC_reset := RMC_RESET ()	WAIT_RESET_CNF
3	WAIT_RESET_CNF	/ ACM_reset = "True" && RMC_reset = "True" => ----- ACM and RMC reset completed RESET_VAR () DLM-Reset.conf { Status := "Success" }	READY
4	READY	DLM-Set-value.req { Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) = "Valid" => SET_VALUE (Variable-name, Desired-value) DLM-Set-value.conf { Status := "Success" }	READY
5	READY	DLM-Set-value.req { Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) <> "Valid" => DLM-Set-value.conf { Status := "Failure" }	READY
6	READY	DLM-Get-value.req { Variable-name } / CHECK_VAR (Variable-name) = "Valid" => Current-value := GET_CURRENT_VAL (Variable-name) DLM-Get-value.conf { Current-value := Current-value, Status := "Success" }	READY
7	READY	DLM-Get-value.req { Variable-name } / CHECK_VAR (Variable-name) <> "Valid" => Current-value := NIL Status := "Failure" DLM-Get-value.conf { Current-value := NIL, Status := "Failure" }	READY
8	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) = "Valid" => SET_PC (Desired-speed-class, Desired-configuration) DLM-Set-publisher-configuration.conf { Status := "Success" }	READY
9	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) <> "Valid" => Status := "Failure" DLM-Set-publisher-configuration.conf { Status }	READY
10	READY	DLM-Get-publisher-configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) = "Valid" => Current-configuration := GET_PC (Desired-speed-class, Desired-configuration) DLM-Get-Publisher-Configuration.conf { Status := "Success", Current-configuration := Current-configuration }	READY

#	Current state	Event /condition =>actions	Next state
11	READY	DLM-Get-Publisher-Configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) <> "Valid" => DLM-Get-Publisher-Configuration.conf { Status := "Failure", Current-configuration := NIL }	READY
12	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) = "Valid" => START_TCC (Desired-speed-class) DLM-Activate-TCC.conf { Status := "Success" }	READY
13	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) <> "Valid" => DLM-Activate-TCC.conf { Status := "Failure" }	READY
14	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) = "Valid" => STOP_TCC (Desired-speed-class) DLM-Deactivate-TCC.conf { Status := "Success" }	READY
15	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) <> "Valid" => DLM-Deactivate-TCC.conf { Status := "Failure" }	READY

7.4.2.4 Function

The DLM Functions are summarized in Table 95.

Table 95 – DLM function table

Function name	Description and operation
CHECK_PC (Desired-speed-class, Desired-configuration)	Check that the requested Desired-speed-class and Desired-configuration, specifically on the DLCEP-address value and total amount of Desired-configuration, is valid. Possible value of Desired-speed-class is "High-speed" or "Medium-speed". when TCC data transmission of the requested Desired-speed-class is in activity, the value "Valid" is returned, otherwise "Invalid" is returned
CHECK_SPEED (Desired-speed-class)	Check that the requested Desired-speed-class is valid. Possible Desired-speed-class is "High-speed" or "Medium-speed"
CHECK_VALUE (Variable-name, Desired-value)	Check that the requested Variables with Desired value are valid. Possible variables with the value are defined in 4.6
CHECK_VAR (Variable-name)	Check that the requested Variables are valid. Possible variables are defined in 4.6
CHECK_ATCC (Desired-speed-class)	Check that the requested Desired-speed-class is valid. Possible Desired-speed-class is "High-speed" or "Medium-speed". When TCC data transmission of the requested Desired-speed-class is in activity, the value "Valid" is returned, otherwise "Invalid" is returned
CHECK_DTCC (Desired-speed-class)	Check that the requested Desired-speed-class is valid. Possible Desired-speed-class is "High-speed" or "Medium-speed". When TCC data transmission of the requested Desired-speed-class is in activity, the value "valid" is returned, otherwise "Invalid" is returned
SET_PC (Desired-speed-class, Desired-configuration)	Set the Desired-configuration specified by the Desired-speed-class
SET_VALUE (Variable-name, Desired-value)	Set the Desired-value to the Variable specified by the Variable-name
GET_CURRENT_VAL (Variable-name)	Get the value of requested Variable
GET_PC (Desired-speed-class, Desired-configuration)	Get the Desired-configuration specified by the Desired-speed-class
RESET_VAR ()	Initialize all variables and the set of the Publisher-configuration for each Speed-class. TCC data transfer service is deactivated
ACM_RESET ()	Initialize the ACM. "True" is returned after the ACM initialized
RMC_RESET ()	Initialize the RMC. "True" is returned after the RMC initialized
START_TCC (Desired-speed)	Activate the TCC data transmission of requested Desired-speed-class
STOP_TCC (Desired-speed)	Deactivate the TCC data transmission of the requested Desired-speed-class

Bibliography

IEC 61784-1, *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 3309:1993 ², *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 – Specific requirements*

ISO/IEC 9314-2:1989, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 2: Token Ring Media Access Control (MAC)*

ISO/TR 13283:1998, *Industrial automation – Time-critical communications architectures – User requirements and network management for time-critical communications systems*

² This publication has been withdrawn.

SOMMAIRE

AVANT-PROPOS	167
INTRODUCTION.....	169
1 Domaine d'application	171
1.1 Généralités.....	171
1.2 Spécifications.....	171
1.3 Procédures.....	171
1.4 Applicabilité.....	172
1.5 Conformité	172
2 Références normatives.....	172
3 Termes, définitions, symboles et abréviations.....	173
3.1 Termes et définitions du modèle de référence	173
3.2 Termes et définitions de convention de service	175
3.3 Termes et définitions.....	176
3.4 Symboles et abréviations	180
4 Présentation du protocole DL	181
4.1 Généralités.....	181
4.2 Présentation du contrôle d'accès au support	182
4.3 Service supposé provenir de PhL	183
4.4 Architecture DLL	184
4.5 Machine de contrôle d'accès et fonctions de support de planification	189
4.6 Paramètres locaux, variables, compteurs, temporisateurs et files d'attente	189
5 Structure générale et codage de PhIDU et DLPDU et des éléments de procédure connexes.....	211
5.1 Présentation.....	211
5.2 Structure PhIDU et codage	211
5.3 Structure, codage et éléments de procédure de la trame MAC commune	211
5.4 Éléments de la trame MAC	212
5.5 Ordre de transmission binaire.....	217
5.6 DLPDU non valide	217
6 Structure, codage et éléments de procédure spécifiques à la DLPDU.....	218
6.1 Généralités.....	218
6.2 DLPDU de synchronisation (SYN)	218
6.3 DLPDU de transmission complète (CMP)	224
6.4 DLPDU de demande dans l'anneau (REQ)	226
6.5 DLPDU de revendication (CLM).....	228
6.6 Commande DLPDU (COM)	229
6.7 Données cycliques et données cycliques avec DLPDU de transmission complète (DT) et (DT-CMP).....	231
6.8 DLPDU RAS (RAS)	232
6.9 DLPDU de demande de répétition de boucle (LRR)	233
6.10 DLPDU de diagnostic de boucle (LPD)	237
7 Éléments de procédure DLE	238
7.1 Éléments de procédure DLE d'une architecture en étoile.....	238
7.2 Éléments de procédure DLE d'une architecture en boucle	264
7.3 Convertisseur parallèle-série et convertisseur série-parallèle	319
7.4 Protocole de gestion DLL	319

Bibliographie.....	330
Figure 1 – Relations des DLSAP, des adresses DLSAP et des adresses DL de groupe	177
Figure 2 – Principe de base du contrôle d'accès au support.....	182
Figure 3 – Interaction des primitives PhS et de la DLE.....	184
Figure 4 – Architecture interne de la couche liaison de données de l'architecture en étoile	186
Figure 5 – Architecture interne de la couche liaison de données de l'architecture en boucle.....	188
Figure 6 – Format de trame MAC commune pour les DLPDU	212
Figure 7 – Structure du champ de contrôle de trame (FC).....	214
Figure 8 – Structure de DLPDU SYN.....	219
Figure 9 – Structure de DLPDU CMP	225
Figure 10 – Structure de DLPDU REQ	226
Figure 11 – Structure de DLPDU CLM	228
Figure 12 – Structure de DLPDU COM.....	230
Figure 13 – Structure de DLPDU DT	231
Figure 14 – Structure de DLPDU RAS.....	232
Figure 15 – Structure des données utilisateur de l'architecture en boucle	233
Figure 16 – Structure de DLPDU LRR.....	234
Figure 17 – Anneau ouvert sous contrôle	236
Figure 18 – Structure de DLPDU LPD	238
Figure 19 – Structure générale de la DLL.....	239
Figure 20 – Transition d'états DLE	240
Figure 21 – Diagramme de transition d'état de la commande CTRC.....	243
Figure 22 – Diagramme de transition d'état de la commande STRC	247
Figure 23 – Diagramme de transition d'état de l'ACM.....	251
Figure 24 – Diagramme de transition d'état de la commande RMC d'envoi et d'arbitrage d'envoi	259
Figure 25 – Diagramme de transition d'état de la commande RMC de réception	262
Figure 26 – Structure générale de la DLL.....	265
Figure 27 – Transition d'états DLE	266
Figure 28 – Diagramme de transition d'état de la commande CTRC.....	269
Figure 29 – Diagramme de transition d'état de la commande STRC	272
Figure 30 – Diagramme de transition d'état de la commande ACM pour le fonctionnement à 100 Mbps	277
Figure 31 – Diagramme de transition d'état de la commande ACM pour le fonctionnement à 1 000 Mbps	278
Figure 32 – Diagramme de transition d'état de la commande RMC pour le fonctionnement à 100 Mbps	298
Figure 33 – Diagramme de transition d'état de la commande RMC pour le fonctionnement à 1 000 Mbps	299
Figure 34 – Diagramme de transition d'état de la commande DLM	322
Figure 35 – Diagramme de transition d'état de la commande DLM	327

Tableau 1 – Composants de la couche liaison de données de l'architecture en étoile	185
Tableau 2 – Composants de la couche liaison de données de l'architecture en boucle	187
Tableau 3 – Variables DLE et valeurs admises de l'architecture en étoile	191
Tableau 4 – Variables observables et leurs plages de valeurs de l'architecture en étoile	193
Tableau 5 – Variables DLE et valeurs admises de l'architecture en boucle	194
Tableau 6 – Variables observables et leurs plages de valeurs de l'architecture en boucle.....	197
Tableau 7 – F-type: Type de DLPDU.....	214
Tableau 8 – Longueur, polynômes et constantes FCS.....	215
Tableau 9 – Paramètre PN: 3 ^{ème} octet.....	220
Tableau 10 – Structure de CW: 4 ^{ème} octet.....	220
Tableau 11 – Paramètre PM	221
Tableau 12 – Paramètre RMSEL.....	221
Tableau 13 – Structure de CW: 4 ^{ème} octet.....	221
Tableau 14 – Paramètre ST: 5 ^{ème} octet.....	222
Tableau 15 – Paramètre Th: 6 ^{ème} , 7 ^{ème} et 8 ^{ème} octets	222
Tableau 16 – Paramètre Tm: 9 ^{ème} et 10 ^{ème} octets.....	222
Tableau 17 – Paramètre Ts: 11 ^{ème} et 12 ^{ème} octet	223
Tableau 18 – Paramètre TI: 13 ^{ème} et 14 ^{ème} octets	223
Tableau 19 – Paramètres LL: 15 ^{ème} à 46 ^{ème} octets	224
Tableau 20 – Paramètre NM	226
Tableau 21 – Paramètre RN	227
Tableau 22 – Paramètre CLM: 4 ^{ème} octet	228
Tableau 23 – Paramètre DT: 3 ^{ème} et 4 ^{ème} octets.....	231
Tableau 24 – Paramètre RAS: 3 ^{ème} et 4 ^{ème} octets	232
Tableau 25 – Format du paramètre PS: 3 ^{ème} octet	234
Tableau 26 – Valeur du paramètre PP	234
Tableau 27 – Valeur des paramètres send-enable-A/-B	235
Tableau 28 – Valeur des paramètres receive-enable-A/-B.....	235
Tableau 29 – Valeur des paramètres forward-enable-A/-B	235
Tableau 30 – Paramètre RN: 4 ^{ème} octet	235
Tableau 31 – Etat de fonctionnement du nœud	236
Tableau 32 – Primitives échangées entre l'utilisateur DLS et la commande CTRC	242
Tableau 33 – Primitives échangées entre la commande CTRC et l'ACM	242
Tableau 34 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC.....	242
Tableau 35 – Table d'états CTRC	244
Tableau 36 – Table de fonctions CTRC.....	245
Tableau 37 – Primitives échangées entre l'utilisateur DLS et la commande STRC	246
Tableau 38 – Primitives échangées entre la commande STRC et l'ACM	246
Tableau 39 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC	246
Tableau 40 – Table d'états STRC	247

Tableau 41 – Table de fonctions STRC.....	248
Tableau 42 – Primitives échangées entre l'ACM et la commande RMC	249
Tableau 43 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC	249
Tableau 44 – Primitives échangées entre l'ACM et la commande CTRC	249
Tableau 45 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTRC	250
Tableau 46 – Primitives échangées entre l'ACM et la commande STRC.....	250
Tableau 47 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC.....	250
Tableau 48 – Table d'états ACM	252
Tableau 49 – Table de fonctions ACM	256
Tableau 50 – Primitives échangées entre l'ACM et la commande RMC	257
Tableau 51 – Primitives échangées entre la commande RMC et le convertisseur parallèle-série/série-parallèle.....	258
Tableau 52 – Primitives échangées entre la commande RMC et la couche Ph	258
Tableau 53 – Paramètres entre la commande RMC et l'ACM	258
Tableau 54 – Paramètres entre la commande RMC et la couche Ph	258
Tableau 55 – Table d'états de la commande RMC d'envoi	260
Tableau 56 – Table d'états de la commande RMC d'arbitrage d'envoi	261
Tableau 57 – Table d'états de la commande RMC de réception	262
Tableau 58 – Table de fonctions RMC	264
Tableau 59 – Primitives échangées entre l'utilisateur DLS et la commande CTRC	268
Tableau 60 – Primitives échangées entre la commande CTRC et l'ACM	268
Tableau 61 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC.....	269
Tableau 62 – Table d'états CTRC	269
Tableau 63 – Table de fonctions CTRC.....	270
Tableau 64 – Primitives échangées entre l'utilisateur DLS et la commande STRC	271
Tableau 65 – Primitives échangées entre la commande STRC et l'ACM	272
Tableau 66 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC	272
Tableau 67 – Table d'états STRC	273
Tableau 68 – Table de fonctions STRC.....	274
Tableau 69 – Primitives échangées entre l'ACM et la commande RMC	275
Tableau 70 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC	275
Tableau 71 – Primitives échangées entre l'ACM et la commande CTRC	275
Tableau 72 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTRC	276
Tableau 73 – Primitives échangées entre l'ACM et la commande STRC.....	276
Tableau 74 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC.....	276
Tableau 75 – Table d'états ACM pour le fonctionnement à 100 Mbps	279
Tableau 76 – Table d'états ACM pour le fonctionnement à 1 000 Mbps.....	286
Tableau 77 – Table de fonctions ACM	295

Tableau 78 – Primitives échangées entre l'ACM et la commande RMC	296
Tableau 79 – Primitives échangées entre la commande RMC et le convertisseur parallèle-série/série-parallèle.....	296
Tableau 80 – Primitives échangées entre la commande RMS et la couche Ph	297
Tableau 81 – Paramètres entre la commande RMC et l'ACM	297
Tableau 82 – Paramètres entre la commande RMC et le convertisseur parallèle-série / série-parallèle, et la couche Ph.....	297
Tableau 83 – Table d'états RCM pour le fonctionnement à 100 Mbps	299
Tableau 84 – Table d'états RCM pour le fonctionnement à 1 000 Mbps	309
Tableau 85 – Table de fonctions RMC	319
Tableau 86 – Primitives échangées entre l'utilisateur DLMS et la commande DLM	320
Tableau 87 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DL et la commande DLM	320
Tableau 88 – Variables de changement d'état relatives aux événements	321
Tableau 89 – Table d'états DLM	322
Tableau 90 – Table de fonctions DLM	324
Tableau 91 – Primitives échangées entre l'utilisateur DLMS et la commande DLM	325
Tableau 92 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DL et la commande DLM	325
Tableau 93 – Variables de changement d'état relatives aux événements	326
Tableau 94 – Table d'états DLM	327
Tableau 95 – Table de fonctions DLM	329

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

**RÉSEAUX DE COMMUNICATION INDUSTRIELS –
SPECIFICATIONS DES BUS DE TERRAIN –****Partie 4-11: Spécification du protocole de la couche liaison de données –
Éléments de type 11**

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 elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
- 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.

L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type.

NOTE Les combinaisons de types de protocole sont spécifiées dans la CEI 61784-1 et la CEI 61784-2.

La Norme internationale CEI 61158-4-11 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition parue en 2010. Cette édition constitue une révision technique.

Les modifications majeures par rapport à l'édition précédente sont l'ajout d'une architecture en (anneau) boucle. Plus de détails:

- les Paragraphes 4.6.1, 4.6.4 et 5.4.6, l'Article 6 et 7.2 pour l'architecture en boucle ont été modifiés pour couvrir les spécifications supplémentaires relatives au débit de données plus élevé dans l'architecture en boucle;
- les spécifications relatives à l'architecture en étoile existante sont conservées en l'état.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65C/762/FDIS	65C/772/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, peut être consultée sur le site web de la CEI.

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.

INTRODUCTION

La présente partie de la CEI 61158 appartient à la série de normes visant à faciliter l'interconnexion des composants du système d'automatisation. Elle est liée aux autres normes de la série telle que définie par le modèle de référence de bus de terrain « à trois couches » décrit dans la CEI 61158-1.

Le protocole de liaison de données fournit le service de liaison de données en utilisant les services disponibles à partir de la couche physique. La présente norme a pour principal objet de préciser un ensemble de règles de communication, exprimées sous la forme de procédures devant être réalisées par des entités de liaison de données homologues (DLE) au moment de la communication. Ces règles de communication ont pour vocation d'assurer une base de développement solide pour atteindre plusieurs objectifs:

- a) guider les implémenteurs et les concepteurs;
- b) réaliser les essais et acquérir l'équipement;
- c) dans le cadre d'un accord d'intégration des systèmes dans l'environnement de systèmes ouverts;
- d) dans le cadre d'une meilleure compréhension des communications prioritaires (ou à contrainte de temps) au sein de l'OSI.

La présente norme porte en particulier sur la communication et l'interfonctionnement des capteurs, des effecteurs et d'autres appareils d'automatisation. Grâce à la présente norme associée à d'autres normes des modèles de référence OSI ou de bus de terrain, des systèmes par ailleurs incompatibles peuvent fonctionner ensemble, quelle que soit leur combinaison.

L'attention est attirée sur le fait que l'utilisation de certains types de protocole associés est limitée par leurs détenteurs de droit à la propriété intellectuelle. Dans tous les cas, l'engagement visant à limiter l'abandon des droits à la propriété intellectuelle prévus par les détenteurs de ces droits permet d'utiliser un type de protocole de couche liaison de données particulier avec les protocoles de couche physique et de couche d'application dans les combinaisons de type, comme spécifié explicitement dans les parties relatives au profil. L'utilisation de différents types de protocole dans d'autres combinaisons peut impliquer d'obtenir l'autorisation auprès de leurs détenteurs de droit de propriété intellectuelle respectifs.

La commission électrotechnique internationale (CEI) attire l'attention sur le fait qu'il est déclaré que la conformité avec le présent document peut impliquer l'utilisation de brevets relatifs aux éléments de Type 11 et éventuellement aux autres types présentés en 4.2, 4.4, 4.5, 5.4, 6.2 à 6.10, 7.1 et 7.2 comme suit:

AU 2007320662(B2) [TO] Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system

ZL 200780042584.7 [TO] Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system

JP 4991254(B2) [TO] Control method for double-ring network, initializing method for double-ring network, transmission station of double-ring network, restructuring method for abnormality occurrence of double-ring network, network system, control method for network system, transmission station, and program of transmission station

KR 101149837(B1) [TO] Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system

RU 2420899(C2) [TO] Double-ring network system and method of controlling communication in said network, transmission station and programme for transmission stations

US 8411559(B2) [TO] Double ring network system and communication control method thereof, and transmission station, and program for transmission stations

EP 2093942(A1) [TO] Double ring network system, communication control method thereof, transmission station, and communication control program of double ring network system

US 6711131(B1) [TO] Data transmitting apparatus, network interface apparatus, and data transmitting system

JP 3461954(B2) [TO] Data transmitter

La CEI ne prend pas position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Les détenteurs de ces droits de propriété ont donné l'assurance à la CEI qu'il consentent à négocier des licences avec des demandeurs du monde entier, soit sans frais soit à des termes et conditions raisonnables et non discriminatoires. À ce propos, la déclaration des détenteurs des droits de propriété est enregistrée à la CEI. Des informations peuvent être demandées à:

[TO] Toshiba Corporation
1, Toshiba -cho
Fuchu-shi
Tokyo 183-8511, Japan
Attention: Intellectual Property Rights Section

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété autres que ceux qui ont été mentionnés ci-dessus. La CEI ne saurait être tenue pour responsable de l'identification de ces droits de propriété en tout ou partie.

L'ISO (www.iso.org/patents) et la CEI (<http://patents.iec.ch>) maintiennent des bases de données, consultables en ligne, des droits de propriété pertinents à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les droits de propriété.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPECIFICATIONS DES BUS DE TERRAIN –

Partie 4-11: Spécification du protocole de la couche liaison de données – Eléments de type 11

1 Domaine d'application

1.1 Généralités

La couche liaison de données assure la communication de messagerie prioritaire entre les appareils d'un environnement d'automatisation.

Ce protocole offre des opportunités de communication à toutes les entités de liaison de données participantes

- a) de manière cyclique et synchrone, selon une planification préalablement établie, et
- b) de manière cyclique ou acyclique et asynchrone, comme demandé par chaque cycle de chacune de ces entités de liaison de données.

Par conséquent, ce protocole peut se caractériser comme assurant un accès cyclique et acyclique asynchrone, mais avec un redémarrage synchrone de chaque cycle.

1.2 Spécifications

La présente norme spécifie

- a) les procédures de transfert opportun des données et des informations de commande entre une entité utilisateur de liaison de données et une entité utilisateur homologue, et parmi les entités de liaison de données formant le fournisseur de service de liaison de données distribué;
- b) les procédures permettant à toutes les entités DL participantes de communiquer, de manière séquentielle et cyclique dans le cadre d'un transfert déterministe et synchronisé, à intervalles cycliques d'une milliseconde au maximum;
- c) les procédures permettant à la transmission de données prioritaires de communiquer avec une transmission de données non prioritaires, sans préjudice pour la première;
- d) les procédures permettant à la transmission de données prioritaires de communiquer de manière cyclique et acyclique, avec un accès en priorité;
- e) les procédures permettant d'assurer la communication en fonction du contrôle d'accès au support ISO/CEI 8802-3, avec des dispositions relatives aux nœuds à ajouter ou retirer pendant le fonctionnement normal;
- f) la structure des DLPDU de bus de terrain utilisée par le protocole de la présente norme pour le transfert des données et des informations de commande, et leur représentation sous forme d'unités de données d'interface physique.

1.3 Procédures

Les procédures sont définies en termes

- a) d'interactions entre les entités DL (DLE) homologues par l'échange de DLPDU de bus de terrain;
- b) d'interactions entre un fournisseur de service 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 service Ph au sein du même système par l'échange de primitives de service Ph.

1.4 Applicabilité

Ces procédures s'appliquent aux instances de communication entre des systèmes qui prennent en charge des services de communications prioritaires dans la couche liaison de données des modèles de référence OSI ou de bus de terrain, et qui peuvent être connectés dans un environnement d'interconnexion de systèmes ouverts.

Les profils sont un moyen simple à plusieurs attributs de récapituler les capacités d'une mise en œuvre, et donc son applicabilité en fonction des différents besoins de communications prioritaires.

1.5 Conformité

La présente norme spécifie également les exigences de conformité relatives aux systèmes mettant en œuvre ces procédures. La présente norme ne comporte aucun essai visant à démontrer la conformité à ces exigences.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. 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 éventuels amendements).

NOTE Toutes les parties de la série CEI 61158, ainsi que la CEI 61784-1 et la CEI 61784-2 font l'objet d'une maintenance simultanée. Les références croisées à ces documents dans le texte se rapportent par conséquent aux éditions datées dans la présente liste de références normatives.

CEI 61158-3-11:2007, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-11: Définition des services de la couche liaison de données – Eléments de type 11*

IEC 61158-5-11:2007, *Industrial communication networks – Fieldbus specifications – Part 5-11: Application layer service definition – Type 11 elements* (disponible en anglais seulement)

ISO/CEI 7498-1, *Technologies de l'information – Modèle de référence de base pour l'interconnexion de systèmes ouverts (OSI): 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/IEC 8802-3:2000, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications* (disponible en anglais seulement)

ISO/CEI 10731, *Technologies de l'information – interconnexion de systèmes ouverts – Modèle de référence de base – Conventions pour la définition des services OSI*

3 Termes, définitions, symboles et abréviations

Pour les besoins du présent document, 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, et utilise les termes suivants.

3.1.1	adresse DL appelée	[ISO/CEI 7498-3]
3.1.2	adresse DL appelante	[ISO/CEI 7498-3]
3.1.3	connexion centralisée à plusieurs extrémités	[ISO/CEI 7498-1]
3.1.4	entités (N) correspondantes entités DL correspondantes (N=2) entités Ph correspondantes (N=1)	[ISO/CEI 7498-1]
3.1.5	démultiplexage	[ISO/CEI 7498-1]
3.1.6	adresse DL	[ISO/CEI 7498-3]
3.1.7	mapping d'adresse DL	[ISO/CEI 7498-1]
3.1.8	connexion DL	[ISO/CEI 7498-1]
3.1.9	extrémité de connexion DL	[ISO/CEI 7498-1]
3.1.10	identifiant d'extrémité de connexion DL	[ISO/CEI 7498-1]
3.1.11	transmission en mode de connexion DL	[ISO/CEI 7498-1]
3.1.12	transmission en mode sans connexion DL	[ISO/CEI 7498-1]
3.1.13	collecteur de données DL	[ISO/CEI 7498-1]
3.1.14	source de données DL	[ISO/CEI 7498-1]
3.1.15	transmission duplex DL	[ISO/CEI 7498-1]
3.1.16	installation DL	[ISO/CEI 7498-1]
3.1.17	vue locale DL	[ISO/CEI 7498-3]
3.1.18	nom DL	[ISO/CEI 7498-3]
3.1.19	protocole DL	[ISO/CEI 7498-1]
3.1.20	identifiant de connexion de protocole DL	[ISO/CEI 7498-1]
3.1.21	informations de commande de protocole DL	[ISO/CEI 7498-1]
3.1.22	unité de données de protocole DL	[ISO/CEI 7498-1]
3.1.23	identifiant de version de protocole DL	[ISO/CEI 7498-1]
3.1.24	relais DL	[ISO/CEI 7498-1]
3.1.25	identifiant de connexion de service DL	[ISO/CEI 7498-1]
3.1.26	unité de données de service DL	[ISO/CEI 7498-1]

3.1.27	transmission simplex DL	[ISO/CEI 7498-1]
3.1.28	sous-système DL	[ISO/CEI 7498-1]
3.1.29	données utilisateur DL	[ISO/CEI 7498-1]
3.1.30	contrôle de flux	[ISO/CEI 7498-1]
3.1.31	gestion de couche	[ISO/CEI 7498-1]
3.1.32	multiplexage	[ISO/CEI 7498-3]
3.1.33	autorité (d'adressage) de dénomination	[ISO/CEI 7498-3]
3.1.34	domaine (d'adressage) de dénomination	[ISO/CEI 7498-3]
3.1.35	sous-domaine (d'adressage) de dénomination	[ISO/CEI 7498-3]
3.1.36	entité (N) entité DL entité Ph	[ISO/CEI 7498-1]
3.1.37	unité de données d'interface (N) unité de données de service DL (N=2) unité de données d'interface Ph (N=1)	[ISO/CEI 7498-1]
3.1.38	couche (N) couche DL (N=2) couche Ph (N=1)	[ISO/CEI 7498-1]
3.1.39	service-(N) service DL (N=2) service Ph (N=1)	[ISO/CEI 7498-1]
3.1.40	point d'accès au service (N) point d'accès au service DL (N=2) point d'accès au service Ph (N=1)	[ISO/CEI 7498-1]
3.1.41	adresse de point d'accès au service (N) adresse de point d'accès au service DL (N=2) adresse de point d'accès au service Ph (N=1)	[ISO/CEI 7498-1]
3.1.42	entités homologues	[ISO/CEI 7498-1]
3.1.43	informations de contrôle d'interface Ph	[ISO/CEI 7498-1]
3.1.44	données d'interface Ph	[ISO/CEI 7498-1]
3.1.45	nom de la primitive	[ISO/CEI 7498-3]
3.1.46	réassemblage	[ISO/CEI 7498-1]
3.1.47	recombinaison	[ISO/CEI 7498-1]
3.1.48	réinitialisation	[ISO/CEI 7498-1]
3.1.49	adresse DL de réponse	[ISO/CEI 7498-3]
3.1.50	acheminement	[ISO/CEI 7498-1]
3.1.51	segmentation	[ISO/CEI 7498-1]
3.1.52	séquencement	[ISO/CEI 7498-1]

3.1.53	éclatement	[ISO/CEI 7498-1]
3.1.54	nom synonyme	[ISO/CEI 7498-3]
3.1.55	gestion-système	[ISO/CEI 7498-1]

3.2 Termes et définitions de convention de service

La présente norme utilise également les termes définis dans l'ISO/CEI 10731 tels qu'ils s'appliquent à la couche liaison de données:

3.2.1	accepteur
3.2.2	service asymétrique
3.2.3	confirmation (primitive); requestor.deliver (primitive)
3.2.4	deliver (primitive);
3.2.5	installation DL confirmée
3.2.6	installation DL
3.2.7	vue locale DL
3.2.8	installation DL obligatoire
3.2.9	installation DL non confirmée
3.2.10	installation DL lancée par le fournisseur
3.2.11	installation DL facultative du fournisseur
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	installation DL facultative de l'utilisateur
3.2.16	indication (primitive) acceptor.deliver (primitive)
3.2.17	multihomologue
3.2.18	request (primitive); requestor.submit (primitive)
3.2.19	demandeur
3.2.20	réponse (primitive); acceptor.submit (primitive)
3.2.21	submit (primitive)
3.2.22	service symétrique

3.3 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

3.3.1

mémoire commune

mémoire commune virtuelle du bus de terrain de Type 11, partagée par les nœuds de bus de terrain de Type 11 participant, et principalement utilisée pour les communications en temps réel par le service de données cycliques prioritaires

3.3.2

DLPDU de données

DLPDU acheminant une DLSU d'un utilisateur DLS local vers un utilisateur DLS éloigné

3.3.3

adresse DLCEP

adresse DL qui désigne

- a) une extrémité de connexion DL homologue, ou
- b) une extrémité de connexion DL d'éditeur multihomologue et, implicitement, l'ensemble correspondant d'extrémités de connexion DL d'abonné dans lequel chaque extrémité de connexion DL existe dans un DLSAP distinct et est associée à une adresse DLSAP distincte correspondante

3.3.4

segment DL,

liaison,

liaison locale

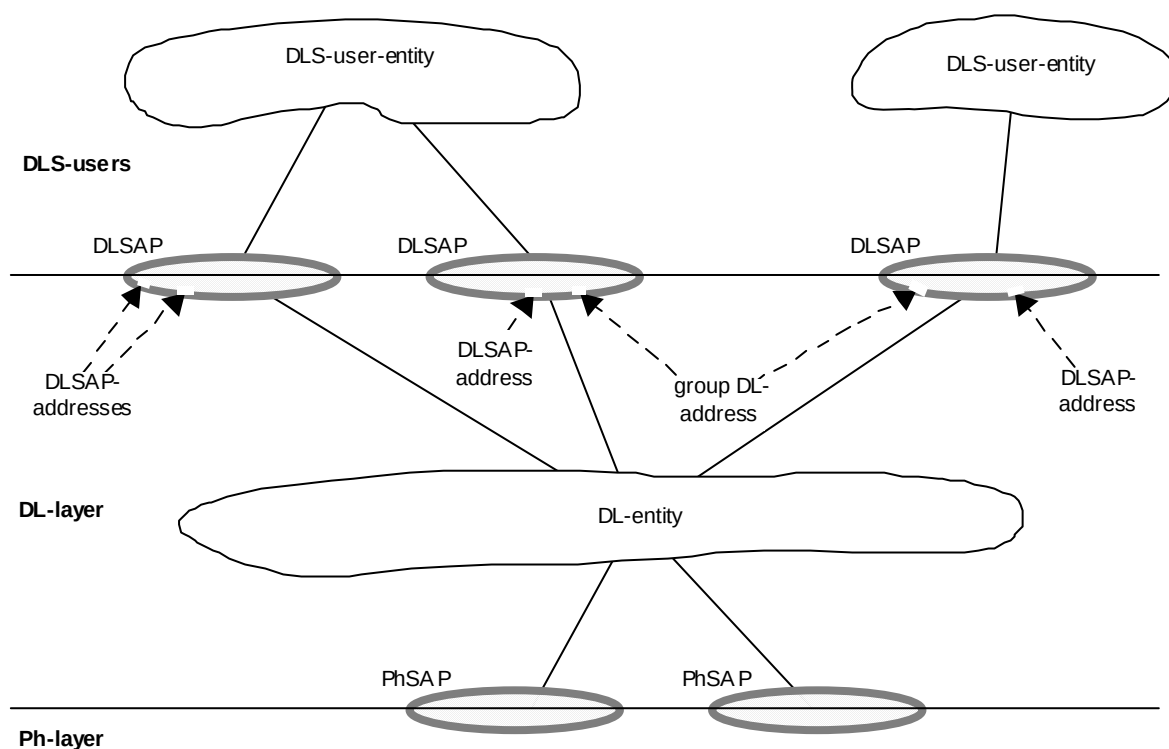
sous-réseau DL dans lequel l'une des DLE connectées peut communiquer directement sans l'intervention d'un relais DL, à chaque fois que toutes les DLE participant à une instance de communication sont simultanément attentives au sous-réseau DL pendant la/les période(s) de tentative de communication

3.3.5

DLSAP

point distinctif au niveau duquel des services DL sont fournis par une seule entité DL à une seule entité de couche supérieure

Note 1 à l'article: Cette définition, déduite de l'ISO/CEI 7498-1, est reprise ici pour faciliter la compréhension de la distinction critique entre les DLSAP et leurs adresses DL. (Voir Figure 1)



Légende

Anglais	Français
DLS-user-entity	Entité utilisateur DLS
DLS-users	Utilisateurs DLS
DLSAP-addresses	Adresses DLSAP
DLSAP-address	Adresse DLSAP
group DL-address	Adresse DL de groupe
DL-layer	Couche DL
DL-entity	Entité DL
Ph-layer	Couche Ph

NOTE 1 Les DLSAP et PhSAP sont décrits comme des éléments ovales à cheval sur la limite entre deux couches adjacentes.

NOTE 2 Les adresses DL sont décrites comme désignant de petits écarts (points d'accès) dans la partie DLL d'un DLSAP.

NOTE 3 Une seule entité DL peut comporter plusieurs adresses DLSAP et adresses DL de groupe associées à un seul DLSAP.

Figure 1 – Relations des DLSAP, des adresses DLSAP et des adresses DL de groupe

3.3.6

DLSAP

point distinctif au niveau duquel des services DL sont fournis par une seule entité DL à une seule entité de couche supérieure

Note 1 à l'article: Cette définition, déduite de l'ISO/CEI 7498-1, est reprise ici pour faciliter la compréhension de la distinction critique entre les DLSAP et leurs adresses DL. (Voir Figure 1).

3.3.7

DL(SAP)-address (adresse DL(SAP))

adresse DLSAP individuelle désignant le DLSAP d'un utilisateur DLS unique ou adresse DL de groupe désignant plusieurs DLSAP, chacun d'eux appartenant à un seul utilisateur DLS

Note 1 à l'article: Cette terminologie a été choisie car l'ISO/CEI 7498-3 ne permet pas d'utiliser le terme adresse DLSAP pour désigner plusieurs DLSAP uniques d'un seul utilisateur DLS.

3.3.8

adresse DLSAP (individuelle)

adresse DL qui désigne un seul DLSAP dans la liaison étendue

Note 1 à l'article: Une seule entité DL peut comporter plusieurs adresses DLSAP associées à un seul DLSAP.

3.3.9

liaison étendue

sous-réseau DL composé d'un nombre maximal de liaisons interconnectées par des relais DL et partageant un espace de noms DL unique (adresse DL) dans lequel toutes les entités DL connectées peuvent communiquer, soit directement soit à l'aide d'une ou de plusieurs de ces entités relais DL

Note 1 à l'article: Une liaison étendue peut être composée d'une seule liaison.

3.3.10

erreur FCS

erreur qui se produit lorsque la valeur de la séquence de contrôle de trame calculée après réception de tous les octets d'une DLPDU ne correspond pas à la valeur résiduelle prévue

3.3.11

trame

synonyme obsolète de DLPDU

3.3.12

adresse DL de groupe

adresse DL qui désigne potentiellement plusieurs DLSAP dans la liaison étendue

Note 1 à l'article: Une entité DL unique peut comporter plusieurs adresses DL de groupe avec un seul DLSAP.

Note 2 à l'article: Une entité DL unique peut également comporter une seule adresse DL de groupe avec plusieurs DLSAP.

3.3.13

données cycliques à grande vitesse

données acheminées au moyen d'une transmission de données cycliques à grande vitesse

3.3.14

transmission de données cycliques à grande vitesse

priorité la plus élevée du service de données cycliques prioritaires

3.3.15

jeton implicite

mécanisme régissant les droits de transmission

Note 1 à l'article: Aucun message de jeton réel n'est transmis sur le support. Chaque nœud garde une trace du nœud dont il suppose détenir les droits de transmission. Les droits de transmission sont transmis d'un nœud à l'autre en conservant le dernier nœud qui les a transmis. Un intervalle de temps est utilisé afin d'ignorer un nœud manquant dans la rotation.

3.3.16

données cycliques à basse vitesse

données acheminées au moyen d'une transmission de données cyclique à basse vitesse

3.3.17**transmission de données cyclique à basse vitesse**

priorité la plus basse du service de données cycliques prioritaires

3.3.18**données cycliques à moyenne vitesse**

données acheminées au moyen d'une transmission de données cycliques à moyenne vitesse

3.3.19**transmission de données cycliques à moyenne vitesse**

priorité moyenne du service de données cycliques prioritaires

3.3.20**DLC multihomologue**

connexion DL centralisée à plusieurs extrémités assurant la transmission duplex DL entre un utilisateur DLS distingué (appelé éditeur ou utilisateur DLS de publication) et un ensemble d'utilisateurs DLS homologues mais non distingués (appelés collectivement abonnés ou utilisateurs DLS d'abonnement)

Note 1 à l'article: L'utilisateur DLS de publication peut envoyer des éléments aux utilisateurs DLS d'abonnement (mais pas individuellement) et inversement (mais pas l'un à l'autre).

3.3.21**connexion multipoint**

connexion entre un nœud et plusieurs nœuds

Note 1 à l'article: Les connexions multipoint permettent de transférer des données d'un éditeur vers plusieurs nœuds d'abonnés.

3.3.22**nœud**

entité DL unique telle qu'elle se présente sur une liaison locale

3.3.23**adresse DL du nœud**

adresse DL qui désigne l'entité DL (unique) associée à un seul nœud d'une liaison locale particulière

3.3.24**id de nœud**

identifiant principal à deux octets de la DLE de la liaison locale, dont les valeurs sont contraintes

Note 1 à l'article: Une valeur admise est comprise entre 1 et 255. Une valeur nulle est spécifiquement utilisée pour le nœud SYN, qui émet une trame SYN.

3.3.25**utilisateur DLS destinataire**

utilisateur du service DL auquel sont destinées les données utilisateur DL

Note 1 à l'article: Un utilisateur de service DL peut être à la fois un utilisateur DLS expéditeur et destinataire.

3.3.26**utilisateur DLS expéditeur**

utilisateur du service DL à la source des données utilisateur DL

3.3.27

intervalle de temps

durée comprise entre 512 bits et 4 096 bits du symbole de signalisation physique du débit de données de 100 Mbps et 1 000 Mbps respectivement spécifié à l'Article 29 de l'ISO/CEI 8802-3:2000

3.3.28

service de données de messagerie sporadiques

transfert de message aperiodique se produisant de manière sporadique lorsque l'utilisateur DLS demande le transfert d'un ou de plusieurs messages et qu'une trame de message Ethernet ISO/CEI 8802-3 normale est transférée dans le cadre de ce transfert de message

3.3.29

nœud SYN

nœud qui transmet une trame SYN

3.3.30

service de données cycliques prioritaires

transfert de données cycliques à trois niveaux de transmission simultanée de données, dont le niveau de transmission de données est fonction de la priorité des données et de la période de transmission des données pour la remise en temps réel, et dont la période de transmission de données et le volume total des données peuvent être spécifiés dans la phase de conception et selon les besoins de l'application

3.3.31

jeton

droit de transmission sur la liaison locale

3.4 Symboles et abréviations

ACM	Access control machine (Machine de contrôle d'accès)
CLM	Claim frame (Trame d'appel)
CMP	Transmission complete frame (Trame complète d'émission)
CTRC	Cyclic-transmission TX/RX control (Commande TX/RX d'émission cyclique)
COM	Command frame (Trame de commande)
DT	Cyclic data frame (Trame de données cycliques)
CW	Control word (Mot de commande) (paramètre de DLPDU SYN)
DT-CMP	DT avec trame complète d'émission
FC	Frame control field (Champ de contrôle de trame)
LL	Live list (Liste opérationnelle) (paramètre de DLPDU SYN)
Mbps	Million de bits par seconde
PhIDU	Ph-interface-data-unit (unité de données d'interface Ph)
PM	Periodic mode (Mode périodique) (paramètre de DLPDU SYN)

PN	Transmission permits node number (Numéro de nœud d'autorisation d'émission) (paramètre de DLPDU SYN)
Pri	Champ de priorité
RAS	Trame RAS
REC	Trame de demande dans l'anneau
RMC	Redundancy medium control (Commande de support de redondance)
RMSEL	Redundant medium selection (Sélection du support de redondance)
RN	Recipient node number (Numéro de nœud du destinataire)
SN	Source node number field (Champ du numéro de nœud source)
ST	Slot time (Intervalle de temps) (paramètre de DLPDU SYN)
SYN	Trame de synchronisation
Th	Période de transmission à grande vitesse (en tant que période de transmission)
Tm	Période de transmission à moyenne vitesse (en tant que période de transmission)
Ts	Période de transmission à vitesse sporadique (en tant que période de transmission)
TI	Période de transmission à basse vitesse (en tant que période de transmission)

4 Présentation du protocole DL

4.1 Généralités

La présente norme remplit les objectifs d'automatisation industrielle du marché visant à assurer le transfert de données déterministes et prioritaires prévisible et à en fournir les moyens, permettant la coexistence avec le transfert de données non prioritaires sur les supports de communication de la série ISO/CEI 8802-3, pour la prise en charge de la coopération et de la synchronisation entre les processus d'automatisation des appareils de terrain dans un système d'application en temps réel. Le terme « prioritaire » est utilisé pour indiquer la présence d'une fenêtre temporelle, dans laquelle une ou plusieurs actions spécifiées sont à réaliser selon un certain niveau de certitude.

4.1.1 Champ des applications

Dans les systèmes de contrôle industriel, plusieurs sortes d'appareils de terrain (lecteurs, capteurs et actionneurs, automates programmables, systèmes à commande distribuée et interface homme/machine, par exemple) sont à connecter aux réseaux de commande. Les données de contrôle de processus et les données d'état sont transférées parmi ces appareils de terrain dans le système, les communications entre ces appareils impliquant la simplicité de la programmation d'application et l'exécution selon un temps de réponse adapté. Dans la plupart des systèmes d'automatisation industrielle (l'alimentation, l'eau, les eaux usées, le papier et l'acier, y compris les laminoirs, par exemple), le réseau de commande est tenu d'assurer une capacité de réponse prioritaire à ses applications, comme l'exige l'ISO/TR 13283 pour les architectures de communication prioritaire.

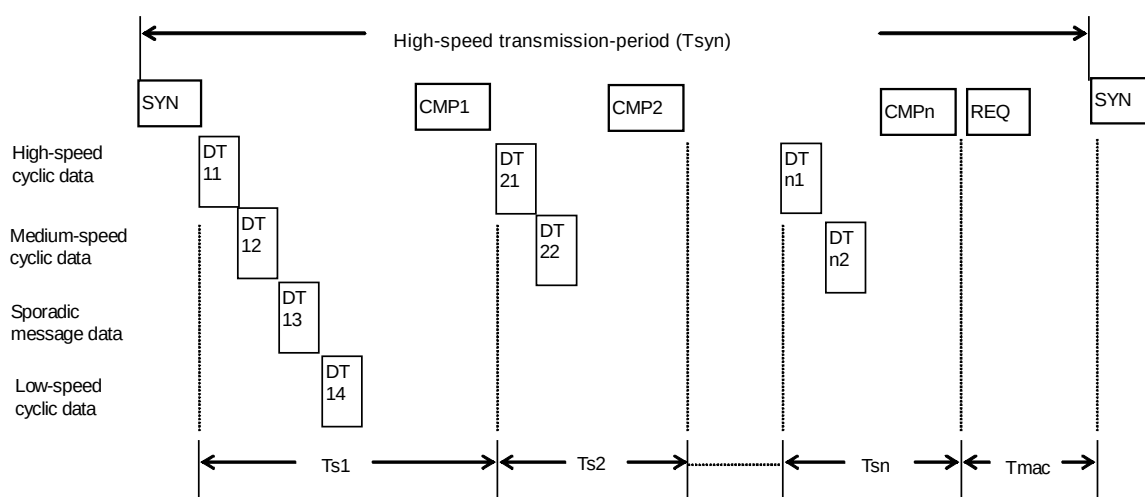
La production de l'usine peut être compromise en raison d'erreurs se produisant dans le système de commande si le réseau n'assure pas un temps de réponse prioritaire. Par conséquent, les caractéristiques suivantes sont requises dans le cas d'un réseau de commande prioritaire:

- temps de réponse déterministe entre les nœuds de l'appareil de commande;
- aptitude à partager les données de processus en continu dans le système de commande.

Ce protocole est applicable à ce type d'environnement d'automatisation industrielle, dans lequel les communications prioritaires sont principalement requises. Le terme « prioritaire » est utilisé pour indiquer la présence d'une fenêtre temporelle, dans laquelle une ou plusieurs actions spécifiées sont à réaliser selon un certain niveau de certitude. Si les actions spécifiées ne sont pas réalisées dans la fenêtre temporelle, les applications demandant les actions risquent de connaître une défaillance, avec les risques que cela comporte pour les équipements, les usines et éventuellement la vie humaine.

4.2 Présentation du contrôle d'accès au support

Le bus de terrain de Type 11 dispose d'un contrôle d'accès au support déterministe afin d'éviter toute collision lorsqu'un certain nombre de nœuds envoie des trames de données simultanément, et d'offrir la possibilité d'envoyer des données à chaque nœud dans un ordre séquentiel et dans une période préalablement déterminée. La Figure 2 illustre le principe de base du contrôle d'accès au support du bus de terrain de Type 11.



Légende

Anglais	Français
High-speed transmission period	période de transmission à grande vitesse
High-speed cyclic data	Données cycliques à grande vitesse
Medium-speed cyclic data	Données cycliques à moyenne vitesse
Low-speed cyclic data	Données cycliques à basse vitesse
Sporadic message data	Données de message sporadiques

Figure 2 – Principe de base du contrôle d'accès au support

Au début de chaque période de transmission à grande vitesse (T_{syn}), la trame SYN est diffusée à tous les nœuds. Lors de la réception de la trame SYN, le nœud portant le numéro séquentiel 1 commence à envoyer ses trames de données, puis diffuse sa trame CMP pour indiquer la fin de la transmission de ses trames de données. Le $N^{ème}$ nœud peut envoyer ses trames de données après avoir reçu la trame CMP du nœud ($N-1$). Lorsque tous les nœuds ont envoyé leurs trames de données, la période de sollicitation de nouveaux nœuds commence. La trame REQ est utilisée pour un nouveau nœud demandant l'intégration du

réseau de bus de terrain de Type 11. Le numéro séquentiel est attribué à un nouveau nœud lorsque l'approbation de jonction est accordée.

Chaque nœud peut disposer du droit de transmission pendant une durée prédéfinie et doit envoyer sa trame CMP afin de transférer ce droit au nœud suivant dans la durée prédéfinie. Les données à envoyer et celles à conserver sont déterminées par priorité.

La transmission comprend les données cycliques prioritaires et la transmission de message Ethernet sporadique. La transmission de données cycliques est réalisée à vitesse élevée, moyenne ou basse. Chaque nœud envoie les trames de données cycliques à grande vitesse dès qu'il obtient le droit de transmission. Les données de priorité basse, c'est-à-dire les données cycliques à moyenne vitesse, les données de message Ethernet sporadique et les données cycliques à basse vitesse respectivement, sont envoyées ou ne le sont pas en fonction des circonstances. La durée de maintien du droit de transmission de chaque nœud est déterminée par les paramètres des périodes de transmission de données cycliques à grande vitesse, des données cycliques à moyenne vitesse, des données de message Ethernet sporadique et des données cycliques à basse vitesse, ainsi que par le volume des données de transmission pour chaque nœud. Après l'envoi de toutes les données cycliques à grande vitesse, le nœud envoie les données cycliques à moyenne vitesse. Si la durée de maintien du droit de transmission se termine pendant l'envoi des données cycliques à moyenne vitesse, la transmission est interrompue et la trame CMP est envoyée. Le n^{ème} nœud obtient de nouveau le droit de transmission au cours de la période de transmission des données à grande vitesse suivante, pendant laquelle les données cycliques à grande vitesse et le reste des données cycliques à moyenne vitesse précédentes sont envoyés. Tmac est la période pendant laquelle un nouveau nœud qui envoie une trame REQ entre dans le réseau.

Deux types d'architecture (en étoile et en boucle) sont spécifiés en 4.4. Plus particulièrement dans l'architecture en boucle, l'ordre de priorité des données de message Ethernet sporadique et des données cycliques à basse vitesse change en fonction de l'aptitude à envoyer les données cycliques à basse vitesse dans le fonctionnement à 100 Mbps et 1 000 Mbps. L'ordre de priorité des données de message Ethernet sporadique est le plus bas, et les données cycliques à basse vitesse ne sont pas en mesure de fonctionner à 100 Mbps.

4.3 Service supposé provenir de PhL

4.3.1 Généralités

Le Paragraphe 4.3 décrit le service physique (PhS) supposé et les contraintes liées à son utilisation par la DLE de Type 11. Le service physique est censé fournir les primitives de service suivantes spécifiées par l'Article 6 de l'ISO/CEI 8802-3:2000.

4.3.2 Primitives supposées de PhS

Le PhS est censé fournir les deux catégories suivantes de primitives au protocole DL de Type 11.

- Primitives de service pour la transmission et la réception de trames entre d'autres DLE homologues.
- Primitives de service donnant les informations nécessaires à la DLE locale pour exécuter les fonctions d'accès au support.

Les primitives supposées de PhS sont regroupées en deux catégories:

- a) Transfert de données à toutes les DLE homologues
 - demande PLS_DATA
 - indication PLS_DATA

b) Gestion d'accès au support par la DLE locale

- indication PLS_CARRIER
- indication PLS_SIGNAL
- indication PLS_DATA_VALID

L'interaction des primitives PhS et de la DLE est présentée à la Figure 3.

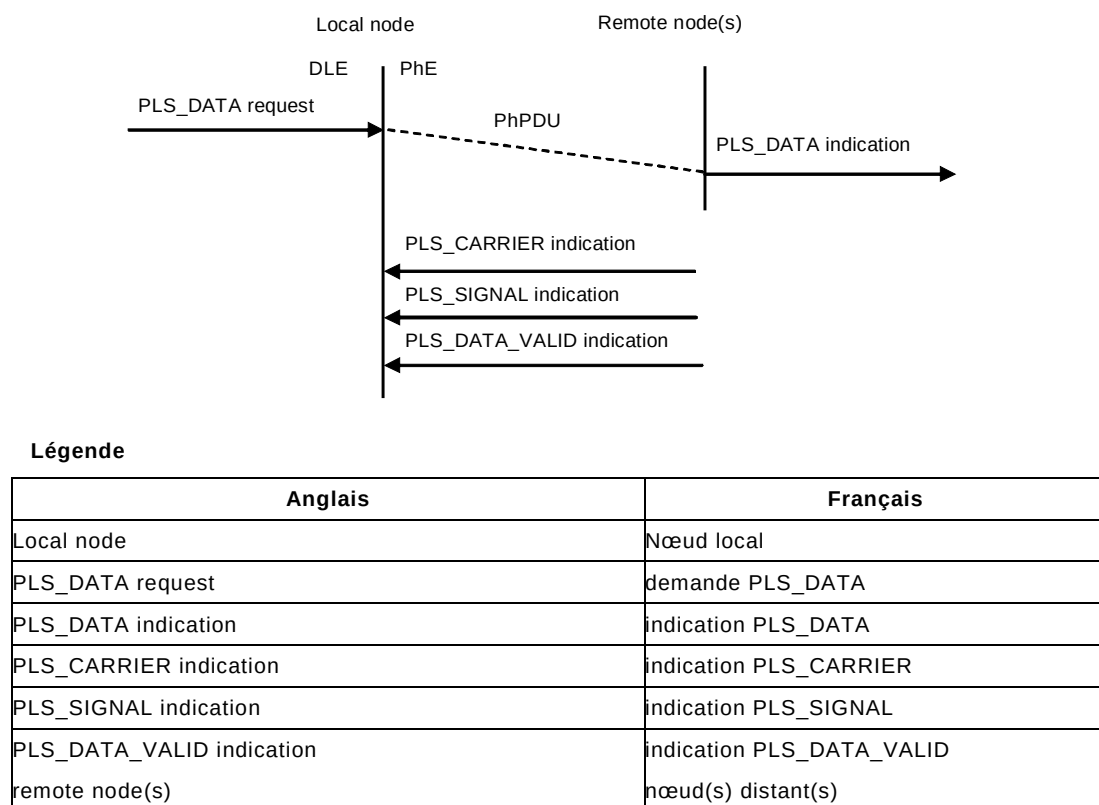


Figure 3 – Interaction des primitives PhS et de la DLE

4.4 Architecture DLL

4.4.1 Présentation

La DLL de bus de terrain de Type 11 est modélisée sous la forme d'une combinaison des composants de commande de la machine de contrôle d'accès (ACM), Commande TX/RX d'émission cyclique (CTRC), Commande TX/RX sporadique (STRC), Commande de support de redondance (RMC), Convertisseur parallèle-série, Convertisseur série-parallèle et Interface de gestion DLL.

La Commande de la machine de contrôle d'accès étant le composant de commande principal, elle assure la fonction de contrôle d'accès au support déterministe en coopération avec la Commande TX/RX d'émission cyclique, la Commande TX/RX sporadique et la Commande de support de redondance, afin d'assurer une prise en charge fiable et efficace des services de transfert de données en mode de connexion de niveau supérieur et en mode sans connexion. De manière spécifique, la machine de contrôle d'accès est essentiellement chargée

- a) de s'assurer que le nœud local détecte et utilise intégralement sa période d'accès attribuée;
- b) de s'assurer que le nœud local ne gêne pas les transmissions des autres nœuds, particulièrement le nœud qui transmet la trame SYNC;

- c) de détecter toute interruption du réseau et de lancer la transmission de la trame SYN pour la restauration du réseau après la durée prescrite au cours de laquelle la trame SYN n'est pas entendue;
- d) d'assurer l'ajout et le retrait d'un nouveau nœud au/du réseau.

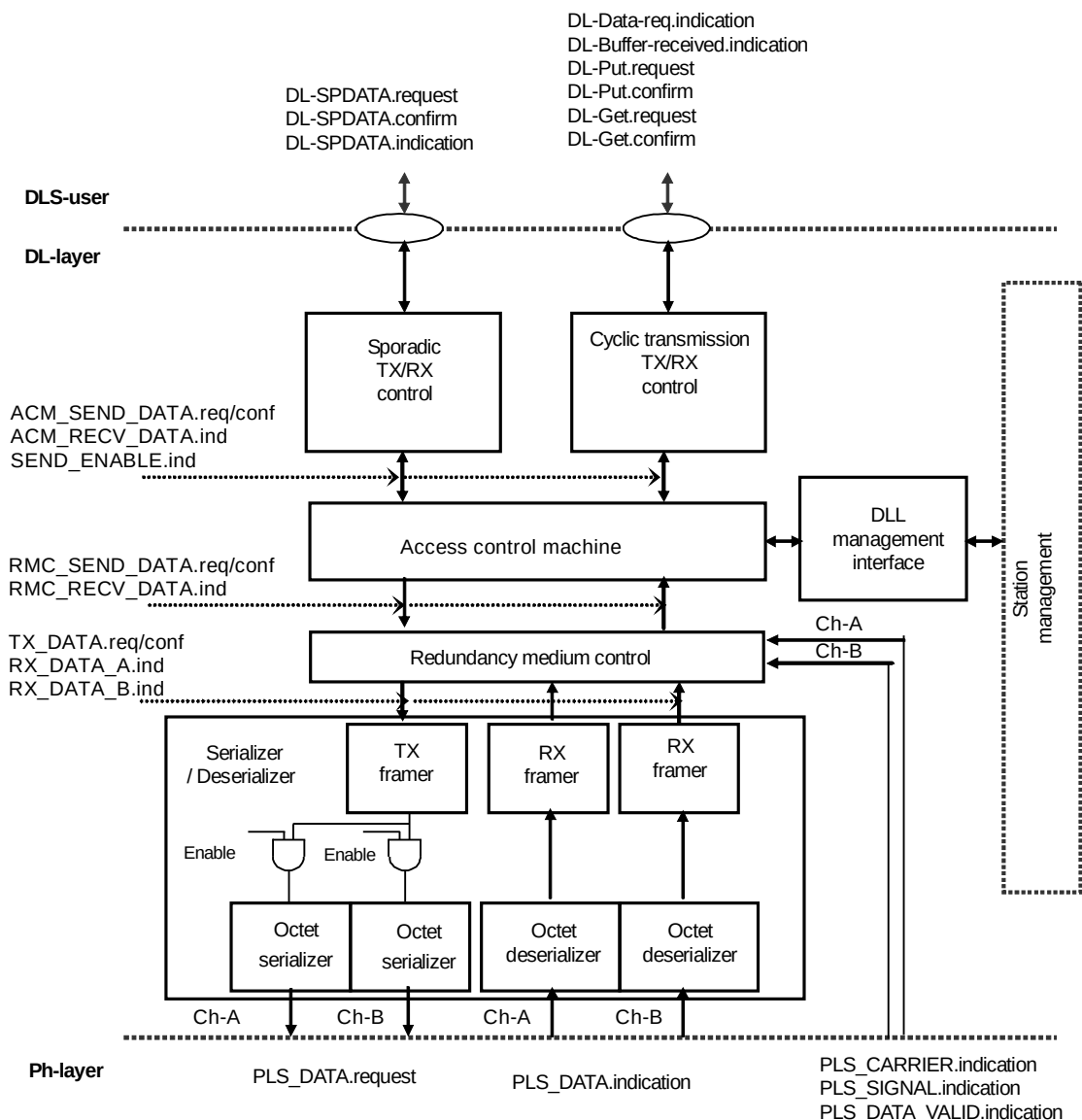
L'interface de gestion DLL offre des fonctions de gestion DLL. La structure PhL et les délimiteurs sont gérés par les fonctions DLL pour la conversion parallèle-série et série-parallèle des demandes M_symbol et des indications.

4.4.2 Architecture en étoile

La couche liaison de données contient les composants figurant au Tableau 1. La disposition interne de ces composants, et leurs interfaces, sont présentées dans la Figure 4. Les pointes de flèche illustrent la direction principale du flux de données et de contrôle.

Tableau 1 – Composants de la couche liaison de données de l'architecture en étoile

Composants	Description
Machine de contrôle d'accès (ACM)	Contrôle d'accès déterministe au support et planification des opportunités d'envoi des DLPDU, contrôle de l'ajout et de la suppression de nœuds, restauration après interruption. Permet d'assembler et de transmettre les DLPDU au verrouilleur de trame TX par l'intermédiaire de la RMC, de recevoir et désassembler les DLPDU avec les informations de commande provenant de la RMC, et de déterminer la temporisation et la durée des transmissions. Permet de demander à la RMC d'envoyer les DLPDU pour le contrôle et le diagnostic du bus de terrain de Type 11. Permet de placer dans la mémoire tampon et de répartir dans le temps les DLPDU reçues entre l'utilisateur DLS et la RMC
Commande TX/RX d'émission cyclique (CTRC)	Permet de placer dans la mémoire tampon et de répartir dans le temps les DLSU reçues pour le transfert de données cycliques prioritaires entre l'utilisateur DLS et l'ACM
Commande TX/RX sporadique (STRC)	Permet de placer dans la mémoire tampon et de répartir dans le temps les DLSU reçues pour les données de message sporadiques entre l'utilisateur DLS et l'ACM
Commande de support de redondance (RMC)	Permet de recevoir les DLPDU provenant de l'ACM et de les décomposer en demandes de symbole d'octets envoyées au convertisseur parallèle-série d'octets, d'assembler les octets reçus du convertisseur série-parallèle d'octets dans les DLPDU et de les transmettre à l'ACM. Permet de sélectionner l'une de deux sorties des convertisseurs série-parallèle d'octets grâce aux verrouilleurs de trame RX de redondance du support
Verrouilleur de trame TX/RX	Permet de recevoir les symboles d'octets provenant de la RMC, de détecter et d'indiquer la temporisation de démarrage d'une DLPDU au convertisseur parallèle-série d'octets, de transmettre les octets reçus du convertisseur série-parallèle d'octets et d'indiquer la temporisation de démarrage d'une DLPDU à la RMC
Convertisseur parallèle-série d'octets	Permet de recevoir les symboles d'octet, de les coder et les convertir, puis de les envoyer en tant que M_symbols à la PhL. Il est également chargé de générer la FCS
Convertisseur série-parallèle d'octets	Permet de recevoir les M_symbols de la PhL, de les convertir en octets et de les envoyer au verrouilleur de trame RX. Il est également chargé de vérifier la FCS
Interface de gestion DLL	Permet de conserver les variables de gestion de station appartenant à la DLL, et de gérer les modifications synchronisées apportées aux paramètres de liaison



Légende

Anglais	Français
DLS user	utilisateur DLS
DL layer	couche DL
Sporadic TX/RX control	Commande TX/RX sporadique
cyclic transmission TX/TX control	Commande TX/RX d'émission cyclique
access control machine	Machine de contrôle d'accès
DLL management interface	interface de gestion DLL
station management	gestion de station
redundancy medium control	Commande de support de redondance
serializer / deserializer	Convertisseur parallèle-série / série-parallèle
TX framer	verrouilleur de trames TX
RX framer	verrouilleur de trames RX
octet serializer	Convertisseur parallèle-série d'octets
octet deserializer	Convertisseur série-parallèle d'octets
Ph-layer	couche Ph

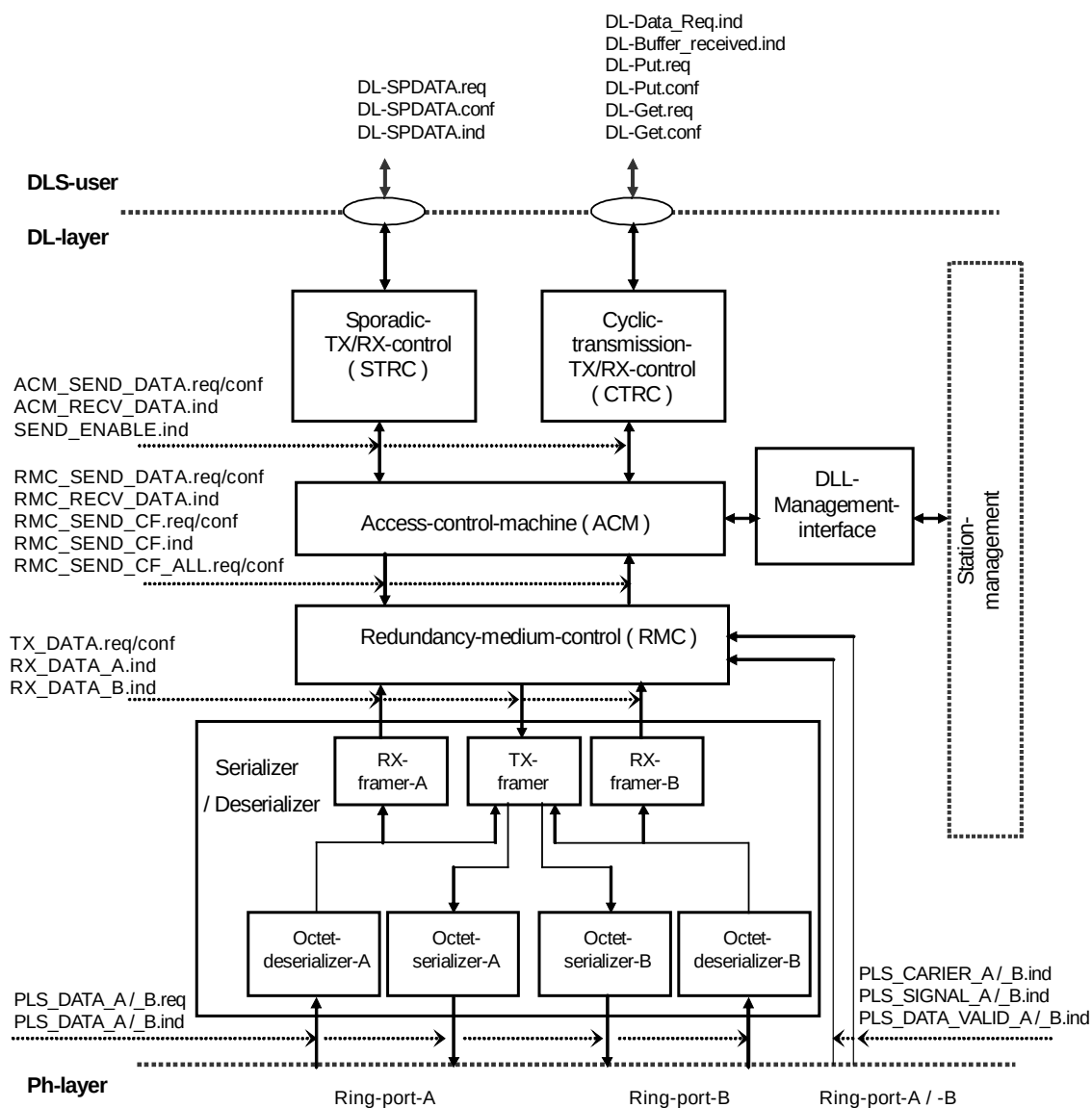
Figure 4 – Architecture interne de la couche liaison de données de l'architecture en étoile

4.4.3 Architecture en boucle

La couche liaison de données contient les composants figurant au Tableau 2. La disposition interne de ces composants, et leurs interfaces, sont présentées dans la Figure 5. Les pointes de flèche illustrent la direction principale du flux de données et de contrôle.

Tableau 2 – Composants de la couche liaison de données de l'architecture en boucle

Composants	Description
Machine de contrôle d'accès (ACM)	Contrôle d'accès déterministe au support et planification des opportunités d'envoi des DLPDU, contrôle de l'ajout et de la suppression de nœuds, restauration après interruption. Permet d'assembler et de transmettre les DLPDU au verrouilleur de trame TX par l'intermédiaire de la RMC, de recevoir et de désassembler les DLPDU avec les informations de commande provenant de la RMC, et de déterminer la temporisation et la durée des transmissions. Permet de demander à la RMC d'envoyer les DLPDU pour le contrôle et le diagnostic du bus de terrain de Type 11. Permet de placer dans la mémoire tampon et de répartir dans le temps les DLPDU reçues entre l'utilisateur DLS et la RMC
Commande TX/RX d'émission cyclique (CTRC)	Permet de placer dans la mémoire tampon et de répartir dans le temps les DLSU reçues pour le transfert de données cycliques prioritaires entre l'utilisateur DLS et l'ACM
Commande TX/RX sporadique (STRC)	Permet de placer dans la mémoire tampon et de répartir dans le temps les DLSU reçues pour les données de message sporadiques entre l'utilisateur DLS et l'ACM
Commande de support de redondance (RMC)	Permet de recevoir les DLPDU provenant de l'ACM et de les décomposer en demandes de symbole d'octets adressées au verrouilleur de trame TX, d'assembler les octets reçus provenant des verrouilleurs de trame RX dans les DLPDU, de sélectionner l'une des deux sorties des verrouilleurs de trame RX et de les soumettre à l'ACM. Permet de recevoir le signal provenant de l'ACM et d'activer la transmission des trames DLPDU pour le contrôle et le diagnostic du bus de terrain de Type 11
Verrouilleur de trame TX/RX	Le verrouilleur de trame TX permet de recevoir les symboles d'octets provenant de la RMC, de détecter et d'indiquer la temporisation de démarrage d'une DLPDU au convertisseur parallèle-série d'octets, de transférer les octets reçus provenant du convertisseur série-parallèle d'octets connecté à un port en anneau vers l'autre convertisseur parallèle-série d'octets connecté à l'autre port en anneau. Le verrouilleur de trame RX transmet les octets reçus du convertisseur série-parallèle d'octets et indique le délimiteur de début d'une DLPDU à la RMC
Convertisseur parallèle-série d'octets	Permet de recevoir les symboles d'octet, de les coder et les convertir, puis de les envoyer en tant que M_symbols à la PhL. Il est également chargé de générer la FCS
Convertisseur série-parallèle d'octets	Permet de recevoir les M_symbols du PhL, de les convertir en octets et de les envoyer aux verrouilleurs de trame RX et TX. Il est également chargé de vérifier la FCS
Interface de gestion DLL	Permet de conserver les variables de gestion de station appartenant à la DLL, et de gérer les modifications synchronisées apportées aux paramètres de liaison



Légende

Anglais	Français
DLS user	utilisateur DLS
DL layer	couche DL
Sporadic TX/RX control	Commande TX/RX sporadique
cyclic transmission TX/RX control	Commande TX/RX d'émission cyclique
access control machine (ACM)	Machine de contrôle d'accès (ACM)
DLL management interface	interface de gestion DLL
station management	gestion de station
redundancy medium control	Commande de support de redondance
serializer / deserializer	Convertisseur parallèle-série / série-parallèle
TX framer	verrouilleur de trames TX
RX framer	verrouilleur de trames RX
octet serializer	Convertisseur parallèle-série d'octets
octet deserializer	Convertisseur série-parallèle d'octets
Ph-layer	couche Ph

Figure 5 – Architecture interne de la couche liaison de données de l'architecture en boucle

4.5 Machine de contrôle d'accès et fonctions de support de planification

Les fonctions ACM planifient toutes les communications entre les DLE participantes du bus de terrain de Type 11, la temporisation de ces communications étant contrôlée pour

- a) satisfaire au contrôle d'accès au support spécifique afin de donner la possibilité à toutes les DLE d'envoyer deux sortes de classe de données cycliques prioritaires et de données de message sporadiques de manière opportune, hiérarchisée et déterministe, de détecter toute interruption du réseau et de lancer la restauration en temps voulu, puis d'ajouter et de retirer des nœuds de la ligne;
- b) offrir plusieurs niveaux de possibilités de transfert de données prioritaires afin d'envoyer des données au nœud dans l'ordre séquentiel et dans chaque période prédéfinie, de manière à procéder au transfert de données de chaque niveau dans la durée préalablement spécifiée (durée de maintien du jeton) et de transférer ou conserver les données des niveaux inférieurs sur une période de temps cyclique ultérieure selon le niveau et l'occasion du niveau supérieur du transfert de données est toujours réalisée à chaque occasion. Par ailleurs, l'ensemble du volume du transfert de données des niveaux inférieurs est transféré dans chaque période préalablement spécifiée;
- c) offrir des possibilités de transfert de données de message sporadiques au nœud indiquant que le transfert s'est déroulé de manière sporadique par l'utilisateur DLS et que les données sont transférées dans la période préalablement spécifiée du niveau de priorité correspondant et selon les applications ISO/CEI 8802-3 normales.

Le déroulement précis de la planification est essentiel à la prise en charge de nombreuses tâches de contrôle et de collecte des données dans le domaine des applications de ce protocole.

4.6 Paramètres locaux, variables, compteurs, temporisateurs et files d'attente

4.6.1 Présentation

4.6.1.1 Généralités

La présente norme utilise les paramètres de demande P(...) et les variables locales V(...) de l'utilisateur DLS afin de clarifier les effets de certaines actions et les conditions de validité de ces actions, les temporisateurs locaux T(...) afin de surveiller les actions du fournisseur DLS distribué et d'apporter une réponse DLE locale à l'absence de ces actions et les compteurs locaux C(...) pour exécuter des fonctions de mesure de fréquence. Elle s'appuie également sur les files d'attente locales Q(...) afin d'ordonner certaines activités, de clarifier les effets de certaines actions et les conditions de validité de ces actions.

Sauf indication contraire, au moment de leur création ou de l'activation de la DLE:

- a) toutes les variables doivent reprendre leur valeur par défaut ou leur valeur minimale admise si aucune valeur par défaut n'est spécifiée;
- b) tous les compteurs doivent être remis à zéro;
- c) tous les temporisateurs doivent être désactivés;
- d) toutes les files d'attente doivent être vides.

La gestion DL peut modifier les valeurs des variables de configuration.

4.6.1.2 Récapitulatif des variables, paramètres, compteurs, temporisateurs de l'architecture en étoile

Le Tableau 3 et le Tableau 4 récapitulent les variables et leur utilisation.

Les types de données utilisés dans les variables de liaison de données, les paramètres et les diagrammes d'états sont spécifiés à l'Article 5 de la CEI 61158-5-11. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire. Les bits de chaque octet sont numérotés de 0 à 7, 0 étant le bit de poids faible de l'octet. Chaque octet est en premier lieu transféré au bit inférieur de la couche Ph et plusieurs octets sont transférés en commençant par l'octet de poids faible (format petit-boutiste).

Tableau 3 – Variables DLE et valeurs admises de l'architecture en étoile

Nom de variable	Description	Valeurs admises	Type de données
BW	Longueur du mot de données TCC dans une DLPDU	La valeur par défaut est 128 (octets). Voir 4.6.2.7	UINT
DR	Vitesse de transmission des données	La valeur par défaut est 100 (Mbps). Voir 4.6.2.11	UINT
HT _h	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à grande vitesse	2 à 2 ¹⁶ -1 (en unités de 1 ms). La valeur par défaut est 3 x V(Th). Voir 4.6.2.12	UDINT
HT _i	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à basse vitesse	2 à 2 ¹⁶ -1 (en unités de 1 ms). La valeur par défaut est 3 x V(Th). Voir 4.6.3.5	UDINT
HT _m	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à moyenne vitesse	2 à 2 ¹⁶ -1 (en unités de 1 ms). La valeur par défaut est 3 x V(Th). Voir 4.6.2.14	UDINT
IA	Adresse individuelle de ce nœud	Adresse individuelle de 48 bits identique à celles de l'ISO/CEI 8802-3. Voir 4.6.2.16	USINT [6]
IGP	Intervalle de temps entre la fin d'une trame précédente et le début de la trame consécutive envoyée par un nœud	La valeur par défaut est 24 (0,96 µs en unités de 0,04 µs). Voir 4.6.2.17	UDINT
IP	Adresse IP de ce nœud	Adresse de 32 bits identique à IPv4. Voir 4.6.2.18	USINT [4]
MD	Distance maximale sur le chemin de connexion entre 2 nœuds	1 à 100 (km). La valeur par défaut est 8. Voir 4.6.2.21	UINT
MDD	Différence maximale de la distance de deux supports physiques redondants sur le chemin de connexion entre 2 nœuds	0 à 2 000 (en m). La valeur par défaut est 500. Voir 4.6.3.7	UINT
MGA	Adresse de groupe de multidiffusion	Adresse de multidiffusion de 48 bits identique à celles de l'ISO/CEI 8802-3, pour les nœuds de bus de terrain de Type 11 associés de manière logique. La valeur par défaut est 0x01-0x00-0x5e-0x50-0x00-0x01. Voir 4.6.2.22	USINT [6]
MN	Nombre maximal de nœuds	254 (valeur par défaut). Voir 4.6.2.23	USINT
MPD	Différence maximale du délai de propagation du signal sur deux supports physiques redondants sur le chemin de connexion entre 2 nœuds	1 à 0x7c (en unités de 0,04 µs). La valeur par défaut est 0x7c (≈ 5 µs). Voir 4.6.3.8	UDINT
MRT	Nombre maximal d'unités de répéteur sur le chemin de connexion entre 2 nœuds	0 à 7. La valeur par défaut est 3. Voir 4.6.2.24	UINT
MTHT	Temps de maintien maximal du jeton pour la transmission de données cycliques à grande vitesse	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est 0x30B4 (= 900 µs). Voir 4.6.2.25	UDINT
PB _h	Liste de 2 048 mémoires tampons pour l'envoi de données TCC à grande vitesse	1 à 2 048. Voir 4.6.2.28	UINT
PB _i	Liste de 2 048 mémoires tampons pour l'envoi de données TCC à basse vitesse	1 à 2 048. Voir 4.6.3.10	UINT
PB _m	Liste de 2 048 mémoires tampons pour l'envoi de données cyclique prioritaires à moyenne vitesse	1 à 2 048. Voir 4.6.2.29	UINT
RCS	Contrôle de commutation du canal de réception des trames de réception. Désigne la commande de commutation des trames de réception sur l'un des deux canaux de réception A et B correspondant à chacun des supports redondants A et B	« Automatic », « Force A », « Force B » « Automatic »: Bascule automatiquement sur le bon canal de réception. « Force A », « Force B »: Force le basculement du canal de réception A ou B, respectivement. La valeur par défaut est « Automatic ». Voir Tableau 10, Tableau 12, 4.6.3.12	BOOL

Nom de variable	Description	Valeurs admises	Type de données
RMGP	Intervalle de temps maximal d'un canal de réception, qui a déjà terminé une trame reçue et a attendu pendant la durée de l'intervalle IGP la fin d'une trame à recevoir sur l'autre canal de réception afin de détecter l'autre canal de réception interrompu	250 (en unités de 0,04 µs). La valeur 250 représente 10 µs. Voir 4.6.3.14	UDINT
ST	L'intervalle de temps est l'unité de temps fondamentale d'observation, utilisée dans la DLME pour observer le lancement de l'action (la réinitialisation de l'envoi de la trame CLM, par exemple) et la trame CMP envoyée par le nœud correspondant. Si le nœud n'a pas pu procéder à l'envoi, cet intervalle est dans la DLME pour lancer l'action dès la réception de la trame CMP	1 à 255 (en unités de temps de symbole physique pour 512 bits, identique à l'ISO/CEI 8802-3). La valeur par défaut est 20 (= 100 µs). Voir Tableau 14, 4.6.2.38	USINT
SCMP	Temps de contrôle d'accès au support du CMP de substitution	1 à 255 (en unités de temps de symbole physique pour 512 bits, identique à l'ISO/CEI 8802-3). La valeur par défaut est 20 (= 100 µs). Voir 4.6.2.31	USINT
SCMPL	Comptage répétitif admissible dans lequel le nœud peut se comporter comme le nœud qui a reçu la trame CMP, même si le nœud correspondant n'a pas été en mesure d'envoyer la trame CMP. Le nœud détermine que le nœud correspondant est hors service si le comptage est terminé	1 à 16. La valeur par défaut est 3. Voir 4.6.2.33	USINT
SEN _A SEN _B	Envoyer la commande activée pour l'envoi de trames hors de l'émetteur -A ou -B	True/Enable ou False/Disable La valeur par défaut est True/Enable. Voir 4.6.3.16	BOOL
Th	Période de transmission à grande vitesse	1 à 160 (en unités de 0,1 ms). Voir Tableau 15, 4.6.2.39	USINT [3]
Tl	Période de transmission à basse vitesse	1 à 10 000 (en unités de 1 ms). Voir Tableau 18, 4.6.3.17	UINT
TISPD	Intervalle de temps traité de manière cyclique pour le service de données de messagerie sporadiques	1 à 1 000 (en unités de 1 ms). La valeur par défaut est 100. Voir Tableau 17, 4.6.2.40, 4.6.2.41	UINT
Tm	Période de transmission à moyenne vitesse	1 à 1 000 (en unités de 1 ms). La valeur par défaut est 100. Voir Tableau 16, 4.6.2.42	UINT
TMAC	Période maximale d'observation pour le nœud SYN afin de solliciter de nouveaux nœuds dans le bus de terrain de Type 11	1 à 100 (en unités de 0,1 ms). La valeur par défaut est 100. Voir 4.6.2.43, 4.6.2.44	USINT
Tsl	Nombre de DLE capables d'envoyer l'emplacement suivant	1 à 255. Voir 4.6.2.46	USINT
TSYN	Intervalle de temps cible entre deux arrivées de trame SYN	1 à 160 (en unités de 0,1 ms). Voir 4.6.2.47	UDINT
TTRT0	Temps de rotation du jeton cible de la classe d'accès 0	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est P(TSYN). Voir 4.6.3.18	UDINT
TTRT1	Temps de rotation du jeton cible de la classe d'accès 1	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est P(TSYN). Voir 4.6.2.48	UDINT
TTRT2	Temps de rotation du jeton cible de la classe d'accès 2	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est P(TSYN). Voir 0	UDINT
TN	Numéro d'identification du nœud. La valeur initiale est 0 à la mise sous tension ou la réinitialisation	1 à 255. Voir 4.6.2.45	USINT

Tableau 4 – Variables observables et leurs plages de valeurs de l'architecture en étoile

Nom de variable	Description	Plage de valeurs	Type de données
AIGP _A AIGP _B	Appareil de surveillance IGP sur les canaux de réception A et B	0 à 24, décrémentation continue. La valeur 24 représente 0,96 µs. Voir 4.6.3.1	UDINT
ARMGP	Période observée d'un canal de réception, qui a déterminé une trame reçue et a attendu pendant la durée IGP (0,96 µs) la fin d'une trame reçue sur l'autre canal de réception afin de détecter l'autre canal de réception interrompu	0 à 250, décrémentation continue. La valeur 250 représente 10 µs. Voir 4.6.3.2	UDINT
ASCMP	Ensemble de compteurs du nombre de nœuds total. Chaque compteur correspondant à un nœud indique le comptage répétitif observé selon lequel le nœud correspondant n'a pas envoyé la trame CMP et l'autre nœud peut se comporter comme celui de la trame CMP reçue sans erreur. Le nœud détermine que le nœud correspondant est hors service lorsque le comptage atteint le nombre préalablement spécifié. Le nombre déterminé est indiqué et est incrémenté par hasard au niveau de chaque DLME	1 à 16. La valeur par défaut est égale à la valeur de SCmpl, soit 3. Voir 4.6.2.1	USINT
ATHT	Période observée du MHT	0 à MHT, décrémentation continue. Voir 4.6.2.2	UDINT
ATSYN	Période observée entre deux arrivées de trame SYN	1 à 2 ¹⁶ (temps octet). Voir 4.6.2.3	UDINT
ATTRT0	Période observée du TTRT0	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.3.3	UDINT
ATTRT1	Période observée du TTRT1	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.2.4	UDINT
ATTRT2	Période observée du TTRT2	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.2.5	UDINT
CDh	Comptage cumulé des trames de données cycliques à grande vitesse transmises	0 à 2 ³² -1. Voir 4.6.2.8	UDINT
CDH _{blk}	Etat des données d'information de la mémoire tampon avec le numéro d'identification _{blk} , indiquant que les données TCC correspondantes sont actives (normales) ou inactives (anormales)	True: actives. False: inactives. Voir 4.6.2.9	BOOL
CDl	Comptage cumulé des trames de données cycliques à basse vitesse transmises	0 à 2 ³² -1. Voir 4.6.3.4	UDINT
CDm	Comptage cumulé des trames de données cycliques à moyenne vitesse transmises	0 à 2 ³² -1. Voir 4.6.2.10	UDINT
LL	Liste opérationnelle indiquant que le nœud correspondant est, à cet instant, connecté et qu'il fonctionne dans le domaine de bus de terrain de Type 11 dans la trame SYN reçue. La liste opérationnelle est un ensemble de 256 éléments booléens, représentés dans 32 octets, dans lequel chaque bit correspond à un nœud dans le domaine de bus de terrain de Type 11 et indique l'État courant de ce nœud	Ensemble d'éléments booléens de 256 bits True: le nœud est connecté et fonctionne. False: le nœud n'est pas connecté et ne fonctionne pas. Voir 4.6.2.19	USINT [16]
LN	Numéro extrait de la liste opérationnelle au niveau de chaque nœud et leur permettant de déterminer si le nœud est en mesure d'envoyer la trame de données sur le support	1 à 255. Voir 4.6.2.20	USINT
NCD _A , NCD _B	Comptage cumulé d'absence de porteuse sur le canal de réception A ou B	0 à 2 ³² -1. Voir 4.6.3.9	UDINT

Nom de variable	Description	Plage de valeurs	Type de données
NONC	Comptage répétitif admissible de trames non CMP reçues par le nœud SYN dans les cycles Tsyn consécutifs correspondants (256 fois par SCMPL) afin de ne détecter aucun autre nœud que le nœud SYN dans le domaine de bus de terrain de Type 11	1 à 16. La valeur par défaut est 3. Voir 4.6.2.26, 4.6.2.27	USINT
RE _A , RE _B	Comptage cumulé des DLPDU reçues par erreur sur le canal de réception A ou B. RE _A est incrémenté lorsqu'une erreur est détectée sur le canal de réception A, le canal de réception B ne présentant aucune erreur. RE _B correspond, mais les rôles du canal sont permutés.	0 à 2 ³² -1. Voir 4.6.3.13	UDINT
ROK _A , ROK _B	Comptage cumulé de DLPDU reçue sans erreur sur le canal de réception A ou B	0 à 2 ³² -1. Voir 4.6.3.15	UDINT
SL	Période observée du SL	0 à V(SL). Voir 4.6.2.35, 4.6.2.36	USINT
SD	Comptage cumulé des trames de données sporadiques transmises	0 à 2 ³² -1. Voir 4.6.2.34	UDINT

4.6.1.3 Récapitulatif des variables, paramètres, compteurs, temporisateurs de l'architecture en boucle

Le Tableau 5 et le Tableau 6 récapitulent les variables et leur utilisation. Les types de données utilisés dans les variables de liaison de données, les paramètres et les diagrammes d'états sont spécifiés à l'Article 5 de la CEI 61158-5-11.

Les types de données utilisés dans les variables de liaison de données, les paramètres et les diagrammes d'états sont spécifiés à l'Article 5 de la CEI 61158-5-11. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire. Les bits de chaque octet sont numérotés de 0 à 7, 0 étant le bit de poids faible de l'octet. Chaque octet est en premier lieu transféré au bit inférieur de la couche Ph et plusieurs octets sont transférés en commençant par l'octet de poids faible (format petit-boutiste).

Tableau 5 – Variables DLE et valeurs admises de l'architecture en boucle

Nom de variable	Description	Valeurs admises	Type de données
BW	Longueur du mot de données TCC dans une DLPDU	16 à 128. La valeur par défaut est 64. Voir 4.6.2.7	UINT
DR	Vitesse de transmission des données	La valeur est 100 ou 1 000 (Mbps). Voir 4.6.2.11	UINT
ESYN	Activer SYN	« Enable » ou « Disable ». "Enable" peut être un nœud SYN. "Disable" ne peut pas être un nœud SYN. La valeur par défaut est Enable. Voir 6.4.3, 4.6.4.5	
FWE _{AB} , FWE _{BA}	Le transfert de commande de la trame de réception permet chaque direction entre le port en anneau A et le port en anneau B. FWE _{AB} contrôle les trames de transfert recevant du port en anneau A vers le port en anneau B. Par ailleurs, FWE _{BA} contrôle le transfert des trames recevant du port en anneau B vers le port en anneau A	« Enable » ou « Disable ». Enable: permet de transférer des trames recevant d'un port en anneau vers un autre port en anneau. Disable: ne permet pas le transfert de trames entre des ports en anneau. La valeur par défaut est Disable. Voir 6.9.3, 4.6.4.6	BOOL
HTH	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à grande vitesse	2 à 2 ¹⁰ -1 (en unités de:). – 1 ms pour le fonctionnement à 100 Mbps; – 0,1 ms pour le fonctionnement à 1 000 Mbps). La valeur par défaut est 3 x V(Th). Voir 4.6.2.12	UDINT

Nom de variable	Description	Valeurs admises	Type de données
HTI	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à basse vitesse	2 à $2^{16}-1$ (en unités de 1 ms). La valeur par défaut est $3 \times V(Th)$. Voir 4.6.4.7	UDINT
HTm	Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à moyenne vitesse	2 à $2^{16}-1$ (en unités de 1 ms). La valeur par défaut est $3 \times V(Th)$. Voir 4.6.2.14	UDINT
IA	Adresse individuelle de ce nœud	Adresse individuelle de 48 bits identique à celles de l'ISO/CEI 8802-3. Voir 4.6.2.16	SWORD[6]
IGP	Intervalle de temps entre la fin d'une trame précédente et le début de la trame consécutive envoyée par un nœud	La valeur par défaut est: – 24 pour le fonctionnement à 100 Mps; – 12 pour le fonctionnement à 1 000 Mbps. Voir 4.6.2.17	UDINT
IP	Adresse IP de ce nœud	Adresse de 32 bits identique à IPv4. Voir 4.6.2.18	USINT [4]
MD	Distance maximale sur le chemin de connexion entre 2 nœuds	1 à 20 (km). La valeur par défaut est 4. Voir 4.6.2.21	UINT
MDAD	Distance maximale entre nœuds adjacents	La valeur est: – 1 à 2 (km) pour le fonctionnement à 100 Mbps; – 1 à 5 (km) pour le fonctionnement à 1 000 Mbps. La valeur par défaut est 1. Voir 4.6.4.11	UINT
MGA	Adresse de groupe de multidiffusion	Adresse de multidiffusion de 48 bits identique à celles de l'ISO/CEI 8802-3, pour les nœuds de bus de terrain de Type 11 associés de manière logique. La valeur par défaut est 0x01-0x00-0x5e-0x50-0x00-0x01. Voir 4.6.2.22	USINT [6]
MN	Nombre maximal de nœuds	254 (valeur par défaut). Voir 4.6.2.23	USINT
MRT	Nombre maximal d'unités de répéteur sur le chemin de connexion entre 2 nœuds	0 à 252. La valeur par défaut est 30. Voir 4.6.2.24	USINT
MTHT	Temps de maintien maximal du jeton pour la transmission de données cycliques à grande vitesse	1 à $2^{16}-1$ (temps octet). La valeur par défaut est 0x30B4 (= 900 µs pour le fonctionnement à 100 Mbps et 90 µs pour le fonctionnement à 1 000 Mbps). Voir 4.6.2.25	UDINT
PBh	Liste de mémoires tampons pour l'envoi de données TCC à grande vitesse	La valeur est: – 1 à 2 048 pour le fonctionnement à 100 Mbps; – 1 à 4 096 pour le fonctionnement à 1 000 Mbps. Voir 4.6.2.28	UINT
PBl	Liste de mémoires tampons pour l'envoi de données TCC à basse vitesse	La valeur est: – 1 à 2 048 pour le fonctionnement à 100 Mbps; – 1 à 4 096 pour le fonctionnement à 1 000 Mbps. Voir 4.6.2.16	UINT
PBm	Liste de mémoires tampons pour l'envoi de données TCC à moyenne vitesse	La valeur est: – 1 à 2 048 pour le fonctionnement à 100 Mbps; – 1 à 4 096 pour le fonctionnement à 1 000 Mbps. Voir 4.6.2.29	UINT
PS _A PS _B	Contrôle de l'état du port pour le blocage ou le non-blocage de trame provenant du port en anneau A ou B	« Blocked » ou « Non-blocked ». La valeur par défaut est Blocked. Voir Tableau 27, 4.6.4.18	BOOL
REN _A REN _B	Contrôle d'activation de réception des trames de réception provenant du port en anneau A ou B.	« Enable » ou « Disable ». La valeur par défaut est Enable. Voir Tableau 28, 4.6.4.20	BOOL

Nom de variable	Description	Valeurs admises	Type de données
ST	L'intervalle de temps est l'unité de temps fondamentale d'observation, utilisée dans la DLME pour observer le lancement de l'action (la réinitialisation de l'envoi de la trame CLM, par exemple) et la trame CMP envoyée par le nœud correspondant. Si le nœud n'a pas pu procéder à l'envoi, cet intervalle est dans la DLME pour lancer l'action dès la réception de la trame CMP	1 à 255 (en unités de temps de symbole physique pour 512 bits pour le fonctionnement à 100 Mbps et pour 4 096 bits pour le fonctionnement à 1 000 Mbps, comme dans l'ISO/CEI 8802-3). La valeur par défaut est 20 (= 100 µs pour le fonctionnement à 100 Mbps et 8 µs pour le fonctionnement à 1 000 Mbps). Voir Tableau 14, 4.6.2.38	USINT
SCMP	Temps de contrôle d'accès au support du CMP de substitution	1 à 255 (en unités de temps de symbole physique pour 512 bits pour le fonctionnement à 100 Mbps et pour 4 096 bits pour le fonctionnement à 1 000 Mbps, comme dans l'ISO/CEI 8802-3). La valeur par défaut est 20 (= 100 µs pour le fonctionnement à 100 Mbps et 8 µs pour le fonctionnement à 1 000 Mbps). Voir 4.6.2.31	USINT
SCMPL	Comptage répétitif admissible dans lequel le nœud peut se comporter comme le nœud qui a reçu la trame CMP, même si le nœud correspondant n'a pas été en mesure d'envoyer la trame CMP. Le nœud détermine que le nœud correspondant est hors service si le comptage est terminé	1 à 16. La valeur par défaut est 3. Voir 4.6.2.33	USINT
SEN _A SEN _B	Envoyer la commande activée pour l'envoi de trames hors du port en anneau A ou B	True/Enable ou False/Disable. La valeur par défaut est True/Enable. Voir Tableau 25, 4.6.4.22	BOOL
Th	Période de transmission à grande vitesse	1 à 160 (en unités de 0,1 ms). Voir Tableau 15, 4.6.2.39	USINT [3]
Tl	Période de transmission à basse vitesse	1 à 10 000 (en unités de 1 ms). La valeur par défaut est 100. Voir Tableau 18, 4.6.4.23	UINT
TISPD	Intervalle de temps traité de manière cyclique pour le service de données de messagerie sporadiques	1 à 1 000 (en unités de 1 ms). La valeur par défaut est 100. Voir Tableau 17, 4.6.2.40, 4.6.2.41	UINT
Tm	Période de transmission à moyenne vitesse	10 à 1 000 (en unités de 1 ms). La valeur par défaut est 100. Voir Tableau 16, 4.6.2.42	UINT
TMAC	Période maximale d'observation pour le nœud SYN afin de solliciter de nouveaux nœuds dans le bus de terrain de Type 11	1 à 100 (en unités de 0,1 ms pour le fonctionnement à 100 Mbps et de 0,01 ms pour le fonctionnement à 1 000 Mbps). La valeur par défaut est 100. Voir 4.6.2.43, 4.6.2.44	USINT
Tsl	Nombre de DLE capables d'envoyer l'emplacement suivant	1 à 255. Voir 4.6.2.46	USINT
TSYN	Intervalle de temps cible entre deux arrivées de trame SYN	1 à 160 (en unités de 0,1 ms). Voir 4.6.2.47	UDINT
TTRT0	Temps de rotation du jeton cible de la classe d'accès 0	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est P(TSYN). Voir 4.6.2.24	UDINT
TTRT1	Temps de rotation du jeton cible de la classe d'accès 1	1 à 2 ¹⁶ -1 (temps octet). La valeur par défaut est P(TSYN). Voir 4.6.2.48	UDINT
TTRT2	Temps de rotation du jeton cible de la classe d'accès 2	1 à 2 ¹⁶ -1 (temps octet) La valeur par défaut est P(TSYN). Voir 0	UDINT
TN	Numéro d'identification du nœud. La valeur initiale est 0 à la mise sous tension ou la réinitialisation	1 à 255. Voir 4.6.2.45	USINT

Tableau 6 – Variables observables et leurs plages de valeurs de l'architecture en boucle

Nom de variable	Description	Plage de valeurs	Type de données
AN _A AN _B	Numéro du nœud adjacent avoisinant le port en anneau A ou B	1 à 254. Voir 4.6.4.1	USINT
ASCMP	Ensemble de compteurs du nombre de nœuds total. Chaque compteur correspondant à un nœud indique le comptage répétitif observé selon lequel le nœud correspondant n'a pas envoyé la trame CMP et l'autre nœud peut se comporter comme celui de la trame CMP reçue sans erreur. Le nœud détermine que le nœud correspondant est hors service lorsque le comptage atteint le nombre préalablement spécifié. Le nombre déterminé est indiqué et est incrémenté par hasard au niveau de chaque DLME	1 à 16. La valeur par défaut est égale à la valeur de SCMP, soit 3. Voir 4.6.2.1	USINT
ATHT	Période observée du MHT	0 à MHT, décrémentation continue. Voir 4.6.2.2	UDINT
ATSYN	Période observée entre deux arrivées de trame SYN	1 à 2 ¹⁶ (temps octet). Voir 4.6.2.3	UDINT
ATRT0	Période observée du TTRT0	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.4.2	UDINT
ATRT1	Période observée du TTRT1	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.2.4	UDINT
ATRT2	Période observée du TTRT2	1 2 ¹⁶ -1, décrémentation continue. Voir 4.6.2.5	UDINT
CDh	Comptage cumulé des trames de données cycliques à grande vitesse transmises	0 à 2 ³² -1. Voir 4.6.2.8	UDINT
CDH _{blk}	Etat des données d'information de la mémoire tampon avec le numéro d'identification _{blk} , indiquant que les données TCC correspondantes sont actives (normales) ou inactives (anormales)	True: actives. False: inactives. Voir 4.6.2.9	BOOL
CDm	Comptage cumulé des trames de données cycliques à moyenne vitesse transmises	0 à 2 ³² -1. Voir 4.6.2.10	UDINT
CRE _A CRE _B	Comptage répétitif observé en cas d'erreur de réception anormale détectée lors de la réception de la trame entrante provenant du port en anneau A ou B, respectivement. Le comptage est diminué à chaque détection de réception anormale répétitive, mais réinitialisé lors de la suppression de la réception anormale. Si la valeur est nulle, une erreur de réception anormale a été détectée	0 à 15. La valeur par défaut est 2. Voir 4.6.4.4	USINT
LINK _A LINK _B	Etat de la liaison indiquant la condition de la connexion de liaison physique au niveau du port en anneau A ou B	Abnormal ou Normal. Voir 4.6.4.9	BOOL
LSYN	Numéro du nœud à l'état maître en cours	1 à 254. Voir 4.6.4.10	USINT
LL	Liste opérationnelle indiquant que le nœud correspondant est, à cet instant, connecté et qu'il fonctionne dans le domaine de bus de terrain de Type 11 dans la trame SYN reçue. La liste opérationnelle est un ensemble de 256 éléments booléens, représentés dans 32 octets, dans lequel chaque bit correspond à un nœud dans le domaine de bus de terrain de Type 11 et indique l'État courant de ce nœud	Ensemble d'éléments booléens de 256 bits. True: le nœud est connecté et fonctionne. False: le nœud n'est pas connecté ou ne fonctionne pas. Voir 4.6.2.19	USINT [16]

Nom de variable	Description	Plage de valeurs	Type de données
LN	Numéro extrait de la liste opérationnelle au niveau de chaque nœud et leur permettant de déterminer si le nœud est en mesure d'envoyer la trame de données sur le support	1 à 255. Voir 4.6.2.20	USINT
NCD _A , NCD _B	Comptage cumulé d'absence de porteuse sur le port en anneau A ou B	0 à $2^{32}-1$. Voir 4.6.4.12	UDINT
NONC	Comptage répétitif admissible de trames non CMP reçues par le nœud SYN dans les cycles Tsyn consécutifs correspondants (256 fois par SCMPL) afin de ne détecter aucun autre nœud que le nœud SYN dans le domaine de bus de terrain de Type 11	1 à 16. La valeur par défaut est 3. Voir 4.6.2.26, 4.6.2.27	USINT
NOS	Etat de fonctionnement du nœud indiquant la condition de fonctionnement du nœud à l'état normal ou ouvert	Normal ou Open. La valeur par défaut est Open. Voir 4.6.4.13	BOOL
NRF _A , NRF _B	Période observée dans laquelle aucune trame n'a été reçue du port en anneau A ou B après qu'une trame a été reçue d'un autre port en anneau B ou A, respectivement	0 à 15 (en unités de temps Th). La valeur par défaut est 2. Voir 4.6.4.14	USINT
NS	Etat du nœud indiquant que le nœud fonctionne en tant que maître ou qu'esclave	Master ou Slave. La valeur par défaut est Slave. Voir 4.6.4.15	BOOL
PP	Port principal indiquant le port en anneau A ou B par lequel a eu lieu la première réception de SYN ou CLM depuis l'activation de ce nœud, le démarrage ou l'initialisation du système	Ring-port-A ou Ring-port-B. La valeur par défaut est Ring-port-A. Voir Tableau 26, 4.6.4.17	BOOL
RE _A , RE _B	Comptage cumulé des DLPDU reçues par erreur sur le port en anneau A ou B. RE _A est incrémenté lorsqu'une erreur est détectée sur le port en anneau A, le port en anneau B ne présentant aucune erreur. RE _B correspond, mais les rôles du canal sont permutés.	0 à $2^{32}-1$. Voir 4.6.4.20	UDINT
ROK _A , ROK _B	Comptage cumulé de DLPDU reçue sans erreur sur le port en anneau A ou B	0 à $2^{32}-1$. Voir 4.6.4.21	UDINT
SL	Période observée du SL	0 à V(SL). Voir 4.6.2.35, 4.6.2.36	USINT
SD	Comptage cumulé des trames de données sporadiques transmises	0 à $2^{32}-1$. Voir 4.6.2.34	UDINT

4.6.2 Variables, paramètres, compteurs, temporisateurs et files d'attente communs de Type 11

4.6.2.1 C(ASCMP): Comptage ASCMP

Ensemble de compteurs, dont chacun indique le nombre de comptages répétitifs observés selon lequel le nœud correspondant n'a pas envoyé la trame CMP et l'autre nœud peut se comporter comme celui de la trame CMP reçue sans erreur. L'incrémentation du nombre C(ASCMP) coïncide avec chaque nœud du bus de terrain de Type 11. La valeur est comprise entre 1 et V(SCMPL), la valeur par défaut étant 3.

4.6.2.2 T(ATHT): Appareil de surveillance MTHT

La DLE utilise la variable T(ATHT) pour mesurer le temps écoulé de MTHT. La valeur est réduite dans la plage comprise entre P(MTHT) et 0.

4.6.2.3 T(ATSYN): Appareil de surveillance TSYN

La DLE utilise la variable T(ATSYN) pour surveiller la période comprise entre deux arrivées de trame SYN. La valeur est indiquée en temps octet par le débit binaire équivalent.

4.6.2.4 T(ATTRT1): Appareil de surveillance TTRT1

La DLE utilise la variable T(ATTRT1) pour surveiller la période de TTRT1. La valeur est réduite dans la plage comprise entre $2^{16}-1$ et 1, dont l'unité est donnée en temps octet par le débit binaire équivalent.

4.6.2.5 T(ATTRT2): Appareil de surveillance TTRT2

La DLE utilise la variable T(ATTRT2) pour surveiller la période de TTRT2. La valeur est réduite dans la plage comprise entre $2^{16}-1$ et 1, dont l'unité est donnée en temps octet par le débit binaire équivalent.

4.6.2.6 Q(BF_{blk}): Registre tampon utilisé pour l'envoi et la réception d'une DLPDU pour une DLPDU de données cycliques prioritaires

Le nombre total de registres tampons est équivalent au nombre total de DLCEP. Chaque registre tampon avec le numéro "blk" d'identifiant correspond à un DLCEP avec le numéro d'identifiant "identifiant".

4.6.2.7 V(BW), P(BW): Longueur du mot de données cycliques prioritaires dans une DLPDU

Cette variable comporte et désigne la longueur du mot de données cycliques prioritaires dans une DLPDU.

La valeur possible de la longueur en octets est la suivante:

- a) architecture en étoile: La valeur par défaut est 128;
- b) architecture en boucle: 16 à 128. La valeur par défaut est 64.

4.6.2.8 C(CDh): Comptage cumulé de la trame de données cycliques à grande vitesse transmise

C(CDh) indique le nombre de comptages cumulés de la trame de données cycliques à grande vitesse transmise sur le support. C(CDh) est un compteur binaire de remplacement d'une longueur de 32 bits.

4.6.2.9 V(CDH_{blk}) Etat des données d'information du registre tampon avec numéro d'identifiant "blk"

Cette variable indique l'état des données cycliques prioritaires correspondantes actives (saines) ou inactives (non saines). La valeur "True" (vrai) indique l'état actif, et la valeur "False" (faux) indique l'état inactif.

4.6.2.10 C(CDm): Comptage cumulé de la trame de données cycliques à moyenne vitesse transmise

C(CDm) indique le nombre de comptages cumulés de la trame de données cycliques à moyenne vitesse transmise sur le support. C(CDm) est un compteur binaire de remplacement d'une longueur de 32 bits.

4.6.2.11 V(DR): Vitesse de transmission des données

La valeur de cette variable indique le débit binaire équivalent en Mbps. La valeur est 100 pour l'architecture en étoile et 100 ou 1 000 pour l'architecture en boucle.

4.6.2.12 V(HTh), P(HTh): Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à grande vitesse

Cette variable comporte et désigne la valeur de la période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à grande vitesse. La valeur de cette variable se situe dans la plage comprise entre 2 et $2^{16}-1$, dont l'unité est 1 ms pour un fonctionnement à 100 Mbps, et 0,1 ms pour un fonctionnement à 1 000 Mbps. La valeur par défaut est $3 \times V(Th)$.

4.6.2.13 T(HTh): Appareil de surveillance HTh

La DLE utilise la variable T(HTh) pour mesurer le temps HTh écoulé pour détecter l'échec du cycle de transmission continu à grande vitesse. La valeur est réduite dans la plage comprise entre V(HTh) et 1.

4.6.2.14 V(HTm), P(HTm): Période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à moyenne vitesse

Cette variable comporte et désigne la valeur de la période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à moyenne vitesse. La valeur de cette variable se situe dans la plage comprise entre 2 et $2^{16}-1$, dont l'unité est 1 ms. La valeur par défaut est $3 \times V(Th)$.

4.6.2.15 T(HTm): Appareil de surveillance HTm

La DLE utilise la variable T(HTm) pour mesurer le temps HTm écoulé pour détecter l'interruption du cycle de transmission continu à moyenne vitesse. La valeur est réduite dans la plage comprise entre V(HTm) et 1.

4.6.2.16 V(IA): Individual-address-of-this-node (adresse individuelle de ce nœud)

Cette variable comporte l'adresse individuelle de ce nœud dans la longueur de 48 bits spécifiée par l'ISO/CEI 8802-3. La valeur est définie par le DLMS.

4.6.2.17 V(IGP): Inter-frame-time-gap (Intervalle de temps entre trames)

Cette variable indique la valeur de l'intervalle de temps entre la fin d'une trame précédente et le début de la trame consécutive envoyée par un nœud. La valeur est équivalente à 0,96 μ s pour un fonctionnement à 100 Mbps, et à 0,096 μ s pour un fonctionnement à 1 000 Mbps et identique à celle spécifiée par l'ISO/CEI 8802-3.

4.6.2.18 V(IP), P(IP): Adresse IP de ce nœud

Cette variable comporte et désigne la valeur DLSAP attribuée pour le service de données de messagerie sporadiques de bus de terrain de Type 11.

4.6.2.19 V(LL): Liste opérationnelle

Cette variable indique l'état de fonctionnement actuel, qu'un nœud correspondant soit relié ou fonctionne normalement dans le bus de terrain de Type 11. La valeur potentielle est "True" ou "False", "True" signifiant que le nœud est relié et fonctionne normalement et "False" signifiant le contraire. V(LL) est utilisée par la DLE et générée par les informations acheminées par la trame SYN. La liste opérationnelle est un ensemble de 8 mots d'une longueur de 32 bits, chaque bit correspondant au nœud dans le bus de terrain de Type 11 et indiquant l'état de fonctionnement actuel. Chaque bit correspond au nombre de nœuds V(TN) dans un ordre séquentiel compris entre 0 et 255 au format petit-boutiste.

4.6.2.20 V(LN): Live-node-number (Nombre de nœuds actifs)

Cette variable indique le nombre TN de nœuds, extraits de la variable V(LL), reliés et fonctionnant normalement dans le bus de terrain de Type 11 à ce stade. La DLE utilise la variable V(LN) pour déterminer si le nœud est en mesure d'envoyer la trame de données sur le support. Cette valeur se situe dans la plage comprise entre 1 et 255.

4.6.2.21 V(MD), P(MD): Maximum-distance (Distance maximale)

La valeur de cette variable, définie par le DLMS, comporte la distance maximale en kilomètres sur le chemin de connexion situé entre 2 nœuds.

Cette valeur se situe dans une plage définie comme suit:

- a) architecture en étoile: 1 à 100; la valeur par défaut est 8;
- b) architecture en boucle: 1 à 20; la valeur par défaut est 4.

4.6.2.22 V(MGA), P(MGA): Adresse de groupe de multidiffusion

Cette variable comporte et désigne l'adresse de groupe de multidiffusion, utilisée par la DLE pour constituer un groupe d'association logique des nœuds de bus de terrain de Type 11, d'une longueur de 48 bits spécifiée et identique à celle définie dans l'ISO/CEI 8802-3. La valeur par défaut est 0x01-0x00-0x5e-0x50-0x00-0x01.

4.6.2.23 V(MN), P(MN): Maximum-node-number (Nombre de nœuds maximal)

Cette variable comporte le nombre de nœuds maximal, et est définie par le DLMS. La valeur par défaut de cette variable est 254.

4.6.2.24 V(MRT), P(MRT): Maximum-number-of-the-repeater-unit (Nombre maximal de répéteurs)

Cette variable, définie par le DLMS, comporte le nombre maximal de répéteurs placés sur le chemin de connexion situé entre 2 nœuds.

La valeur potentielle est la suivante:

- a) architecture en étoile: 0 à 7; et la valeur par défaut est 3;
- b) architecture en boucle: 0 à 252; la valeur par défaut est 30.

4.6.2.25 V(MTHT), P(MTHT): Maximum-token-hold-time (temps de maintien maximal des jetons)

Cette variable comporte et désigne la valeur du maximum-token-hold-time pour une transmission de données cycliques à grande vitesse. La plage de cette valeur est comprise entre 1 et $2^{16}-1$, dont l'unité est donnée en temps octet par le débit binaire équivalent. La valeur par défaut est 0x30B4, soit environ 900 μ s pour le fonctionnement à 100 Mbps et 90 μ s pour le fonctionnement à 1 000 Mbps.

4.6.2.26 V(NONC), P(NONC): Comptage répétitif admissible de non CMP

Cette variable comporte et désigne la valeur de comptage répétitif admissible de trames non CMP reçues par le nœud SYN dans les cycles Tsyn consécutifs correspondants, c'est-à-dire 256 fois par la valeur V(SCMPL), afin de ne détecter aucun autre nœud que le nœud SYN dans le bus de terrain de Type 11. La plage de cette valeur est comprise entre 1 et 16, et la valeur par défaut est 3.

4.6.2.27 C(NONC): Comptage NONC

La valeur C(NONC) indique le nombre de comptages répétitifs de trames non CMP reçues par le nœud SYN compris dans les valeurs V(MN) x V(SCMPL) des cycles Tsyn consécutifs. La valeur C(NONC) parvient au nombre de V(NONC), le nœud SYN détecte et reconnaît l'accrochage d'aucun autre nœud au bus de terrain de Type 11.

4.6.2.28 V(PBh), P(PBh) Nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à grande vitesse

Cette variable comporte et désigne le nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à grande vitesse.

4.6.2.29 V(PBm), P(PBm) Nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à moyenne vitesse

Cette variable comporte et désigne le nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à moyenne vitesse.

4.6.2.30 Q(QBF_{SPD}): Registre de file d'attente pour données sporadiques

Le registre de file d'attente dédié au service de données de messagerie sporadiques comporte la DLPDU à transmettre par l'utilisateur DL. La taille du registre de file d'attente dédié au transfert, ainsi qu'à la réception des données de messagerie sporadiques, dépend de la mise en œuvre.

4.6.2.31 V(SCMP), P(SCMP): Medium-access-control-time for substitute CMP (Temps de contrôle d'accès au support du CMP de substitution)

Cette variable comporte et désigne la valeur de la période d'observation maximale nécessaire pour que le nœud intervienne tout comme le nœud ayant reçu la trame CMP, même si le nœud correspondant n'est pas parvenu à envoyer la trame CMP.

4.6.2.32 T(SCMP): Appareil de surveillance SCMP

La DLE du nœud utilise la variable T(SCMP) pour mesurer le temps SCMP écoulé pour déclencher l'action semblable à celle déclenchée par le nœud ayant reçu la trame CMP, même si le nœud correspondant n'est pas parvenu à envoyer la trame CMP. La valeur est réduite dans la plage comprise entre V(SCMP) et 0.

4.6.2.33 V(SCMPL), P(SCMPL): Comptage répétitif admissible du CMP de substitution

Cette variable comporte et désigne la valeur du comptage répétitif admissible au sein duquel le nœud peut se comporter comme le nœud ayant reçu la trame CMP sans erreur, même si le nœud correspondant n'est pas parvenu à envoyer la trame CMP. La plage de cette valeur est comprise entre 1 et 16, et la valeur par défaut est 3.

4.6.2.34 C(SD): Comptage cumulé de la trame de données sporadiques transmise

C(SD) indique le nombre de comptages cumulés des trames de données sporadiques transmises sur le support. C(SD) est un compteur binaire de remplacement d'une longueur de 32 bits.

4.6.2.35 V(SL), P(SL): Temps de silence

Cette variable comporte et désigne la valeur du temps de silence ou de la période d'inactivité permettant de détecter le nœud SYN actuel défaillant, suite au déclenchement de l'envoi de la trame CLM à l'extérieur de ce nœud, afin de revendiquer un nouveau nœud SYN si celui-ci est désigné et admis comme un nœud SYN. La valeur est réglée par la DLE sur le temporisateur T(SL) de la manière suivante:

$$T(SL) = V(TSYN) + V(ST) \times V(TN) \times 2$$

4.6.2.36 T(SL): Appareil de surveillance SL

La DLE utilise la variable T(SL) pour mesurer le temps écoulé de SL. La valeur est réduite dans la plage comprise entre V(SL) et 0.

4.6.2.37 V(SM), P(SM): Masque d'adresse de sous réseau

Cette variable comporte et désigne la valeur de masque d'adresse de sous réseau pour la variable V(IP) attribuée pour le service de données de messagerie sporadiques de bus de terrain de Type 11.

4.6.2.38 V(ST), P(ST), T(ST): Créneau de bus de terrain de Type 11

Le créneau est l'unité de temps d'observation fondamentale, utilisée dans la DLME pour l'observation du déclenchement de l'action telle que la réinitialisation de l'envoi de la trame CLM, pour l'observation du déclenchement de l'action selon laquelle le nœud se comporte comme la trame CMP reçue, même si le nœud correspondant n'est pas parvenu à envoyer la trame CMP.

Cette variable comporte et désigne la valeur de temps d'observation fondamentale du créneau. La plage de la valeur est comprise entre 1 et 255, dont le temps est compris dans le temps de symbole physique de 512 bits pour le fonctionnement à 100 mbps et de 4 096 bits pour le fonctionnement à 1 000 Mbps spécifié et identique à celui défini dans l'ISO/CEI 8802-3. La valeur par défaut est 20, ce qui équivaut à environ 100 µs pour le fonctionnement à 100 Mbps et à 8 µs pour le fonctionnement à 1 000 Mbps, c'est-à-dire $20 \times 512 \times 1/100 \mu s$ pour le fonctionnement à 100 Mbps.

La formule mathématique suivante permet de calculer la valeur de créneau:

$$V(ST) = \text{arrondi} [2 \times V(MD) + V(MRT)/2 + 2]$$

4.6.2.39 V(Th), P(Th): Période de transmission à grande vitesse

Cette variable comporte et désigne la période cyclique de transmission de données prioritaires à grande vitesse. La DLE utilise la variable P(Th) dont la valeur est réglée sur V(SYN). La plage de cette valeur est comprise entre 1 et 160, dont l'unité est 0,1 ms. La valeur par défaut est 100.

4.6.2.40 V(TISPD), P(TISPD): Time-interval for sporadic-message-data-service (Intervalle de temps pour le service de données de messagerie sporadiques)

Cette variable comporte et désigne la valeur de l'intervalle de temps à traitement cyclique pour le service Sporadic-message-data. La plage de cette valeur est comprise entre 1 et 1 000, dont l'unité est 1 ms. La valeur par défaut est 100.

4.6.2.41 T(TISPD): Appareil de surveillance TISPD

La DLE utilise la variable T(TISPD) pour mesurer l'intervalle de temps de TISPD.

4.6.2.42 V(Tm), P(Tm): Période de transmission à moyenne vitesse

Cette variable comporte et désigne la période cyclique de transmission de données prioritaires à moyenne vitesse. La DLE utilise la variable P(Tm). La plage de cette valeur est comprise entre 10 et 1 000, dont l'unité est 1 ms. La valeur par défaut est 100.

4.6.2.43 V(TMAC), P(TMAC): Temps de contrôle d'accès au support pour la sollicitation de nouveaux nœuds

Cette variable comporte et désigne la valeur de la période maximale d'observation pour le nœud SYN afin de solliciter de nouveaux nœuds dans le bus de terrain de Type 11. Au cours de la période, les nouveaux nœuds tentent de transmettre la trame REQ au nœud SYN. La valeur par défaut est 100, dont l'unité est 0,1 ms pour le fonctionnement à 100 Mbps et 0,01 ms pour le fonctionnement à 1 000 Mbps.

4.6.2.44 T(TMAC): Appareil de surveillance TMAC

La DLE du nœud SYN utilise la variable T(TMAC) pour mesurer le temps écoulé de TMAC dans l'attente des trames REQ transmises par le nœud qui tente d'entrer dans le bus de terrain de Type 11. La valeur est réduite dans la plage comprise entre V(TMAC) et 0.

4.6.2.45 V(TN), P(TN): This-node (Ce nœud)

Cette variable comporte et désigne le numéro d'identifiant de ce nœud. La valeur initiale est 0 à la mise sous tension ou réinitialisation de ce nœud. Cette valeur se situe dans la plage comprise entre 1 et 255.

4.6.2.46 V(TSI): Nombre de DLE capables d'envoyer l'emplacement suivant

La DLE utilise cette variable pour comporter et indiquer le numéro d'identifiant de nœud de V(TN) dont le nœud peut envoyer ses données suivantes dans un ordre séquentiel successif à un nœud qui vient d'envoyer toutes ses données et DLPDU CMP ou DT-CMP. La valeur de cette variable est actualisée et réglée à la valeur de V(TN) du nœud suivant à envoyer, à chaque réception d'une DLPDU CMP ou DT-CMP. Cette variable se situe dans la plage comprise entre 1 et 255.

4.6.2.47 V(TSYN): Temps périodique cible de synchronisation

Cette variable indique la période cible entre deux arrivées de trame SYN. La valeur est égale à V(Th), et la valeur de P(Th) est utilisée par le DLMS et réglée sur V(Th). La plage potentielle de cette période est comprise entre 0,1 ms et 160 ms. V(SYN) et V(Th) indiquent la période cyclique au cours de laquelle tous les nœuds contenant des données cycliques à grande vitesse peuvent obtenir le droit de transmettre la totalité de ces données.

4.6.2.48 V(TTTR1), P(TTTR1): Temps de rotation du jeton cible de la classe d'accès 1

Cette variable comporte et désigne la valeur du temps de rotation du jeton cible de la classe d'accès 1, notamment pour la transmission de données de messagerie sporadiques. La plage de cette valeur est comprise entre 1 et $2^{16}-1$, dont l'unité est donnée en temps octet par le débit binaire équivalent. La valeur par défaut est P(TSYN). Cette valeur est utilisée par la DLE, et il convient qu'elle soit égale et commune à toutes les DLE du bus de terrain de Type 11.

La DLME utilise la variable P(TTTR1), la valeur correspondant à la période au cours de laquelle il est supposé qu'un nœud contenant des données de messagerie sporadiques peut obtenir un droit de transmission.

4.6.2.49 **V(TTTR2), P(TTTR2): Temps de rotation du jeton cible de la classe d'accès 2**

Cette variable comporte et désigne la valeur du temps de rotation du jeton cible de la classe d'accès 2, notamment pour la transmission de données cycliques à moyenne vitesse. La plage de cette valeur est comprise entre 1 et $2^{16}-1$, dont l'unité est donnée en temps octet par le débit binaire équivalent. La valeur par défaut est P(TSYN). Cette valeur est utilisée par la DLE, et il convient qu'elle soit égale et commune à toutes les DLE du bus de terrain de Type 11.

La DLME utilise la variable P(TTR2), la valeur correspondant à la période prévue au cours de laquelle tous les nœuds contenant des données cycliques à moyenne vitesse peuvent obtenir un droit de transmission et envoyer la totalité des données cycliques à moyenne vitesse.

4.6.3 **Variables, paramètres, compteurs, temporisateurs et files d'attente spécifiques à l'architecture en étoile**

4.6.3.1 **T(AIGP_A), T(AIGP_B): Appareil de surveillance IGP sur les canaux de réception A et B**

La DLE utilise les variables T(AIGP_A) et T(AIGP_B) pour mesurer le temps écoulé de IGP sur les canaux de réception A et B respectivement. La valeur est réduite dans la plage comprise entre 24 et 0, la valeur 24 équivalant à 0,96 µs.

4.6.3.2 **T(ARMGP): Appareil de surveillance RMGP**

La DLE utilise la variable T(ARMGP) pour surveiller le temps écoulé de RMGP. La valeur est réduite dans la plage comprise entre 250 et 0, la valeur 250 équivalant à 10 µs.

4.6.3.3 **T(ATTRT0): Appareil de surveillance TTRT0**

La DLE utilise la variable T(ATTRT0) pour surveiller la période de TTRT0. La valeur est réduite dans la plage comprise entre $2^{16}-1$ et 1, dont l'unité est donnée en temps octet par le débit binaire équivalent.

4.6.3.4 **C(CDI): Comptage cumulé de la trame de données cycliques à basse vitesse transmise**

C(CDI) indique le nombre de comptages cumulés des trames de données cycliques à basse vitesse transmise sur le support. C(CDI) est un compteur binaire de remplacement d'une longueur de 32 bits.

4.6.3.5 **V(HTI), P(HTI): Période d'observation maximale permettant de détecter le cycle de transmission continu à basse vitesse**

Cette variable comporte et désigne la valeur de la période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à basse vitesse. La valeur se situe dans la plage comprise entre 2 et $2^{16}-1$, dont l'unité est 1 ms. La valeur par défaut est $3 \times V(\text{Th})$.

4.6.3.6 **T(HTI): Appareil de surveillance HTI**

La DLE utilise la variable T(HTI) pour mesurer le temps HTI écoulé pour détecter l'interruption du cycle de transmission continu à basse vitesse. La valeur est réduite dans la plage comprise entre V(HTI) et 1.

4.6.3.7 V(MDD), P(MDD): Maximum-difference-of-the-distance (Différence maximale de la distance)

Cette variable, définie par le DLMS, comporte la différence maximale en mètres de la distance de deux supports physiques redondants placés sur le chemin de connexion situé entre 2 nœuds. La plage de cette valeur est comprise entre 0 et 2 000, et la valeur par défaut est 500.

4.6.3.8 V(MPD), P(MPD): Maximum-distance-of-the-signal-propagate-delay (Distance maximale du délai de propagation de signal)

Cette variable, définie par le DLMS, comporte la différence maximale en microsecondes du délai de propagation de signal sur deux supports physiques redondants placés sur le chemin de connexion situé entre 2 nœuds. La plage de cette valeur est comprise entre 1 et 0x7c, et la valeur par défaut est 0x7c, ce qui équivaut à environ 5 µs, c'est-à-dire 0x7c (= 124) x 0,04 µs.

4.6.3.9 C(NCD_A), C(NCD_B): Comptage cumulé d'absence de porteuse sur le canal de réception A ou B

C(NCD_A) ou C(NCD_B) indique le nombre de comptages cumulés d'absence de porteuse sur un canal de réception, tandis que ladite porteuse est présente de l'autre côté du canal de réception. C(NCD_A) et C(NCD_B) représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.3.10 V(PBI), P(PBI): Nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à basse vitesse

Cette variable comporte et désigne le nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à basse vitesse. La valeur est comprise entre 0 et 2 048.

4.6.3.11 V(RCS), P(RCS): Contrôle de commutation du canal de réception des trames de réception

Cette variable comporte et désigne la commande de commutation dédiée à la réception des trames par l'un des deux canaux de réception A et B correspondant à chacun des supports de redondance A et B.

La valeur potentielle est "Automatic" ("Automatique"), "Force A" ou "Force B". "Automatic" indique que la commande de commutation des trames de réception bascule de manière automatique sur le canal de réception approprié. "Force A" ou "Force B" indique que la commande de commutation est contrainte de basculer sur le canal de réception A ou B respectivement. La valeur par défaut est "Automatic".

4.6.3.12 V(RCS_A), V(RCS_B): Canal de réception A ou B choisi pour la réception des trames

Cette variable indique que le canal de réception correspondant est choisi pour la réception des trames provenant de l'autre nœud. La valeur "True" indique que le canal de réception correspondant est choisi, et la valeur "False" indique le contraire.

4.6.3.13 C(RE_A), C(RE_B): Comptage cumulé de DLPDU reçues sans erreur sur le canal de réception A ou B

C(RE_A) ou C(RE_B) indique le nombre de comptages cumulés de DLPDU reçues avec erreur, tandis qu'aucune erreur ne se produit de l'autre côté du canal de réception. C(RE_A) et C(RE_B) représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.3.14 **V(RMGP), P(RMGP): IGP sur d'autres supports de redondance**

Cette variable comporte et désigne la valeur de l'intervalle de temps maximal d'un canal de réception, qui a déjà terminé une trame reçue et a attendu pendant la durée de l'intervalle IGP la fin d'une trame à recevoir sur l'autre canal de réception afin de détecter l'autre canal de réception interrompu. La valeur par défaut est 250, dont l'unité est en 0,04 μ s, et équivaut à 10 μ s.

4.6.3.15 **C(ROK_A), C(ROK_B): Comptage cumulé de DLPDU reçues sans erreur sur le canal de réception A ou B**

C(ROK_A) ou C(ROK_B) indique le nombre de comptages cumulés de DLPDU reçues sans erreur sur le canal de réception A ou B respectivement. C(ROK_A) et C(ROK_B) représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.3.16 **V(SEN_A), P(SEN_A); V(SEN_B), P(SEN_B): Emetteur A ou B activé pour l'envoi de trames**

Cette variable désigne l'émetteur A ou B activé pour l'envoi des trames, et indique l'état de l'émetteur A ou B activé ou désactivé. La valeur "True" est à activer, la valeur "False" ne devant pas être activée pour l'émetteur A ou B correspondant. La valeur par défaut est "True".

4.6.3.17 **V(TI), P(TI): Période de transmission à basse vitesse**

Cette variable comporte et désigne la période cyclique de transmission de données prioritaires à basse vitesse. La DLE utilise la variable P(TI). La plage de cette valeur est comprise entre 100 et 1 000, dont l'unité est 1 ms, et la valeur par défaut est 100.

4.6.3.18 **V(TTRT0), P(TTRT0): Temps de rotation du jeton cible de la classe d'accès 0**

Cette variable comporte et désigne la valeur du temps de rotation du jeton cible de la classe d'accès 0, notamment pour la transmission de données cycliques à basse vitesse. La plage de cette valeur est comprise entre 1 et $2^{16}-1$, dont l'unité est donnée en temps octet par le débit binaire équivalent. La valeur par défaut est P(TSYN). Cette valeur est utilisée par la DLE, et il convient qu'elle soit égale et commune à toutes les DLE du bus de terrain de Type 11.

La DLME utilise la variable P(TTR0), la valeur correspondant à la période prévue au cours de laquelle tous les nœuds contenant des données cycliques à basse vitesse peuvent obtenir un droit de transmission et envoyer la totalité des données cycliques à basse vitesse.

4.6.4 **Variables, paramètres, compteurs, temporisateurs et files d'attente spécifiques à l'architecture en boucle**

4.6.4.1 **V(AN_A), V(AN_B): Numéro de nœud du nœud adjacent A, B**

Cette variable comporte le numéro de nœud du nœud adjacent. Les variables V(AN_A) et V(AN_B) indiquent le numéro de nœud du nœud adjacent au voisinage des ports en anneau A et B de ces nœuds respectivement. La valeur initiale à la mise sous tension ou réinitialisation de ce nœud est 0.

4.6.4.2 **T(ATTRT0): Appareil de surveillance TTRT0**

La DLE utilise la variable T(ATTRT0) pour surveiller la période de TTRT0 pour le fonctionnement à 1 000 Mbps. La valeur est réduite dans la plage comprise entre $2^{16}-1$ et 1, dont l'unité est donnée en temps octet par le débit binaire équivalent.

4.6.4.3 C(CDI): Comptage cumulé de la trame de données cycliques à basse vitesse transmise

C(CDI) indique le nombre de comptages cumulés des trames de données cycliques à basse vitesse transmise sur le support. C(CDI) est un compteur binaire de remplacement d'une longueur de 32 bits.

4.6.4.4 V(CRE_A), P(CRE_A), C(CRE_A); V(CRE_B), P(CRE_B), C(CRE_B): Comptage répétitif maximal admissible pour une erreur de réception anormale A, B

Cette variable, définie par la DLE, comporte et indique le nombre de comptages répétitifs observés de l'erreur de réception anormale détectée lors de la réception des trames entrantes provenant des ports en anneau A et B respectivement. La réception anormale se produit dans des conditions de réception anormale et de silence de réception de longue durée. Cette valeur se situe dans la plage comprise entre 15 et 0. La valeur par défaut est 2. Le comptage est réduit à chaque détection de réception anormale répétitive, mais réglé à la valeur par défaut lors de la suppression de la réception anormale. Lorsque la valeur atteint 0, la RMC reconnaît effectivement la défaillance.

4.6.4.5 V(ESYN), P(ESYN): Activation du nœud SYN

Cette variable est définie par la DLE et indique que la DLE peut être un nœud SYN. La valeur "Enable" est utilisée pour l'activation, à l'inverse de la valeur "Disable". La valeur par défaut est "Enable".

4.6.4.6 V(FWE_{AB}), P(FWE_{AB}), V(FWE_{BA}), P(FWE_{BA}): Etat d'acheminement AB, BA

Cette variable comporte et désigne le port en anneau A ou B activé pour acheminer les trames reçues du port en anneau A au port en anneau B ou inversement, respectivement.

La valeur "Enable" (activer) est à activer, la valeur "Disable" (désactiver) ne devant pas être activée pour le port en anneau A ou B correspondant. La valeur par défaut est "Disable".

4.6.4.7 V(HTI), P(HTI): Période d'observation maximale permettant de détecter le cycle de transmission continu à basse vitesse

Cette variable comporte et désigne la valeur de la période d'observation maximale permettant de détecter l'interruption du cycle de transmission continu à basse vitesse pour le fonctionnement à 1 000 Mbps. La valeur se situe dans la plage comprise entre 2 et $2^{16}-1$, dont l'unité est 1 ms. La valeur par défaut est $3 \times V(Th)$.

4.6.4.8 T(HTI): Appareil de surveillance HTI

La DLE utilise la variable T(HTI) pour mesurer le temps HTI écoulé pour détecter l'interruption du cycle de transmission continu à basse vitesse pour le fonctionnement à 1 000 Mbps. La valeur est réduite dans la plage comprise entre V(HTI) et 1.

4.6.4.9 V(LINK_A), V(LINK_B): Etat de liaison A, B

Cette variable indique l'état de liaison des ports en anneau A et B de ce nœud. La valeur de cette variable est "normal" ou "abnormal" (anormal) selon la connexion de liaison physique à la ligne principale spécifiée par l'ISO/CEI 8802-3.

4.6.4.10 V(LSYN): Numéro de nœud à l'état maître

Cette variable indique le numéro de nœud à l'état maître actuel.

4.6.4.11 **V(MDAD), P(MDAD): Distance maximale entre nœuds adjacents**

La valeur de cette variable, définie par la DLE, comporte la distance maximale en kilomètres entre les nœuds adjacents. La plage de cette valeur est comprise entre 1 et 2 pour le fonctionnement à 100 Mbps et entre 1 et 5 pour le fonctionnement à 1 000 Mbps. La valeur par défaut est 1.

4.6.4.12 **C(NCD_A), C(NCD_B): Comptage cumulé d'absence de porteuse détectée A, B**

C(NCD_A) ou C(NCD_B) indique le nombre de comptages cumulés d'absence de porteuse détectée sur le port en anneau A ou B respectivement, tandis que ladite porteuse est présente sur l'autre port. C(NCD_A) et C(NCD_B) représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.4.13 **V(NOS): Etat de fonctionnement du nœud**

Cette variable comporte et indique l'état de fonctionnement du nœud, à savoir l'état normal ou ouvert. La valeur de cette variable est soit "Normal", soit "Open" (ouvert) selon l'état normal ou l'état ouvert respectivement. La valeur initiale à la mise sous tension ou réinitialisation de ce nœud est "Open" (ouvert).

4.6.4.14 **V(NRF_A), P(NRF_A), T(NRF_A); V(NRF_B), P(NRF_B), T(NRF_B): Période d'observation maximale pour aucune trame reçue A, B**

Cette variable est réglée et utilisée par la DLE pour surveiller la période au cours de laquelle aucune trame n'a été reçue du port en anneau A ou B, en provenance de l'autre port, c'est-à-dire le port en anneau B ou A respectivement, après réception d'une trame. Cette valeur se situe dans la plage comprise entre 15 et 0. La valeur par défaut est 2, dont l'unité est égale à V(TSYN). Le temporisateur est réglé sur 2 et commence à chaque réception d'une trame provenant de l'autre port, et est diminué pendant la phase au cours de laquelle le port concerné ne transmet aucune trame.

4.6.4.15 **V(NS): Etat de nœud**

Cette variable comporte et indique l'état de nœud, à savoir l'état maître ou l'état esclave. La valeur de cette variable est "Master" (maître) lorsque l'état est l'état maître, ou "slave" (esclave) lorsque l'état est l'état esclave. La valeur initiale à la mise sous tension ou réinitialisation de ce nœud est "Slave".

4.6.4.16 **V(PBI), P(PBI): Nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à basse vitesse**

Cette variable comporte et désigne le nombre maximal de registres de données utilisés pour l'envoi de données cycliques prioritaires à basse vitesse pour le fonctionnement à 1 000 Mbps. Cette valeur se situe dans la plage comprise entre 1 et 4 096.

4.6.4.17 **V(PP): Port primaire**

Cette variable comporte et indique le port par lequel a eu lieu la première réception de trame SYN depuis l'activation de ce nœud, le démarrage ou l'initialisation du système. La valeur de cette variable est "Ring-port-A" (port en anneau A) ou "Ring-port-B" (port en anneau B).

4.6.4.18 **V(PS_A), P(PS_A); V(PS_B), P(PS_B): Etat de port en anneau A, B**

Ces variables comportent et désignent l'état du port de ce nœud. Les variables V(PS_A), P(PS_A) et V(PS_B), P(PS_B) sont définies par la DLE et comportent l'état du port en anneau A ou B, respectivement. La valeur est soit "Blocked" (bloqué) ou "Non-blocked" (non bloqué). Chaque valeur initiale à la mise sous tension ou réinitialisation de ce nœud est "Blocked".

4.6.4.19 $C(RE_A)$, $C(RE_B)$: Comptage cumulé de la trame reçue avec erreur A, B

$C(RE_A)$ ou $C(RE_B)$ indique le nombre de comptages cumulés de la trame reçue avec erreur provenant du port en anneau A ou B respectivement, aucune erreur ne se produisant sur l'autre port. $C(RE_A)$ et $C(RE_B)$ représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.4.20 $V(REN_A)$, $P(REN_A)$; $V(REN_B)$, $P(REN_B)$: Activation de réception A, B

Ces variables comportent et désignent l'activation de réception du port en anneau A ou B de ce nœud. Les variables $V(REN_A)$, $P(REN_A)$ et $V(REN_B)$, $P(REN_B)$ sont définies par la DLME et comportent l'état de l'activation de réception de trames en provenance du port en anneau A ou B, respectivement.

La valeur de cette variable est "Enable" (activer) ou "Disable" (désactiver). La valeur par défaut est "Enable".

4.6.4.21 $C(ROK_A)$, $C(ROK_B)$: Comptage cumulé de la trame reçue sans erreur A, B

$C(ROK_A)$ ou $C(ROK_B)$ indique le nombre de comptages cumulés de la trame reçue sans erreur en provenance du port en anneau A ou B respectivement. $C(ROK_A)$ et $C(ROK_B)$ représentent des compteurs binaires de remplacement d'une longueur de 32 bits.

4.6.4.22 $V(SEN_A)$, $P(SEN_A)$; $V(SEN_B)$, $P(SEN_B)$: Activation d'envoi A, B

Ces variables comportent et désignent l'activation d'envoi hors du port en anneau A ou B de ce nœud. Les variables $V(SEN_A)$, $P(SEN_A)$ et $V(SEN_B)$, $P(SEN_B)$ sont définies par la DLME et comportent l'état de l'activation d'envoi de trames hors du port en anneau A ou B, respectivement.

La valeur de cette variable est "Enable" (activer) ou "Disable" (désactiver). La valeur par défaut est "Enable".

4.6.4.23 $V(TI)$, $P(TI)$: Période de transmission à basse vitesse

Cette variable comporte et désigne la période cyclique de transmission de données prioritaires à basse vitesse pour le fonctionnement à 1 000 Mbps. La DLE utilise la variable $P(TI)$. La plage de cette valeur est comprise entre 1 et 10 000, dont l'unité est 1 ms, et la valeur par défaut est 100.

4.6.4.24 $V(TTROT)$, $P(TTROT)$: Temps de rotation du jeton cible de la classe d'accès 0

Cette variable comporte et désigne la valeur du temps de rotation du jeton cible de la classe d'accès 0, notamment pour la transmission de données cycliques à basse vitesse. La plage de cette valeur est comprise entre 1 et $2^{16}-1$, dont l'unité est donnée en temps octet par le débit binaire équivalent. La valeur par défaut est $P(TSYN)$. Cette valeur est utilisée par la DLE, et il convient qu'elle soit égale et commune à toutes les DLE du bus de terrain de Type 11.

La DLME utilise la variable $P(TTROT)$, la valeur correspondant à la période prévue au cours de laquelle tous les nœuds contenant des données cycliques à basse vitesse peuvent obtenir un droit de transmission et envoyer la totalité des données cycliques à basse vitesse.

5 Structure générale et codage de PhIDU et DLPDU et des éléments de procédure connexes

5.1 Présentation

La DLL et ses procédures sont nécessaires pour fournir les services proposés à l'utilisateur DLS à l'aide des services disponibles auprès de PhL. L'Article 5 décrit la structure et la sémantique des PhIDU, MA_PDU, DLPDU, ainsi que la procédure couramment utilisée dans la présente norme. Cette partie est néanmoins identique et totalement conforme à l'ISO/CEI 8802-3.

Dans l'Article 5, toute référence au bit k d'un octet est une référence au bit dont le poids dans un octet non signé est 2^k , ceci étant parfois désigné comme une numérotation de bits "petit-boutiste".

5.2 Structure PhIDU et codage

La sous-couche MAC locale utilise les primitives de service fournies par la sous-couche PLS spécifiée par l'Article 6 de l'ISO/CEI 8802-3:2000. Toutes les primitives de service fournies par la sous-couche PLS se présentent comme suit et sont considérées comme obligatoires:

- a) demande PLS_DATA;
- b) indication PLS_DATA;
- c) indication PLS_CARRIER;
- d) indication PLS_SIGNAL;
- e) indication PLS_DATA_VALID.

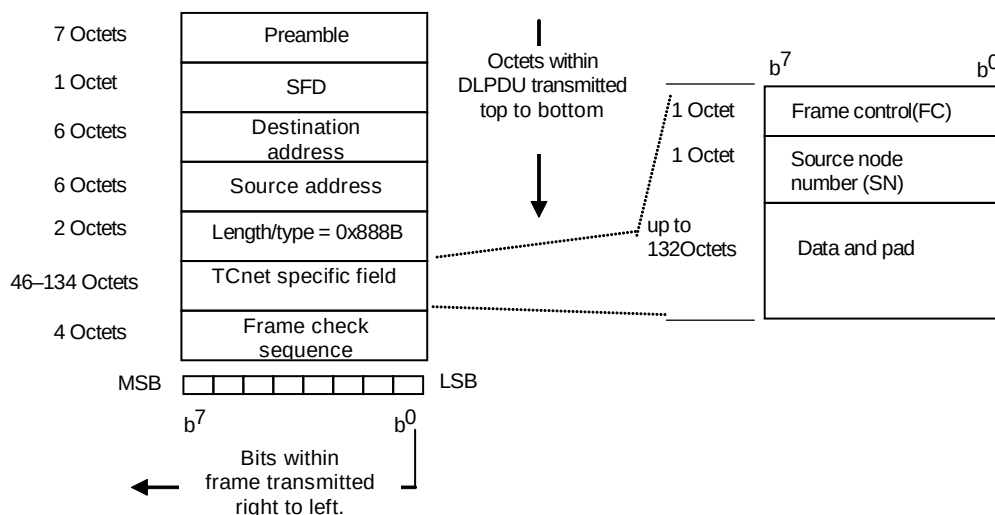
Dans le cas où une spécification d'au moins 100 Mbps est appliquée, la sous-couche de réconciliation met en correspondance les signaux fournis au niveau de la MII et de la GMII pour le fonctionnement à 1 000 Mbps avec les primitives de service PLS (voir l'Article 22 de l'ISO/CEI 8802-3:2000 et l'Article 35, respectivement). Les primitives de service PLS fournies par la sous-couche de réconciliation se comportent exactement tel que défini à l'Article 6 de l'ISO/CEI 8802-3:2000.

5.3 Structure, codage et éléments de procédure de la trame MAC commune

5.3.1 Structure de trame MAC

5.3.1.1 Format de trame MAC commune pour la DLPDU de bus de terrain de Type 11

La structure de la trame MAC dédiée à la DLPDU du bus de terrain de Type 11 est intégrée dans le format de trame de Ethernet V2.0 spécifié par l'Article 3 de l'ISO/CEI 8802-3:2000. La valeur du champ Longueur/Type est définie comme étant égale à 0x888B, qui est une valeur autorisée et enregistrée comme le numéro d'identification de protocole par la IEEE Registration Authority, devant être identifiée comme la trame de bus de terrain de Type 11. La Figure 6 illustre le format de trame MAC utilisé, commun à la DLPDU de bus de terrain de Type 11, sauf pour la transmission de données sporadiques de ce même bus.



Légende

Anglais	Français
Preamble	préambule
Octets within DLPDU transmitted top to bottom	Transmission des octets de la DLPDU du haut vers le bas
Destination address	Adresse de destination
Source address	Adresse source
Frame control (FC)	Contrôle de trame (FC)
Source node number (SN)	Numéro de nœud de la source (SN)
Length/type	Longueur/type
Up to	Jusqu'à
Data and pad	Données et assembleur-désassembleur
TCnet specific field	Champ spécifique TCnet
Frame check sequence	Séquence de contrôle de trame
Bits within frame transmitted right to left	Transmission des bits dans la trame de droite à gauche

Figure 6 – Format de frame MAC commune pour les DLPDU

5.3.1.2 Format de frame MAC pour la DLPDU sporadique de bus de terrain de Type 11

Le format de frame MAC utilisé pour la transmission de données sporadiques du bus de terrain de Type 11 doit être conforme à l'ISO/CEI 8802-3:2000, Figure 3-1.

5.4 Éléments de la trame MAC

5.4.1 Généralités

Les éléments de la trame MAC sont le préambule, le délimiteur de trame de début, l'adresse de destination, l'adresse source, le code de longueur/type et la séquence de contrôle de trame, tous ces éléments étant tel que spécifié par l'Article 3 de l'ISO/CEI 8802-3:2000.

5.4.2 Champ de préambule

Le préambule de la trame MAC est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. Le champ de préambule est un champ de 7 octets utilisé pour permettre aux circuits de la partie de signalisation physique de parvenir à leur synchronisation de régime permanent avec la temporisation de la trame de réception.

Le modèle de préambule est le suivant:

“10101010 10101010 10101010 10101010 10101010 10101010 10101010”

5.4.3 Délimiteur de trame de début (SFD)

Le délimiteur de trame de début (SFD) est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. Le champ SFD est la séquence de configuration binaire "10101011". Il suit immédiatement le modèle de préambule et indique le début d'une trame.

5.4.4 Champ d'adresse

5.4.4.1 Généralités

Les champs d'adresse (adresse de destination et adresse source) ont une structure et une sémantique identiques au champ d'adresse de la trame MAC de base, décrit dans l'Article 3 de l'ISO/CEI 8802-3:2000. Chaque champ d'adresse doit avoir une longueur de 48 bits.

5.4.4.2 Champ d'adresse de destination (DA)

Le champ d'adresse de destination (DA) est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. Ce champ spécifie la (les) station(s) à laquelle/lesquelles la trame est destinée. Il peut s'agir d'une adresse individuelle ou multidiffusion (y compris diffusion). La valeur de DA pour la DLPDU de bus de terrain de Type 11, sauf pour la transmission de données sporadiques de bus de terrain de Type 11, est toujours réglée sur la valeur V(MGA).

5.4.4.3 Champ d'adresse source (SA)

Le champ d'adresse source (SA) est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. Ce champ spécifie la station de transmission de la trame. Le champ d'adresse source n'est pas interprété par la DLE, ainsi que la sous-couche MAC CSMA/CD. La valeur de SA est toujours réglée sur la valeur V(IA).

5.4.5 Champ de longueur/type

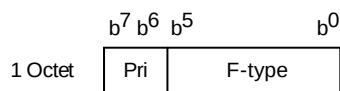
Le champ de longueur/type est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. La valeur du champ de longueur/type, pour pouvoir être identifiée comme une trame de bus de terrain de Type 11, est réglée sur 0x888B, valeur autorisée et enregistrée comme numéro d'identification de protocole pour le bus de terrain de Type 11 par la IEEE Registration Authority. Chaque trame dont la valeur est pratiquement toujours 0x888B est identique à la trame conformément à l'Article 3 de l'ISO/CEI 8802-3:2000, et est traitée comme une trame de données sporadiques de bus de terrain de Type 11.

5.4.6 Champ de contrôle de trame (FC)

5.4.6.1 Structure du champ de contrôle de trame

La structure du champ de contrôle de trame est présentée dans la Figure 7.

La DLE utilise le champ de type F pour identifier et désigner le type de trame du bus de terrain de Type 11 pour la commande d'accès au support. La DLE utilise le champ Pri pour désigner et identifier la classe de service de la trame de données cycliques prioritaires.



Légende

Pri = Priorité

F-type = Type de trame

Figure 7 – Structure du champ de contrôle de trame (FC)

5.4.6.2 Champ de type de trame (type F)

Le Tableau 7 présente la liste des types F, les valeurs et les trames de bus de terrain de Type 11 correspondants. Tous les types F sont réservés pour une utilisation ultérieure, sauf ceux figurant dans le Tableau 7.

Tableau 7 – F-type: Type de DLPDU

Valeur de type F	Type de trame	Nom de trame
0x00	CLM	Trame d'appel
0x01	SYN	Trame de synchronisation
0x02, 0x22	REQ	Trame de demande dans l'anneau -Architecture en étoile: la valeur est 0x02 -Architecture en boucle: la valeur est 0x22
0x04	COM	Trame de commande
0x05	RAS	Trame RAS
0x07	DT	Trame de données cycliques
0x08	CMP	Trame complète d'émission
0x0F	DT-CMP	DT avec trame complète d'émission
0x23	LPD	Trame de diagnostic de boucle
0x26	LRR	Trame de demande de répétition de boucle

5.4.6.3 Champ de priorité (Pri)

La DLE utilise le champ de priorité pour désigner et identifier la classe de service de la trame de données cycliques prioritaires, appliquée à la fois à DT et à DT-CMP. 3 niveaux de classe de service dédiés à l'architecture en étoile et à l'architecture en boucle sont prévus dans le fonctionnement à 1 000 Mbps. L'architecture en boucle pour le fonctionnement à 100 Mbps offre 2 niveaux de classe de service. Le niveau maximal à minimal étant le niveau 3 à 1. La transmission de données prioritaires à grande vitesse est attribuée au niveau 3 de la classe de service, la transmission de données prioritaires à moyenne vitesse est attribuée au niveau 2 et la transmission de données prioritaires à basse vitesse au niveau 1. La transmission de données prioritaires à basse vitesse n'est pas attribuée pour l'architecture en boucle pour le fonctionnement à 100 Mbps. Un autre type de trame, à l'exception du type de trame dédié à DT et DT-CMP, est attribué au niveau 3 de la classe de service.

5.4.7 Champ du numéro de nœud source (SN)

La valeur de SN est le numéro d'identifiant de nœud, égale à la valeur de V(TN) du nœud ayant transmis la DLPDU de bus de terrain de Type 11.

5.4.8 Champ de données et de l'assembleur-désassembleur de paquets

Le champ de données contient une séquence de N octets qui prévoit la transparence parfaite des données au sens où toute séquence arbitraire de valeurs d'octets peut apparaître dans le champ de données jusqu'à un nombre maximal spécifié par l'ISO/CEI 8802-3. Une taille de trame minimale est requise, c'est-à-dire la *mimFrameSize* définie par l'ISO/CEI 8802-3, et si une taille de trame est inférieure à la *mimFrameSize*, le champ de données est étendu par ajout de bits supplémentaires en unités d'octets. La trame, qui s'étend du champ DA au champ FCS inclus, correspond au moins aux bits *mimFrameSize*.

La structure de ce champ pour la DLPDU de bus de terrain de Type 11 est décrite à l'Article 6.

5.4.9 Séquence de contrôle de trame (FCS)

5.4.9.1 Généralités

La construction de séquence de contrôle de trame (FCS), le polynôme et la valeur résiduelle sont identiques à l'Article 3 de l'ISO/CEI 8802-3:2000.

Dans l'Article 5, toute référence au bit K d'un octet fait référence au bit dont le poids dans un entier non signé d'un octet est 2^K .

Parfois appelé numérotation de bit "petit-boutiste".

Pour la plupart des types de protocole définis dans la présente norme, comme dans les autres normes internationales (l'ISO/CEI 3309:1993, l'ISO/CEI 8802 et l'ISO/CEI 9314-2:1989, par exemple), la détection d'erreur au niveau de la DLPDU est assurée en calculant et ajoutant une séquence de contrôle de trame (FCS) à plusieurs bits aux autres champs DLPDU pendant la transmission. Il s'agit de former un "mot de code systématique"¹ de longueur n , composé de k bits de message DLPDU suivis de $n - k$ (égal à 32) bits redondants, puis de déterminer pendant la réception que le message et la FCS concaténée forment un mot de code (n,k) légal. Le mécanisme adapté à cette vérification se présente comme suit:

La forme générique du polynôme générateur de cette construction FCS est spécifiée dans l'équation (6) et le polynôme des données résiduelles du récepteur est spécifié dans l'équation (11). Les polynômes spécifiques pour chaque type de protocole DL sont spécifiés dans le Tableau 8.

Tableau 8 – Longueur, polynômes et constantes FCS

Elément	Valeur
$n-k$	32
$G(X)$	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$ a, b, c
$R(X)$	$X^{31} + X^{30} + X^{26} + X^{25} + X^{24} + X^{18} + X^{15} + X^{14} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$ c, d
a Les mots de code $D(X)$ construits à partir de ce polynôme $G(X)$ ont une distance de Hamming 4 pour les longueurs $\leq 22\ 901$ octets, une distance de Hamming 5 pour les longueurs ≤ 371 octets, une distance de Hamming 6 pour les longueurs ≤ 33 octets, une distance de Hamming 7 pour les longueurs ≤ 21 octets, et une distance de Hamming 8 pour les longueurs ≤ 11 octets. b Ce polynôme $G(X)$ est relativement important pour toutes les situations, et n'est ainsi pas remis en cause par l'un des polynômes couramment utilisés dans les DCE (modems): le polynôme de codage différentiel $1 + X^{-1}$ et tous les polynômes d'embrouillage primitifs de la forme $1 + X^{-i} + X^{-k}$. c Il s'agit des mêmes polynômes et méthodes que ceux spécifiés dans l'ISO/CEI 8802 (Ethernet). d Il convient que le $R(x)$ restant soit 1100 0111 0000 0100 1101 1101 0111 1011 (X^{31} à X^0, respectivement) en l'absence d'erreur.	

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

5.4.9.2 Au niveau de la DLE émettrice

Le message d'origine (c'est-à-dire la DLPDU sans FCS), la FCS et le mot de code de message composite (la DLPDU et la FCS concaténées) sont 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 $GF(2)$. Si les bits de message sont $m_1 \dots m_k$ et que les bits FCS sont $f_{n-k-1} \dots f_0$, où

- $m_1 \dots m_8$ forment le premier octet envoyé,
- $m_{8N-7} \dots m_{8N}$ forment le $N^{\text{ème}}$ octet envoyé,
- $f_7 \dots f_0$ forment le dernier octet envoyé, et
- 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 trame PhL),

Cet ordre "tel que transmis" est critique en fonction des propriétés de détection d'erreur de la FCS.

le vecteur de message $M(X)$ est alors le suivant

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

et le vecteur FCS $F(X)$ est le suivant

$$\begin{aligned} (X) &= f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= f_{31}X^{31} + \dots + f_0 \end{aligned} \quad (\text{si } k = 32) \quad (2)$$

Le vecteur composite $D(X)$, de la DLPDU complète, est construit comme la concaténation du message et des vecteurs FCS

$$\begin{aligned} D(X) &= M(X)X^{n-k} + F(X) \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{32} + f_{31}X^{31} + \dots + f_0 \end{aligned} \quad (\text{si } k = 32) \quad (3)$$

La DLPDU présentée à la PhL doit être composée d'une séquence d'octets dans l'ordre indiqué.

Les bits de contrôle redondants $f_{n-k-1} \dots f_0$ de la FCS sont 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 $n-k$ des mots de code

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \dots + 1 \quad (4)$$

et $L(X)$ est le polynôme de poids maximal (tous) de degré $n-k-1$

$$\begin{aligned} L(X) &= \frac{X^{n-k} + 1}{X + 1} = X^{n-k-1} + X^{n-k-2} + \dots + X + 1 \\ &= X^{31} + X^{14} + X^{13} + X^{12} + \dots + X^2 + X + 1 \end{aligned} \quad (\text{si } k = 32) \quad (5)$$

En d'autres termes,

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

Les termes $L(X)$ sont inclus dans le calcul afin de détecter la troncature ou l'extension initiale ou terminale de message en ajoutant un facteur dépendant de la longueur à la FCS.

Comme mise en œuvre classique lorsque $n-k = 32$, le reste initial de la division est prédéfini sur chacun d'eux. Le train de bits de message transmis est multiplié par X^{n-k} et divisé (module 2) par le polynôme générateur $G(X)$, spécifié dans la formule (4). Les compléments du reste résultant sont transmis comme FCS de $(n-k)$ bits, avec le coefficient de X^{n-k-1} transmis en premier.

5.4.9.3 Au niveau de la DLE de réception

La séquence d'octets indiquée par la PhE est concaténée dans la DLPDU et la FCS reçues, puis considérée comme un vecteur $V(X)$ de dimension u

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

En raison d'erreurs, u peut être différent de n , la dimension du vecteur de code transmis.

Un reste $R(X)$ est calculé pour $V(X)$, la DLPDU et la FCS reçues, par une méthode analogue à celle utilisée par la DLE émettrice (voir 5.4.9.2) dans le calcul de $F(X)$

$$\begin{aligned} R(X) &= L(X) X^u + V(X) X^{n-k} \text{ (module } G(X)) \\ &= r_{n-k-1}X^{n-k-1} + \dots + r_0 \end{aligned} \quad (8)$$

Définir $E(X)$ comme le vecteur de code d'erreur des différences (module 2) supplémentaires entre le vecteur de code transmis $D(X)$ et le vecteur reçu $V(X)$ résultant d'erreurs rencontrées (dans le fournisseur PhS) entre les DLE émettrices et destinataires.

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

Si aucune erreur ne s'est produite, de sorte que $E(X) = 0$, alors $R(X)$ est égal à un polynôme de reste d'une constante non nulle

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

dont la valeur est indépendante de $D(X)$. Malheureusement, $R(X)$ est aussi égal à $R_{ok}(X)$, lorsque $E(X)$ est un multiple exact non nul de $G(X)$, auquel cas des erreurs sont "indétectables". Dans tous les autres cas, $R(X)$ n'est pas égal à $R_{ok}(X)$. Ces DLPDU sont erronées et peuvent être écartées sans autre forme d'analyse.

Comme mise en œuvre classique, le reste initial de la division est prédéfini sur chacun d'eux. Le train de bits reçu est multiplié par X^{n-k} et divisé (module 2) par le polynôme générateur $G(X)$, spécifié dans la formule (4).

5.5 Ordre de transmission binaire

L'ordre de transmission binaire est identique à l'Article 3 de l'ISO/CEI 8802-3:2000. Chaque octet de la DLPDU, à l'exception de la FCS, doit être transmis avec le bit de poids faible en premier, et la FCS doit être transmise dans l'ordre spécifié en 5.4.9.2.

5.6 DLPDU non valide

Une DLPDU non valide doit être définie comme une DLPDU satisfaisant au moins à l'une des conditions suivantes, et est pratiquement identique à l'Article 3 de l'ISO/CEI 8802-3:2000:

- a) La longueur de trame n'est pas compatible avec une valeur de longueur spécifiée dans le champ de longueur/type. Si le champ de longueur/type contient une valeur de type définie par l'ISO/CEI 8802-3:2000, 3.2.6, la longueur de trame est alors supposée conforme à ce

champ, et il convient de ce fait de ne pas la considérer comme une DLPDU non valide sur cette base.

- b) La longueur n'est pas un nombre d'octets intégral.
- c) Les bits de la DLPDU entrante (à l'exclusion toutefois du champ FCS) ne génèrent aucune valeur CRC identique à celle reçue.
- d) Cette DLPDU n'est pas compatible avec une valeur de type F de DLPDU de bus de terrain de Type 11.

Le contenu de la DLPDU non valide ne doit pas être transféré à l'utilisateur DL ou à la DLE. L'occurrence d'une DLPDU non valide peut être communiquée à la gestion du réseau.

Une DLPDU non valide peut être ignorée, refusée ou utilisée de manière privée par un utilisateur DL autre qu'un utilisateur DL RTE. L'utilisation de ce type de DLPDU n'entre pas dans le domaine d'application de la présente norme.

6 Structure, codage et éléments de procédure spécifiques à la DLPDU

6.1 Généralités

L'Article 6 définit la structure, le contenu et le codage de chaque type et chaque format de DLPDU, sauf la DLPDU sporadique de bus de terrain de Type 11, et spécifie les éléments de procédure applicables à la DLPDU.

La structure, le contenu, les paramètres et le codage de la DLPDU sont décrits dans chaque paragraphe de l'Article 6, la partie spécifique de bus de terrain de Type 11 de la structure de DLPDU, présentée en 5.3.1.1 étant par ailleurs spécifiée. Les aspects relatifs à l'envoi et à la réception des utilisateurs DLS et de leurs DLE sont décrits plus en détail. L'Article 6 décrit l'ensemble des formats et codages de données au format petit-boutiste.

Dans l'Article 6, toute référence au bit K d'un octet est une référence au bit dont le poids dans un octet non signé est 2^K , ceci étant parfois désigné comme une numérotation de bits "petit-boutiste".

Lorsqu'une action conditionnelle est spécifiée et lorsque la condition d'activation spécifique ne se produit pas, l'action correspondante ne se produit également pas. L'effet qui en découle est que les événements ne satisfaisant pas aux conditions d'activation spécifiées dans une procédure de service, n'entraîneront aucune action conséquente par rapport à cette procédure de service spécifique.

6.2 DLPDU de synchronisation (SYN)

6.2.1 Généralités

Une DLE qui équivaut au nœud SYN ou qui fonctionne actuellement à l'état maître transmet la DLPDU SYN de manière périodique aux intervalles de temps spécifiques de $V(Th)$ pour la synchronisation de la transmission de données cycliques prioritaires par toutes les DLE associées dans le bus de terrain de Type 11.

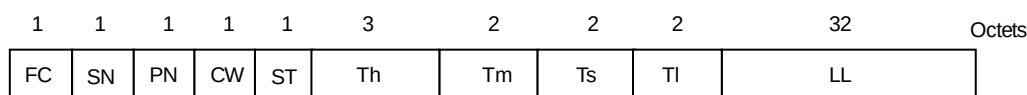
Une DLE qui essaie d'être active (en ligne), transmet une DLPDU REQ pour demander au nœud SYN de l'ajouter au bus de terrain de Type 11 dans les conditions dans lesquelles la DLE a reçu une DLPDU SYN, avec la valeur de PN archivée égale à la valeur $V(TN)$. Le délai, au cours duquel la DLPDU REQ est transmise, s'inscrit dans la période de contrôle MAC de T_{mac} . Cette période débute immédiatement après que la DLE a entièrement reçu une DLPDU CMP ou une DLPDU DT-CMP par le nœud, avec le nombre le plus élevé de la valeur de $V(TN)$.

Une DLE en ligne et fonctionnant comme un nœud SYN, commande la transmission de données cycliques prioritaires des DLPDU de bus de terrain de Type 11, sollicite l'ajout d'un

nouveau nœud, gère la perte de nœuds, commande et gère tous les paramètres associés afin de maintenir une transmission de données cycliques stable. Une DLE qui ne fonctionne pas comme un nœud SYN doit s'efforcer de devenir un nouveau nœud SYN afin de maintenir une transmission de données cycliques stable suite au nœud SYN actuel en cas de dysfonctionnement de ce dernier. Une DLE qui ne fonctionne pas comme un nœud SYN détermine l'ordre et la temporisation de la transmission de données cycliques prioritaires, en utilisant la valeur de V(LL) dans la DLPDU SYN reçue.

6.2.2 Structure de DLPDU SYN

La structure de DLPDU SYN est présentée dans la Figure 8.



Légende

FC = Contrôle de trame (11000001)	SN = Numéro de nœud de la source
PN = Numéro de nœud d'autorisation d'émission	CW = Mot de commande
ST = Créneau	Th = Temps périodique à grande vitesse
Tm = Temps périodique à moyenne vitesse	Ts = Temps de rotation de messages sporadiques
TI = Temps périodique à basse vitesse 2	LL = Liste opérationnelle

Figure 8 – Structure de DLPDU SYN

6.2.3 Paramètres de DLPDU SYN

6.2.3.1 Numéro de nœud d'autorisation d'émission (PN)

Le fonctionnement à 100 Mbps d'une architecture en étoile et en boucle liée au PN se présente comme suit:

- Une DLE qui s'efforce d'être en ligne est autorisée à transmettre une DLPDU REQ demandant au nœud SYN d'entrer dans le bus de terrain de Type 11 lorsque la valeur de PN dans la DLPDU SYN devient égale à la valeur de V(TN). La DLE peut transmettre une DLPDU REQ au nœud SYN dans le délai de la période de contrôle MAC au cours du créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN avec une valeur de PN égale à celle de V(TN).

Le fonctionnement à 1 000 Mbps d'une architecture en boucle liée au PN se présente comme suit:

- Une DLE est autorisée à envoyer au nœud SYN une DLPDU RAS à chaque fois et une DLPDU COM en cas d'événement lorsque la valeur de PN dans la DLPDU SYN devient égale à la valeur de V(TN). La DLE peut envoyer une DLPDU RAS et une DLPDU COM dans le délai de V(LN) égal à la valeur de V(TN) au cours du créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN avec une valeur de PN égale à celle de V(TN), et la DLE envoie la DLPDU RAS et la DLPDU COM avant d'envoyer DLPDU DT.

Cette valeur se situe dans la plage comprise entre 0x1 et 0xFF. La valeur est augmentée par chaque DLPDU SYN et passe de 0xFF à 0x1.

Le Tableau 9 illustre la structure du paramètre PN.

Tableau 9 – Paramètre PN: 3^{ème} octet

Numéro de nœud d'autorisation d'émission							
PN (1 à 255)							
7	6	5	4	3	2	1	0

6.2.3.2 Mot de commande (CW)

6.2.3.2.1 CW pour l'architecture en étoile

6.2.3.2.1.1 Structure de CW

La structure du mot de commande (CW) est présentée dans le Tableau 10.

Tableau 10 – Structure de CW: 4^{ème} octet

Mot de commande							
Mode périodique	Réservé					Sélection du support de redondance	
PM	0					RMSEL	
7	6	5	4	3	2	1	0

6.2.3.2.1.2 Paramètre de mode périodique (PM)

Le paramètre "PM" indique le mode de synchronisation pour la transmission de données cycliques prioritaires.

Le mode "exécution autonome" (free-run) est basé sur une transmission de données périodiques simple, c'est-à-dire que le nœud SYN transmet la trame SYN immédiatement après qu'il détecte que tous les nœuds ont transmis leurs données cycliques prioritaires dans la période de V(TSYN). D'autre part, le mode "période constante" (constant period) est basé sur la transmission de données à synchronisation temporelle avec émission d'un signal par une source horloge de période constante.

Le Tableau 11 illustre le mode du paramètre PN.

Tableau 11 – Paramètre PM

Mode PM	Valeur	Description
Exécution autonome	0	Transmission de données cycliques basée sur une exécution autonome
Période constante	1	Transmission de données cycliques basée sur une synchronisation temporelle

6.2.3.2.1.3 Sélection du support de redondance (RMSEL)

Ce paramètre indique et désigne le mode de sélection du support de redondance pour tous les nœuds. La DLE utilise la valeur de RMSEL et la règle sur V (RCS) pour la commande de commutation dédiée à la réception des trames par l'un des deux canaux de réception A et B correspondant à chacun des supports de redondance A et B. La valeur potentielle est "Automatic", "Force A" et "Force B".

"Automatic" indique que la commande de commutation des trames de réception bascule de manière automatique sur le canal de réception approprié. "Force A" ou "Force B" indique que la commande de commutation est contrainte de basculer sur le canal de réception A ou B respectivement. La valeur initiale est "Automatic".

Le mode du paramètre RMSEL est récapitulé dans le Tableau 12.

Tableau 12 – Paramètre RMSEL

Mode RMSEL	Valeur	Description
Automatic	00	Bascule automatiquement sur le canal de réception approprié.
–	01	Réservé à un usage ultérieur
Force A	10	Contraint de basculer sur le canal de réception A
Force B	11	Contraint de basculer sur le canal de réception B

6.2.3.2.2 CW de l'architecture en boucle

6.2.3.2.2.1 Structure de CW

La structure de CW est présentée dans le Tableau 13.

Tableau 13 – Structure de CW: 4^{ème} octet

Mot de commande							
Réservé (0x0)				PM	Réservé (0x0)		
7	6	5	4	3	2	1	0

6.2.3.2.2.2 Paramètre de mode périodique (PM)

Le paramètre "PM" indique le mode de synchronisation pour la transmission de données cycliques prioritaires. Le paramètre "PM" est égal à celui de l'architecture en étoile spécifié en 6.2.3.2.1.2.

6.2.3.3 Créneau (ST)

Ce paramètre est l'unité de temps d'observation fondamentale, utilisée dans la DLE pour observer l'action d'initiation telle que la réinitialisation pour la transmission de la trame CLM, et pour l'envoi de la trame CMP spécifiquement par le nœud SYN actuel en remplacement du nœud n'ayant pu transmettre ladite trame.

V(ST) a la valeur de ST. La plage de la valeur est comprise entre 1 et 255, dont le temps est compris dans le temps de symbole physique de 512 bits pour le fonctionnement à 100 mbps des architectures en étoile et en boucle, et de 4 096 bits pour le fonctionnement à 1 000 Mbps de l'architecture en boucle, spécifié et identique à celui défini dans l'ISO/CEI 8802-3. La valeur par défaut est 20, ce qui équivaut à environ 100 μ s pour le fonctionnement à 100 Mbps et à 8 μ s pour le fonctionnement à 1 000 Mbps, c'est-à-dire $20 \times 512 \times 1/100 \mu$ s pour le fonctionnement à 100 Mbps.

Le Tableau 14 illustre la structure du paramètre ST.

Tableau 14 – Paramètre ST: 5^{ème} octet

V(ST): Créneau							
7	6	5	4	3	2	1	0

6.2.3.4 Période de transmission à grande vitesse (Th)

Ce paramètre désigne la période cyclique de transmission de données prioritaires à grande vitesse. La DLE utilise la valeur de Th et la règle finalement sur V(Th), la valeur V(TSYN) étant égale pour chaque nœud en ligne. La plage de cette valeur est comprise entre 1 250 et 2 500 000, dont l'unité est 80 ns.

Le Tableau 15 illustre la structure du paramètre Th.

Tableau 15 – Paramètre Th: 6^{ème}, 7^{ème} et 8^{ème} octets

Période de transmission à grande vitesse																							
Th ($2^7 - 2^0$)								Th ($2^8 - 2^{15}$)								Th ($2^{16} - 2^{23}$)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.5 Période de transmission à moyenne vitesse (Tm)

Ce paramètre désigne la période cyclique de transmission de données prioritaires à moyenne vitesse. La DLE utilise la valeur de Tm et la règle sur V(Tm) de manière égale pour chaque nœud actif (en ligne). La plage de cette valeur est comprise entre 10 et 1 000, dont l'unité est 1 ms.

Le Tableau 16 illustre la structure du paramètre Tm.

Tableau 16 – Paramètre Tm: 9^{ème} et 10^{ème} octets

Période de transmission à moyenne vitesse															
Tm (2 ⁷ - 2 ⁰)								Tm (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.6 Valeur du temps de rotation du jeton cible de transmission de message sporadique (Ts)

Ce paramètre désigne la valeur du temps de rotation du jeton cible pour la transmission de données de messagerie sporadiques. La DLE utilise cette valeur et la règle sur $V(TTRTn)$ de manière égale pour chaque nœud en ligne. La plage de cette valeur est comprise entre 1 et 1 000, dont l'unité est 1 ms. $V(TTRTn)$ est $V(TTR1)$ pour le fonctionnement à 100 Mbps des architectures en étoile et en boucle, et $V(TTRT0)$ pour le fonctionnement à 1 000 Mbps de l'architecture en boucle.

Le Tableau 17 illustre la structure du paramètre Ts.

Tableau 17 – Paramètre Ts: 11^{ème} et 12^{ème} octet

Valeur du temps de rotation cible de transmission de message sporadique															
Ts (2 ⁷ - 2 ⁰)								Ts (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.7 Période de transmission à basse vitesse (TI)

Ce paramètre désigne la période cyclique de transmission de données prioritaires à basse vitesse dans le cas d'une architecture en étoile, et pour le fonctionnement à 1 000 Mbps de l'architecture en boucle, et n'est pas utilisé pour le fonctionnement à 100 Mbps de l'architecture en boucle. La DLE utilise la valeur de TI pour une architecture en étoile, et la règle sur $V(TI)$ de manière égale pour chaque nœud actif (en ligne). La plage de cette valeur est comprise entre 10 et 10 000, dont l'unité est 1 ms.

Le Tableau 18 illustre la structure du paramètre TI. Le champ TI est réservé et la valeur est réglée sur 0x0 pour le fonctionnement à 100 Mbps de l'architecture en boucle.

Tableau 18 – Paramètre TI: 13^{ème} et 14^{ème} octets

Période de transmission à basse vitesse															
TI (2 ⁷ - 2 ⁰)								TI (2 ⁸ - 2 ¹⁵)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.2.3.8 Liste opérationnelle (LL)

Ce paramètre indique l'état de fonctionnement actuel, qu'un nœud correspondant soit en ligne ou fonctionne normalement dans le bus de terrain de Type 11. Chaque bit de valeur "1" indique le nœud correspondant actif (en ligne) et normal, un nœud de valeur "0" étant inactif (hors-ligne).

$V(LL)$ est utilisée par la DLE et générée par les informations acheminées par la trame SYN. La liste opérationnelle est un ensemble de 8 mots d'une longueur de 32 bits, chaque mot correspondant à chaque nœud dans le bus de terrain de Type 11 et indiquant l'état de fonctionnement actuel. Chaque bit dans le mot LL correspond au nombre de nœuds $V(TN)$ dans un ordre séquentiel compris entre 0 et 255 au format petit-boutiste.

Le Tableau 19 illustre la structure du paramètre LL.

Tableau 19 – Paramètres LL: 15^{ème} à 46^{ème} octets

Liste opérationnelle																	
LL ($2^7 - 2^0$)									LL ($2^8 - 2^{15}$)								
15 ^{ème}	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	16 ^{ème}
17 ^{ème}	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24	18 ^{ème}
19 ^{ème}	39	38	37	36	35	34	33	32	47	46	45	44	43	42	41	40	20 ^{ème}
21 ^{ème}	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56	22 ^{ème}
23 ^{ème}	71	70	69	68	67	66	65	64	79	78	77	76	75	74	73	72	24 ^{ème}
⋮																	⋮
⋮																	⋮
41 ^{ème}	215	214	213	212	211	210	209	208	223	222	221	220	219	218	217	216	42 ^{ème}
43 ^{ème}	231	230	229	228	227	226	225	224	239	238	237	236	235	234	233	232	44 ^{ème}
45 ^{ème}	247	246	245	244	243	242	241	240	255	254	253	252	251	250	249	248	46 ^{ème}

6.2.4 Données utilisateur

La DLPDU SYN n'achemine aucune donnée utilisateur.

6.2.5 Envoi de DLPDU SYN

Une DLE d'un nœud SYN actuel transmet une DLPDU SYN avec un ensemble de paramètres, tel qu'indiqué ci-dessus, qui sont gérés, maintenus et réglés sur la valeur de DLPDU SYN par ladite DLE.

6.2.6 Réception de DLPDU SYN

Chaque DLE du nœud, qui est actif ou devant l'être, sauf le nœud SYN, exécute les actions suivantes à réception de DLPDU SYN.

- Dans le cas de l'architecture en étoile, la valeur de RMSEL permet de déterminer la commande de commutation dédiée à la réception des trames par l'un des deux canaux de réception A et B, correspondant à chacun des supports de redondance A et B. La valeur V(RCS) reflète la valeur de RMSEL.
- Chaque valeur de ST, Th, Ts et TI est utilisée. La valeur de TI n'est toutefois pas utilisée pour le fonctionnement à 100 Mbps de l'architecture en boucle, mais est reçue dans la DLPDU SYN, la valeur correspondante de V(ST),V(Th),V(Tm) et V(TI) dans un nœud faisant l'objet d'une comparaison respectivement. Si la valeur est différente, chaque valeur de ST, Th, Ts et TI est alors tenue d'être une nouvelle valeur de chaque variable V(ST), V(Th), V(Tm) et V(TI) respectivement.
- La valeur de LL est tenue d'être une nouvelle valeur de V(LL).
- La DLE, qui a transmis une DLPDU REQ, doit confirmer que le bit correspondant de V(TN) de LL devient "true". Si la valeur du bit correspondant de LL est "true", le nœud de V(TN) est alors actif. La DLE de la V(TN) doit commencer à transmettre ses données cycliques prioritaires de manière systématique et immédiatement après réception d'une DLPDU CMP ou DT-CMP émise par un nœud actif et dont l'identifiant peut être obtenu à partir de la LL et est inférieur en ordre séquentiel dans la LL.

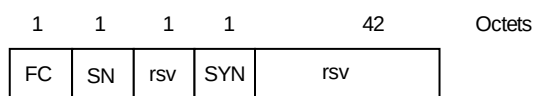
6.3 DLPDU de transmission complète (CMP)

6.3.1 Généralités

L'envoi d'une DLPDU CMP permet d'indiquer que la transmission de toutes les données cycliques prioritaires est complète au terme de la transmission de données. Deux types de DLPDU permettant d'indiquer que la transmission de données est complète sont spécifiés, l'un étant DLPDU CMP, et l'autre DLPDU DT.

6.3.2 Structure de DLPDU CMP

La Figure 9 illustre la structure de DLPDU CMP.



Légende

FC = Contrôle de trame (11001000) SN = Numéro de nœud de la source
 rsv = Réservé (0x0) SYN = Numéro de nœud SYN

Figure 9 – Structure de DLPDU CMP

6.3.3 Paramètre de numéro de nœud SYN (SYN)

Ce paramètre indique le numéro de V(TN) du nœud SYN actuel.

6.3.4 Données utilisateur de DLPDU CMP

La DLPDU CMP n'achemine aucune donnée utilisateur.

6.3.5 Envoi de DLPDU CMP

La DLE transmet la DLPDU CMP dans les délais et conditions spécifiés comme suit.

- a) La DLE comprise dans le délai attribué pour la transmission de ses données, transmet simplement ses données dans la période V(TSYN), et doit transmettre DLPDU CMP au moment où elle a terminé l'envoi de l'ensemble des données, tant cycliques prioritaires que de messagerie sporadiques. La dernière trame transmet DLPDU CMP à l'extérieur du nœud afin de signaler à tous les nœuds de bus de terrain de Type 11 que la transmission des données est terminée.

Lorsqu'une DLE transmet des données cycliques prioritaires en tant que dernière DLPDU, elle doit transmettre une DLPDU DT-CMP en remplacement d'une DLPDU CMP.

- b) Une DLE d'un nœud SYN doit attendre l'envoi d'une DLPDU CMP ou DT-CMP par la DLE qui transmet ses données, au cours de V(SCMP). Lorsque la DLE d'un nœud SYN n'est pas parvenu à recevoir une DLPDU CMP ou DT-CMP, elle doit lancer la transmission d'une DLPDU CMP en remplacement de la DLE qui n'a pas envoyé sa CLPDU CMP ou DT-CMP.

6.3.6 Réception de DLPDU CMP

Chaque DLE du nœud, qui est actif, doit activer les éléments suivants à réception d'une DLPDU CMP ou DT-CMP:

- a) La valeur de V(Tsl) est actualisée sur le numéro de V(TN) dont le nœud est tenu de transmettre ses données, après la recherche du nœud suivant émis par V(LL) et la conversion de ce dernier en une valeur définie, et obtenir le numéro d'identifiant de nœud réel de V(TN). La DLE, dans sa recherche du nœud suivant, vérifie et extrait le premier bit de V(LL) "true" dans un ordre supérieur à la position de bit correspondante du nœud ayant terminé la transmission des données. Lorsque le premier bit "true" est identifié, et selon la position de bit déterminée dans la V(LL), la DLE doit convertir la position de bit en numéro d'identifiant de nœud réel de V(TN),
- b) Sous réserve que la valeur du numéro d'identifiant de nœud, traduite de la position de bit dans la V(LL), est égale au numéro de V(TN) de ce nœud, la DLE doit commencer à transmettre des éléments de l'ensemble des données cycliques prioritaires à grande et moyenne vitesses, des données de messagerie sporadiques et des données cycliques prioritaires à basse vitesse, dans cet ordre dans le cas de l'architecture en étoile. D'autre part, dans le cas de l'architecture en boucle, l'ordre est celui des données cycliques prioritaires à grande vitesse, des données cycliques prioritaires à moyenne vitesse, des

données cycliques prioritaires à basse vitesse et des données de messagerie sporadiques. Toutefois, les données cycliques prioritaires à basse vitesse ne sont pas utilisées pour le fonctionnement à 100 Mbps de l'architecture en boucle.

Les données cycliques prioritaires à moyenne et basse vitesses, ainsi que les données de messagerie sporadiques, sont capables de transmettre des éléments du nœud si les conditions sont satisfaites respectivement, et chaque transmission de données doit s'achever lorsque l'une des conditions suivantes est satisfaite:

- Toutes les données avec chaque niveau de priorité ont terminé leur transmission,
- Le temporisateur d'observation correspondant a expiré,
- Le T(MTHT) dédié à la transmission de données cycliques prioritaires à grande vitesse a expiré.

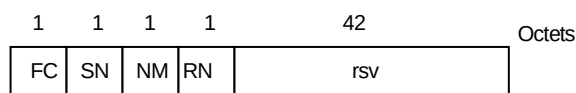
6.4 DLPDU de demande dans l'anneau (REQ)

6.4.1 Généralités

La DLE transmet une DLPDU REQ au nœud SYN afin de revendiquer un ajout dans le bus de terrain de Type 11.

6.4.2 Structure de DLPDU REQ

La Figure 10 illustre la structure de DLPDU REQ.



Légende

FC = Contrôle de trame (11000010) SN = Numéro de nœud de la source
 NM = Mode du nœud RN = Numéro de nœud du destinataire
 rsv = Réserve (0x0)

Figure 10 – Structure de DLPDU REQ

6.4.3 Mode du nœud (NM) – Paramètre ESYN

Ce paramètre indique le mode de fonctionnement de la DLE d'un nœud qui revendique un ajout au bus de terrain de Type 11. Le NM est utilisé pour le fonctionnement à 1 000 Mbps de l'architecture en boucle, dans le cas contraire, la valeur est réglée sur 0x0.

Le Tableau 20 présente le format du mode du nœud (NM).

Le paramètre "ESYN" indique que la DLE d'un nœud peut être un nœud SYN. La valeur "Enjable"/"true" peut être un nœud SYN, à l'inverse de la valeur "Disable"/"false". V(ESYN) a la valeur de ESYN et est utilisé par la DLE du nœud.

Tableau 20 – Paramètre NM

NM: Mode du nœud							
Réserve (0x0)			ESYN	Réserve (0x0)			
7	6	5	4	3	2	1	0

6.4.4 Paramètre de numéro de nœud du destinataire (RN)

Ce paramètre désigne le numéro de nœud du destinataire, ledit nœud devant recevoir la DLPDU REQ. RN est utilisé pour l'architecture en boucle, dans le cas contraire, la valeur est réglée sur 0x0.

Le Tableau 21 illustre le format du paramètre RN.

Tableau 21 – Paramètre RN

RN: Numéro de nœud du destinataire							
V(RN) (1 à 254)							
7	6	5	4	3	2	1	0

6.4.5 Données utilisateur de DLPDU REQ

La DLPDU REQ n'achemine aucune donnée utilisateur.

6.4.6 Envoi de DLPDU REQ

Dans le cas du fonctionnement à 100 Mbps des architectures en étoile et en boucle, la DLPDU REQ est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLE d'un nœud qui tente de faire partie intégrante du bus de terrain de Type 11, transmet une DLPDU REQ qui revendique l'ajout et l'activation du nœud, sous réserve que la DLE ait reçu une DLPDU SYN avec la valeur du champ PN égale à la valeur V(TN), et ultérieurement dans le délai de la période de contrôle MAC de Tmac.
- Cette période débute immédiatement après que la DLE a entièrement reçu une DLPDU CMP ou une DLPDU DT-CMP par le nœud, avec le nombre le plus élevé de la valeur de V(TN) dans le bus de terrain de Type 11.

D'autre part, dans le cas du fonctionnement à 1 000 Mbps de l'architecture en boucle, la DLPDU REQ est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLE d'un nœud qui tente de faire partie intégrante du bus de terrain de Type 11, transmet une DLPDU REQ qui revendique l'ajout et l'activation du nœud, sous réserve que la DLE ait reçu une DLPDU SYN sur V(PP), et ultérieurement dans le délai de la période de contrôle MAC pendant le créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN.
- La période de contrôle MAC débute immédiatement après que la DLE a entièrement reçu une DLPDU CMP ou une DLPDU DT-CMP par le nœud, avec le nombre le plus élevé de la valeur de V(TN) dans le bus de terrain de Type 11. La DLE envoie une DLPDU REQ du V(PP) du nœud.
- La DLPDU REQ d'origine est bloquée une fois au niveau du nœud avoisinant le nœud de tentative dans la direction de V(PP). Le nœud avoisinant envoie la DLPDU REQ au nœud SYN dans le délai de V(LN) égal à la valeur de V(TN) au cours du créneau de V(TSYN) avant d'envoyer la DLPDU DT.

6.4.7 Réception de DLPDU REQ

La DLE du nœud SYN actuel en fonctionnement reçoit une DLPDU REQ. Une fois la DLPDU REQ reçue, la DLE extrait le numéro de V(TN) de la DLE d'envoi, de la valeur du champ SN dans ladite DLPDU REQ, et règle sur "1" (mode actif ou opérationnel) le bit correspondant de V(TN) dans V(LL). Immédiatement après la période de contrôle MAC de Tmac au cours de laquelle la DLE a reçu une DLPDU REQ, cette même DLE transmet DLPDU SYN avec la valeur du numéro extrait de V(TN) dans le champ LL.

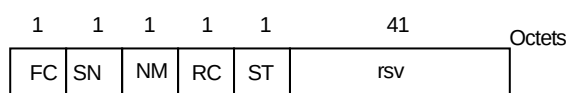
6.5 DLPDU de revendication (CLM)

6.5.1 Généralités

La DLE qui revendique la resynchronisation du bus de terrain de Type 11 transmet une DLPDU CLM, sous réserve d'un dysfonctionnement du nœud SYN actuel et de l'expiration du temporisateur T(SL) de surveillance du nœud SYN inactif. Une DLPDU CLM peut être envoyée par le nœud désigné et admis comme nœud SYN. La valeur de T(SL) de chaque nœud est différente et dépend de la valeur de V(ST) et V(TN).

6.5.2 Structure de DLPDU CLM

La Figure 11 illustre la structure de DLPDU CLM.



Légende

FC = Contrôle de trame (11000000) SN = Numéro de nœud de la source
 NM = Mode du nœud RC = Comptages résiduels de DLPDU CLM
 ST = Créneau rsv = Réserve (0x0)

Figure 11 – Structure de DLPDU CLM

6.5.3 Paramètre de DLPDU CLM

6.5.3.1 Paramètre de comptages résiduels DLPDU CLM (RC)

Ce paramètre indique les comptages résiduels nécessaires pour que la DLE transmette une DLPDU CLM. La valeur est réduite dans la plage comprise entre V(ST) et 0. La valeur par défaut est 20. La DLE, parvenue au comptage "0", devient une DLE SYN.

Le Tableau 22 illustre la structure du paramètre CLM.

Tableau 22 – Paramètre CLM: 4^{ème} octet

Comptages résiduels de DLPDU CLM							
V(RC) ($2^7 - 2^0$)							
7	6	5	4	3	2	1	0

6.5.3.2 Paramètre de créneau (ST)

La valeur de ce paramètre est réglée sur celle définie en 0.

6.5.3.3 Paramètre de mode du nœud (NM)

La valeur de ce paramètre est réglée sur celle définie en 6.4.3.

6.5.4 Données utilisateur de DLPDU CLM

La DLPDU CLM n'achemine aucune donnée utilisateur.

6.5.5 Envoi et réception d'une DLPDU CLM

La DLE, qui comporte la fonction de nœud SYN, transmet une DLPDU CLM revendiquant la resynchronisation du bus de terrain de Type 11.

Après mise sous tension et initialisation de la DLE, une fois le temporisateur T(SL) expiré, le temporisateur T(ST) est déclenché de manière successive. Une fois le temporisateur T(ST) expiré, la DLE transmet une DLPDU CLM.

La DLE doit transmettre une DLPDU CLM selon un nombre d'occurrences prédéfini, qui dépend de la valeur de V(ST). Le nombre d'occurrences est calculé par la formule suivante, dans laquelle "N" est le nombre d'occurrences.

$$N = \text{arrondi} [2 \times V(\text{MD}) + V(\text{MRT})/2 + 2]$$

Lors de la transmission d'une DLPDU CLM, et lors de l'occurrence de la collision, la DLE interrompt immédiatement la transmission d'une DLPDU CLM et commence à surveiller et à vérifier les conditions de fonctionnement du support commun.

Lorsque la DLE a reçu avec succès une DLPDU CLM émise par une autre DLE, elle doit extraire la valeur de SN intégrée dans la DLPDU CLM et comparer ladite valeur avec sa V(TN). Lorsque la valeur de SN est inférieure à la V(TN), la DLE ne doit pas poursuivre la transmission d'une autre DLPDU CLM. A l'inverse, lorsque la V(TN) est inférieure, la DLE doit une nouvelle fois tenter de transmettre la DLPDU CLM après expiration du T(SL) et du T(ST) consécutif.

Lorsque la DLE a transmis avec succès la DLPDU CLM par nombre de comptages total de V(ST), elle se transforme en DLE SYN qui fonctionne comme un nouveau nœud SYN. Dans le cas d'une DLPDU SYN transmise par une autre DLE, la DLE doit cesser de transmettre la DLPDU CLM pour resynchronisation.

La DLE non fournie avec la fonction de nœud SYN ou non désignée pour être un nœud SYN, ne doit pas transmettre la DLPDU CLM, et par ailleurs ne doit pas répondre à la DLPDU CLM reçue par un autre nœud.

6.6 Commande DLPDU (COM)

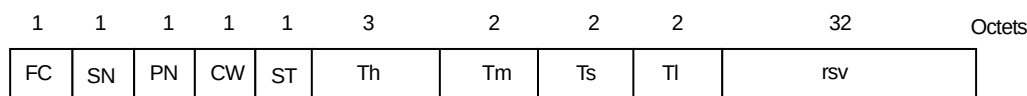
6.6.1 Généralités

La DLE transmet la DLPDU COM afin de demander à la DLE du nœud SYN d'égaliser et de modifier la valeur des paramètres de bus de terrain de Type 11 de tous les nœuds membres, et la modification doit être intégrée à la DLPDU SYN à transmettre dans la période V(Th) suivante. Les paramètres de bus de terrain de Type 11 à égaliser sont les suivants:

- a) V(ST): Créneau (ST),
- b) V(Th), V(Tm), V(Ts), V(Tl): Les périodes cycliques, la valeur du temps de rotation du jeton cible,
- c) CW: Mot de commande de nœud.

6.6.2 Structure de DLPDU COM

La Figure 12 illustre la structure de DLPDU COM.



Légende

FC = Contrôle de trame (11000100) SN = Numéro de nœud de la source
 PN = Numéro de nœud d'autorisation d'émission CW = Mot de commande
 ST = Créneau Th = Temps périodique à grande vitesse
 Tm = Temps périodique à moyenne vitesse Ts = Temps de rotation de messages sporadiques
 TI = Temps périodique à basse vitesse rsv = Réservé (0x0)

Figure 12 – Structure de DLPDU COM

6.6.3 Paramètre de DLPDU COM

Les paramètres de DLPDU COM sont équivalents à ceux de la DLPDU SYN spécifiés en 6.2.3.

6.6.4 Données utilisateur de DLPDU COM

La DLPDU COM n'achemine aucune donnée utilisateur.

6.6.5 Envoi et réception d'une DLPDU COM

La DLE est activée pour transmettre une DLPDU COM à réception de la primitive DLM_SET_Value, requérant plus particulièrement la modification de chaque valeur de V(ST), V(Th), V(Tm), V(Ts), V(TI) et CW.

Dans le cas du fonctionnement à 100 Mbps des architectures en étoile et en boucle, la DLPDU COM est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLE transmet une DLPDU COM au nœud SYN dans le délai de la période de contrôle MAC au cours du créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN avec une valeur de PN égale à celle de V(TN) après réception de la primitive DLM_SET_Value.

D'autre part, dans le cas du fonctionnement à 1 000 Mbps de l'architecture en boucle, la DLPDU COM est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLE transmet une DLPDU COM au nœud SYN dans le délai de la période de V(LN) égale à la valeur de V(TN) au cours du créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN avec une valeur de PN égale à celle de V(TN) après réception de la primitive DLM_SET_Value.
- La DLE envoie DLPDU COM avant d'envoyer DLPDU DT et/ou DLPDU RAS.

Lorsque la DLE du nœud SYN a reçu avec succès la DLPDU COM de la DLE d'un autre nœud, elle transmet l'exigence de la DLPDU COM à la DLPDU SYN à transmettre dans la période V(Th) suivante.

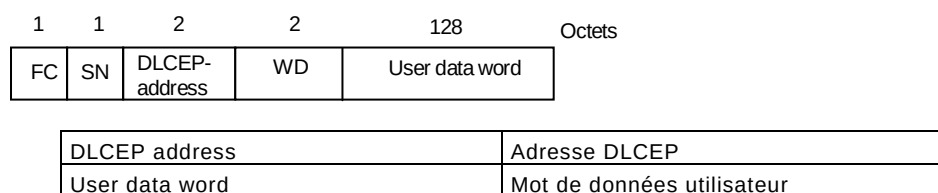
6.7 Données cycliques et données cycliques avec DLPDU de transmission complète (DT) et (DT-CMP)

6.7.1 Généralités

La DLPDU DT ou la DLPDU DT-CMP permet d'acheminer les données cycliques prioritaires vers tous les nœuds membres de bus de terrain de Type 11. La différence entre la DLPDU DT et la DLPDU DT-CMP est l'indication d'une transmission complète ou non.

6.7.2 Structure de DLPDU DT

La Figure 13 illustre la structure de DLPDU DT.



Légende

FC = Contrôle de trame (DT: pp0001111, DT-CMP: pp0011111

"pp" correspond au niveau de priorité).

SN = Numéro de nœud de la source

WD = Longueur de mot

Figure 13 – Structure de DLPDU DT

6.7.3 Paramètres de DLPDU DT

6.7.3.1 Paramètre d'adresse DLCEP

Ce paramètre indique le DLCEP pour une connexion DL éditeur multipoint prédéfinie.

Le Tableau 23 illustre la structure du paramètre DT.

Tableau 23 – Paramètre DT: 3^{ème} et 4^{ème} octets

adresse DLCEP															
adresse DLCEP ($2^7 - 2^0$)								adresse DLCEP ($2^8 - 2^{15}$)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.7.3.2 Paramètre de longueur de mot (WD)

6.7.3.2.1 WD pour l'architecture en étoile

Ce paramètre indique la longueur du User-data-word. La valeur est 0x40, et est intégrée à la valeur V(BW).

6.7.3.2.2 WD pour l'architecture en boucle

Ce paramètre indique la longueur du User-data-word. La valeur est choisie parmi les valeurs 0x8, 0x0c, 0x10, 0x14, 0x18, 0x24, 0x30 et 0x40 pour le fonctionnement à 100 Mbps, et 0x14 et 0x40 pour le fonctionnement à 1 000 Mbps. La valeur est intégrée à la valeur V(BW).

6.7.4 Envoi de DLPDU DT ou DT-CMP

La CTRC (commande TX/RC d'émission cyclique) se charge de la transmission d'une DLPDU DT ou DLPDU DT-CMP à réception de la primitive DL-PUT de l'utilisateur DL.

Le type de transmission de données cycliques à grande, à moyenne ou à basse vitesse doit être désigné pour le sous-champ pp de FC de la DLPDU DT et de la DLPDU DT-CMP. La transmission de données cycliques à basse vitesse n'est pas utilisée pour le fonctionnement à 100 Mbps de l'architecture en boucle.

6.7.5 Réception de DLPDU DT ou DT-CMP

La DLE ayant reçu la DLPDU DT ou DLPDU DT-CMP doit actualiser la Receive_buffer correspondante associée au champ d'adresse DLCEP dans la DLPDU DT ou DLPDU DT-CMP, et notifier à l'utilisateur DLS, qui utilise la primitive d'indication DL-Buffer-Received, l'actualisation et la disponibilité de lecture des données contenues dans la Receive-Buffer spécifiée.

6.8 DLPDU RAS (RAS)

6.8.1 Généralités

La DLPDU RAS permet de transférer les informations RAS (Fiabilité, Disponibilité, Aptitude au service) à tous les nœuds membres.

6.8.2 Structure de DLPDU RAS

La Figure 14 illustre la structure de DLPDU RAS.

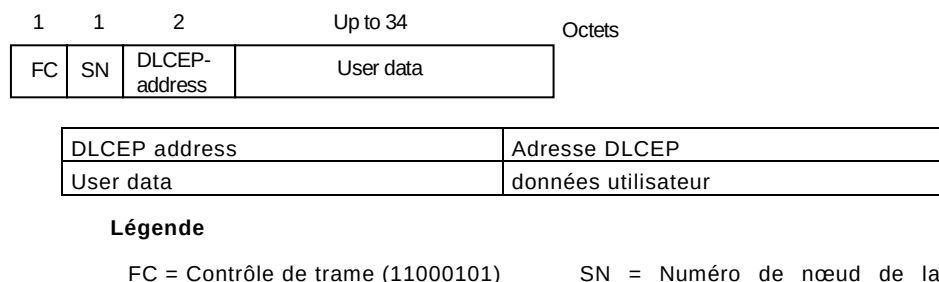


Figure 14 – Structure de DLPDU RAS

6.8.3 Paramètre d'adresse DCEP

Ce paramètre indique le DLCEP pour une connexion DL éditeur multipoint prédéfinie.

Le Tableau 24 illustre la structure du paramètre RAS.

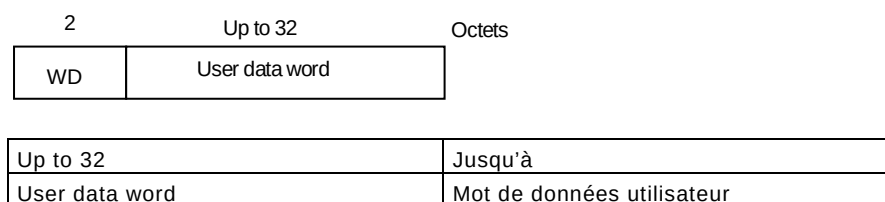
Tableau 24 – Paramètre RAS: 3^{ème} et 4^{ème} octets

adresse DLCEP															
adresse DLCEP (27 - 20)								adresse DLCEP (28 - 215)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.8.4 Données utilisateur

La DLPDU RAS achemine les données utilisateur, qui sont spécifiquement associées aux informations RAS de la DLE locale, jusqu'à 34 octets simultanément, le paramètre WD de la longueur du 2nd octet étant toutefois inclus dans cette DLE, et utilisé pour l'architecture en boucle, de sorte que l'acheminement des données utilisateur peut s'effectuer jusqu'à 32 octets. La Figure 15 illustre la structure des données utilisateur de l'architecture en boucle.

La DLE de chaque nœud local peut traiter les informations RAS sur 34 octets, le nombre total de ces informations pour chaque nœud étant ensuite réparti en plusieurs fragments, chaque fragment étant acheminé sur la DLPDU RAS.



Légende

WD = Longueur de mot de données utilisateur

Figure 15 – Structure des données utilisateur de l'architecture en boucle

6.8.5 Envoi et réception d'une DLPDU RAS

La DLE de nœud local peut diffuser les informations RAS à tous les nœuds membres de bus de terrain de Type 11, à l'aide de la DLPDU RAS.

Dans le cas du fonctionnement à 100 Mbps des architectures en étoile et en boucle, la DLPDU RAS est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLE de chaque nœud local, l'un après l'autre dans l'ordre de la valeur V(TN), transmet la DLPDU RAS pendant la période T_{MAC}. Par conséquent, la transmission de la DLPDU RAS ou la diffusion des informations RAS de nœud local est effectuée sur le fond.

D'autre part, dans le cas du fonctionnement à 1 000 Mbps de l'architecture en boucle, la DLPDU RAS est envoyée par la DLE au moment et dans les conditions spécifiées ci-dessous.

- La DLPDU RAS est envoyée dans le délai de V(LN) égal à la valeur de V(TN) au cours du créneau de V(TSYN) dans lequel la DLE a reçu la DLPDU SYN avec une valeur de PN égale à celle de V(TN), et la DLE envoie la DLPDU RAS avant d'envoyer DLPDU DT.

La DLE ayant reçu la DLPDU RAS doit actualiser la Receive_buffer correspondante associée au champ d'adresse DLCEP dans la DLPDU RAS, et notifier à l'utilisateur DLS, qui utilise la primitive d'indication DL-Buffer-Receive, l'actualisation et la disponibilité de lecture des données contenues dans la Receive-Buffer spécifique.

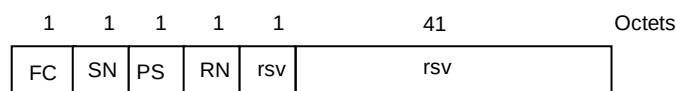
6.9 DLPDU de demande de répétition de boucle (LRR)

6.9.1 Généralités

La DLPDU LRR est utilisée spécifiquement pour l'architecture en boucle et est transmise pour récupération de l'anneau ouvert en cas de défaillance.

6.9.2 Structure de DLPDU LRR

La Figure 16 illustre la structure de DLPDU LRR.



Légende

FC = Contrôle de trame (11100110) SN = Numéro de nœud de la source

PS = État du port

RN = Numéro de nœud du destinataire

rsv = Réserve (0x0)

Figure 16 – Structure de DLPDU LRR

6.9.3 Paramètres de DLPDU LRR

6.9.3.1 Structure de l'état du port (PS)

6.9.3.1.1 Généralités

Ce paramètre indique l'état du nœud. Le Tableau 25 illustre la structure de l'état du port.

Tableau 25 – Format du paramètre PS: 3^{ème} octet

PP: État du port							
Réserve	Port primaire (PP)	Send-enable-A/-B		Receive-enable-A/B		Forward enable-A/-B	
7	6	5	4	3	2	1	0

6.9.3.1.2 Port primaire (PP)

Ce paramètre indique le port en anneau par lequel a eu lieu la première réception de trame SYN depuis l'activation de ce nœud, le démarrage ou l'initialisation du système.

Le Tableau 26 montre la valeur du paramètre de port primaire.

Tableau 26 – Valeur du paramètre PP

Port primaire (PP)	Valeur	Description
Ring-port-A	0	Le Ring-port-A est le port primaire.
Ring-port-B	1	Le Ring-port-B est le port primaire.

6.9.3.1.3 Paramètres Send- Receive- et Forward-enable

Ces paramètres indiquent l'état opérationnel du port en anneau, que l'état de ce dernier soit ou non send-, receive- ou forward-enabled.

Le Tableau 27, le Tableau 28 et le Tableau 29 montrent la valeur des paramètres send-, receive-, forward-enable-A / -B.

Tableau 27 – Valeur des paramètres send-enable-A/-B

Send enable-A/-B	Valeur	Description
Send-disable	00	Les ports en anneau A et B sont en mode désactivé
Send-enable-A	01	Le port en anneau A est en mode activé
Send-enable-B	10	Le port en anneau B est en mode activé
Send-enable-A and -B	11	Les ports en anneau A et B sont en mode activé

Tableau 28 – Valeur des paramètres receive-enable-A/-B

Receive enable-A/-B	Valeur	Description
Receive-disable	00	Les ports en anneau A et B sont en mode désactivé
Receive-enable-A	01	Le port A est en mode activé
Receive-enable-B	10	Le port B est en mode activé
Receive-enable- A and -B	11	Les ports A et B sont en mode activé

Tableau 29 – Valeur des paramètres forward-enable-A/-B

Forward enable-A/-B	Valeur	Description
Forward-disable	00	L'acheminement entre les ports A et B est en mode désactivé. Les ports en anneau A et B sont en mode bloqué
Forward-enable-A	01	L'acheminement entre le port A et le port B est en mode activé. Ring-port-A est en mode non bloqué et Ring-port-B est en mode bloqué
Forward-enable-B	10	L'acheminement entre le port B et le port A est en mode activé. Ring-port-B est en mode non bloqué et Ring-port-A est en mode bloqué
Forward-enable-A to -B	11	L'acheminement entre les ports A et B est en mode activé. Les ports en anneau A et B sont en mode non bloqué

6.9.3.2 Numéro de nœud du destinataire (RN)

Ce paramètre désigne le numéro de nœud du destinataire, ledit nœud devant recevoir la DLPDU LRR. Le format est présenté dans le Tableau 30.

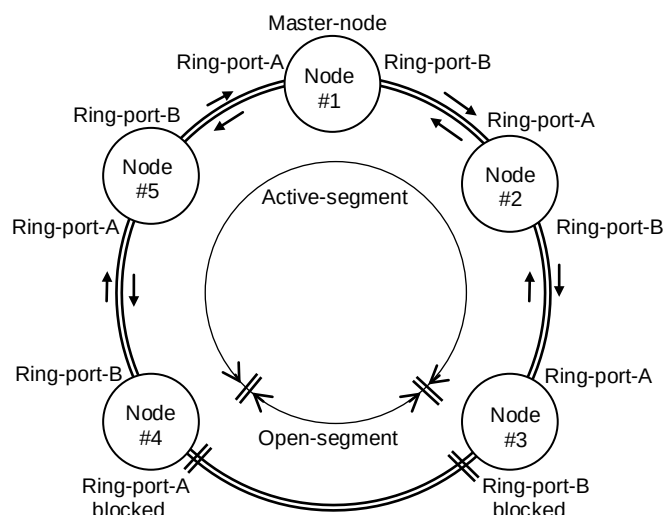
Tableau 30 – Paramètre RN: 4^{ème} octet

RN: Numéro de nœud du destinataire							
V(RN) (1 à 254)							
7	6	5	4	3	2	1	0

6.9.4 Envoi et réception d'une DLPDU LRR

Le bus de terrain de Type 11 de l'architecture en boucle est intégré à une structure en anneau, dans laquelle chaque nœud est relié en série à deux autres nœuds. La DLPDU est transmise dans les deux directions à l'extérieur du nœud à tour de rôle, de sorte que les données transmises y circulent de manière séquentielle entre les nœuds. Il existe des points ouverts délibérément sous contrôle du bus de terrain de Type 11, et qui permettent de se soustraire à la DLPDU dans laquelle circulent en permanence des données contenues dans l'anneau, et ce, de sorte qu'aucune DLPDU ne soit plus renvoyée au nœud source le long du chemin circulaire.

La Figure 17 illustre un exemple de structure en anneau ouvert sous contrôle, et le Tableau 31 récapitule l'état de fonctionnement de chaque nœud.



Légende

Anglais	Français
Master-node	Nœud-maître
Ring-port-A	Port en anneau A
Ring-port-B	Port en anneau B
Node	Nœud
Active-segment	Segment actif
Open-segment	Segment ouvert
Ring-port-A blocked	Port en anneau A bloqué
Ring-port-B blocked	Port en anneau B bloqué

Figure 17 – Anneau ouvert sous contrôle

Tableau 31 – Etat de fonctionnement du nœud

Nœud #	État du nœud			
	Maître/Esclave - état du nœud	État normal/Ouvert	Etat de Ring-port-A/-B	
			Ring-port-A	Ring-port-B
1	Maître	Normal	Non bloqué	Non bloqué
2	Esclave	Normal	Non bloqué	Non bloqué
3	Esclave	Ouvert	Non bloqué	Bloqué
4	Esclave	Ouvert	Bloqué	Non bloqué
5	Esclave	Normal	Non bloqué	Non bloqué

La DLE prévoit une fonction de gestion de l'action de réparation en collaboration avec la DLE d'autres nœuds, à l'occurrence d'un point de défaillance unique, et ce, afin de maintenir le bon fonctionnement des communications entre les nœuds.

Les DLPDU LRR d'envoi et de réception destinées à compenser l'occurrence d'un point de défaillance unique se présentent comme suit:

- a) Une DLE à l'état ouvert (désignée ci-après DLE1) située à une extrémité du segment actif actuel et subissant un point de défaillance unique sur le chemin du nœud à l'état maître, doit détecter tout silence de réception de longue durée;
- b) La DLE1 à l'état ouvert, qui détecte le silence de réception de longue durée, doit transmettre la DLPDU LRR avec le RN équivalent au node-identifiant-number du nœud adjacent, à savoir un autre nœud (ou une DLE, désignée ci-après DLE2) à l'état ouvert et situé à l'extrémité du segment actif actuel, et ce, dans les deux directions simultanément, selon la temporisation fournie par l'ACM;
- c) La DLE1:
 - doit attendre de recevoir la DLPDU LRR avec le RN équivalent à la V(TN);
 - doit passer à l'état normal si elle reçoit, dans une période spécifiée, une DLPDU LRR avec le RN équivalent à la V(TN) transmise par la DLE2;
 - doit modifier l'état de port "Bloqué" du port en anneau du segment actif actuel, puis l'état de port "Non bloqué" du port en anneau sur le segment ouvert actuel, à défaut de réception de la DLPDU LRR avec le RN équivalent à la V(TN) dans la période spécifiée, transmise par la DLE, qui est le nœud adjacent du segment actif actuel;
- d) La DLE2 qui reçoit la DLPDU LRR avec le RN équivalent à la V(TN) doit passer à l'état normal;
- e) La DLE (désignée ci-après DLE3) à l'état normal et située sur le chemin compris entre le point de défaillance unique et le nœud à l'état ouvert (DLE1), doit transmettre la DLPDU LRR avec le RN équivalent au node-identifiant-number du nœud adjacent à l'extrémité du port en anneau de réception d'une DLPDU LRR, dans les deux directions simultanément;
- f) La DLE3:
 - doit attendre, pendant une période spécifiée, de recevoir la DLPDU LRR avec le RN équivalent à la V(TN);
 - doit rester à l'état normal si elle reçoit dans une période spécifiée une DLPDU LRR avec le RN équivalent à la V(TN) en provenance de l'extrémité opposée du port en anneau par lequel la DLE3 a reçu une DLPDU LRR pour la première fois;
 - doit modifier l'état "Bloqué" d'un port en anneau à l'extrémité opposée d'un port de même nature par lequel la DLE3 a reçu une DLPDU LRR pour la première fois, en l'absence de réception d'une DLPDU LRR avec le RN équivalent à la V(TN) en provenance du port en anneau dans la période spécifique, et doit passer à l'état ouvert.

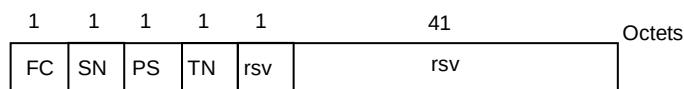
6.10 DLPDU de diagnostic de boucle (LPD)

6.10.1 Généralités

La transmission de la DLPDU LPD permet un diagnostic de l'état des liaisons entre les nœuds adjacents.

6.10.2 Structure de DLPDU LPD

La Figure 18 illustre la structure de DLPDU LPD.



Légende

FC = Contrôle de trame (11100011) SN = Numéro de nœud de la source

PS = État du port

TN = Ce numéro de nœud

rsv = Réserve (0x0)

Figure 18 – Structure de DLPDU LPD

6.10.3 Paramètres de DLPDU LPD

6.10.3.1 État du port (PS)

Ce paramètre équivaut au paramètre PS défini en 6.9.3.1.

6.10.3.2 Ce numéro de nœud (TN)

Ce paramètre indique le node-identifier-number de la valeur V(TN), dont le nœud a transmis la DLPDU LPD.

6.10.4 Envoi et réception d'une DLPDU LPD

Chaque DLE transmet une DLPDU LPD de manière périodique avec le numéro d'identifiant V(TN) du nœud et l'état du port actif, de sorte que les nœuds voisins soient capables de recueillir ces informations.

- Chaque DLE doit transmettre la DLPDU LPD dans les deux directions simultanément, au moment spécifique indiqué périodiquement par l'ACM de l'architecture en boucle;
- Tous les nœuds doivent transmettre simultanément la DLPDU LPD;
- Chaque DLE qui reçoit une DLPDU LPD ne doit acheminer aucune DLPDU LPD reçue d'un port en anneau à l'autre.

7 Éléments de procédure DLE

7.1 Éléments de procédure DLE d'une architecture en étoile

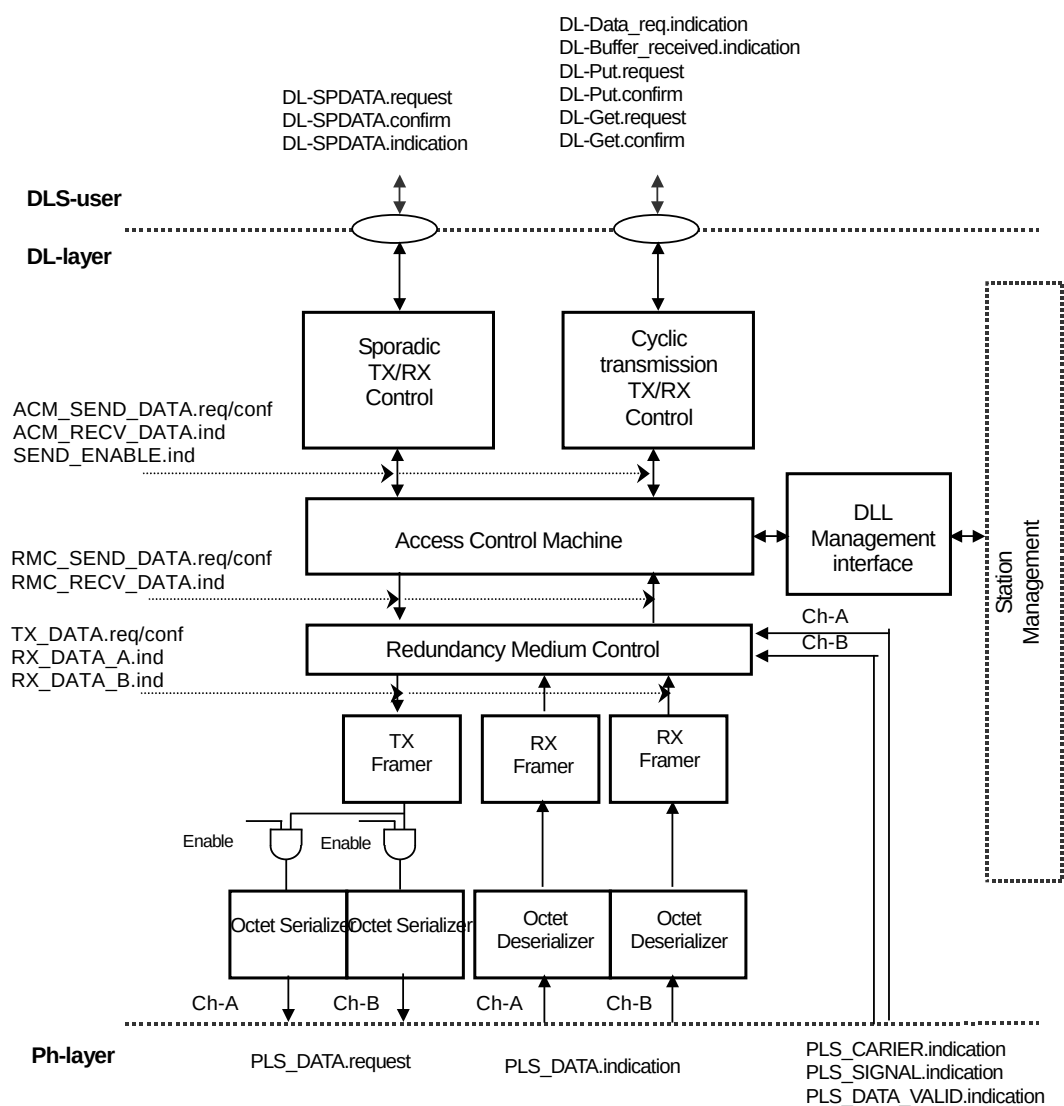
7.1.1 Structure générale

La DLL est constituée des composants de commande suivants: Commande TX/RX d'émission cyclique (CTRC), Commande TX/RX sporadique (STRC), Machine de contrôle d'accès (ACM), Commande de support de redondance (RMC), Verrouilleur de trames TX/RX, Convertisseur parallèle-série d'octets, Convertisseur série-parallèle d'octets et interface de gestion DLL.

La machine de contrôle d'accès (ACM), en tant que composant de commande principal, assure la fonction de contrôle d'accès au support déterministe en coopération avec les commandes CTRC, STRC et RMC, afin d'assurer une prise en charge fiable et efficace des services de transfert de données en mode de connexion de niveau supérieur et en mode sans connexion.

L'interface de gestion DLL offre des fonctions de gestion DLL.

La Figure 19 illustre la structure générale de la DLL.



Légende

Anglais	Français
DLS-user	Utilisateur DLS
DL-layer	Couche DL
Sporadic TX/RX control	Commande TX/RX sporadique
Cyclic transmission TX/RX control	Commande TX/RX d'émission cyclique
Access control machine	Machine de contrôle d'accès
DLL management interface	Interface de gestion DLL
Station management	Gestion de station
Redundancy medium control	Commande de support de redondance
TX framer	Verrouilleur de trames TX
RX framer	Verrouilleur de trames RX
Octet serializer	Convertisseur parallèle-série d'octets
Octet deserializer	Convertisseur série-parallèle d'octets
Ph-layer	Couche Ph

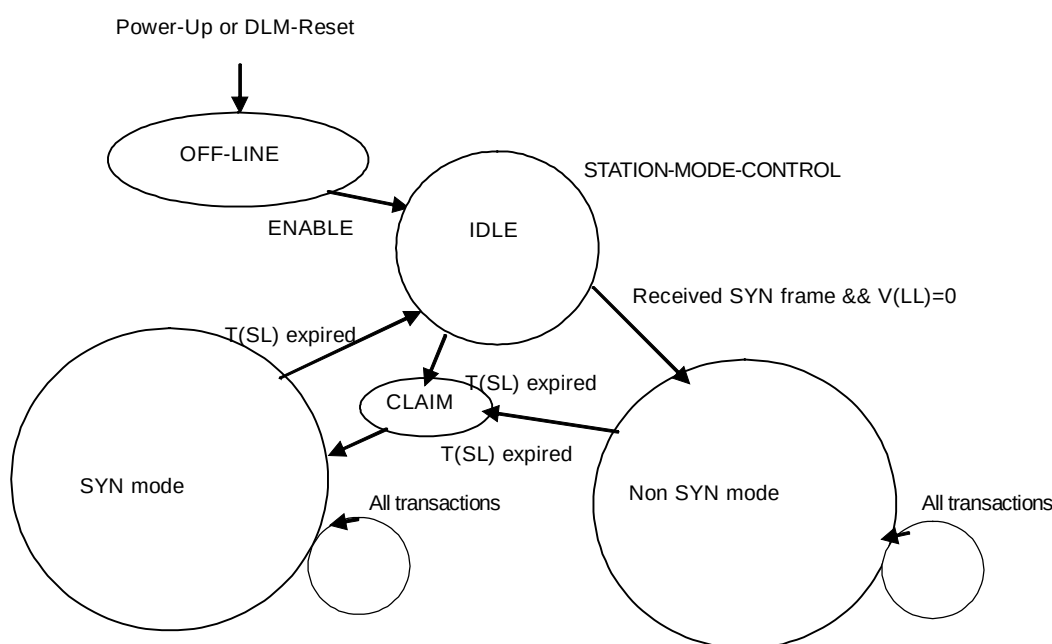
Figure 19 – Structure générale de la DLL

7.1.2 Initialisation

La DLE doit passer à l'état OFF-LINE à la mise sous tension ou après réception de la primitive de demande DLM-RESET. A l'état OFF-LINE, la DLE ne doit pas transmettre et doit ignorer toutes les DLPDU reçues.

Lorsque toutes les variables relatives à un fonctionnement DLE normal sont définies par la primitive de demande DLM-Set-Value, l'état passe de OFF-LINE à STATION MODE CONTROL, dans lequel la DLE doit initier un fonctionnement DL normal.

La Figure 20 illustre la transition des états DLE.



Légende

Anglais	Français
Power-up or DLM-Reset	Mise sous tension ou DLM-Reset
T(SL) expired	T(SL) expiré
SYN mode	Mode SYN
Non SYN mode	Mode non SYN
All transactions	Toutes transactions

Figure 20 – Transition d'états DLE

7.1.3 Commande TX/RX d'émission cyclique (CTRC)

7.1.3.1 Présentation générale

La commande TX/RX d'émission cyclique (CTRC) est chargée de la mise en mémoire tampon et de la répartition de la DLSDU reçue pour le transfert de données cycliques prioritaires entre l'utilisateur DLS et l'ACM.

La primitive d'indication DL-Data-Req de la commande CTRC informe l'utilisateur DLS qu'il peut commencer le transfert de données entre chaque mémoire tampon d'utilisateur DLS et la mémoire tampon transmise de la DLE, en utilisant la primitive de demande DL-Put. La mémoire tampon de l'utilisateur DLS est adressée avec le paramètre d'identifiant DLCEP dans la primitive d'indication DL-Data-req de la commande CTRC.

Chaque mémoire tampon de l'utilisateur DLS contient des données dédiées à la transmission de données cycliques prioritaires. Après réception des données de l'utilisateur DLS, la CTRC émet une primitive de demande ACM_SEND_DATA afin que l'ACM commence à transmettre lesdites données sous forme de DLPDU DT ou DT-CMP.

La CTRC émet une primitive d'indication DL-Data-Req en réponse à la primitive d'indication SEND_ENABLE de l'ACM. Le paramètre de classe SPEED de la primitive d'indication SEND_ENABLE permet à la CTRC d'activer le transfert de la classe correspondante de la mémoire tampon des données par l'utilisateur DLS. Les classes désignées par le paramètre de classe de vitesse correspondent aux classes de transfert de données cycliques à grande, moyenne et basse vitesses.

Le transfert de données de chaque classe de vitesse désignée est fonction du niveau de priorité de transfert:

- a) Pour ce qui concerne la classe de priorité 3, qui correspond au transfert de données cycliques à grande vitesse, et à réception de la primitive d'indication SEND_ENABLE, la CTRC doit traiter l'ensemble des données de chaque mémoire tampon d'utilisateur DLS correspondant au transfert de données cycliques à grande vitesse, ou la classe à grande vitesse est à fournir à l'ACM et être transmise sur le support d'émission dans son intégralité et en une seule fois dans tous les cas;
- b) Pour ce qui concerne les classes de priorité 2 et 0, qui correspondent au transfert de données cycliques à moyenne vitesse, ou la classe à moyenne vitesse et le transfert de données cycliques à basse vitesse ou la classe à basse vitesse respectivement, la CTRC doit traiter dans tous les cas les données de chaque mémoire tampon de l'utilisateur DLS correspondant à la classe à moyenne vitesse, la classe à basse vitesse étant transmise ou non selon les conditions en cours. Le temps de maintien du jeton de chaque classe 2 et 0 régit ces conditions. Les données cycliques à moyenne vitesse sont transmises une fois toutes les données cycliques à grande vitesse transmises. Lorsque le temps de maintien du jeton s'achève lors de l'envoi des données cycliques à moyenne vitesse, le transfert de données est interrompu et la DLPDU CMP ou DT-CMP est transmise dans ce cas. Le reste des données cycliques à moyenne vitesse précédentes est à transmettre dès que possible selon les mêmes conditions. Pour ce qui concerne les données cycliques à basse vitesse, le même contrôle que celui applicable au transfert de données cycliques à moyenne vitesse est à effectuer. Le cas se présente à chaque période V(TSYN).

Lorsque l'ACM a émis la primitive d'indication ACM_RECV_DATA, la commande CTRC archive la DLPDU transmise par ladite ACM dans chaque mémoire tampon de réception, adressée avec le paramètre d'identifiant DLCEP dans cette même primitive. Par ailleurs, la CTRC émet une primitive d'indication DL-Buffer-received pour informer l'utilisateur DLS de l'actualisation et de la disponibilité de lecture de la mémoire tampon de réception de la DLE. L'utilisateur DLS peut lire les données contenues dans cette mémoire en utilisant la primitive de demande DL-Get en réponse à la primitive d'indication DL-Buffer-received émise par la CTRC.

7.1.3.2 Définitions de primitive

7.1.3.2.1 Primitives échangées entre l'utilisateur DLS et la commande CTRC

Les primitives échangées entre l'utilisateur DLS et la commande CTRC sont présentées dans le Tableau 32 et les primitives échangées entre la commande CTRC et l'ACM sont récapitulées dans le Tableau 33.

Tableau 32 – Primitives échangées entre l'utilisateur DLS et la commande CTRC

Nom de la primitive	Source	Paramètres associés
DL-Data-req.indication	CTRC	Identifiant DLCEP
DL-Put.request	Utilisateur DL	identifiant DLCEP, longueur DLSDU, DLSDU
DL-Put.confirm	CTRC	Identifiant DLCEP, Etat
DL-Buffer-received.indication	CTRC	Identifiant DLCEP
DL-Get.request	Utilisateur DL	Identifiant DLCEP
DL-Get.confirm	CTRC	Identifiant DLCEP, longueur DLSDU DLSDU Etat

Tableau 33 – Primitives échangées entre la commande CTRC et l'ACM

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_RECV_DATA.ind	ACM	DLPDU

7.1.3.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC

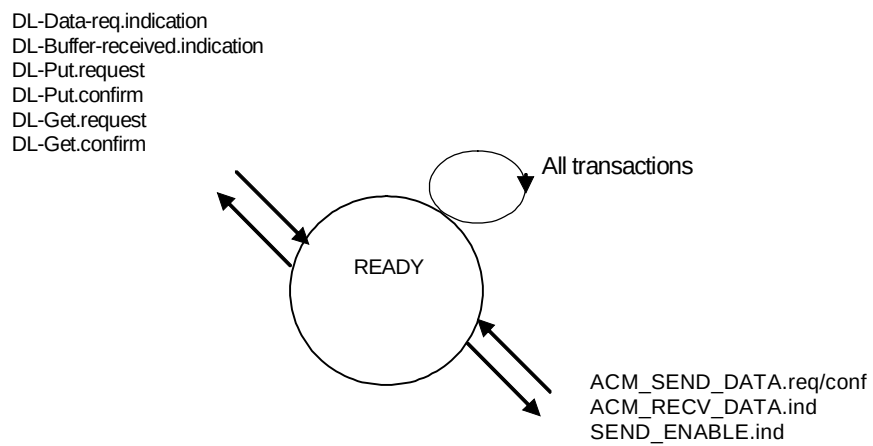
Les paramètres utilisés pour l'interaction entre l'utilisateur DLS et la commande CTRC sont récapitulés dans le Tableau 34.

Tableau 34 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC

Nom du paramètre	Description
Identifiant DLCEP	Identifiant permettant de désigner Send-Buffer ou Receive-Buffer
DLSDU	Contenu de Send-Buffer or Receive-Buffer, qui correspond aux données cycliques prioritaires traitées par la commande CTRC
Longueur DLSDU	Longueur de DLSDU
Classe de vitesse	Classe de vitesse permettant de désigner la classe de transfert de données cycliques prioritaires, applicable aux données cycliques à grande, moyenne et basse vitesses
Etat	Rapport d'état indiquant si le service demandé est fourni avec succès ou s'il est défaillant pour le motif spécifié

7.1.3.3 Table d'états CTRC

Le diagramme de transition d'état de la commande CTRC est illustré à la Figure 21, et la table d'états CTRC est présentée dans le Tableau 35.



Anglais	Français
All transactions	Toutes transactions

Figure 21 – Diagramme de transition d'état de la commande CTRC

Tableau 35 – Table d'états CTRC

#	État courant	Événement /condition ⇒ action	État suivant
1	READY	DL-Put.request {DLCEP-identifiant, DLSDU-length, DLSDU} / CHECK_PAR_PUT (DLCEP-identifiant, DLSDU) = "True" ⇒ PUT_BUFFER (DLCEP-identifiant, DLSDU) DL-Put.confirm { DLCEP-identifiant, DLSDU-length, Status:= "success" }	READY
2	READY	DL-Put.request {DLCEP-identifiant, DLSDU-length, DLSDU} / CHECK_PAR_PUT (DLCEP-identifiant, DLSDU) = "False" ⇒ DL-Put.confirm { DLCEP-identifiant, DLSDU-length, Status:= "failure" }	READY
3	READY	DL-Get.request {DLCEP-identifiant } / CHECK_PAR_GET (DLCEP-identifiant) = "True" ⇒ DL-Get.confirm { DLCEP-identifiant, DLSDU-length, DLSDU:= GET_BUFFER(DLCEP-identifiant), Status:= "success" }	READY
4	READY	DL-Get.request {DLCEP-identifiant} / CHECK_PAR_GET (DLCEP-identifiant) = "False" ⇒ DL-Get.confirm { DLCEP-identifiant, DLSDU-length, DLSDU:= null, Status:= "failure" }	READY
5	READY	SEND_ENABLE.ind {Speed-class} / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "True" ⇒ DL-Data-req indication { DLCEP-identifiant:= GET_NEXT_ID(Speed-class) } DLSDU:= GET_BUFFER (DLCEP-identifiant) DLPDU: = BUILD_DT (Speed-class, DLCEP-identifiant, DLSDU) ACM_DATA.req { DLPDU } ACM_DATA.conf { } -- immediate ⇒ NEXT(Speed-class)	READY
6	READY	SEND_ENABLE.ind {Speed-class} / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "False" ⇒ ACM_DATA.req { null -- no data } ACM_DATA.conf { } -- immediate ⇒ NEXT(Speed-class)	READY
7	READY	ACM_DATA.ind {DLPDU} / Class <> "SCOPRADIC" && CHECK_PAR_DT (DLPDU) ⇒ DL-Buffer-received.indication { DLPDU.DLCEP-address }	READY

7.1.3.4 Fonctions de la commande CTRC

Toutes les fonctions de la commande CTRC sont récapitulées dans le Tableau 36.

Tableau 36 – Table de fonctions CTRC

Nom de la fonction	Entrée	Sortie	Fonctionnement
CHECK_PAR_PUT	Identifiant DLCEP, DLSDU	Vrai/Faux	Vérifier la validité de tous les paramètres de l'identifiant DLCEP et de la DLSDU de la primitive DL-put.request. Si la validité est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
PUT_BUFFER	Identifiant DLCEP, DLSDU	(aucun)	Archiver la DLSDU dans la mémoire tampon Send-Buffer associée à l'identifiant DLCEP
CHECK_PAR_GET	Identifiant DLCEP	Vrai/Faux	Vérifier la validité de l'identifiant DLCEP de la primitive DL-get.request. Si la validité est confirmée, la valeur "Vrai" est renvoyée
GET_BUFFER	Identifiant DLCEP	DLSDU	Placer la DLSDU dans la mémoire tampon Receive-Buffer associée à l'identifiant DLCEP
CHECK_NEXT_SEND	Classe de vitesse	Vrai/Faux	Vérifier l'existence de la DLSDU spécifiée par la classe de vitesse. Si cette existence est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
GET_NEXT_ID	Classe de vitesse	Identifiant DLCEP	Obtenir l'identifiant DLCEP suivant de la mémoire tampon de l'utilisateur DLS spécifié par la classe de vitesse
BUILD_DT	Classe de vitesse, identifiant DLCEP, DLSDU	DLPDU	Intégrer dans la DLPDU des données cycliques prioritaires issues des données contenues dans la mémoire tampon Send-Buffer. La DLPDU est assemblée comme suit. DLPDU.DA:= V(MGA) DLPDU.SA:= V(IA) DLPDU.Len/Type:= 0x888b DLPDU.FC:= Speed-class + "DT" DLPDU.SN:= V(TN) DLPDU.DLCEP.address:= DLCEP-identifiant DLPDU.DLSDU:= DLSDU
CHECK_PAR_DT	DLPDU	Vrai/Faux	Vérifier la validité de la DLPDU spécifiée. Si la validité est confirmée, la valeur "Vrai" est renvoyée
NEXT	Classe de vitesse		Vérifier le maintien de la DLSDU supplémentaire spécifiée par la classe de vitesse. Si tel est le cas, la commande revient au sommet de l'État courant

7.1.4 Commande TX/RX sporadique (STRC)

7.1.4.1 Présentation générale

La commande TX/RX sporadique (STRC) est chargée de la mise en mémoire tampon et de la répartition dans le temps de la DLSDU reçue pour le transfert de données de messagerie sporadiques entre l'utilisateur DLS et l'ACM.

7.1.4.2 Définitions de primitive

7.1.4.2.1 Primitives échangées entre l'utilisateur DLS et la commande STRC

Le Tableau 37 montre toutes les primitives échangées entre l'utilisateur DLS et la commande STRC et le Tableau 38 montre toutes les primitives échangées entre la commande STRC et l'ACM.

Tableau 37 – Primitives échangées entre l'utilisateur DLS et la commande STRC

Nom de la primitive	Source	Paramètres associés
DL-SPDATA.request	Utilisateur DLS	DA, MSDU
DL-SPDATA.confirm	STRC	DA, Etat
DL-SPDATA.indication	STRC	DA, SA, MSDU, Rec-Status

Tableau 38 – Primitives échangées entre la commande STRC et l'ACM

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_SEND_DATA.ind	ACM	DLPDU

7.1.4.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC

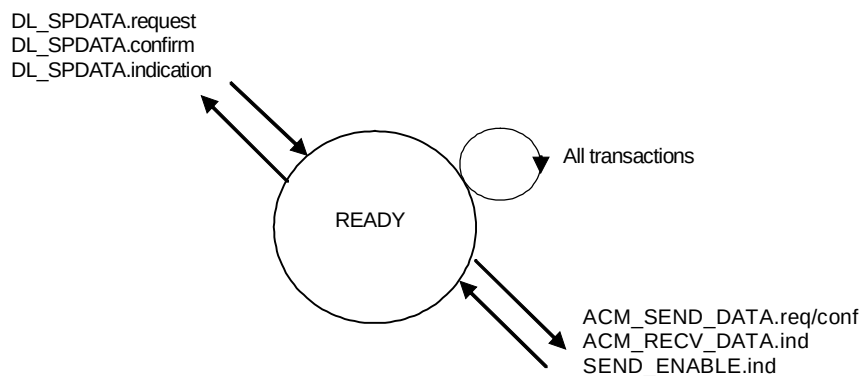
Tous les paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC sont présentés dans le Tableau 39.

Tableau 39 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC

Nom du paramètre	Description
DA	Adresse de destination
SA	Adresse source
MSDU	Unité de données de service MAC
Etat	Indique si le service demandé de DL-SPDATA.request est fourni avec succès ou s'il est défaillant pour le motif spécifié
Rec-Status	Indique si la réception de la DLPDU s'est effectuée sans erreur ou non
DLPDU	DLPDU pour transfert de données de messagerie sporadiques
Classe de vitesse	Classe de vitesse demandée par l'utilisateur DLS. Dans le service de transfert de données de messagerie sporadiques, la classe de vitesse est spécifiée comme "sporadique" ou classe 1.

7.1.4.3 Table d'états STRC

Le diagramme de transition d'état de la commande STRC est illustré à la Figure 22, et la table d'états STRC est présentée dans le Tableau 40.



Légende

Anglais	Français
All transactions	Toutes transactions

Figure 22 – Diagramme de transition d'état de la commande STRC

Tableau 40 – Table d'états STRC

#	État courant	Événement /condition ⇒ action	État suivant
1	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () <> "Full" ⇒ QUEUE_SPDATA (DA, MSDU) DL-SPDATA.confirm { DA, MSDU, Status:= "success" }	READY
2	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () = "Full" ⇒ DL-SPDATA.confirm { DA, MSDU, Status:= "failure - The SPDATA Queue is full" }	READY
3	READY	DL-SPDATA.request {DA, MSDU} / CHECK_PAR_SPDATA (DA, MSDU) = "False" ⇒ DL-SPDATA.confirm { DA, MSDU, Status:= "Failure - Invalid parameter" }	READY
4	READY	SEND_ENABLE.ind {Speed-class} / Speed-class = "SPORADIC" && CHECK_SPDATAQ () <> "Empty" ⇒ DLPDU:= BUILD_SPDATA (DA, MSDU:= DEQUEUE_SPDATA ()) ACM_SEND_DATA.req { DLPDU } ACM_SEND_DATA.conf { } -- immediate response ⇒ (no action taken)	READY
5	READY	ACM_SEND_DATA.ind {DLPDU} / Speed-class = "SPORADIC" && CHECK_SPDATA (DLPDU) ⇒ DL-Buffer-received.indication { DLPDU.DLCEP-address }	READY

7.1.4.4 Fonctions de la commande STRC

Toutes les fonctions de la commande STRC sont récapitulées dans le Tableau 41.

Tableau 41 – Table de fonctions STRC

Nom de la fonction	Entrée	Sortie	Fonctionnement
CHECK_PAR_SPDATA	DA, MSDU	Vrai/Faux	Vérifier la validité de tous les paramètres DA et DSDU fournis avec la DL-SPDATA.request. Si la validité est confirmée, la valeur "Vrai" est renvoyée
CHECK_SPDATAQ	(aucun)	état	Vérifier que la condition File d'attente pour SPDATA est File d'attente maximale. L'état renvoyé est "Complet", "Vide" ou "En file d'attente"
QUEUE_SPDATA	DA, MSDU	(aucun)	Place les données d'entrée dans la file d'attente SPDATA sur une base FIFO
DEQUEUE_SPDATA	(aucun)	DA, MSDU	Retire de la file d'attente SPDATA sur une base FIFO
BUILD_SPDATA	DA, MSDU	DLPDU	Intégrer à la DLPDU des données de messagerie sporadiques. La DLPDU est assemblée comme suit. DLPDU.DA:= MSDU.DA DLPDU.SA:= V(IA) DLPDU.Len/Type:= MSDU.LEN/Type DLPDU.MSDU:= MSDU.DLSDU
CHECK_RCV_SPDATA	DLPDU	Vrai/Faux	Vérifier la validité de la DLPDU spécifiée. Si la validité est confirmée, la valeur "Vrai" est renvoyée

7.1.5 Machine de contrôle d'accès (ACM)

7.1.5.1 Présentation générale

La machine de contrôle d'accès (ACM) est chargée du contrôle d'accès au support déterministe et des possibilités de planification de l'envoi des DLPDU, du contrôle d'ajout et de retrait de nœuds, et de la restauration du réseau suite à une interruption ayant entraîné un arrêt du système.

L'ACM est chargée principalement

- de s'assurer que le nœud local détecte et utilise intégralement sa période d'accès attribuée;
- de s'assurer que le nœud local ne gêne pas les transmissions des autres nœuds, particulièrement le nœud qui transmet la DLPDU SYN;
- de détecter toute interruption du réseau et de lancer la transmission de la DLPDU SYN pour la restauration du réseau après la durée prescrite au cours de laquelle la DLPDU SYN n'est pas entendue;
- d'assurer l'ajout et le retrait d'un nouveau nœud au/du réseau.

7.1.5.2 ACM et fonctions de support de planification

Les fonctions ACM planifient toutes les communications entre les DLE participantes du bus de terrain de Type 11, la temporisation de ces communications étant contrôlée pour:

- satisfaire au contrôle d'accès au support spécifique afin de donner la possibilité à toutes les DLE d'envoyer 2 sortes de classe de données cycliques prioritaires et de données de messagerie sporadiques de manière opportune, hiérarchisée et déterministe, de détecter toute interruption du réseau et de lancer la restauration en temps voulu, puis d'ajouter et de retirer des nœuds de la ligne;

- b) offrir 3 niveaux de possibilités de transfert de données prioritaires afin d'envoyer des données au nœud dans l'ordre séquentiel et dans chaque période prédéfinie, de manière à procéder au transfert de données de chaque niveau dans la durée préalablement spécifiée (temps de maintien du jeton) et de transférer ou conserver les données des niveaux inférieurs sur une période de temps cyclique ultérieure, selon le niveau et l'occasion du niveau supérieur du transfert de données est toujours réalisée à chaque occasion. Par ailleurs, l'ensemble du volume du transfert de données des niveaux inférieurs est transféré dans chaque période préalablement spécifiée;
- c) offrir des possibilités de transfert de données de messagerie sporadiques au nœud indiquant que la demande de transfert se déroule de manière sporadique par l'utilisateur DLS et que les données sont transférées dans la période préalablement spécifiée du niveau de priorité correspondant et selon les applications ISO/CEI 8802-3 normales.

7.1.5.3 Définitions de primitive

7.1.5.3.1 Primitives échangées entre l'ACM et la commande RMC

Le Tableau 42 récapitule toutes les primitives échangées entre l'ACM et la commande RMC.

Tableau 42 – Primitives échangées entre l'ACM et la commande RMC

Nom de la primitive	Source	Paramètres associés
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU

Les paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC sont décrits dans le Tableau 43.

Tableau 43 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC

Nom du paramètre	Description
RMSDU	SDU de RMC
RMC-status	Etat indiquant les résultats de la demande à la commande RMC

7.1.5.3.2 Primitives échangées entre l'ACM et la commande CTRC

Le Tableau 44 récapitule toutes les primitives échangées entre l'ACM et la commande CTRC.

Tableau 44 – Primitives échangées entre l'ACM et la commande CTRC

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	ACM_status
ACM_RECV_DATA.ind	ACM	DLPDU

Les paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTRC sont décrits dans le Tableau 45.

Tableau 45 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTCR

Nom du paramètre	Description
ACM_Status	Etat indiquant les résultats
DLPDU	SDU de ACM
Classe de vitesse	Classe de vitesse indiquant la transmission de données cycliques prioritaires

7.1.5.3.3 Primitives échangées entre l'ACM et la commande STRC

Le Tableau 46 énumère toutes les primitives échangées entre l'ACM et la commande STRC.

Tableau 46 – Primitives échangées entre l'ACM et la commande STRC

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_SEND_DATA.ind	ACM	DLPDU

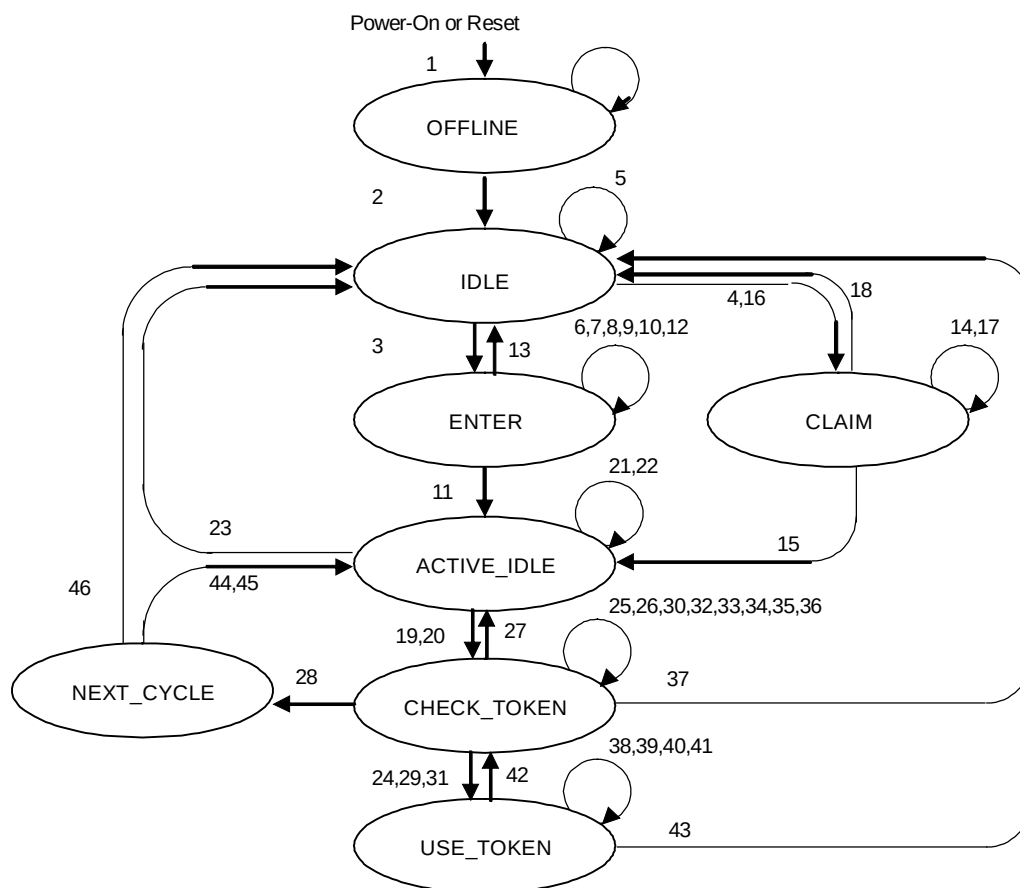
Les paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC sont décrits dans le Tableau 47.

Tableau 47 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC

Nom du paramètre	Description
ACM_Status	Etat indiquant les résultats
DLPDU	SDU de ACM
Classe de vitesse	Classe de vitesse indiquant la transmission de données de messagerie sporadiques

7.1.5.4 Table d'états ACM

Le diagramme de transition d'état est présenté à la Figure 23.



Légende

Anglais	Français
Power-on or Reset	Mise sous tension ou réinitialisation

Figure 23 – Diagramme de transition d'état de l'ACM

L'état "OFFLINE" est activé lors de la mise sous tension ou lors de l'émission de la primitive DLM-Reset.request. Dans cet état, l'ensemble des variables, paramètres et informations de configuration sont définis par la DLE. Lorsque la variable "In_Ring_Desire" devient vraie, l'état passe à "IDLE".

Cet état consiste à attendre pour être prêt à rejoindre les membres du bus de terrain de Type 11. Lorsqu'un autre nœud SYN a transmis une DLPDU SYN, l'état passe à l'état "ENTER" dans lequel est transmise la DLPDU REQ revendiquant l'ajout du nœud SYN actuel aux membres du bus de terrain de Type 11. De plus, à l'état "IDLE", lorsque les signaux émis sur le support de transmission demeurent inactifs pendant une période spécifiée, l'état devient "CLAIM", dans lequel le nœud tente d'être un nouveau nœud SYN.

L'état "ACTIVE_IDLE" fait partie des membres du bus de terrain de Type 11, le nœud parvenant à obtenir le droit de transmission par la commande d'accès au support du bus de terrain de Type 11, soit lorsque le nœud fonctionne comme mode SYN par transmission de la DLPDU SYN, ou comme mode non SYN de réception de cette même DLPDU par un autre nœud SYN, l'état étant "CHECK_TOKEN".

L'état "CHECK_TOKEN" attend jusqu'à ce que la valeur $V(Tsl)$ soit égale à $V(TN)$ afin d'envoyer les données sur le support de transmission. Lorsque $V(Tsl)$ est égale à $V(TN)$, l'état passe à l'état "USE_TOKEN".

L'état "USE_TOKEN" correspond à l'état dans lequel le nœud est capable d'envoyer les données d'émission cyclique prioritaire et les données de transmission sporadiques. Lorsque toutes les données ont été transmises ou à l'expiration du temps de maintien du jeton, l'état passe à l'état "NEXT_CYCLE".

Cet état "NEXT_CYCLE" consiste à solliciter un nouveau nœud comme fonction principale.

La table d'états ACM est présentée au Tableau 48.

Tableau 48 – Table d'états ACM

#	État courant	Événement /condition ⇒ action	État suivant
1	Any states	POWER-ON or RESET =>	OFFLINE
2	OFFLINE	In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && RMSDU.PN = V(TN) && V (LL)<V(TN)> = 0 => -- receiving SYN frame and LL this node not live -- SYN_frame:= RMSDU V(LN):= 1 START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	EXPIRED_TIMER (T(SL)) = "True" => RMSDU:= BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / => -- receiving a frame except the above -- START_TIMER (T(SL), V(SL))	IDLE
6	ENTER	V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(Tsl))	ENTER
7	ENTER	V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER
8	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN) = V(MN) => -- Live Node = Maximum Node No. -- RMSDU:= BUILD_PDU (REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
9	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++ START_TIMER (T(SL), V(SL))	ENTER
10	ENTER	RMC_SEND_DATA.conf { } => (none)	ENTER

#	État courant	Événement /condition ⇒ action	État suivant
11	ENTER	RMC_RECV_DATA.ind { RMSDU } && FC = SYN / V (LL)<V(TN)> = 1 => SYN_frame:= RMSDU V(LN):= 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
12	ENTER	EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
13	ENTER	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
14	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time <= V(MN) => All_Slot_Time++ START_TIMER (T(SL), V(SL))	CLAIM
15	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = V(MN) => V(LL) = 0 V(LL)<V(TN)>:= 1 RMSDU:= BUILD_PDU (SYN) SYN_frame:= RMSDU RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
16	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => START_TIMER (T(SL), V(SL))	IDLE
17	CLAIM	RMC_RECV_DATA.ind { RMSDU } => -- receiving a frame except the above -- START_TIMER (T(SL), V(SL))	CLAIM
18	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
19	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN => -- receiving SYN frame-- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1	CHECK_TOKEN
20	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = SYN => -- sent SYN frame-- SYN_mode:= "True" V(LN):= 1	CHECK_TOKEN
21	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = DT => -- sending DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
22	ACTIVE_IDLE	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC <> Type 11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE

#	État courant	Événement /condition ⇒ action	État suivant
23	ACTIVE_IDLE	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
24	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) => -- Get token -- Speed-class:= 3 – High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(MAC), V(MAC))	USE_TOKEN
25	CHECK_TOKEN	V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN
26	CHECK_TOKEN	V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	CHECK_TOKEN
27	CHECK_TOKEN	/V(LN) = V(MN) && SYN_frame .PN = V(TN) && SYN_mode = "False" => -- Non-SYN mode, on MAC control time -- IF Required COM frame THEN RMSDU:= BUILD_PDU (COM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) ENDIF	ACTIVE_IDLE
28	CHECK_TOKEN	/V(LN) = V(MN) && SYN_frame .PN <> V(TN) && SYN_mode = "True" => -- in SYN mode, on MAC control time -- START_TIMER (T(MAC), V(MAC))	NEXT_CYCLE
29	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN)++ = V(TN) => -- Get token -- -- matching V(LN) and on receiving CMP frame -- STOP_TIMER (T(SCMP)) Speed-class:= 3 -- High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
30	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CMP && V(LN)++ <> V(TN) => -- not matching V(LN) and on receiving CMP frame -- START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN
31	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT-CMP && V(LN)++ = V(TN) => -- matching V(LN) and on receiving DT-CMP frame -- Speed-class:= 3 -- High Priority START_TOKEN_HOLD_TIMER () SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
32	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT-CMP && V(LN)++ <> V(TN) => -- not matching V(LN) and on receiving DT-CMP frame START_TIMER (T(SCMP), V(Tsl))	CHECK_TOKEN

#	État courant	Événement /condition ⇒ action	État suivant
33	CHECK_TOKEN	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC = DT => -- sending DT frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
34	CHECK_TOKEN	RMC_RECV_DATA.conf { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
35	CHECK_TOKEN	EXPIRED_TIMER (T(SCMP)) / => V(LN)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif START_TIMER (T(SL), V(SL))	CHECK_TOKEN
36	CHECK_TOKEN	RMC_SEND_DATA.conf { RMSDU } / RMSDU.FC = CMP => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
37	CHECK_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
38	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send DT or Sporadic frame -- RMPDU:= BUILD_PDU (DLPDU) RMC_SEND_DATA.req { DLPDU } START_TIMER (T(SL), V(SL))	USE_TOKEN
39	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (DLPDU.FC = DT) (DLPDU.FC <> Type-11 fieldbus && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
40	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (DLPDU.FC = DT) (DLPDU.FC <> Type-11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> 0 => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class:= Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
41	USE_TOKEN	ACM_SEND_DATA.req { NIL } / Speed-class <> 0 => -- Next Speed-class -- Speed-class:= Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	État courant	Événement /condition ⇒ action	État suivant
42	USE_TOKEN	ACM_SEND_DATA.req { NIL } / Speed-class = 0 => -- All sent -- RMSDU:= BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
43	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
44	NEXT_CYCLE	/EXIPERED_TIMER (T(MAC)) = "True" =>	ACTIVE_IDLE
45	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ => -- receiving REQ from non-SYN node -- V(LL)<RMSDU.SN>:= 1 RMSDU:= BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } SYN_frame:= RMSDU START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
46	NEXT_CYCLE	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE

7.1.5.5 Fonctions utilisées par l'ACM

Toutes les fonctions utilisées par l'ACM sont récapitulées dans le Tableau 49.

Tableau 49 – Table de fonctions ACM

Nom de la fonction	Entrée	Sortie	Fonctionnement
BUILD_PDU	FC	RMSDU	Conformément au FC (Contrôle de trames) demandé, assembler la trame de bus de terrain de Type 11
EXPIRED_TIMER	T(x)	Vrai/Faux	Lorsque le temporisateur demandé x a expiré, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
MAINT_LL	V(LN)	(aucun)	Le nœud spécifié V(LN) est lu à l'expiration de T(SCMP). A l'expiration de T(SCMP), à tous les moments de prolongation de C(ASCMP) dans la période précédente, le bit correspondant de V(LL) est supprimé
START_TIMER	T(x), V(y)	(aucun)	Le temporisateur x est défini par la valeur de y, puis est activé
START_THT	Classe de vitesse	(aucun)	T(ATHT) est activée selon la valeur restante V(TTRT) de la classe d'accès spécifiée par la classe de vitesse
START_TTRT	Classe de vitesse	(aucun)	T(ATTRT) est activée selon la valeur V(TTRT) de la classe d'accès spécifiée par la classe de vitesse
STOP_TIMER	T(x)	(aucun)	Le temporisateur x est désactivé

7.1.6 Commande de support de redondance (RMC)

7.1.6.1 Présentation générale

La commande de support de redondance (RMC) gère la sélection de deux entrées issues de deux verrouilleurs de trames RX, ainsi que de deux sorties pour deux convertisseurs parallèle-série d'octets pour un support fiable, sans interruption de services de transfert de données de niveau supérieur. La fonction de RMC est la suivante:

- a) Lorsque l'ACM a transmis les DLPDU, la commande RMC les répartit en demandes de symboles d'octets à l'intention des deux convertisseurs parallèle-série d'octets, de sorte que les DLPDU sont transmises sur les deux supports de transmission A et B simultanément;
- b) Lorsqu'une collision s'est produite lors de l'envoi des DLPDU sur un des deux supports de transmission, la transmission de données ayant subi ladite collision est interrompue immédiatement et le signal JAM spécifié par l'ISO/CEI 8802-3 est transmis pour la DLPDU suivante prête à être transmise. Néanmoins, en l'absence de collision sur un autre support de transmission, la transmission des données sur le support correspondant se poursuit sans interruption;
- c) La commande RMC sélectionne une sortie parmi deux sorties des convertisseurs série-parallèle d'octets. La sélection s'effectue sur une base de premier-arrivé d'une trame sans erreur, la commande RMC soumettant les octets de la trame à l'ACM;
- d) La commande RMC prévoit une fonction destinée à sélectionner et désigner une entrée sur deux entrées issues de deux verrouilleurs de trames RX. Cette fonction est gérée selon la valeur du paramètre RMSEL dans le champ CW de la DLPDU SYN reçue. La relation entre le choix du support de redondance et la valeur du paramètre RMSEL est spécifiée en 6.2.3.2.1.3. L'ordre de commande par le paramètre RMSEL est ignoré dans le cas d'un système à support unique.

7.1.6.2 Définitions de primitive

7.1.6.2.1 Primitives échangées entre l'ACM et la commande RMC

Le Tableau 50 présente toutes les primitives échangées entre l'ACM et la commande RMC. Le Tableau 51 présente les primitives échangées entre la commande RMC et le convertisseur parallèle-série/série-parallèle; par ailleurs, les primitives échangées entre la commande RMC et la couche Ph sont présentées dans le Tableau 52.

Tableau 50 – Primitives échangées entre l'ACM et la commande RMC

Nom de la primitive	Source	Paramètres associés
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU

Tableau 51 – Primitives échangées entre la commande RMC et le convertisseur parallèle-série/série-parallèle

Nom de la primitive	Source	Paramètres associés
TX_DATA_A.req	RMC	RMPDU
TX_DATA_B.req	RMC	RMPDU
TX_DATA_A.conf	Serializer_A	Etat
TX_DATA_B.conf	Serializer_B	Etat
RX_DATA_A.ind	Deserializer_A	RMPDU, Etat
RX_DATA_B.ind	Deserializer_B	RMPDU, Etat

Tableau 52 – Primitives échangées entre la commande RMC et la couche Ph

Nom de la primitive	Source	Paramètres associés
PLS_CARRIER_A.indication	Ph-layer_A	CARRIER_STATUS(ON/ OFF)
PLS_SIGNAL_A.indication	Ph-layer_A	SIGNAL_STATUS(ERROR/NO _ERROR)
PLS_DATA_VALID_A.indication	Ph-layer_A	DATA_VALID_STATUS(DATA_VALID/DATA_NOT_VALID)
PLS_CARRIER_B.indication	Ph-layer_B	CARRIER_STATUS(CARRIER_ON/CARRIER_OFF)
PLS_SIGNAL_B.indication	Ph-layer_B	SIGNAL_STATUS(SIGNAL_ERROR/NO_SIGNAL_ERROR)
PLS_DATA_VALID_B.indication	Ph-layer_B	DATA_VALID_STATUS(DATA_VALID/DATA_NOT_VALID)

7.1.6.2.2 Paramètres utilisés entre la commande RMC et l'ACM, et la couche Ph

Le Tableau 53 et le Tableau 54 présentent les paramètres utilisés pour l'interaction entre la commande RMC et l'ACM ou la couche Ph respectivement.

Tableau 53 – Paramètres entre la commande RMC et l'ACM

Nom du paramètre	Description
RMSDU	Unité de données de service utilisée entre la commande RMC et l'ACM
RMC-status	Etat de résultat de demande, condition de réception, etc.
TX_ENABLE	Commande Activer/Désactiver pour la transmission de données

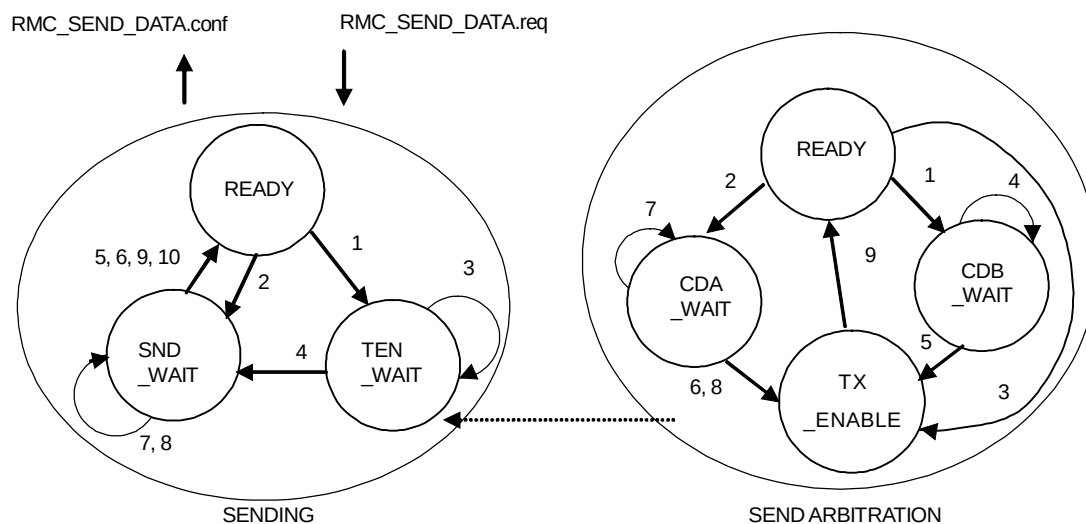
Tableau 54 – Paramètres entre la commande RMC et la couche Ph

Nom du paramètre	Description
CARRIER_STATUS	Etat de condition d'application du signal porteur sur le support ou non
SIGNAL_STATUS	Etat d'occurrence de collision
DATA_VALID_STATUS	Etat de données reçues disponibles

7.1.6.3 Table d'états RMC

7.1.6.3.1 Table d'états d'envoi et arbitrage d'envoi

La Figure 24 présente le diagramme de transition d'état de la commande RMC d'envoi et d'arbitrage d'envoi. La table d'états de la commande RMC d'envoi est présentée dans le Tableau 55 et la table d'états de la commande RMC d'arbitrage d'envoi est présentée dans le Tableau 56.

**Légende**

Anglais	Français
Sending	Envoi
Send arbitration	Arbitrage d'envoi

Figure 24 – Diagramme de transition d'état de la commande RMC d'envoi et d'arbitrage d'envoi

Tableau 55 – Table d'états de la commande RMC d'envoi

#	État courant	Événement /condition ⇒ action	État suivant
1	READY	RMC_SEND_DATA.req { RMSDU } / RMC_STATE <> "TX_ENABLE" ⇒ (no action)	TEN_WAIT
2	READY	RMC_SEND_DATA.req { RMSDU } / RMC_STATE = "TX_ENABLE" ⇒ ----- Under transmission enable SNDC = TX_DATA_AB (RMSDU)	SND_WAIT
3	TEN_WAIT	/ RMC-STATE <> "TX_ENABLE" ⇒ (no action)	TEN_WAIT
4	TEN_WAIT	/ RMC_STATE = "TX_ENABLE" ⇒ ----- Channel A into transmission enable SNDC = TX_DATA_AB (RMSDU)	SND_WAIT
5	SND_WAIT	TX_DATA_A.conf { } / SNDC = 1 ⇒ -----Transmission completed through Channel A RMC_SEND_DATA.conf { RMC-status:= "success"}	READY
6	SND_WAIT	TX_DATA_B.conf { } / SNDC = 1 ⇒ -----Transmission completed through Channel B RMC_SEND_DATA.conf { RMC-status:= "success"}	READY
7	SND_WAIT	TX_DATA_A.conf { } / SNDC = 2 ⇒ ---- Transmission completed on Channel A but B incomplete SNDC:= SNDC - 1	SND_WAIT
8	SND_WAIT	TX_DATA_B.conf { } / SNDC = 2 ⇒ ---- Transmission completed on Channel B but A incomplete SNDC:= SNDC - 1	SND_WAIT
9	SND_WAIT	TX_DATA_A.conf { } / SNDC = 0 ⇒ -----Transmission completed through Channel A RMC_SEND_DATA.conf { RMC-status:= "fail"}	READY
10	SND_WAIT	TX_DATA_B.conf { } / SNDC = 0 ⇒ -----Transmission completed through Channel B RMC_SEND_DATA.conf { RMC-status:= "fail"}	READY

Tableau 56 – Table d'états de la commande RMC d'arbitrage d'envoi

#	État courant	Événement /condition ⇒ action	État suivant
1	READY	PLS_CARRIER_A.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over medium A START_TIMER (T(AIGP _A)) START_TIMER (T(ARMGP))	CDB_WAIT
2	READY	PLS_CARRIER_B.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over medium B START_TIMER (T(AIGP _B)) START_TIMER (T(ARMGP))	CDA_WAIT
3	CDB_WAIT	EXPIRED_TIMER (T(AIGPT)) = "True" / ⇒ ----- No transmission from Channel A	TX_ENABLE
4	CDB_WAIT	PLS_CARRIER_B.indication { CARRIER_STATUS (OFF) } / ⇒ ----- No transmission from both Channel A and B START_TIMER (T(AIGP _B)) STOP_TIMER (T(ARMGP))	CDB_WAIT
5	CDB_WAIT	EXPIRED_TIMER (T(AIGP _B)) = "True" / ⇒ ----- Inter-packet-gap time elapsed over channel B	TX_ENABLE
6	CDA_WAIT	EXPIRED_TIMER (T(ARMGP)) = "True" / ⇒ ----- No transmission from Channel A	TX_ENABLE
7	CDA_WAIT	PLS_CARRIER_A.indication { CARRIER_STATUS (OFF) } / ⇒ ----- Carrier Off over both medium A and B START_TIMER (T(AIGP _A)) STOP_TIMER (T(ARMGP))	CDB_WAIT
8	CDB_WAIT	EXPIRED_TIMER (T(AIGP _A)) = "True" / ⇒ ----- Inter-packet-gap time elapsed over channel A	TX_ENABLE
9	TX_ENABLE	/ ⇒ ----- Signal TX_ENABLE to Sending state machine	READY

Dans les transitions d'état, la commande RMC fonctionne comme suit:

- Lorsque l'ACM a transmis les DLPDU, la commande RMC les répartit en demandes de symboles d'octets à l'intention des deux convertisseurs parallèle-série d'octets, de sorte que les DLPDU sont transmises sur les deux supports de transmission A et B simultanément;
- Lorsqu'une collision s'est produite lors de l'envoi des DLPDU sur un des deux supports de transmission, la transmission de données ayant subi ladite collision est interrompue immédiatement et le signal JAM spécifié par l'ISO/CEI 8802-3:2000, Article 4, est transmis pour la DLPDU suivante prête à être transmise. Néanmoins, en l'absence de collision sur un autre support de transmission, la transmission des données sur le support correspondant se poursuit sans interruption;
- Lorsqu'une collision s'est produite sur les deux supports de transmission, le signal JAM est transmis sur ces deux supports. Dans ce cas, aucune nouvelle transmission ne se produit;
- Les deux cas de collision se produisant sur un ou les deux supports de transmission sont réalisés sur la base de la DLPDU par la DLE;
- Le signal TX_ENABLE est effectivement activé par la valeur ultérieure de la temporisation expirée de T(GPA) et T(GPB);

- f) Après expiration de l'une des valeurs de T(AIGP*) et attente supplémentaire de l'expiration de T(ARMGP), et en l'absence de réception à l'expiration de l'autre valeur T(AIGP*), la commande RMC détermine que le canal correspondant adopte un état de fonctionnement anormal. Le signal d'expiration de la valeur T(AIGP*) reçue signifie la restauration de l'état normal.

7.1.6.3.2 Table d'états pour la commande RMC de réception

La Figure 25 illustre le diagramme de transition d'état de la commande RMC de réception, et la table d'états de la commande RMC de réception est présentée dans le Tableau 57.

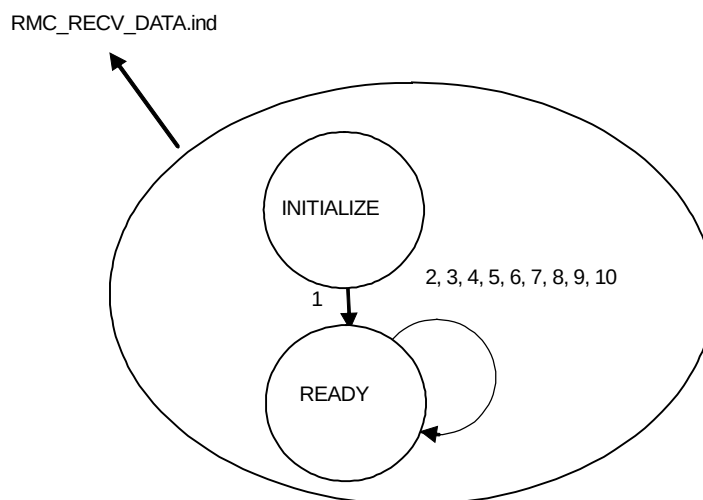


Figure 25 – Diagramme de transition d'état de la commande RMC de réception

Tableau 57 – Table d'états de la commande RMC de réception

#	État courant	Événement /condition ⇒ action	État suivant
1	INITIALIZE	POWER-ON or RESET / ⇒ ----- Determine Receive channel RCVM:= DECIDE_RCVM ()	READY
2	READY	FORCE_RCVM(RCS) / ⇒ RCVM:= DECIDE_RCVM ()	READY
3	READY	RX_DATA_A.ind { RMPDU } / RCVM = MEDIA-A && Status:= NORMAL && RMPDU = "SYN" frame ⇒ LAST_RMA:= NORMAL RMSDU:= RMPDU RMC_RECV_DATA.ind { RMSDU } RCVM:= DECIDE_RCVM ()	READY
4	READY	RX_DATA_A.ind { RMPDU } / RCVM = MEDIA-A && Status:= "NORMAL" ⇒ LAST_RMA:= "NORMAL" RMSDU:= RMPDU RMC_RECV_DATA.ind { RMSDU }	READY
5	READY	RX_DATA_A.ind { RMPDU } / RCVM <> MEDIA-A && Status:= "NORMAL" ⇒ LAST_RMA:= "NORMAL"	READY

#	État courant	Événement /condition ⇒ action	État suivant
6	READY	RX_DATA_A.ind { RMPDU } / Status:= "ERROR" ⇒ LAST_RMA:= ERROR RCVM:= DECIDE_RCVM ()	READY
7	READY	RX_DATA_B.ind { RMPDU } / RCVM = MEDIA-B && Status:= "NORMAL" && RMPDU = "SYN" frame ⇒ LAST_RMB:= "NORMAL" RMSDU:= RMPDU RMC_RECV_DATA.ind { RMSDU } RCVM:= DECIDE_RCVM ()	READY
8	READY	RX_DATA_B.ind { RMPDU } / RCVM = MEDIA-B && Status:= "NORMAL" ⇒ LAST_RMB:= "NORMAL" RMSDU:= RMPDU RMC_RECV_DATA.ind { RMSDU }	READY
9	READY	RX_DATA_B.ind { RMPDU } / RCVM <> MEDIA-B && Status:= "NORMAL" ⇒ LAST_RMB:= "NORMAL"	READY
10	READY	RX_DATA_A.ind { RMPDU } / Status:= "ERROR" ⇒ LAST_RMB:= "ERROR" RCVM:= DECIDE_RCVM ()	READY

Dans les transitions d'état, la commande RMC fonctionne comme suit:

- a) Le support de réception (Receive-Media) A est choisi à la phase initiale;
- b) La commande de commutation par "FORCE_RCV()" est exécutée immédiatement après réception de l'ordre d'exécution;
- c) La politique de commutation est la suivante:
 - Le support de réception A ou B est choisi selon "Manual-select";
 - Le support de réception A ou B est choisi selon le mode spécifié par le paramètre RMSEL dans la DLPDU SYN reçue;
 - Une fois choisi et en l'absence d'erreur, le support de réception choisi est maintenu;
 - En cas d'erreur, l'autre support de réception est choisi.

7.1.6.4 Fonctions utilisées par la commande RMC

Les fonctions utilisées par la commande RMC sont récapitulées dans le Tableau 58.

Tableau 58 – Table de fonctions RMC

Nom de la fonction	Description et fonctionnement
EXPIRED_TIMER (T(X))	Lorsque le temporisateur demandé x a expiré, la valeur "True" (vrai) est renvoyée, la valeur "False" (faux) étant renvoyée dans le cas contraire
FORCE_RCV (RCS)	<pre> if RCS:= Force_A then RCV:= MEDIA-A endif if RCS:= Force_B then RCV = MEDIA-B if RCS:= Automatic then RCV:= DECIDE_RCV() </pre>
START_TIMER (v(t))	Activer le temporisateur spécifié. STOP_TIMER() contraint le temporisateur spécifié et en fonctionnement à s'arrêter. L'événement "Temporisateur expiré" se produit à l'expiration d'un temporisateur désigné
STOPT_TIMER(v(t))	Contraint le temporisateur spécifié à s'arrêter
DECIDE_RCV ()	Le support de réception est choisi par "FORCE_RECV", "Manual-select", "RMSEL dans la DLPDU SYN", puis est commuté sur une base conditionnelle de la DLPDU précédente reçue sous mode "Automatic" La commande de commutation est exécutée dans l'ordre de priorité descendante de "FORCE_RCV", "Manual-select", "RMSEL dans la DLPDU SYN", puis est commutée sur une base conditionnelle sous mode "Automatic". <pre> If FORCE_RCV(RCS:= MEDIA-A or:= MEDIA-B) then return MEDIA-A or MEDIA-B f else if "Manual-select":= MEDIA-A then return MEDIA-A endif else if "Manual-select":= MEDIA-B then return MEDIA-B endif else if RMSEL of SYNDLPDU:= MEDIA-A then return MEDIA-A endif else if RMSEL of SYN DLPDU:= MEDIA-B then return MEDIA-B endif else if RCV:= MEDIA-A && LAST_RMA <> NORMAL then return MEDIA-B endif else if RCV:= MEDIA-B && LAST_RMB <> NORMAL then return MEDIA-A endif endif </pre>
TX_DATA_AB (PDU)	Transmettre la PDU sur les deux supports A et B simultanément; cette transmission est toutefois interdite sur le support rejeté. Les comptages sollicités de TX_DATA_A ou B. request sont renvoyés. <pre> sndc:= 0; if Media-A is permitted to send out then TX_DATA_A.req { PDU } sndc:= sndc + 1 endif if Media-B is permitted to send out then TX_DATA_B.req { PDU } sndc:= sndc + 1 endif return sndc </pre>

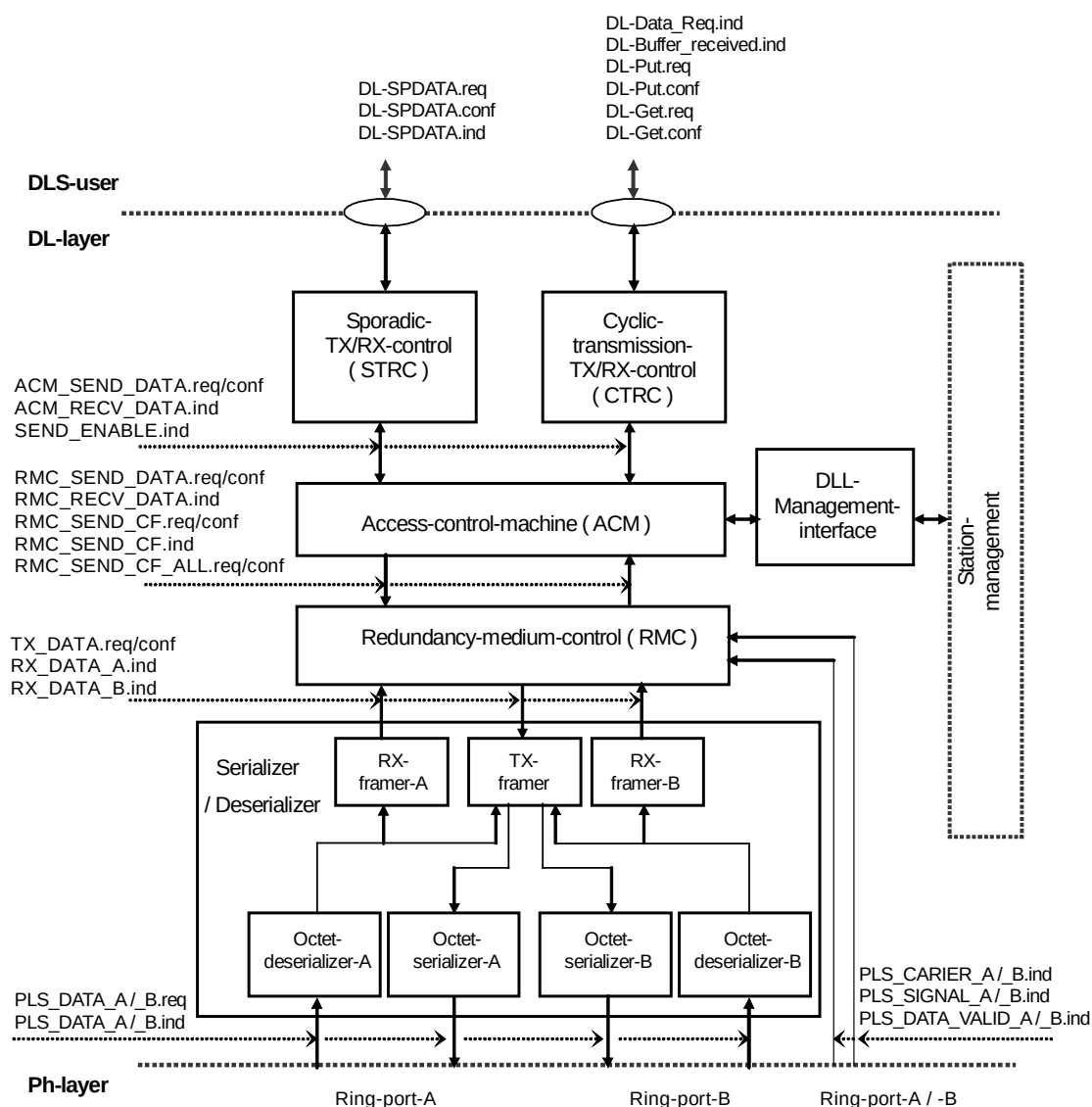
7.2 Éléments de procédure DLE d'une architecture en boucle

7.2.1 Structure générale

La Figure 26 illustre la structure générale de la DLL. Cette dernière est constituée des éléments de commande suivants: commande TX/RX d'émission cyclique (CTRC), commande TX/RX sporadique (STRC), machine de contrôle d'accès (ACM), commande de support de redondance (RMC), verrouilleur de trames TX/RX, convertisseur parallèle-série d'octets, convertisseur série-parallèle d'octets et interface de gestion DLL.

La machine de contrôle d'accès (ACM), en tant que composant de commande principal, assure la fonction de contrôle d'accès au support déterministe en coopération avec les commandes CTRC, STRC et RMC, afin d'assurer une prise en charge fiable et efficace des services de transfert de données en mode de connexion de niveau supérieur et en mode sans connexion.

L'interface de gestion DLL offre des fonctions de gestion DLL.



Légende

Anglais	Français
DLS user	utilisateur DLS
DL layer	couche DL
Sporadic TX/RX control (STRC)	Commande TX/RX sporadique (STRC)
cyclic transmission TX/TX control (CTRC)	Commande TX/RX d'émission cyclique (CTRC)
access control machine (ACM)	Machine de contrôle d'accès (ACM)
DLL management interface	interface de gestion DLL
station management	gestion de station
redundancy medium control (RMC)	Commande de support de redondance (RMC)
serializer / deserializer	Convertisseur parallèle-série / série-parallèle
TX framer	verrouilleur de trames TX
RX framer	verrouilleur de trames RX
octet serializer	Convertisseur parallèle-série d'octets
octet deserializer	Convertisseur série-parallèle d'octets
Ph-layer	couche Ph

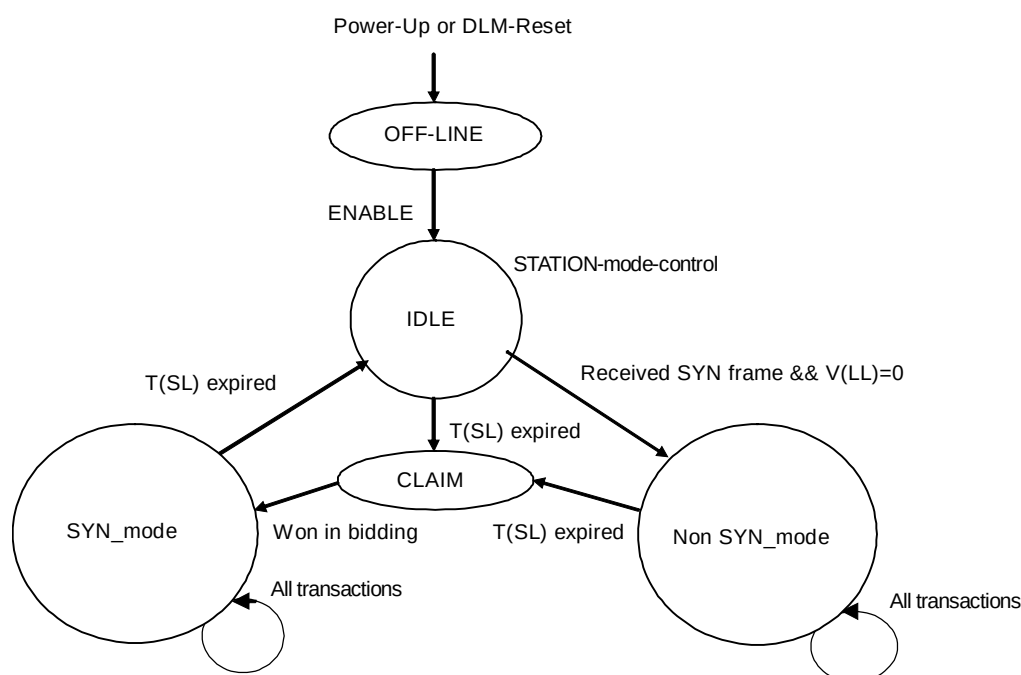
Figure 26 – Structure générale de la DLL

7.2.2 Initialisation

La DLE doit passer à l'état OFF-LINE à la mise sous tension ou après réception de la primitive de demande DLM-RESET. A l'état OFF-LINE, la DLE ne doit pas transmettre et doit ignorer toutes les DLPDU reçues.

Lorsque toutes les variables relatives à un fonctionnement DLE normal sont définies par la primitive de demande DLM-Set-Value, l'état passe de OFF-LINE à STATION MODE CONTROL, dans lequel la DLE doit initier un fonctionnement DL normal.

La Figure 27 illustre la transition des états DLE.



Légende

Anglais	Français
Power-up or DLM-Reset	Mise sous tension ou DLM-Reset
T(SL) expired	T(SL) expiré
Received SYN frame && V(LL)=0	Trame SYN reçue && V(LL)=0
Won in bidding	Acquis par soumission
Non SYN_mode	Non SYN-mode
All transactions	Toutes transactions

Figure 27 – Transition d'états DLE

7.2.3 Commande TX/RX d'émission cyclique (CTRC)

7.2.3.1 Présentation générale

La commande TX/RX d'émission cyclique (CTRC) est chargée de la mise en mémoire tampon et de la répartition de la DLSDU reçue pour le transfert de données cycliques prioritaires entre l'utilisateur DLS et l'ACM.

La primitive d'indication DL-Data-Req de la commande CTRC informe l'utilisateur DLS qu'il peut commencer le transfert de données entre chaque mémoire tampon d'utilisateur DLS et la mémoire tampon transmise de la DLE, en utilisant la primitive de demande DL-Put. La

mémoire tampon de l'utilisateur DLS est adressée avec le paramètre d'identifiant DLCEP dans la primitive d'indication DL-Data-Req de la commande CTRC.

Chaque mémoire tampon de l'utilisateur DLS contient des données dédiées à la transmission de données cycliques prioritaires. Après réception des données de l'utilisateur DLS, la CTRC émet une primitive de demande ACM_SEND_DATA afin que l'ACM commence à transmettre lesdites données sous forme de DLPDU DT ou DT-CMP.

La CTRC émet une primitive d'indication DL-Data-Req en réponse à la primitive d'indication SEND_ENABLE de l'ACM. Le paramètre de classe de vitesse de la primitive d'indication SEND_ENABLE permet à la CTRC d'activer le transfert de la classe correspondante de la mémoire tampon des données par l'utilisateur DLS. Les classes désignées par le paramètre de classe de vitesse correspondent aux classes de transfert de données cycliques à grande, moyenne et basse vitesses.

Le transfert de données de chaque classe de vitesse désignée est fonction du niveau de priorité de transfert:

- a) Pour ce qui concerne la classe de priorité 3, qui correspond au transfert de données cycliques à grande vitesse, et à réception de la primitive d'indication SEND_ENABLE, la CTRC doit traiter l'ensemble des données de chaque mémoire tampon d'utilisateur DLS correspondant au transfert de données cycliques à grande vitesse, ou la classe à grande vitesse est à fournir à l'ACM et être transmise sur le support d'émission dans son intégralité et en une seule fois dans tous les cas;
- b) Pour ce qui concerne la classe de priorité 2, qui correspond au transfert de données cycliques à moyenne vitesse ou à la classe à moyenne vitesse respectivement, la CTRC doit traiter dans tous les cas les données de chaque mémoire tampon de l'utilisateur DLS correspondant à la classe à moyenne vitesse, transmise ou non selon les conditions en cours. Le temps de maintien du jeton de chaque classe de priorité 2 régit ces conditions. Les données cycliques à grande vitesse sont transmises après l'envoi de toutes les données cycliques à grande vitesse. Lorsque le temps de maintien du jeton s'achève lors de l'envoi des données cycliques à moyenne vitesse, le transfert de données est interrompu et la DLPDU CMP ou DT-CMP est transmise dans ce cas. Le reste des données cycliques à moyenne vitesse précédentes est à transmettre dès que possible selon les mêmes conditions. Pour ce qui concerne la classe de priorité inférieure n, le même contrôle que celui applicable au transfert des données cycliques à moyenne vitesse est à effectuer. Le cas se présente à chaque période $V(TSYN)$.

Dans le cas du fonctionnement à 100 Mbps, 3 classes de la classe de vitesse doivent être prises en charge.

- La classe de vitesse 3 est dédiée au transfert de données cycliques à grande vitesse, la classe de vitesse 2 est dédiée au transfert de données cycliques à moyenne vitesse, et la classe de vitesse 1 est dédiée au transfert de données de messagerie sporadiques.

D'autre part, dans le cas du fonctionnement à 1 000 Mbps, 4 classes de la classe de vitesse doivent être prises en charge, comme suit.

- La classe de vitesse 3 est dédiée au transfert de données cycliques à grande vitesse, la classe de vitesse 2 est dédiée au transfert de données cycliques à moyenne vitesse, la classe de vitesse 1 est dédiée au transfert de données cycliques basse vitesse et la classe de vitesse 0 est dédiée au transfert de données de messagerie sporadiques.

Lorsque l'ACM a émis la primitive d'indication ACM_RECV_DATA, la commande CTRC archive la DLPDU transmise par ladite ACM dans chaque mémoire tampon de réception, adressée avec le paramètre d'identifiant DLCEP dans cette même primitive. Par ailleurs, la CTRC émet une primitive d'indication DL-Buffer-received pour informer l'utilisateur DLS de l'actualisation et de la disponibilité de lecture de la mémoire tampon de réception de la DLE. L'utilisateur DLS peut lire les données contenues dans cette mémoire en utilisant la primitive de demande DL-Get en réponse à la primitive d'indication DL-Buffer-received émise par la CTRC.

7.2.3.2 Définitions de primitive

7.2.3.2.1 Primitives échangées entre l'utilisateur DLS et la commande CTRC

Les primitives échangées entre l'utilisateur DLS et la commande CTRC, et les primitives échangées entre la commande CTRC et l'ACM sont identiques à celles applicables à l'architecture en étoile, et sont récapitulées dans le Tableau 59 et le Tableau 60 respectivement.

Tableau 59 – Primitives échangées entre l'utilisateur DLS et la commande CTRC

Nom de la primitive	Source	Paramètres associés
DL-Data-Req.ind	CTRC	Identifiant DLCEP
DL-Put.req	Utilisateur DL	identifiant DLCEP, longueur DLSDU, DLSDU
DL-Put.conf	CTRC	Identifiant DLCEP, Etat
DL-Buffer-received.ind	CTRC	Identifiant DLCEP
DL-Get.req	Utilisateur DL	Identifiant DLCEP
DL-Get.conf	CTRC	Identifiant DLCEP, longueur DLSDU DLSDU Etat

Tableau 60 – Primitives échangées entre la commande CTRC et l'ACM

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_RECV_DATA.ind	ACM	DLPDU

7.2.3.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC

Les paramètres utilisés pour l'interaction entre l'utilisateur DLS et la commande CTRC sont récapitulés dans le Tableau 61.

Tableau 61 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande CTRC

Nom du paramètre	Description
Identifiant DLCEP	Identifiant permettant de désigner le Send-Buffer ou le Receive-Buffer
DLSDU	Contenu de Send-Buffer ou Receive-Buffer, qui correspond aux données cycliques prioritaires traitées par la commande CTRC
Longueur DLSDU	Longueur de DLSDU
Classe de vitesse	Classe de vitesse permettant de désigner la classe de transfert des données cycliques prioritaires, c'est-à-dire les données cycliques à grande, à moyenne et à basse vitesses La valeur de la classe de vitesse pour les données cycliques à grande vitesse est 3, et de 2 pour les données cycliques à moyenne vitesse. Pour les données cycliques à basse vitesse pour le fonctionnement à 1 000 Mbps, la valeur de la classe de vitesse est 1
Etat	Rapport d'état indiquant si le service demandé est fourni avec succès ou s'il est défaillant pour le motif spécifié

7.2.3.3 Table d'états CTRC

Le diagramme de transition d'état de la commande CTRC est illustré à la Figure 28, et la table d'états CTRC est présentée dans le Tableau 62.

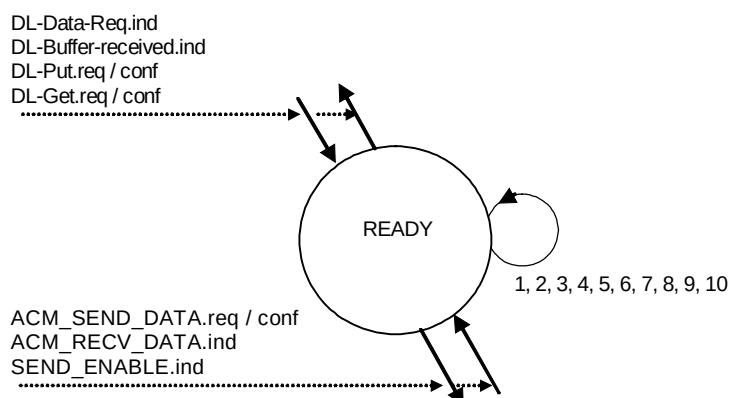


Figure 28 – Diagramme de transition d'état de la commande CTRC

Tableau 62 – Table d'états CTRC

#	État courant	Événement /Condition => action	État suivant
1	READY	DL-Put.req { DLCEP-identifiant, DLSDU-length, DLSDU } / CHECK_PAR_PUT (DLCEP-identifiant, DLSDU) = "True" => Status:= "success" PUT_BUFFER (DLCEP-identifiant, DLSDU) DL-Put.conf { DLCEP-identifiant, DLSDU-length, Status }	READY
2	READY	DL-Put.req { DLCEP-identifiant, DLSDU-length, DLSDU } / CHECK_PAR_PUT (DLCEP-identifiant, DLSDU) = "False" => Status:= "failure" DL-Put.conf { DLCEP-identifiant, DLSDU-length, Status }	READY
3	READY	DL-Get.req { DLCEP-identifiant } / CHECK_PAR_GET (DLCEP-identifiant) = "True" => Status:= "success" DLSDU:= GET_BUFFER (DLCEP-identifiant) DL-Get.conf { DLCEP-identifiant, DLSDU-length, DLSDU, Status }	READY

#	État courant	Événement /Condition => action	État suivant
4	READY	DL-Get.req { DLCEP-identifiant } / CHECK_PAR_GET (DLCEP-identifiant) = "False" => Status:= "failure" DLSDU:= "null" DL-Get.conf { DLCEP-identifiant, DLSDU-length, DLSDU, Status }	READY
5	READY	SEND_ENABLE.ind { Speed-class } / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND (Speed-class) = "True" => DLCEP-identifiant:= GET_NEXT_ID (Speed-class) DL-Data-req ind { DLCEP-identifiant } DLSDU:= GET_BUFFER (DLCEP-identifiant) DLPDU:= BUILD_DT (Speed-class, DLCEP-identifiant, DLSDU) ACM_SEND_DATA.req { DLPDU }	READY
6	READY	SEND_ENABLE.ind { Speed-class } / Speed-class <> "SPORADIC" && CHECK_NEXT_SEND(Speed-class) = "False" => ACM_SEND_DATA.req { null } ----- no data remained to send out	READY
7	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class <> "SPORADIC" && CHECK_PAR_DT (DLPDU) = "True" => DLCEP-identifiant:= DLPDU.DLCEP-address DL-Buffer-received.ind { DLCEP-identifiant }	READY
8	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class <> "SPORADIC" && CHECK_PAR_DT (DLPDU) = "False" => none	READY
9	READY	ACM_SEND_DATA.conf { } / DLPDU <> "null" => none	READY
10	READY	ACM_SEND_DATA.conf { } / DLPDU = "null" => NEXT (Speed-class)	READY

7.2.3.4 Fonctions utilisées par la commande CTCR

Toutes les fonctions de la commande CTCR sont récapitulées dans le Tableau 63.

Tableau 63 – Table de fonctions CTCR

Nom de la fonction	Entrée	Sortie	Fonctionnement
CHECK_PAR_PUT	Identifiant DLCEP, DLSDU	Vrai/Faux	Vérifier la validité de tous les paramètres de la primitive DL-Put.req. Si la validité est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
PUT_BUFFER	Identifiant DLCEP, DLSDU	(aucun)	Archiver la DLSDU dans la mémoire tampon Send-Buffer associée à l'identifiant DLCEP
CHECK_PAR_GET	Identifiant DLCEP	Vrai/Faux	Vérifier la validité de l'identifiant DLCEP de la primitive DL-Get.req. Si la validité est confirmée, la valeur "Vrai" est renvoyée
GET_BUFFER	Identifiant DLCEP	DLSDU	Placer la DLSDU dans la mémoire tampon associée à l'identifiant DLCEP
CHECK_NEXT_SEND	Classe de vitesse	Vrai/Faux	Vérifier l'existence de la DLSDU spécifiée par la classe de vitesse. Si cette existence est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant

Nom de la fonction	Entrée	Sortie	Fonctionnement
			renvoyée dans le cas contraire
GET_NEXT_ID	Classe de vitesse	Identifiant DLCEP	Obtenir l'identifiant DLCEP suivant de la mémoire tampon de l'utilisateur DLS spécifiée par la classe de vitesse
BUILD_DT	Classe de vitesse, identifiant DLCEP, DLSDU	DLPDU	Intégrer dans une DLPDU issue des données contenues dans la mémoire tampon Send-Buffer. La DLPDU est assemblée comme suit: DLPDU.DA:= V(MGA) DLPDU.SA:= V(IA) DLPDU.Len/Type:= 0x888b DLPDU.FC:= Speed-class + "DT" DLPDU.SN:= V(TN) DLPDU.DLCEP.address:= DLCEP-identifiant DLPDU.DLSDU:= DLSDU
CHECK_PAR_DT	DLPDU	Vrai/Faux	Vérifier la validité de la DLPDU. Si la validité est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
NEXT	Classe de vitesse		Vérifier l'existence des DLSDU spécifiées par la classe de vitesse dans la mémoire tampon d'envoi (Send-buffer). Si cette existence est confirmée, placer l'identifiant DLCEP dans la mémoire tampon d'envoi, émettre la primitive DL_Data_Req.ind avec l'identifiant DLCEP obtenu pour l'utilisateur DLS, puis répéter la même séquence avec le nouvel identifiant DLCEP d'une DLSDU proche de la DLSDU désignée dans la séquence précédente. Le processus doit fonctionner selon des cycles applicables à toutes les DLSDU spécifiées par la classe de vitesse dans la mémoire tampon d'envoi

7.2.4 Commande TX/RX sporadique (STRC)

7.2.4.1 Présentation générale

La commande TX/RX sporadique (STRC) est chargée de la mise en mémoire tampon et de la répartition dans les délais de la DLSDU reçue pour le transfert de données de messagerie sporadiques entre l'utilisateur DLS et l'ACM.

7.2.4.2 Définitions de primitive

7.2.4.2.1 Primitives échangées entre l'utilisateur DLS et la commande STRC

Le Tableau 64 montre toutes les primitives échangées entre l'utilisateur DLS et la commande STRC et le Tableau 65 montre toutes les primitives échangées entre la commande STRC et l'ACM. Ces primitives sont identiques à celles applicables à l'architecture en étoile.

Tableau 64 – Primitives échangées entre l'utilisateur DLS et la commande STRC

Nom de la primitive	Source	Paramètres associés
DL-SPDATA.req	Utilisateur DLS	DA, MSDU
DL-SPDATA.conf	STRC	DA, Etat
DL-SPDATA.ind	STRC	DA, SA, MSDU, Rec-status

Tableau 65 – Primitives échangées entre la commande STRC et l'ACM

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_RECV_DATA.ind	ACM	DLPDU

7.2.4.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC

Tous les paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC sont présentés dans le Tableau 66. Ces paramètres sont identiques à ceux applicables à l'architecture en étoile.

Tableau 66 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DLS et la commande STRC

Nom du paramètre	Description
DA	Adresse de destination
SA	Adresse source
MSDU	Unité de données de service MAC
Etat	Indique si le service demandé de DL-SPDATA.req est fourni avec succès ou s'il est défaillant pour le motif spécifié
Rec-Status	Indique si la réception de la DLPDU s'est effectuée sans erreur ou non
DLPDU	DLPDU pour transfert de données de messagerie sporadiques
Classe de vitesse	Classe de vitesse demandée par l'utilisateur DLS. Dans le service de transfert de données de messagerie sporadiques, la classe de vitesse est spécifiée comme "sporadique" ou "classe 1" pour le fonctionnement à 100 Mbps et comme "sporadique" ou "classe 0" pour le fonctionnement à 1 000 Mbps

7.2.4.3 Table d'états STRC

Le diagramme de transition d'état de la commande STRC est illustré à la Figure 29, et la table d'états STRC est présentée dans le Tableau 67.

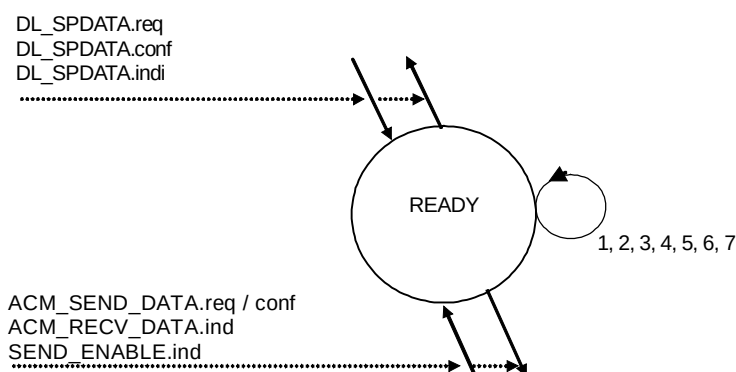


Figure 29 – Diagramme de transition d'état de la commande STRC

Tableau 67 – Table d'états STRC

#	État courant	Événement /Condition => action	État suivant
1	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () <> "Full" => QUEUE_SPDATA (DA, MSDU) DL-SPDATA.confirm { DA, MSDU, Status:= "success" }	READY
2	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "True" && CHECK_SPDATAQ () = "Full" => Status:= "Failure – The SPDATA-queue is full" DL-SPDATA.conf { DA, MSDU, Status }	READY
3	READY	DL-SPDATA.req { DA, MSDU } / CHECK_PAR_SPDATA (DA, MSDU) = "False" => Status:= "Failure – Invalid parameter" DL-SPDATA.confirm { DA, MSDU, Status }	READY
4	READY	SEND_ENABLE.ind { Speed-class } / Speed-class = "SPORADIC" && CHECK_SPDATAQ () <> "Empty" => MSDU:= DEQUEUE_SPDATA () DLPDU:= BUILD_SPDATA (DA, MSDU) ACM_SEND_DATA.req { DLPDU }	READY
5	READY	ACM_SEND_DATA.conf { } / => none	READY
6	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class = "SPORADIC" && CHECK_RECV_SPDATA (DLPDU) = "True" => DL-SPDATA.ind { DLPDU }	READY
7	READY	ACM_RECV_DATA.ind { DLPDU } / Speed-class = "SPORADIC" && CHECK_RECV_SPDATA (DLPDU) = "False" => none	READY

7.2.4.4 Fonctions utilisées par la commande STRC

Toutes les fonctions de la commande STRC sont récapitulées dans le Tableau 68.

Tableau 68 – Table de fonctions STRC

Nom de la fonction	Entrée	Sortie	Fonctionnement
CHECK_PAR_SPDATA	DA, MSDU	Vrai/Faux	Vérifier la validité de tous les paramètres de la primitive DL-SPDATA.req. Si la validité est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
CHECK_SPDATAQ	(aucun)	Etat	Vérifier que la condition File d'attente pour SPDATA est File d'attente maximale. L'état renvoyé est "Complet", "Vide" ou "En file d'attente"
QUEUE_SPDATA	DA, MSDU	(aucun)	Place les données d'entrée dans la file d'attente SPDATA sur une base FIFO
DEQUEUE_SPDATA	(aucun)	DA, MSDU	Retirer de la file d'attente SPDATA sur une base FIFO
BUILD_SPDATA	DA, MSDU	DLPDU	Intégrer dans une DLPDU de données de messagerie sporadiques. La DLPDU est assemblée comme suit: DLPDU.DA:= MSDU.DA DLPDU.SA:= V(IA) DLPDU.Len/Type:= MSDU.LEN/Type DLPDU.MSDU:= MSDU.DLSDU
CHECK_RCV_SPDATA	DLPDU	Vrai/Faux	Vérifier la validité de la DLPDU spécifiée. Si la validité est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire

7.2.5 Machine de contrôle d'accès (ACM)

7.2.5.1 Présentation générale

La machine de contrôle d'accès (ACM) est chargée du contrôle d'accès au support déterministe et des possibilités de planification de l'envoi des DLPDU, du contrôle d'ajout et de retrait de nœuds, et de la restauration du réseau suite à une interruption ayant entraîné un arrêt du système.

L'ACM est chargée principalement

- de s'assurer que le nœud local détecte et utilise intégralement sa période d'accès attribuée;
- de s'assurer que le nœud local ne gêne pas les transmissions des autres nœuds, particulièrement le nœud qui transmet la DLPDU SYN;
- de détecter toute interruption du réseau et de lancer la transmission de la DLPDU SYN pour la restauration du réseau après la durée prescrite au cours de laquelle la DLPDU SYN n'est pas entendue;
- d'assurer l'ajout et le retrait d'un nouveau nœud au/du réseau.

7.2.5.2 ACM et fonctions de support de planification

Les fonctions ACM planifient toutes les communications entre les DLE participantes du bus de terrain de Type 11, la temporisation de ces communications étant contrôlée pour:

- satisfaire au contrôle d'accès au support spécifique afin de donner la possibilité à toutes les DLE d'envoyer deux sortes de classe de données cycliques prioritaires et de données de messagerie sporadiques de manière opportune, hiérarchisée et déterministe, de détecter toute interruption du réseau et de lancer la restauration en temps voulu, puis d'ajouter et de retirer des nœuds de la ligne;

- b) offrir plusieurs niveaux de possibilités de transfert de données prioritaires afin d'envoyer des données au nœud dans l'ordre séquentiel et dans chaque période prédéfinie, de manière à procéder au transfert de données de chaque niveau dans la durée préalablement spécifiée (temps de maintien du jeton) et de transférer ou conserver les données des niveaux inférieurs sur une période de temps cyclique ultérieure selon le niveau et l'occasion du niveau supérieur du transfert de données est toujours réalisée à chaque occasion. Par ailleurs, l'ensemble du volume du transfert de données des niveaux inférieurs est transféré dans chaque période préalablement spécifiée;
- c) offrir des possibilités de transfert de données de messagerie sporadiques au nœud indiquant que la demande de transfert se déroule de manière sporadique par l'utilisateur DLS et que les données sont transférées dans la période préalablement spécifiée du niveau de priorité correspondant et selon les applications ISO/CEI 8802-3 normales.

7.2.5.3 Définitions de primitive

7.2.5.3.1 Primitives échangées entre l'ACM et la commande RMC

Le Tableau 69 énumère toutes les primitives et le Tableau 70 présente les paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC.

Tableau 69 – Primitives échangées entre l'ACM et la commande RMC

Nom de la primitive	Source	Paramètres associés
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU
RMC_SEND_CF.req	ACM	RMSDU
RMC_SEND_CF.conf	RMC	RMC-status
RMC_SEND_CF.ind	RMC	RMC-status
RMC_SEND_CF_ALL.req	ACM	RMSDU
RMC_SEND_CF_ALL.conf	RMC	RMC-status

Tableau 70 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande RMC

Nom du paramètre	Description
RMSDU	SDU de RMC
RMC-status	Etat indiquant les résultats de la demande à la commande RMC

7.2.5.3.2 Primitives échangées entre l'ACM et la commande CTRC

Le Tableau 71 énumère toutes les primitives et le Tableau 72 présente les paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTRC.

Tableau 71 – Primitives échangées entre l'ACM et la commande CTRC

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	CTRC	DLPDU
ACM_SEND_DATA.conf	ACM	ACM-status
ACM_RECV_DATA.ind	ACM	DLPDU

Tableau 72 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande CTRC

Nom du paramètre	Description
ACM-status	Etat indiquant les résultats de la demande à la commande ACM
DLPDU	SDU de AMC
Classe de vitesse	Classe de vitesse indiquant la transmission de données cycliques prioritaires

7.2.5.3.3 Primitives échangées entre l'ACM et la commande STRC

Le Tableau 73 énumère toutes les primitives et le Tableau 74 présente les paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC.

Tableau 73 – Primitives échangées entre l'ACM et la commande STRC

Nom de la primitive	Source	Paramètres associés
SEND_ENABLE.ind	ACM	Classe de vitesse
ACM_SEND_DATA.req	STRC	DLPDU
ACM_SEND_DATA.conf	ACM	Etat
ACM_SEND_DATA.ind	ACM	DLPDU

Tableau 74 – Paramètres utilisés avec les primitives échangées entre l'ACM et la commande STRC

Nom du paramètre	Description
ACM-Status	Etat indiquant les résultats de la demande à la commande ACM
DLPDU	SDU de ACM
Classe de vitesse	Classe de vitesse indiquant la transmission de données cycliques prioritaires

7.2.5.4 Table d'états ACM

Les diagrammes de transition d'état pour les fonctionnements à 100 Mbps et 1 000 Mbps sont présentés à la Figure 30 et à la Figure 31 respectivement.

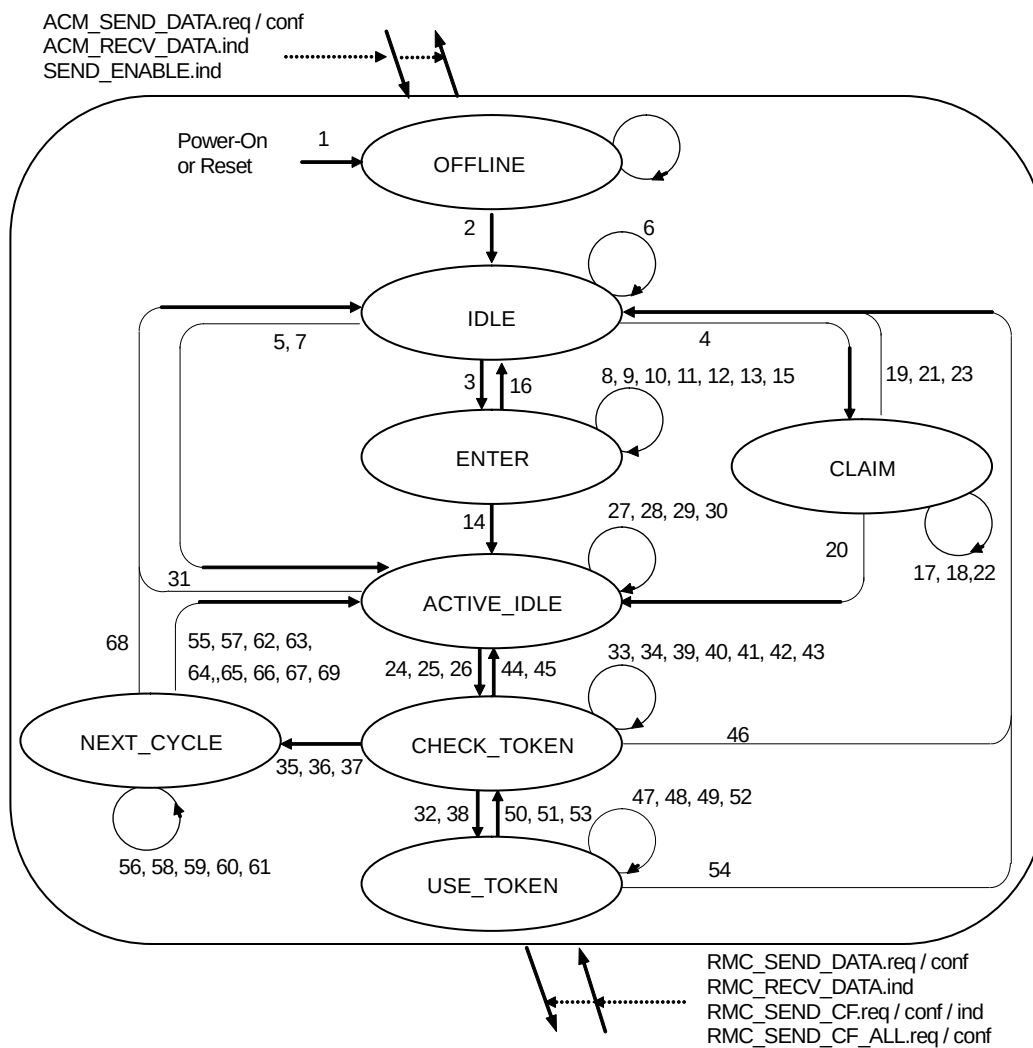


Figure 30 – Diagramme de transition d'état de la commande ACM pour le fonctionnement à 100 Mbps

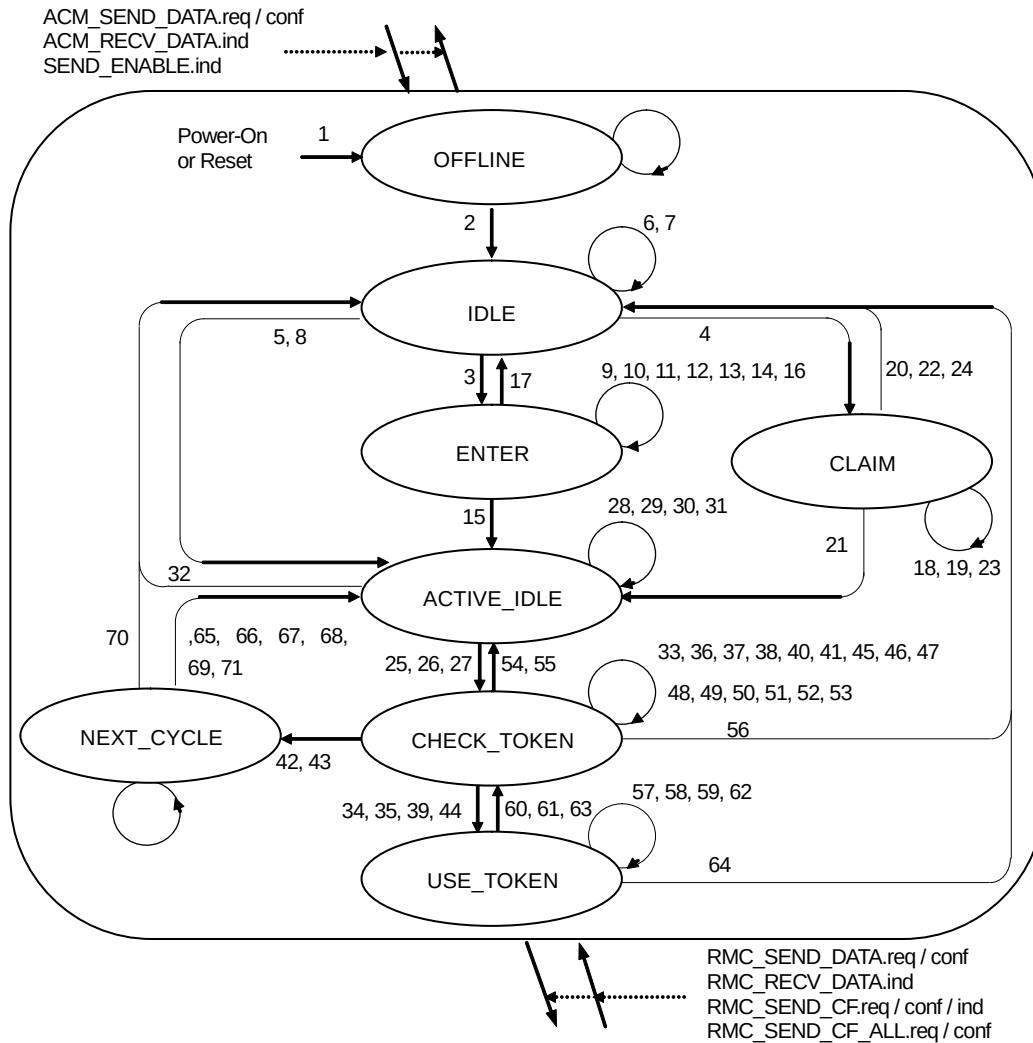


Figure 31 – Diagramme de transition d'état de la commande ACM pour le fonctionnement à 1 000 Mbps

L'état "OFFLINE" est activé lors de la mise sous tension ou lors de l'émission de la primitive DLM-Reset.request. Dans cet état, l'ensemble des variables, paramètres et informations de configuration sont définis par la DLE. Lorsque la variable "In_Ring_Desire" devient vraie, l'état passe à "IDLE".

Cet état consiste à attendre pour être prêt à rejoindre les membres du bus de terrain de Type 11. Lorsqu'un autre nœud SYN a transmis une DLPDU SYN, l'état passe à l'état "ENTER" dans lequel est transmise la DLPDU REQ revendiquant l'ajout du nœud SYN actuel aux membres du bus de terrain de Type 11. De plus, à l'état "IDLE", lorsque les signaux émis sur le support de transmission demeurent inactifs pendant une période spécifiée, l'état devient "CLAIM", dans lequel le nœud tente d'être un nouveau nœud SYN.

L'état "ACTIVE_IDLE" fait partie des membres du bus de terrain de Type 11, le nœud parvenant à obtenir le droit de transmission par la commande d'accès au support du bus de terrain de Type 11, soit lorsque le nœud fonctionne comme mode SYN par transmission de la DLPDU SYN, ou comme mode non SYN de réception de cette même DLPDU par un autre nœud SYN, l'état étant "CHECK_TOKEN".

L'état "CHECK_TOKEN" attend jusqu'à ce que la valeur $V(LN)$ soit égale à $V(TN)$ afin d'envoyer les données sur le support de transmission. Lorsque $V(LN)$ est égale à $V(TN)$, l'état passe à l'état "USE_TOKEN".

L'état "USE_TOKEN" correspond à l'état dans lequel le nœud est capable d'envoyer les données d'émission cyclique prioritaire et les données de transmission sporadiques. Lorsque toutes les données ont été transmises ou à l'expiration du temps de maintien du jeton, l'état passe à l'état "NEXT_CYCLE".

Cet état "NEXT_CYCLE" consiste à solliciter un nouveau nœud comme fonction principale.

Les tables d'état ACM pour les fonctionnements à 100 Mbps et 1 000 Mbps sont présentées au Tableau 75 et au Tableau 76 respectivement.

Tableau 75 – Table d'états ACM pour le fonctionnement à 100 Mbps

#	État courant	Événement /Condition => action	État suivant
1	Any states	/ POWER-ON or RESET =>	OFFLINE
2	OFFLINE	/ In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) = 0 => -- receiving SYN and LL is not alive -- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	/ EXPIRED_TIMER (T(SL)) = "True" && V(LL) = 0 => ----sending out CLM All_Slot_Time:= V(ST) RMSDU:= BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "True" => -- receiving SYN and this node is alive -- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
6	IDLE	RMC_SEND_CF.ind { } / V(LL) <> 0 && V(LL)<V(TN)> = "True" && V(LN) = V(TN) => -----activate sending out LRR RMSDU:= BUILD_PDU (LRR) RMS_SEND_CF.req { RMSDU } START_TIMER (T(SL), V(SL))	IDLE
7	IDLE	RMS_SEND_CF.conf { } / RMSDU.FC = LRR => none	ACTIVE_IDLE
8	ENTER	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	ENTER
9	ENTER	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER

#	État courant	Événement /Condition => action	État suivant
10	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) = V(MN) && V(LL)<V(TN)> = "False" && SYN_frame.PN = V(TN) => RMSDU:= BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
11	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++	ENTER
12	ENTER	/ V(LN) = V(MN) && V(LL)<V(TN)> = "False" && SYN_frame.PN = V(TN) => RMSDU:= BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
13	ENTER	RMC_SEND_DATA.conf { } => none	ENTER
14	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V (LL)<V(TN)> = "True" => SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
15	ENTER	/ EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
16	ENTER	/ EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
17	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time > 0 => All_Slot_Time-- START_TIMER (T(SL), V(SL)) START_TIMER (T(IGP), V(IGP))	CLAIM
18	CLAIM	/ EXPIRED_TIMER (T(IGP))="True" => RMSDU:=BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSD } START_TIMER (T(SL), V(SL))	CLAIM
19	CLAIM	RMC_SEND_DATA.ind {RMSDU} / RMSDU.FC = (SYN CLM) => START_TIMER (T(SL), V(SL)) STOP_TIMER (T(IGP))	IDLE
20	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = 0 => All_Slot_Time:= V(ST) V(LL) = 0 V(LL)<V(TN)>:= 1 RMSDU:= BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE

#	État courant	Événement /Condition => action	État suivant
21	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => All_Slot_Time:= V(ST) START_TIMER (T(SL), V(SL))	IDLE
22	CLAIM	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> (SYN CLM) => -- receiving a frame except SYN, CLM -- START_TIMER (T(SL), V(SL))	CLAIM
23	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- V(LL):= 0 START_TIMER (T(SL), V(SL))	IDLE
24	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "False" => --- receiving SYN in non-SYN_mode---- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1	CHECK_TOKEN
25	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "True" => ---Receiving SYN in SYN_mode----- SYN_Frame:= RMSDU if SYN_frame.SN < V(TN) then SYN_mode:= 0 V(LL):= SYN_frame.LL V(LN):= 1 endif	CHECK_TOKEN
26	ACTIVE_IDLE	RMC_SEND_DATA.conf { } / RMSDU.FC = SYN && CHECK_COL () = "False" => -- sent SYN ---- SYN_mode:= 1 V(LN):= 1	CHECK_TOKEN
27	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
28	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
29	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (DT DT-CMP) => -- receiving DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
30	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
31	ACTIVE_IDLE	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE

#	État courant	Événement /Condition => action	État suivant
32	CHECK_TOKEN	/ V(LL)<V(TN)> = 1 && V(LN) = V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) Speed-class:= 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
33	CHECK_TOKEN	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	CHECK_TOKEN
34	CHECK_TOKEN	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	CHECK_TOKEN
35	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN <> V(TN) && SYN_mode = "True" => ---- in SYN-mode, on MAC-control-time-period -- START_TIMER (T(TMAC), V(TMAC))	NEXT_CYCLE
36	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN = 255 && V(LL)<V(TN)> = "True" => RMSDU:= BUILD_PDU (LPD) -----send LPD RMC_SEND_CF_ALL.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL))	NEXT_CYCLE
37	CHECK_TOKEN	/ V(LN) = V(MN) && SYN_frame.PN = V(TN) && V(LL)<V(TN)> = "True" => RMSDU:= BUILD_PDU (RAS) -----send RAS RMC_SEND_DATA.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL))	NEXT_CYCLE
38	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) => -- Get token -- ----- matching V(LN), on receiving CMP or DT-CMP STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN>:= 0 Speed-class:= 3 -- Highest Priority START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif	USE_TOKEN
39	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ <> V(TN) =>--not matching V(LN).on receiving CMP or DT-CMP START_TIMER (T(SCMP), V(SCMP)) C(ASCMP)<RMSDU.SN>:=0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif	CHECK_TOKEN
40	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT => ---- receiving DT ---- C(ASCMP)<RMSDU.SN>:= 0 ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN

#	État courant	Événement /Condition => action	État suivant
41	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
42	CHECK_TOKEN	/ EXPIRED_TIMER (T(SCMP)) => C(ASCMP)<V(LN)>++ V(ANONC)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif V(LN)++	CHECK_TOKEN
43	CHECK_TOKEN	RMC_SEND_DATA.conf { } / RMSDU.FC = (CMP DT-CMP) => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
44	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
45	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
46	CHECK_TOKEN	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
47	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send out DT or Sporadic frame -- RMSDU:= BUILD_PDU (DLPDU) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	USE_TOKEN
48	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
49	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> 1 => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class:= Speed-class – 1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	État courant	Événement /Condition => action	État suivant
50	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class = 1 => ---Expired THT and Speed-class =1 ---- ACM_SEND_DATA.conf { RMC-status } RMSDU:= BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
51	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / RMSDU.FC = DT-CMP => ACM_SEND_DATA.conf { RMC-status } STOP_TIMER (T(ATHT)) Speed-class:= Speed-class – 1 if Speed-class = 2 then START_TTRT (Speed-class) START_TTRT (Speed-class –1) else if Speed-class = 1 then START_TTRT (Speed-class) endif	CHECK_TOKEN
52	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class <> 1 => -- Next Speed-class -- Speed-class:= Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
53	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class = 1 => -- All sent -- RMSDU:= BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
54	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
55	NEXT_CYCLE	RMC_SEND_CF_ALL.conf { } / RMSDU.FC = LPD => none	ACTIVE_IDLE
56	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && SYN_frame.PN = V(TN) && SYN_mode = "True" => START_TIMER (T(IGP), V(IGP))	NEXT_CYCLE
57	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && SYN_frame.PN = V(TN) && SYN_mode = "False" => if ACM_SEND_DATA.req { COM } then START_TIMER (T(IGP), V(IGP)) else none endif	ACTIVE_IDLE
58	NEXT_CYCLE	/EXPIRED_TIMER (T(IGP)) = "True" && SYN_mode = "False" => RMSDU:= BUILD_PDU (COM) ----send COM RMC_SEND_DATA.req { RMSDU }	NEXT_CYCLE

#	État courant	Événement /Condition => action	État suivant
59	NEXT_CYCLE	RMC_SEND_DATA.conf { } / RMSDU.FC = COM => none	NEXT_CYCLE
60	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "True" => -- receiving REQ at SYN-node-- V(LL)<RMSDU.SN>:= 1 C(NONC):= 0 RMSDU:= BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
61	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "False" => --receiving REQ at Non-SYN-node-- none	ACTIVE_IDLE
62	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = COM && SYN_mode = "True" => -- receiving COM -- V(CW):= RMSDU.CW V(ST):= RMSDU.ST V(TSYN):= RMSDU.Th V(TTRT2):= RMSDU.Tm DLM_Set_value.req { Variable-name, Desired-value } C(NONC):= 0 RMSDU:= BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
63	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
64	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
65	NEXT_CYCLE	/ EXIPERED_TIMER (T(TMAC)) = "True" && SYN.mode = "True" && V(ANONC) < (V(MN) – 1) => C(NONC):= 0 V(ANONC):= 0 RMSDU:= BUILD_PDU (SYN) ----send SYN RMS_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
66	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "True" && (V(ANONC) >= (V(MN) – 1)) && (C(NONC)++ >= V(NONC)) => C(NONC):= 0 V(ANONC):= 0 SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
67	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "False" => none	ACTIVE_IDLE

Tableau 76 – Table d'états ACM pour le fonctionnement à 1 000 Mbps

#	État courant	Événement /Condition => action	État suivant
1	Any states	/ POWER-ON or RESET =>	OFFLINE
2	OFFLINE	/ In_Ring_Desired = "True" => START_TIMER (T(SL), V(SL))	IDLE
3	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) = 0 => -- receiving SYN and LL is not alive -- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ENTER
4	IDLE	/ EXPIRED_TIMER (T(SL)) = "True" && V(LL) = 0 => ----sending out CLM All_Slot_Time:= V(ST) if V(ESYN) = "True" then RMSDU:= BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif	CLAIM
5	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "True" => -- receiving SYN and this node is alive -- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
6	IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V(LL) <> 0 && V(LL)<V(TN)> = "False" => -- receiving SYN and this node is not alive -- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	IDLE
7	IDLE	RMC_SEND_CF.ind { } / V(LL) <> 0 && V(LL)<V(TN)> = "True" && V(LN) = V(TN) => -----activate sending out LRR RMSDU:= BUILD_PDU (LRR) RMS_SEND_CF.req { RMSDU } START_TIMER (T(SL), V(SL))	IDLE
8	IDLE	RMS_SEND_CF.conf { } / RMSDU.FC = LRR => none	ACTIVE_IDLE
9	ENTER	/ V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP))	ENTER
10	ENTER	/ V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++	ENTER

#	État courant	Événement /Condition => action	État suivant
11	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) = V(MN) && V(LL)<V(TN)> = "False" => RMSDU:= BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
12	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN) <> V(MN) => -- Live Node less than Maximum Node No. -- V(LN)++	ENTER
13	ENTER	/ V(LN) = V(MN) && V(LL)<V(TN)> = "False" => RMSDU:= BUILD_PDU (REQ) -----send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ENTER
14	ENTER	RMC_SEND_DATA.conf { } => none	ENTER
15	ENTER	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && V (LL)<V(TN)> = "True" => SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
16	ENTER	/ EXPIRED_TIMER (T(SCMP)) = "True" => V(LN)++	ENTER
17	ENTER	/ EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- START_TIMER (T(SL), V(SL))	IDLE
18	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time > 0 => All_Slot_Time-- START_TIMER (T(SL), V(SL)) START_TIMER (T(IGP), V(IGP))	CLAIM
19	CLAIM	/ EXPIRED_TIMER (T(IGP))="True" => if V(ESYN) = "True" then RMSDU:=BUILD_PDU (CLM) RMC_SEND_DATA.req { RMSD } START_TIMER (T(SL), V(SL)) endif	CLAIM
20	CLAIM	RMC_SEND_DATA.ind {RMSDU} / RMSDU.FC = (SYN CLM) => START_TIMER (T(SL), V(SL)) STOP_TIMER (T(IGP))	IDLE
21	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "False" && All_Slot_Time = 0 => All_Slot_Time:= V(ST) V(LL) = 0 V(LL)<V(TN)>:= 1 RMSDU:= BUILD_PDU (SYN) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE

#	État courant	Événement /Condition => action	État suivant
22	CLAIM	RMC_SEND_DATA.conf { } / CHECK_COL () = "True" => All_Slot_Time:= V(ST) START_TIMER (T(SL), V(SL))	IDLE
23	CLAIM	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> (SYN CLM) => -- receiving a frame except SYN, CLM -- START_TIMER (T(SL), V(SL))	CLAIM
24	CLAIM	EXPIRED_TIMER (T(SL)) = "True" => -- Expired Silence Timer -- V(LL):= 0 START_TIMER (T(SL), V(SL))	IDLE
25	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "False" => --- receiving SYN in non-SYN_mode---- SYN_frame:= RMSDU V(LL):= SYN_frame.LL V(LN):= 1	CHECK_TOKEN
26	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = SYN && SYN_mode = "True" => ---Receiving SYN in SYN_mode----- SYN_Frame:= RMSDU if SYN_frame.SN < V(TN) then SYN_mode:= 0 V(LL):= SYN_frame.LL V(LN):= 1 endif	CHECK_TOKEN
27	ACTIVE_IDLE	RMC_SEND_DATA.conf { } / RMSDU.FC = SYN && CHECK_COL () = "False" => -- sent SYN ---- SYN_mode:= 1 V(LN):= 1	CHECK_TOKEN
28	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
29	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
30	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (DT DT-CMP) => -- receiving DT frame-- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
31	ACTIVE_IDLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
32	ACTIVE_IDLE	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE

#	État courant	Événement /Condition => action	État suivant
33	CHECK_TOKEN	<pre> / V(LL)<V(TN)> = 1 && V(LN) = V(TN) SYN_frame.PN <> V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) if CHECK_SDU_BF (FC:= REQ) = "False" then Speed-class:= 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) endif </pre>	USE_TOKEN
34	CHECK_TOKEN	<pre> / V(LL)<V(TN)> = 1 && V(LN) = V(TN) SYN_frame.PN <> V(TN) => ----- Get token ----- if CHECK_SDU_BF (FC:= REQ) = "True" then RMSDU:= GET_SDU_BF (FC:= REQ) --send REQ RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
35	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = REQ && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN <> V(TN) => Speed-class:= 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) </pre>	USE_TOKEN
36	CHECK_TOKEN	<pre> / V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => ----- Get token ----- STOP_TIMER (T(SCMP)) if CHECK_SDU_BF (FC:= REQ) = "True" then RMSDU:= GET_SDU_BF (FC:= REQ) RMC_SEND_DATA.req { RMSDU } --- send REQ START_TIMER (T(SL), V(SL)) els if CHECK_SDU_BF (FC:= COM) = "True" then RMSDU:= GET_SDU_BF (FC:= COM) RMC_SEND_DATA.req { RMSDU } ---send COM START_TIMER (T(SL), V(SL)) else RMSDU:= BUILD_PDU (RAS) RMC_SEND_DATA.req { RMSDU } ---send RAS START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
37	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = REQ && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => if CHECK_SDU_BF (FC:= COM) = "True" then RMSDU:= GET_SDU_BF (FC:= COM) RMC_SEND_DATA.req { RMSDU } ---send COM START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN

#	État courant	Événement /Condition => action	État suivant
38	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = COM && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => RMSDU:= BUILD_PDU (RAS) RMC_SEND_DATA.req { RMSDU } ---send RAS START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) </pre>	CHECK_TOKEN
39	CHECK_TOKEN	<pre> RMC_SEND_DATA.conf { } / RMSDU.FC = RAS && V(LL)<V(TN)> = 1 && V(LN) = V(TN) && SYN_frame.PN = V(TN) => Speed-class:= 3 -----Highest-priority-class START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) </pre>	USE_TOKEN
40	CHECK_TOKEN	<pre> / V(LL)<V(LN)> = 1 && V(LN) <> V(TN) => START_TIMER (T(SCMP), V(SCMP)) </pre>	CHECK_TOKEN
41	CHECK_TOKEN	<pre> / V(LL)<V(LN)> = 0 && V(LN) <> V(MN) => V(LN)++ </pre>	CHECK_TOKEN
42	CHECK_TOKEN	<pre> / V(LN) = V(MN) && V(LL)<V(TN)> = "False" => ---- MAC-control-time-period -- START_TIMER (T(TMAC), V(TMAC)) </pre>	NEXT_CYCLE
43	CHECK_TOKEN	<pre> / V(LN) = V(MN) && V(LL)<V(TN)> = "True" => RMSDU:= BUILD_PDU (LPD) -----send LPD RMC_SEND_CF_ALL.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) </pre>	NEXT_CYCLE
44	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN <> V(TN) => -- Get token -- ----- matching V(LN), on receiving CMP or DT-CMP STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN>:= 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC:= REQ) = "False" then Speed-class:= 3 -- Highest Priority START_THT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL)) endif </pre>	USE_TOKEN

#	État courant	Événement /Condition => action	État suivant
45	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN <> V(TN) => -- Get token -- STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN>:= 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC:= REQ) = "True" then RMSDU:= GET_SDU_BF (FC:= REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
46	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ = V(TN) && SYN_frame.PN = V(TN) => -- Get token -- STOP_TIMER (T(SCMP)) C(ASCMP)<RMSDU.SN>:= 0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif if CHECK_SDU_BF (FC:= REQ) = "True" then RMSDU:= GET_SDU_BF (FC:= REQ) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) else if CHECK_SDU_BF (FC:= COM) = "True" then RMSDU:= GET_SDU_BF (FC:= COM) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL)) else RMSDU:= BUILD_PDU (RAS) -----send RAS RMC_SEND_DATA.req { RMSDU } START_TIMER (T(TMAC), V(TMAC)) START_TIMER (T(SL), V(SL)) endif </pre>	CHECK_TOKEN
47	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = (CMP DT-CMP) && V(LN)++ <> V(TN) =>--not matching V(LN).on receiving CMP or DT-CMP START_TIMER (T(SCMP), V(SCMP)) C(ASCMP)<RMSDU.SN>:=0 if RMSDU.FC = DT-CMP then ACM_RECV_DATA.ind { RMSDU } endif </pre>	CHECK_TOKEN
48	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = DT => ---- receiving DT ---- C(ASCMP)<RMSDU.SN>:= 0 ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL)) </pre>	CHECK_TOKEN
49	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC <> Type-11 fieldbus FRAME => -- Receiving SPORADIC Frame -- ACM_RECV_DATA.ind { RMSDU } START_TIMER (T(SL), V(SL)) </pre>	CHECK_TOKEN
50	CHECK_TOKEN	<pre> RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "True" => -- receiving REQ at SYN-node-- V(LL)<RMSDU.SN>:= 1 C(NONC):= 0 </pre>	CHECK_TOKEN

#	État courant	Événement /Condition => action	État suivant
51	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = COM && SYN_mode = "True" => -- receiving COM -- V(CW):= RMSDU.CW V(ST):= RMSDU.ST V(TSYN):= RMSDU.Th V(TTTRT2):= RMSDU.Tm DLM_Set_value.req { Variable-name, Desired-value } C(NONC):= 0	CHECK_TOKEN
52	CHECK_TOKEN	/ EXPIRED_TIMER (T(SCMP)) => C(ASCMP)<V(LN)>++ V(ANONC)++ if SYN_mode = "True" then MAINT_LL(V(LN)) endif V(LN)++	CHECK_TOKEN
53	CHECK_TOKEN	RMC_SEND_DATA.conf { } / RMSDU.FC = (CMP DT-CMP) => -- sending CMP frame-- V(LN)++	CHECK_TOKEN
54	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
55	CHECK_TOKEN	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE
56	CHECK_TOKEN	/ EXPIRED_TIMER (T(SL)) = "True" => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
57	USE_TOKEN	ACM_SEND_DATA.req { DLPDU } / => -- send out DT or Sporadic frame -- RMSDU:= BUILD_PDU (DLPDU) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	USE_TOKEN
58	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "False" => ACM_SEND_DATA.conf { RMC-status } SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
59	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class <> "Sporadic" => -- Expired THT, Next Speed class -- ACM_SEND_DATA.conf { RMC-status } Speed-class:= Speed-class – 1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN

#	État courant	Événement /Condition => action	État suivant
60	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / (RMSDU.FC = DT RMSDU.FC <> Type 11 fieldbus) && EXPIRED_TIMER (T(ATHT)) = "True" && Speed-class = "Sporadic" => ---Expired THT and Speed-class="Sporadic"----- ACM_SEND_DATA.conf { RMC-status } RMSDU:= BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
61	USE_TOKEN	RMC_SEND_DATA.conf { RMC-status } / RMSDU.FC = DT-CMP => ACM_SEND_DATA.conf { RMC-status } STOP_TIMER (T(ATHT)) Speed-class:= Speed-class – 1 if Speed-class = 2 then START_TTRT (Speed-class) START_TTRT (Speed-class –1) else if Speed-class = 1 then START_TTRT (Speed-class) endif	CHECK_TOKEN
62	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class <> "Sporadic" => -- Next Speed-class -- Speed-class:= Speed-class -1 START_THT (Speed-class) START_TTRT (Speed-class) SEND_ENABLE.ind { Speed-class } START_TIMER (T(SL), V(SL))	USE_TOKEN
63	USE_TOKEN	ACM_SEND_DATA.req { null } / Speed-class = "Sporadic" => -- All sent -- RMSDU:= BUILD_PDU (CMP) RMC_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	CHECK_TOKEN
64	USE_TOKEN	EXPIRED_TIMER (T(SL)) = "True" / => -- expired Silence Timer - SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
65	NEXT_CYCLE	RMC_SEND_CF_ALL.conf { } / RMSDU.FC = LPD => none	ACTIVE_IDLE
66	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = REQ && SYN_mode = "False" => --receiving REQ at end node of active-segment-- PUT_SDU_BF (RMSDU.FC, RMSDU)	ACTIVE_IDLE
67	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "True" => SYN_mode:= 0 START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
68	NEXT_CYCLE	RMC_RECV_DATA.ind { RMSDU } / RMSDU.FC = CLM && SYN_mode = "False" => none	ACTIVE_IDLE

#	État courant	Événement /Condition => action	État suivant
69	NEXT_CYCLE	/ EXIPERED_TIMER (T(TMAC)) = "True" && SYN.mode = "True" && V(ANONC) < (V(MN) – 1) => C(NONC):= 0 V(ANONC):= 0 RMSDU:= BUILD_PDU (SYN) ----send SYN RMS_SEND_DATA.req { RMSDU } START_TIMER (T(SL), V(SL))	ACTIVE_IDLE
70	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "True" && (V(ANONC) >= (V(MN) – 1)) && (C(NONC)++ >= V(NONC)) => C(NONC):= 0 V(ANONC):= 0 SYN_mode:= "False" START_TIMER (T(SL), V(SL))	IDLE
71	NEXT_CYCLE	/ EXPIRED_TIMER (T(TMAC)) = "True" && SYN_mode = "False" => none	ACTIVE_IDLE

7.2.5.5 Fonctions utilisées par l'ACM

Toutes les fonctions utilisées par l'ACM sont récapitulées dans le Tableau 77.

Tableau 77 – Table de fonctions ACM

Nom de la fonction	Entrée	Sortie	Fonctionnement
BUILD_PDU	FC	RMSDU	Conformément au FC (Contrôle de trames) demandé, assembler la trame de bus de terrain de Type 11
EXPIRED_TIMER	T(x)	Vrai/Faux	Lorsque le temporisateur demandé x a expiré, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
MAINT_LL	V(LN)	(aucun)	L'état de fonctionnement en cours du nœud correspondant spécifié V(LN) est issu de la valeur V(LL) à l'expiration de T(SCMP). A l'expiration de T(SCMP), à tous les moments de prolongation de C(ASCMP) dans la période précédente, le bit correspondant de V(LL) est supprimé
START_TIMER	T(x), V(y)	(aucun)	Le temporisateur x est défini par la valeur de y, puis est activé
START_THT	Classe de vitesse	(aucun)	T(ATHT) est activée selon la valeur restante V(TTRT) de la classe d'accès spécifiée par la classe de vitesse
START_TTRT	Classe de vitesse	(aucun)	T(ATTRT) est activée selon la valeur V(TTRT) de la classe d'accès spécifiée par la classe de vitesse
STOP_TIMER	T(x)	(aucun)	Le temporisateur x est désactivé
CHECK_COL ()	(aucun)	Vrai/Faux	En cas de collision, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
PUT_SDU_BF	FC, RMSDU	(aucun)	Archiver RMSDU dans la mémoire tampon RMSDU-Buffer associée au FC (Contrôle de trame)
GET_SDU_BF	FC	RMSDU	Obtenir RMSDU dans la mémoire tampon RMSDU-Buffer associée au FC (Contrôle de trame)
CHECK_SDU_BF	FC	Vrai/Faux	Vérifier l'existence de la RMSDU spécifiée par le FC (Contrôle de trame) dans la mémoire tampon RMSDU-Buffer. Si cette existence est confirmée, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire

7.2.6 Commande de support de redondance (RMC)

7.2.6.1 Présentation générale

La commande de support de redondance (RMC) assure la fonction de redondance de support déterministe dans le temps du bus de terrain de Type 11 en forme d'anneau, en collaboration avec l'ACM, en faisant appel à la fonction du convertisseur parallèle-série / série-parallèle constitué du verrouilleur de trames TX, du verrouilleur de trames RX A/B, du convertisseur parallèle-série d'octets A/B et du convertisseur série-parallèle d'octets A/B, ainsi que des services fournis par la couche physique. La fonction de RMC est la suivante:

- La commande RMC gère la connexion entre les nœuds voisins reliés en série à deux autres nœuds dans l'anneau, et les données transmises simultanément dans les deux directions de l'anneau sont gérées de sorte qu'elles circulent entre les nœuds de manière séquentielle;
- La commande RMC génère et gère délibérément l'emplacement des points ouverts afin de se soustraire aux données qui circulent en permanence dans l'anneau, et ce, de sorte que les données issues d'un nœud ne soient plus renvoyées au nœud source le long du chemin circulaire;
- La commande RMC gère l'emplacement des points ouverts de répartition de l'anneau en deux parties des segments actif et ouvert, un nœud placé sur le segment actif étant géré de manière à être un nœud membre qui participe et communique au sein du bus de terrain de Type 11. D'autre part, un nœud placé sur le segment ouvert n'est pas un nœud membre et est retiré de ce même bus de terrain ou est à y ajouter;

- d) La commande RMC gère l'emplacement des points ouverts dans l'anneau, par l'initialisation du réseau en vue de son démarrage, par la modification de l'emplacement dédié à l'action de réparation comme redondance de support en cas de défaillance, et ce, afin de prévenir l'arrêt total du système et de maintenir la capacité de communication entre les nœuds en bon état de fonctionnement, ainsi que par la modification de l'emplacement dédié à l'insertion d'un nouveau nœud et au retrait d'un nœud membre;
- e) Lorsque l'ACM transmet les DLPDU, la commande RMC les répartit en demandes de symboles d'octets à l'intention des deux convertisseurs parallèle-série d'octets qui transitent par le verrouilleur de trames TX, de sorte que les DLPDU sont transmises hors des deux ports en anneau simultanément;
- f) Lorsque l'ACM transmet les signaux de commande et de diagnostic, la commande RMC active les transmissions des PDU MAC hors des ports en anneau;
- g) Lorsque le convertisseur série-parallèle d'octets transmet les indications de symboles d'octets, les verrouilleurs de trames RX les regroupent dans les PDU MAC, et la commande RMC sélectionne l'une des deux sorties des verrouilleurs de trames RX de sorte que les PDU MAC sont présentées à l'ACM.

7.2.6.2 Définitions de primitive

7.2.6.2.1 Primitives échangées entre l'ACM et la commande RMC

Le Tableau 78 présente toutes les primitives échangées entre l'ACM et la commande RMC. Le Tableau 79 présente les primitives échangées entre la commande RMC et les convertisseurs parallèle-série/série-parallèle constitués du convertisseur parallèle-série / série-parallèle et du verrouilleur de trames TX/RX. Par ailleurs, les primitives échangées entre le convertisseur parallèle-série / série-parallèle et la couche Ph sont présentées dans le Tableau 80.

Tableau 78 – Primitives échangées entre l'ACM et la commande RMC

Nom de la primitive	Source	Paramètres associés
RMC_SEND_DATA.req	ACM	RMSDU
RMC_SEND_DATA.conf	RMC	RMC-status
RMC_RECV_DATA.ind	RMC	RMSDU
RMC_SEND_CF.req	ACM	RMSDU -----LRR specific
RMC_SEND_CF.conf	RMC	RMC-status
RMC_SEND_CF.ind	RMC	RMC-status
RMC_SEND_CF_All.req	ACM	RMSDU -----LPD specific
RMC_SEND_CF_All.conf	RMC	RMC-status

Tableau 79 – Primitives échangées entre la commande RMC et le convertisseur parallèle-série/série-parallèle

Nom de la primitive	Source	Paramètres associés
TX_DATA.req	RMC	RMPDU
TX_DATA.conf	Convertisseur parallèle-série -A / -B	Etat
RX_DATA_A.ind	Convertisseur série-parallèle -A	RMPDU, Etat
RX_DATA_B.ind	Convertisseur série-parallèle -B	RMPDU, Etat

Tableau 80 – Primitives échangées entre la commande RMS et la couche Ph

Nom de la primitive	Source	Paramètres associés
PLS_CARRIER_A.ind	couche Ph: Ring-port-A	CARRIER_STATUS (CARRIER_ON / CARRIER_OFF)
PLS_SIGNAL_A.ind	couche Ph: Ring-port-A	SIGNAL_STATUS (SIGNAL_ERROR / NO-SIGNAL_ERROR)
PLS_DATA_VALID_A.ind	couche Ph: Ring-port-A	DATA_VALID_STATUS (DATA_VALID / DATA_NOT_VALID)
PLS_CARRIER_B.ind	couche Ph: Ring-port-B	CARRIER_STATUS (CARRIER_ON / CARRIER_OFF)
PLS_SIGNAL_B.ind	couche Ph: Ring-port-B	SIGNAL_STATUS (SIGNAL_ERROR / NO-SIGNAL_ERROR)
PLS_DATA_VALID_B.ind	couche Ph: Ring-port-B	DATA_VALID_STATUS (DATA_VALID / DATA_NOT_VALID)

7.2.6.2.2 Paramètres utilisés entre la commande RMC et l'ACM, et la commande RMC et le convertisseur parallèle-série / série-parallèle

Le Tableau 81 et le Tableau 82 présentent les paramètres utilisés pour l'interaction entre la commande RMC et l'ACM ou la couche Ph respectivement.

Tableau 81 – Paramètres entre la commande RMC et l'ACM

Nom du paramètre	Description
RMSDU	Unité de données de service utilisée entre la commande RMC et l'ACM
RMC-status	Etat de résultat de demande, condition de réception, etc.

Tableau 82 – Paramètres entre la commande RMC et le convertisseur parallèle-série / série-parallèle, et la couche Ph

Nom du paramètre	Description
RMPDU	Unité de données de protocole utilisée entre la commande RMC et le convertisseur parallèle-série / série-parallèle
CARRIER_STATUS	Etat de condition d'application du signal porteur sur le support ou non
SIGNAL_STATUS	Etat de condition d'occurrence d'une collision ou non
DATA_VALID_STATUS	Etat de données reçues valides et disponibles

7.2.6.3 Table d'états RMC

La Figure 32 et la Figure 33 illustrent les diagrammes de transition d'état de la commande RMC pour les fonctionnements à 100 Mbps et 1 000 Mbps respectivement. Les tables d'état de la commande RMC pour les fonctionnements à 100 Mbps et 1 000 Mbps sont présentées au Tableau 83 et au Tableau 84 respectivement.

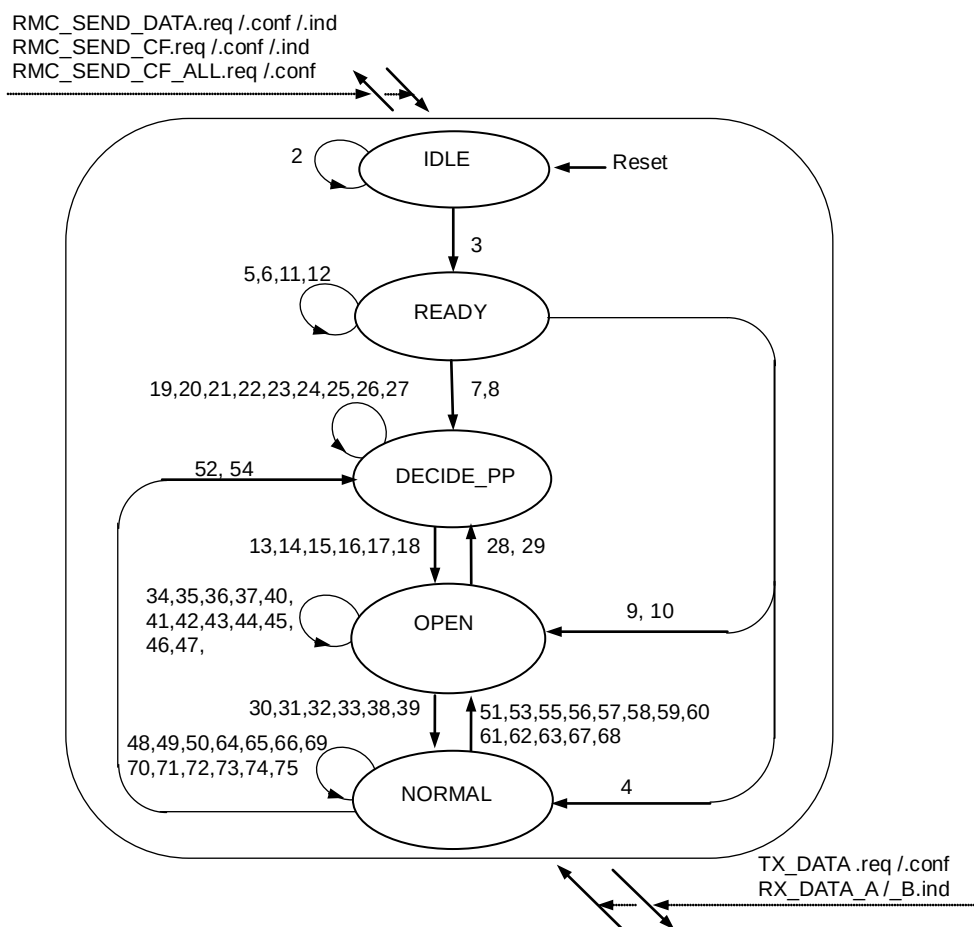


Figure 32 – Diagramme de transition d'état de la commande RMC pour le fonctionnement à 100 Mbps

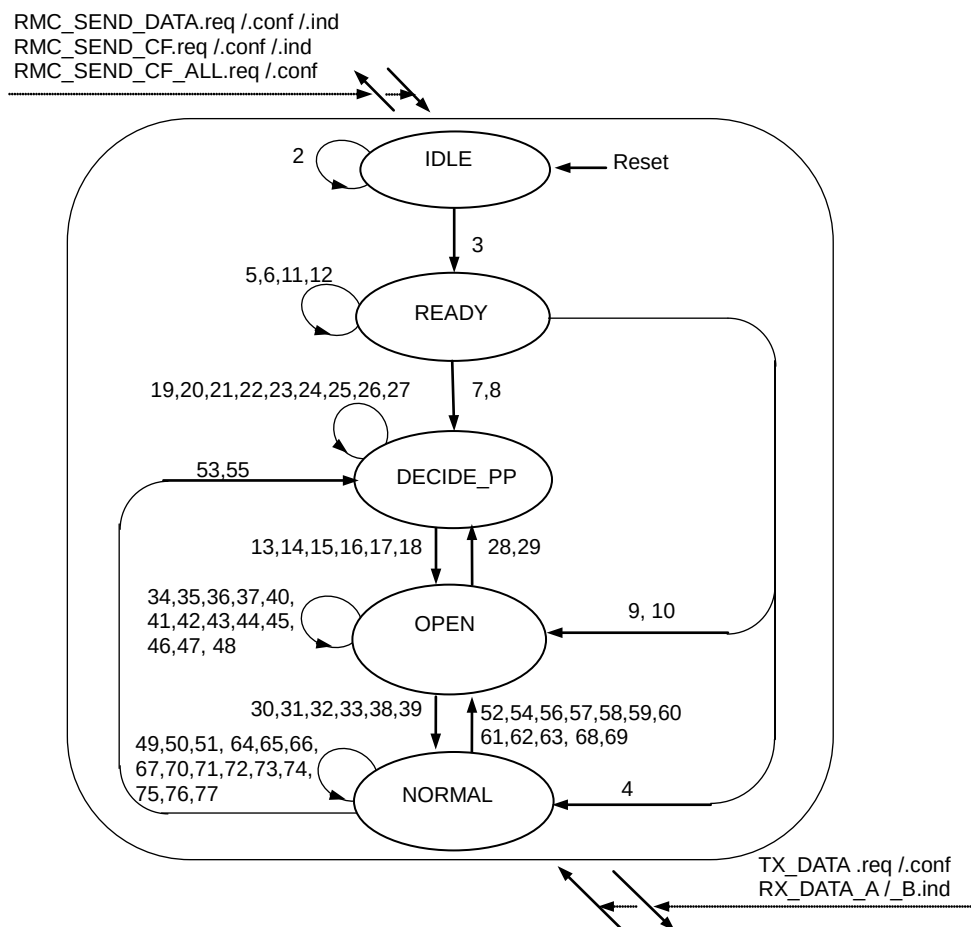


Figure 33 – Diagramme de transition d'état de la commande RMC pour le fonctionnement à 1 000 Mbps

Tableau 83 – Table d'états RCM pour le fonctionnement à 100 Mbps

#	État courant	Événement / Condition => action	État suivant
1	Any states	/ RESET = "True" => RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable")	IDLE
2	IDLE	/ ACTIVE_ACM () = "False" => ----- Wait for ACM being activated none	IDLE
3	IDLE	/ ACTIVE_ACM () = "True" => -----ACM being activated RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Disable", "Disable")	READY

#	État courant	Événement /Condition => action	État suivant
4	READY	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN && V(LINK _A) := "Normal" && V(LINK _B) != "Normal" => -----send out CLM or SYN SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B (" Enable ", "Disable") V(LSYN) := V(TN) RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
5	READY	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMC-status := "Failure" RMC_SEND_DATA.conf { RMC-Status }	READY
6	READY	RMC_SEND_DATA.req { RMSDU } / V(LINK _A) := "Abnormal" && V(LINK _B) := "Abnormal" => RMC-status := "LINK _{AB} -abnormal" RMC_SEND_DATA.conf { RMC-status }	READY
7	READY	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => ----- receiving CLM or SYN from Ring-port-A SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-Port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP
8	READY	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => -----receiving CLM/SYN-frame from Ring-port-B SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP) := "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN) := RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
9	READY	/ V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" => SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) := "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) := null RMC-status := "LINK _B -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN

#	État courant	Événement /Condition => action	État suivant
10	READY	/ V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN):= null RMC-status:= "LINK _A -abnormal" RMC_REC_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
11	READY	/ V(LINK _A):= "Abnormal" && V(LINK _B):= "Abnormal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable") V(LSYN):= null RMC-status:= "LINK _{AB} -abnormal" RMC_REC_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	READY
12	READY	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	READY
13	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----again receiving from Ring-port-A => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
14	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----again receiving from Ring-port-B => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B))	OPEN
15	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----receiving from different Ring-port => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN
16	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----receiving from different Ring-port => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN

#	État courant	Événement /Condition => action	État suivant
17	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" -----LINK _B into abnormal => V(LSYN):= RMPDU.SN RMC-status:= "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU:= RMPDU, RMC-status } START_TIMER (T(NRF _A))	OPEN
18	DECIDE_PP	RX_DATA_B.ind { RMPDU } /(RMPDU.FC = CLM SYM) && V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" -----LINK _A into abnormal => V(LSYN):= RMPDU.SN RMC-status:= "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU:= RMPDU, RMC-status } START_TIMER (T(NRF _B))	OPEN
19	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A):= RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	DECIDE_PP
20	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B):= RMPDU.SN ----- Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	DECIDE_PP
21	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	DECIDE_PP
22	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	DECIDE_PP
23	DECIDE_PP	RMC_SEND_CF.req { RMSDU } / RMSDU = LRR => -----wrong request for LRR RMC_SEND_DATA.conf (RMC-status:= "Failure")	DECIDE_PP
24	DECIDE_PP	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => -----request for LPD SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) TX_DATA.req { RMPDU } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	DECIDE_PP
25	DECIDE_PP	RMC_SEND_DATA.req { RMPDU } / => -----request for data-frame RMC_SEND_DATA.conf (RMC-status:= "Failure")	DECIDE_PP
26	DECIDE_PP	TX_DATA.conf { RMPDU } / => -----conformation of LPD sent out RESTORE_FWE () SEND_COMP (RMPDU)	DECIDE_PP
27	DECIDE_PP	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	DECIDE_PP

#	État courant	Événement /Condition => action	État suivant
28	OPEN	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") -----open-point at Ring-port-B V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A)) STOP_TIMER (T(NRF _B))	DECIDE_PP
29	OPEN	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") -----open-point at Ring-port-A V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B)) STOP_TIMER (T(NRF _A))	DECIDE_PP
30	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-B" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
31	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-A" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
32	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-B" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU }	NORMAL
33	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-A" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU }	NORMAL
34	OPEN	/ EXPIRED_TIMER (T(NRF _A)) && V(PP) = "Ring-port-A" => RMC-status:= "No frame received from Ring-port-A" RMC_SEND_CF.ind { RMC-status } -----activate new ring configuration	OPEN
35	OPEN	/ EXPIRED_TIMER (T(NRF _B)) && V(PP) = "Ring-port-B" => RMC-status:= "No frame received from Ring-port-B" RMC_SEND_CF.ind { RMC-status } ----- activate new ring configuration	OPEN

#	État courant	Événement /Condition => action	État suivant
36	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _A)) => RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
37	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _B)) => RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
38	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _A)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _A))	NORMAL
39	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _B)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _B))	NORMAL
40	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A):= RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	OPEN
41	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B):= RMPDU.SN -----Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	OPEN
42	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	OPEN
43	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	OPEN
44	OPEN	RMC_SEND_DATA.req { RMSDU } / => if RMSDU.FC = (SYN CLM) then SET_NODE_COND ("Master", "Open", V(PS _A), V(PS _B)) V(LSYN):= V(TN) endif RMPDU:= BUILD_RMPDU (RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but LPD and LRR	OPEN

#	État courant	Événement /Condition => action	État suivant
45	OPEN	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => RMPDU:= BUILD_RMPDU (RMPDU.FC) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TX_DATA.req { RMPDU } ----- sending LPD	OPEN
46	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = REQ => RMPDU:= BUILD_RMPDU (RMSDU.FC) if V(PP) = "Ring-port-A" then SET_SEN ("Enable", "Disable") else SET_SEN ("Disable", "Enable") endif SET_NODE_COND ("Slave", "Open", V(PS _A), V(PS _B)) SAVE_FWE () TX_DATA.req { RMPDU } -----sending REQ	OPEN
47	OPEN	TX_DATA.conf {Status } / => RESTORE_FWE () SEND_COMP (RMPDU) --issuing RMC_SEND_DATA /_CF /_CF_ALL.conf	OPEN
48	NORMAL	TX_DATA.conf { RMPDU } / RMPDU.FC = CLM SYN => RMC_SEND_DATA.conf { RMSDU:= RMPDU }	NORMAL
49	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN => SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") V(LSYN):= V(TN) RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending CLM or SYN	NORMAL
50	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMPDU:= BUILD_RMPDU { RMSDU.FC } TX_DATA.req { RMPDU } -----sending all but CLM and SYN	NORMAL
51	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
52	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP

#	État courant	Événement /Condition => action	État suivant
53	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN
54	NORMAL	RX_DATA_B.ind {RMPDU-B} / (RMPDU-B.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
55	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = Ring-port-B" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
56	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = Ring-port-A" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN
57	NORMAL	/ V(LINK _A) = "Abnormal" => -----Link fault on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
58	NORMAL	/ V(LINK _B) = "Abnormal" => -----Link fault on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
59	NORMAL	/ C(CRE _A) >= P(CRE _A) => -----abnormal-reception on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN

#	État courant	Événement /Condition => action	État suivant
60	NORMAL	/ $C(CRE_B) \geq P(CRE_B)$ => -----abnormal-reception on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
61	NORMAL	/ EXPIRED_TIMER (T(NRF_A)) = "True" => -----long-silence on Ring-port-A V(NOS):= "Open"	OPEN
62	NORMAL	/ EXPIRED_TIMER (T(NRF_B)) = "True" => -----long-silence on Ring-port-B V(NOS):= "Open"	OPEN
63	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU:= BUILD_PDU(FC:= LRR) RMPDU.RN:= V(AN_A) SAVE_FWE () SET_FWE_AB_BA ("Enable", "Disable") SET_SEN ("Enable", "Disable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF_A))	NORMAL
64	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU:= BUILD_PDU(FC:= LRR) RMPDU.RN = V(AN_B) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Enable") SET_SEN ("Disable", "Enable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF_B))	NORMAL
65	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF_A))	NORMAL
66	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF_B))	NORMAL
67	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF_A)) = "True" => -----No receipt from neighboring node to Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF_A))	OPEN

#	État courant	Événement /Condition => action	État suivant
68	NORMAL	/ EXPIRED_TIMER (T(ST) = "True" && ACTIVE_TIMER (T(NRF _B)) = "True" => -----No receipt from neighboring node to Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _B))	OPEN
69	NORMAL	RMC_SEND_CF_ALL.req { RMSDU } / => RMPDU:= BUILD_RMPDU (FC:= LPD) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TIME_DELAY (V(IGP)) TX_DATA.req { RMPDU } -----sending LPD	NORMAL
70	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-A V(AN _A):= RMPDU.SN V(FWE _{AB}):= "Enable" START_TIMER (T(NRF _A))	NORMAL
71	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-B V(AN _B):= RMPDU.SN V(FWE _{BA}):= "Enable" START_TIMER (T(NRF _B))	NORMAL
72	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	NORMAL
73	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU-B, REN _B) START_TIMER (T(NRF _B))	NORMAL
74	NORMAL	RMC_SEND_DATA.req { RMSDU } / => -----sending all but LPD or LRR if RMSDU.FC = SYN CLM then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(LSYN):= V(TN) endif SET_FWE_AB_BA ("Enable", "Enable") RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
75	NORMAL	TX_DATA.conf { RMPDU } / RMPDU <> CLM SYN => RESTORE_FWE () SEND_COMP (RMPDU)	NORMAL

Tableau 84 – Table d'états RCM pour le fonctionnement à 1 000 Mbps

#	État courant	Événement /Condition => action	État suivant
1	Any states	/ RESET = "True" => RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable")	IDLE
2	IDLE	/ ACTIVE_ACM () = "False" => ----- Wait for ACM being activated none	IDLE
3	IDLE	/ ACTIVE_ACM () = "True" => -----ACM being activated RESET_VAR () SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Disable", "Disable")	READY
4	READY	RMC_SEND_DATA.req { RMSDU } / (RMSDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => -----send out CLM or SYN if V(ESYN) = "True" then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP) = "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) = V(TN) RMPDU := BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } else RMC-status := "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	NORMAL
5	READY	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMC-status := "Failure" RMC_SEND_DATA.conf { RMC-Status }	READY
6	READY	RMC_SEND_DATA.req { RMSDU } / V(LINK _A) = "Abnormal" && V(LINK _B) = "Abnormal" => RMC-status := "LINK _{AB} -abnormal" RMC_SEND_DATA.conf { RMC-status }	READY
7	READY	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => ----- receiving CLM or SYN from Ring-port-A SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP) = "Ring-Port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN) = RMPDU.SN RMC_RECV_DATA.ind { RMSDU := RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP

#	État courant	Événement /Condition => action	État suivant
8	READY	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" => -----receiving CLM/SYN-frame from Ring-port-B SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
9	READY	/ V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" => SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") V(LSYN):= null RMC-status:= "LINK _B -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
10	READY	/ V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") V(LSYN):= null RMC-status:= "LINK _A -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	OPEN
11	READY	/ V(LINK _A):= "Abnormal" && V(LINK _B):= "Abnormal" => SET_NODE_COND ("Slave", "Open", "Blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Disable", "Disable") SET_REN_A_B ("Disable", "Disable") V(LSYN):= null RMC-status:= "LINK _{AB} -abnormal" RMC_RECV_DATA.ind { RMC-status } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	READY
12	READY	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	READY
13	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----again receiving from Ring-port-A => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN

#	État courant	Événement /Condition => action	État suivant
14	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----again receiving from Ring-port-B => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B))	OPEN
15	DECIDE_PP	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-A" -----receiving from different Ring-port => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN
16	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Normal" && V(PP) = "Ring-port-B" -----receiving from different Ring-port => V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
17	DECIDE_PP	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(LINK _A) = "Normal" && V(LINK _B) = "Abnormal" -----LINK _B into abnormal => V(LSYN):= RMPDU.SN RMC-status:= "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU:= RMPDU, RMC-status } START_TIMER (T(NRF _A))	OPEN
18	DECIDE_PP	RX_DATA_B.ind { RMPDU } /(RMPDU.FC = CLM SYM) && V(LINK _A) = "Abnormal" && V(LINK _B) = "Normal" -----LINK _A into abnormal => V(LSYN):= RMPDU.SN RMC-status:= "LINK _B -abnormal" RMC_RECV_DATA.ind { RMSDU:= RMPDU, RMC-status } START_TIMER (T(NRF _B))	OPEN
19	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A):= RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	DECIDE_PP
20	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B):= RMPDU.SN ----- Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	DECIDE_PP
21	DECIDE_PP	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	DECIDE_PP
22	DECIDE_PP	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (CLM SYN) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	DECIDE_PP

#	État courant	Événement /Condition => action	État suivant
23	DECIDE_PP	RMC_SEND_CF.req { RMSDU } / RMSDU = LRR => -----wrong request for LRR RMC_SEND_DATA.conf (RMC-status:= "Failure")	DECIDE_PP
24	DECIDE_PP	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => -----request for LPD SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) TX_DATA.req { RMPDU } START_TIMER (T(NRF _A)) START_TIMER (T(NRF _B))	DECIDE_PP
25	DECIDE_PP	RMC_SEND_DATA.req { RMPDU } / => -----request for data-frame RMC_SEND_DATA.conf (RMC-status:= "Failure")	DECIDE_PP
26	DECIDE_PP	TX_DATA.conf { RMPDU } / => -----conformation of LPD sent out RESTORE_FWE() SEND_COMP (RMPDU)	DECIDE_PP
27	DECIDE_PP	/ EXPIRED_TIMER (T(NRF _A)) EXPIRED_TIMER (T(NRF _B)) => none	DECIDE_PP
28	OPEN	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") -----open-point at Ring-port-B V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A)) STOP_TIMER (T(NRF _B))	DECIDE_PP
29	OPEN	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN => -----receiving CLM or SYN by different node from different Ring-port V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") -----open-point at Ring-port-A V(LSYN):= RMPDU.SN RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _B)) STOP_TIMER (T(NRF _A))	DECIDE_PP
30	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-B" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL
31	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && V(PP) = "Ring-port-A" && RMPDU.RN = V(TS) => -----receiving LRR intended to this node from Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switching into Normal-state	NORMAL

#	État courant	Événement /Condition => action	État suivant
32	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-B" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU }	NORMAL
33	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = REQ && RMPDU.RN = V(TN) && V(PP) = "Ring-port-A" => -----receiving REQ by new node on open-segment SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU }	NORMAL
34	OPEN	/ EXPIRED_TIMER (T(NRF _A)) && V(PP) = "Ring-port-A" => RMC-status:= "No frame received from Ring-port-A" RMC_SEND_CF.ind { RMC-status } -----activate new ring configuration	OPEN
35	OPEN	/ EXPIRED_TIMER (T(NRF _B)) && V(PP) = "Ring-port-B" => RMC-status:= "No frame received from Ring-port-B" RMC_SEND_CF.ind { RMC-status } ----- activate new ring configuration	OPEN
36	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _A)) => RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending LRR out START_TIMER (T(ST))	OPEN
37	OPEN	RMC_SEND_CF.req { RMSDU } / EXPIRED_TIMER (T(NRF _B)) => RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req (RMPDU) -----sending LRR out START_TIMER (T(ST))	OPEN
38	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _A)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-A SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _A))	NORMAL
39	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && EXPIRED_TIMER (T(ST)) && EXPIRED_TIMER (T(NRF _B)) && RMPDU.RN = V(TN) => ---receiving LRR intended to this node by a node neighbor to Ring-port-B SET_NODE_COND (V(NS), "Normal", "Non-blocked", "Non-blocked") SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") -----switch into Normal-state START_TIMER (T(NRF _B))	NORMAL
40	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => V(AN _A):= RMPDU.SN -----Node-number neighboring to Ring-port-A START_TIMER (T(NRF _A))	OPEN

#	État courant	Événement /Condition => action	État suivant
41	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => V(AN _B):= RMPDU.SN -----Node-number neighboring to Ring-port-B START_TIMER (T(NRF _B))	OPEN
42	OPEN	RX_DATA_A.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-A IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	OPEN
43	OPEN	RX_DATA_B.ind { RMPDU } / RMPDU.FC <> (LPD LRR) => -----receiving on Ring-port-B IND_PROC (RMPDU, REN _B) START_TIMER (T(NRF _B))	OPEN
44	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = (SYN CLM) => if V(ESYN) = "True" then SET_NODE_COND ("Master", "Open", V(PS _A), V(PS _B)) V(LSYN):= V(TN) RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but LPD and LRR else RMC-status:= "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	OPEN
45	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> (SYN CLM REQ) => RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) SAVE_FWE () TX_DATA.req { RMPDU } -----sending all but SYN, CLM, LPD, LRR, REQ	OPEN
46	OPEN	RMC_SEND_CF_ALL.req { RMSDU } / RMSDU.FC = LPD => RMPDU:= BUILD_RMPDU (RMPDU.FC) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TX_DATA.req { RMPDU } ----- sending LPD	OPEN
47	OPEN	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = REQ => RMPDU:= BUILD_RMPDU (RMSDU.FC) if V(PP) = "Ring-port-A" then SET_SEN ("Enable", "Disable") else SET_SEN ("Disable", "Enable") endif SET_NODE_COND ("Slave", "Open", V(PS _A), V(PS _B)) SAVE_FWE () TX_DATA.req { RMPDU } -----sending REQ	OPEN
48	OPEN	TX_DATA.conf {Status } / => RESTORE_FWE () SEND_COMP (RMPDU) --issuing RMC_SEND_DATA /_CF /_CF_ALL.conf	OPEN
49	NORMAL	TX_DATA.conf { RMPDU } / RMPDU.FC = CLM SYN => RMC_SEND_DATA.conf { RMSDU:= RMPDU }	NORMAL

#	État courant	Événement /Condition => action	État suivant
50	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC = CLM SYN => if V(ESYN) = "True" then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Enable") SET_REN_A_B ("Enable", "Enable") V(LSYN):= V(TN) RMPDU:= BUILD_RMPDU (RMSDU.FC) TX_DATA.req { RMPDU } -----sending CLM or SYN else RMC-status:= "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	NORMAL
51	NORMAL	RMC_SEND_DATA.req { RMSDU } / RMSDU.FC <> CLM SYN => RMPDU:= BUILD_RMPDU { RMSDU.FC } TX_DATA.req { RMPDU } -----sending all but CLM and SYN	NORMAL
52	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
53	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-B" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => ----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMPDU } START_TIMER (T(NRF _A))	DECIDE_PP
54	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) = RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN

#	État courant	Événement /Condition => action	État suivant
55	NORMAL	RX_DATA_B.ind { RMPDU-B } / (RMPDU-B.FC = CLM SYN) && V(PP) = "Ring-port-A" && V(LSYN) <> RMPDU.SN V(NS) = "Slave" => -----receiving CLM or SYN by new master from different Ring-port SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	DECIDE_PP
56	NORMAL	RX_DATA_A.ind { RMPDU } / (RMPDU.FC = CLM SYN) && V(PP) = Ring-port-B" && V(NS) = "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _A))	OPEN
57	NORMAL	RX_DATA_B.ind { RMPDU } / (RMPDU.FC = CLM SYM) && V(PP) = Ring-port-A" && V(NS):= "Master" => -----receiving CLM or SYN in Master-state SET_NODE_COND ("Slave", "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" V(LSYN):= RMPDU.SN SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") RMC_RECV_DATA.ind { RMSDU:= RMPDU } START_TIMER (T(NRF _B))	OPEN
58	NORMAL	/ V(LINK _A) = "Abnormal" => -----Link fault on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
59	NORMAL	/ V(LINK _B) = "Abnormal" => -----Link fault on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
60	NORMAL	/ C(CRE _A) >= P(CRE _A) => -----abnormal-reception on Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable")	OPEN
61	NORMAL	/ C(CRE _B) >= P(CRE _B) => -----abnormal-reception on Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable")	OPEN
62	NORMAL	/ EXPIRED_TIMER (T(NRF _A)) = "True" => -----long-silence on Ring-port-A V(NOS):= "Open"	OPEN
63	NORMAL	/ EXPIRED_TIMER (T(NRF _B)) = "True" => -----long-silence on Ring-port-B V(NOS):= "Open"	OPEN

#	État courant	Événement /Condition => action	État suivant
64	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU:= BUILD_PDU(FC:= LRR) RMPDU.RN:= V(AN _A) SAVE_FWE () SET_FWE_AB_BA ("Enable", "Disable") SET_SEN ("Enable", "Disable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _A))	NORMAL
65	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && RMPDU.RN <> V(TN) => -----receiving LRR but not intended to this node RMPDU:= BUILD_PDU(FC:= LRR) RMPDU.RN = V(AN _B) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Enable") SET_SEN ("Disable", "Enable") TX_DATA.req { RMPDU } START_TIMER (T(ST)) START_TIMER (T(NRF _B))	NORMAL
66	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _A))	NORMAL
67	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LRR && ACTIVE_TIMER (T(ST)) = "True" && RMPDU.RN = V(TN) => -----receiving LRR intended to this node STOP_TIMER (T(ST)) RESTORE_FWE () SET_SEN ("Enable", "Enable") START_TIMER (T(NRF _B))	NORMAL
68	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF _A)) = "True" => -----No receipt from neighboring node to Ring-port-A SET_NODE_COND (V(NS), "Open", "Blocked", "Non-blocked") V(PP):= "Ring-port-B" SET_FWE_AB_BA ("Disable", "Enable") SET_REN_A_B ("Disable", "Enable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _A))	OPEN
69	NORMAL	/ EXPIRED_TIMER (T(ST)) = "True" && ACTIVE_TIMER (T(NRF _B)) = "True" => -----No receipt from neighboring node to Ring-port-B SET_NODE_COND (V(NS), "Open", "Non-blocked", "Blocked") V(PP):= "Ring-port-A" SET_FWE_AB_BA ("Enable", "Disable") SET_REN_A_B ("Enable", "Disable") SET_SEN ("Enable", "Enable") STOP_TIMER (T(NRF _B))	OPEN
70	NORMAL	RMC_SEND_CF_ALL.req { RMSDU } / => RMPDU:= BUILD_RMPDU (FC:= LPD) SAVE_FWE () SET_FWE_AB_BA ("Disable", "Disable") TIME_DELAY (V(IGP)) TX_DATA.req { RMPDU } -----sending LPD	NORMAL

#	État courant	Événement /Condition => action	État suivant
71	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-A V(AN _A):= RMPDU.SN V(FWE _{AB}):= "Enable" START_TIMER (T(NRF _A))	NORMAL
72	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU.FC = LPD => -----receiving LPD from Ring-port-B V(AN _B):= RMPDU.SN V(FWE _{BA}):= "Enable" START_TIMER (T(NRF _B))	NORMAL
73	NORMAL	RX_DATA_A.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU, REN _A) START_TIMER (T(NRF _A))	NORMAL
74	NORMAL	RX_DATA_B.ind { RMPDU } / RMPDU <> LPD LRR => -----receiving all but LPD or LRR IND_PROC (RMPDU-B, REN _B) START_TIMER (T(NRF _B))	NORMAL
75	NORMAL	RMC_SEND_DATA.req { RMSDU} / RMSDU.FC = (SYN CLM) => -----sending SYN or CLM if V(ESYN) = "True" then SET_NODE_COND ("Master", "Normal", "Non-blocked", "Non-blocked") V(LSYN):= V(TN) SET_FWE_AB_BA ("Enable", "Enable") RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) TX_DATA.req { RMPDU } else RMC-status:= "SYN disable" RMC_SEND_DATA.conf { RMC-status } endif	NORMAL
76	NORMAL	RMC_SEND_DATA.req { RMSDU} / RMSDU.FC <> (SYN CLM) => -----sending all but LPD or LRR SET_FWE_AB_BA ("Enable", "Enable") RMPDU:= BUILD_RMPDU (FC:= RMSDU.FC) TX_DATA.req { RMPDU }	NORMAL
77	NORMAL	TX_DATA.conf { RMPDU } / RMPDU <> CLM SYN => RESTORE_FWE () SEND_COMP (RMPDU)	NORMAL

7.2.6.4 Fonctions utilisées par la commande RMC

Les fonctions utilisées dans la table d'états RMC sont récapitulées dans le Tableau 85.

Tableau 85 – Table de fonctions RMC

Nom de la fonction	Description et fonctionnement
ACTIVE_ACM ()	Lorsque l'ACM est active, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
ACTIVE_TIMER (v(t))	Lorsque le temporisateur demandé v(t) est actif, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
BUILD_RMPDU (FC)	Selon l'information de commande (FC) de trames demandée, une trame de commande d'anneau est assemblée de façon correspondante
EXPIRED_TIMER (v(t))	Lorsque le temporisateur demandé v(t) a expiré, la valeur "Vrai" est renvoyée, la valeur "Faux" étant renvoyée dans le cas contraire
IND_PROC (PDU, ren)	if ren = "Enabled" && (PDU <> (LPD LRR)) then RMC_RECV_DATA.ind { PDU } endif
RESTORE_FWE ()	Restaurer V(FWE _{AB}) et V(FWE _{BA})
RESET_VAR ()	Réinitialiser les variables sur la valeur par défaut
SAVE_FWE ()	Archiver V(FWE _{AB}) et V(FWE _{BA}) provisoirement
SEND_COMP (RMPDU)	if RMPDU.FC = LRR then RMC_SEND_CF.conf { RMC-status:= "Success" } elseif RMPDU.FC = LPD then RMC_SEND_CF_ALL.conf { RMC-status:= "Success" } else RMC_SEND_DATA.conf { RMC-status:= "Success" } endif
SET_FWE_AB_BA (fweab, fweba)	V(FWE _{AB}):= fweab -----value is either "Enable" or "Disable" V(FWE _{BA}):= fweba -----value is either "Enable" or "Disable"
SET_NODE_COND (Node-state, Operation-state, Ring-port-A-state, Ring-port-B-state)	V(NS):= Node-state -----value is either "Master" or "Slave" V(NOS):= Node-operation-state -----value is either "Normal" or "Open" V(PS _A):= Ring-port-A-state -----value is either "Non-blocked" or "Blocked" V(PS _B):= Ring-port-B-state -----value is either "Non-blocked" or "Blocked"
SET_REN_A_B (rena, renb)	V(REN _A):= rena -----value is either "Enable" or "Disable" V(REN _B):= renb -----value is either "Enable" or "Disable"
SET_SEN (sena, senb)	V(SEN _A):= sena -----value is either "Enable" or "Disable" V(SEN _B):= senb -----value is either "Enable" or "Disable"
START_TIMER (v(t))	Activer le temporisateur demandé V(t); l'événement "Expired Timer" se produit à l'expiration du temporisateur
STOP_TIMER (v(t))	Contraindre le temporisateur demandé V(T) à s'arrêter
TIME_DELAY (v(x))	Attendre le délai spécifié par v(x)

7.3 Convertisseur parallèle-série et convertisseur série-parallèle

Les convertisseurs parallèle-série et série-parallèle sont identiques à ceux définis dans l'ISO/CEI 8802-3.

7.4 Protocole de gestion DLL

7.4.1 Protocole de gestion DLL d'une architecture en étoile

7.4.1.1 Présentation générale

Le protocole d'interface entre la commande DLM et l'utilisateur DLMS est décrit. Le protocole de gestion DLL fournit les services de gestion DLL spécifiés à l'Article 5 de la CEI 61158-3-11 en faisant appel aux services à disposition de l'utilisateur DLMS.

En 7.4.1, la mise en œuvre intégrale et le caractère local sont exclus de manière intentionnelle.

7.4.1.2 Définitions de primitive

7.4.1.2.1 Primitives échangées entre l'utilisateur DLMS et la commande DLM

Le Tableau 86 présente toutes les primitives échangées entre l'utilisateur DLMS et la commande DLM.

Tableau 86 – Primitives échangées entre l'utilisateur DLMS et la commande DLM

Nom de la primitive	Source	Paramètres associés
demande DLM-Reset	Utilisateur DLMS	(aucun)
confirmation DLM-Reset	DLM	Etat
demande DLM-Set-value	Utilisateur DLMS	Variable-name, Desired-value
confirmation DLM-Set-value	DLM	Etat
demande DLM-Get-variable	Utilisateur DLMS	Variable-name
confirmation DLM-Get-variable	DLM	Etat, Current_value
indication DLM-Event	DLM	Event-identifiant, Additional-information
demande DLM-Set-publisher-configuration	Utilisateur DLMS	Desired-speed-class, Desired-configuration
confirmation DLM-Set-publisher-configuration	DLM	Etat
demande DLM-Get-publisher-configuration	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Get-publisher-configuration	DLM	Status, Current-configuration
demande DLM-Activate-TCC	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Activate-TCC	DLM	Etat
demande DLM-Deactivate-TCC	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Deactivate-TCC	DLM	Etat

7.4.1.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLMS et la commande DLM

Tous les paramètres utilisés avec les primitives échangées entre l'utilisateur DLMS et la commande DLM sont récapitulés dans le Tableau 87. Le Tableau 88 récapitule les variables associées aux événements de changement d'état.

Tableau 87 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DL et la commande DLM

Nom du paramètre	Description
Etat	Etat de l'exécution du service
Variable-name	nom de variable DL à adresser
Desired-value	valeur de variable DL à définir
Current-value	Valeur de variable DL actuelle à lire
Event-identifiant / Additional-information	Événement dont l'occurrence est annoncée avec variable DL associée et changement d'état
Desired-speed	Classe de vitesse pour la transmission de données cycliques prioritaires

Nom du paramètre	Description
Desired-configuration	Ensemble d'identifiants DLCEP à définir comme éditeur pour la transmission de données cycliques prioritaires avec classe de vitesse souhaitée
Current-configuration	Ensemble actuel d'identifiants DLCEP défini comme éditeur pour la transmission de données cycliques prioritaires avec classe de vitesse souhaitée

Les types de données sont spécifiés à l'Article 5 de la CEI 61158-5-11. Le bit de poids fort de l'octet correspondant est toujours utilisé comme bit de poids fort du nombre binaire.

Tableau 88 – Variables de changement d'état relatives aux événements

Paramètres d'exploitation			
Nom de variable	Description	Information supplémentaire	Type de données
RCS _A , RCS _B	Canal de réception A ou B choisi pour la réception des paquets	Vrai: canal de réception A ou B choisi pour la réception des paquets respectivement Faux: non choisi pour la réception des paquets sur le canal A ou B respectivement	BOOL
SEN _A , SEN _B	Emetteur A ou B activé pour l'envoi de paquets	Vrai: activé pour l'envoi de paquets depuis l'émetteur-récepteur A ou B respectivement. Faux: désactivé pour l'envoi de paquets depuis l'émetteur-récepteur A ou B respectivement. La valeur par défaut est "Vrai"	BOOL
IS	Fonctionnement du nombre total de nœuds IN ou hors service	Vrai: en service Faux: hors service	BOOL
FE	Erreur fatale effective	Vrai: erreur fatale effective et système hors service. Faux: en fonctionnement normal	BOOL
TCIS	Transmission de données TCC en ou hors service	Vrai: Transmission de données TCC en service Faux: Service de transmission de données TCC hors service	BOOL
CRCC	Modification du canal de réception	Vrai: Modification du canal de réception. RCS _A et RCS _B permettent de vérifier le canal de réception	BOOL
LINK _A , LINK _B	Canal A ou B de liaison Ethernet en fonctionnement ou non	Vrai: Liaison Ethernet en fonctionnement. Faux: Liaison Ethernet hors service	BOOL
CTCC	Modification du canal de transmission	Vrai: Modification du canal de transmission. SEN _A et SEN _B permettent de vérifier le canal de transmission	BOOL
CLLC	Modification du contenu de la liste opérationnelle	Vrai: Modification du contenu LL	BOOL
DDLE	Reproduire les DLE détectées avec le même numéro d'identifiant de nœud de TN	Vrai: Reproduire les DLE détectées avec le même numéro d'identifiant de nœud de TN Faux: une DLE uniquement avec ce numéro de TN	BOOL
CCDHS	Changement de l'état indiquant actif (sain) ou inactif (non sain) des données TCC correspondantes. L'état est mentionné dans le bloc de données d'information "CDH _{blk} "	Vrai: changement d'état d'actif à inactif. Faux: aucun changement d'état	BOOL

7.4.1.3 Table d'états DLM

Les diagrammes de transition d'état sont présentés à la Figure 34 et au Tableau 89.

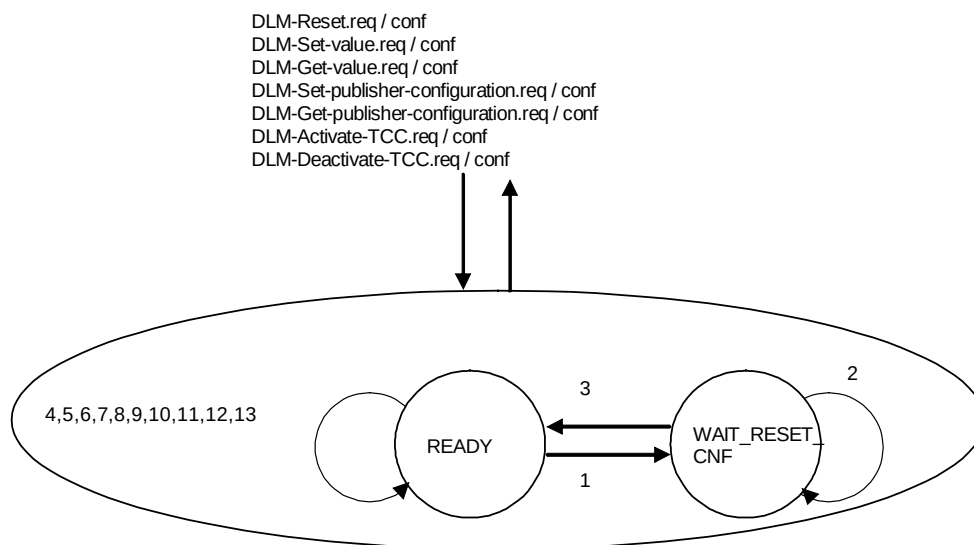


Figure 34 – Diagramme de transition d'état de la commande DLM

Tableau 89 – Table d'états DLM

#	État courant	Événement /condition ⇒ action	État suivant
1	READY	DLM-Reset.req {} ⇒ MAC_reset:= MAC_RESET() RMC_reset:= "FALSE"	WAIT_RESET_CNF
2	WAIT_RESET_CNF	/ MAC_reset:= "TRUE" && RMC_reset:= "FALSE" ⇒ RMC_reset:= RMC_RESET	WAIT_RESET_CNF
3	WAIT_RESET_CNF	/ ⇒ -- MAC and RMC reset completed RESET_VAR () DLM_RESET.cnf { Status:= "Success" }	READY
4	READY	DLM-Set-value.req {Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) = "Valid" ⇒ SET_VALUE (Variable-name, Desired-value) Status:= "Success" DLM-Set-value.conf {Status}	READY
5	READY	DLM-Set-value.req {Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) <> "Valid" ⇒ Status:= "Failure" DLM-Set-value.conf {Status}	READY
6	READY	DLM-Get-value.req {Variable-name } / CHECK_VAR (Variable-name) = "Valid" ⇒ Current-value:= GET_CURRENT_VAL (Variable-name) Status:= "Success" DLM-Get-value.conf { Current-value, Status }	READY
7	READY	DLM-Get-value.req {Variable-name } / CHECK_VAR (Variable-name) <> "Valid" ⇒ Current-value:= NIL Status:= "Failure" DLM-Get-value.conf { Current-value, Status }	READY

#	État courant	Événement /condition ⇒ action	État suivant
8	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) = "Valid" ⇒ SET_PC (Desired-speed-class, Desired-configuration) Status:= "Success" DLM-Set-publisher-configuration.conf { Status }	READY
9	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) <> "Valid" ⇒ Status:= "Failure" DLM-Set-publisher-configuration.conf { Status }	READY
10	READY	DLM-Get-publisher-configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) = "Valid" ⇒ Current-configuration:= GET_PC (Desired-speed-class) Status:= "Success" DLM-Get-Publisher-Configuration.conf { Status, Current- configuration }	READY
11	READY	DLM-Get-Publisher-Configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) <> "Valid" ⇒ Current-configuration:= NIL Status:= "Failure" DLM-Get-Publisher-Configuration.conf { Status, Current- configuration }	READY
12	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) = "Valid" ⇒ STOP_TCC (Desired-speed-class) Status:= "Success" DLM-Activate-TCC.conf { Status }	READY
13	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) = "Valid" ⇒ START_TCC (Desired-speed-class) Status:= "Success" DLM-Deactivate-TCC.conf { Status }	READY
14	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) <> "Valid" ⇒ Status:= "Failure" DLM-Deactivate-TCC.conf { Status }	READY

7.4.1.4 Fonction

Les fonctions DLM sont récapitulées dans le Tableau 90.

Tableau 90 – Table de fonctions DLM

Nom de la fonction	Description et fonctionnement
CHECK_PC (Desired-speed-class, Desired-configuration)	Vérifier la validité des Desired-speed-class et Desired-configuration demandées. Les valeurs Desired-speed-class potentielles sont: - High-speed - Medium-speed ou Low-speed Vérifier la validité de l'adresse DLCEP et du nombre total de Desired-configuration. De plus, lorsque la transmission de données TCC de la Desired-speed-class demandée est inactive, la valeur "valide" est renvoyée
CHECK_SPEED (Desired-speed-class)	Vérifier la validité des Desired-speed-classes demandées. Les valeurs Desired-speed-classes potentielles sont - High-speed - Medium-speed - Low-speed
CHECK_VALUE (Variable-name, Desired-value)	Vérifier la validité des variables demandées avec la valeur souhaitée. Les variables potentielles avec la valeur sont définies au 4.6
CHECK_VAR (Variable-name)	Vérifier la validité des Variables demandées. Les variables potentielles sont définies au 4.6
CHECK_ATCC (Desired-speed-class)	Vérifier la validité des Desired-speed-classes demandées. Les valeurs Desired-speed-classes potentielles sont - High-speed - Medium-speed - Low-speed De plus, lorsque la transmission de données TCC de la Desired-speed-class demandée est inactive, la valeur "valide" est renvoyée
CHECK_DTCC (Desired-speed-class)	Vérifier la validité des Desired-speed-classes demandées. Les valeurs Desired-speed-class potentielles sont - High-speed - Medium-speed - Low-speed De plus, lorsque la transmission de données TCC de la Desired-speed-class demandée est inactive, la valeur "valide" est renvoyée
GET_CURRENT_VAL (Variable-name)	Obtenir la valeur de la variable demandée
RESET_VAR	Initialiser toutes les variables et l'ensemble de Publisher-configuration (Configuration de l'éditeur) pour chaque Speed-class (Classe de vitesse). Le service de transfert de données TCC est désactivé
START_TCC (Desired-speed)	Activer la transmission de données TCC de la Desired-speed-class demandée
STOP_TCC (Desired-speed)	Désactiver la transmission de données TCC de la Desired-speed-class demandée

7.4.2 Protocole de gestion DLL d'une architecture en boucle

7.4.2.1 Présentation générale

Le protocole d'interface entre la commande DLM et l'utilisateur DLMS est décrit. Le protocole de gestion DLL fournit les services de gestion DLL spécifiés à l'Article 5 de la CEI 61158-3-11 en faisant appel aux services à disposition de l'utilisateur DLMS.

En 7.4.2, la mise en œuvre intégrale et le caractère local sont exclus de manière intentionnelle.

7.4.2.2 Définitions de primitive

7.4.2.2.1 Primitives échangées entre l'utilisateur DLMS et la commande DLM

Le Tableau 91 présente toutes les primitives échangées entre l'utilisateur DLMS et la commande DLM.

Tableau 91 – Primitives échangées entre l'utilisateur DLMS et la commande DLM

Nom de la primitive	Source	Paramètres associés
demande DLM-Reset	Utilisateur DLMS	aucun
confirmation DLM-Reset	DLM	Etat
demande DLM-Set-value	Utilisateur DLMS	Variable-name, Desired-value
confirmation DLM-Set-value	DLM	Etat
demande DLM-Get-variable	Utilisateur DLMS	Variable-name
confirmation DLM-Get-variable	DLM	Etat, Current_value
indication DLM-Event	DLM	Event-identifiant, Additional-information
demande DLM-Set-publisher-configuration	Utilisateur DLMS	Desired-speed-class, Desired-configuration
confirmation DLM-Set-publisher-configuration	DLM	Etat
demande DLM-Get-publisher-configuration	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Get-publisher-configuration	DLM	Status, Current-configuration
demande DLM-Activate-TCC	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Activate-TCC	DLM	Etat
demande DLM-Deactivate-TCC	Utilisateur DLMS	Desired-speed-class
confirmation DLM-Deactivate-TCC	DLM	Etat

7.4.2.2.2 Paramètres utilisés avec les primitives échangées entre l'utilisateur DLMS et la commande DLM

Tous les paramètres utilisés avec les primitives échangées entre l'utilisateur DLMS et la commande DLM sont récapitulés dans le Tableau 92.

Tableau 92 – Paramètres utilisés avec les primitives échangées entre l'utilisateur DL et la commande DLM

Nom du paramètre	Description
Etat	Etat de l'exécution du service
Variable-name	nom de variable DL à adresser
Desired-value	valeur de variable DL à définir
Current-value	Valeur de variable DL actuelle à lire
Event-identifiant / Additional-information	Événement dont l'occurrence est annoncée avec variable DL associée et changement d'état
Desired-speed	Classe de vitesse pour la transmission de données cycliques prioritaires
Desired-configuration	Ensemble d'identifiants DLCEP à définir comme éditeur pour la transmission de données cycliques prioritaires avec classe de vitesse souhaitée
Current-configuration	Ensemble actuel d'identifiants DLCEP défini comme éditeur pour la transmission de données cycliques prioritaires avec classe de vitesse souhaitée

Le Tableau 93 récapitule les variables associées aux événements de changement d'état.

Les types de données sont spécifiés à l'Article 5 de la CEI 61158-5-11. Le bit de poids fort de l'octet correspondant est toujours utilisé comme bit de poids fort du nombre binaire.

Tableau 93 – Variables de changement d'état relatives aux événements

Paramètres d'exploitation			
Nom de variable	Description	Information supplémentaire	Type de données
NS	Changement de l'état de nœud en "Maître" ou "Esclave"	Vrai: "Maître" Faux: "Esclave"	BOOL
NOS	Changement de l'état de fonctionnement du nœud en "Normal" ou "Ouvert"	Vrai: "Normal" Faux: "Ouvert"	BOOL
PS _A , PS _B	Changement de l'état de port en anneau A ou B en "Bloqué" ou "Non bloqué"	Vrai: "Non bloqué" Faux: "Bloqué"	BOOL
LINK _A , LINK _B	Modification de la liaison Ethernet sur le port en anneau A ou B en fonctionnement ou non	Vrai: Liaison Ethernet en fonctionnement. Faux: Liaison Ethernet hors service	BOOL
AN _A , AN _B	Modification du nœud adjacent	La valeur se situe dans la plage comprise entre 254 et 0	USINT
NRF _A , NRF _B	Aucune réception de trame sur LINK _A ou B	Vrai: aucune réception de trame au cours de la période d'observation Faux: réception de trames périodique	BOOL
CRE _A , CRE _B	Abnormal-reception-error (erreur de réception anormale) sur LINK _A ou B	Vrai: abnormal-reception-error effective sur les comptages répétitifs admissibles Faux: aucune abnormal-reception-error	BOOL
LSYN	Changement de l'état "Maître" du nœud	La valeur se situe dans la plage comprise entre 254 et 0	USINT
DDLE	Reproduire les DLE détectées avec le même numéro d'identifiant de nœud de TN	Vrai: Reproduire les DLE détectées avec le même numéro d'identifiant de nœud de TN Faux: une DLE uniquement avec ce numéro de TN	BOOL
TCIS	Transmission de données TCC en ou hors service	Vrai: Transmission de données TCC en service Faux: Service de transmission de données TCC hors service	BOOL
IS	Fonctionnement du nombre total de nœuds IN ou hors service	Vrai: en service False: hors service	BOOL
FE	Erreur fatale effective	Vrai: erreur fatale effective et système hors service. Faux: en fonctionnement normal	BOOL

7.4.2.3 Table d'états DLM

Le diagramme de transition d'état de la commande DLM est illustré à la Figure 35 et la table d'états DLM est présentée dans le Tableau 94 respectivement.

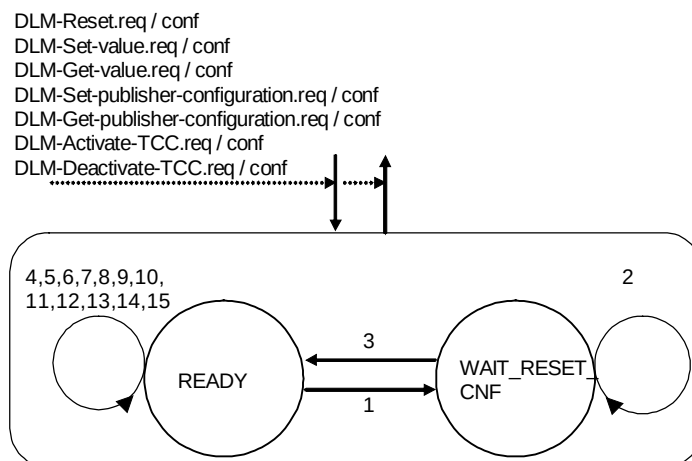


Figure 35 – Diagramme de transition d'état de la commande DLM

Tableau 94 – Table d'états DLM

#	État courant	Événement /Condition => action	État suivant
1	READY	DLM-Reset.req { } / => ACM_reset:= ACM_RESET() RMC_reset:= "FALSE"	WAIT_RESET_CNF
2	WAIT_RESET_CNF	/ ACM_reset = "True" && RMC_reset = "False" => RMC_reset:= RMC_RESET ()	WAIT_RESET_CNF
3	WAIT_RESET_CNF	/ ACM_reset = "True" && RMC_reset = "True" => ----- ACM and RMC reset completed RESET_VAR () DLM-Reset.conf { Status:= "Success" }	READY
4	READY	DLM-Set-value.req { Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) = "Valid" => SET_VALUE (Variable-name, Desired-value) DLM-Set-value.conf { Status:= "Success" }	READY
5	READY	DLM-Set-value.req { Variable-name, Desired-value } / CHECK_VALUE (Variable-name, Desired-value) <> "Valid" => DLM-Set-value.conf { Status:= "Failure" }	READY
6	READY	DLM-Get-value.req { Variable-name } / CHECK_VAR (Variable-name) = "Valid" => Current-value:= GET_CURRENT_VAL (Variable-name) DLM-Get-value.conf { Current-value:= Current-value, Status:= "Success" }	READY
7	READY	DLM-Get-value.req { Variable-name } / CHECK_VAR (Variable-name) <> "Valid" => Current-value:= NIL Status:= "Failure" DLM-Get-value.conf { Current-value:= NIL, Status:= "Failure" }	READY

#	État courant	Événement /Condition => action	État suivant
8	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) = "Valid" => SET_PC (Desired-speed-class, Desired-configuration) DLM-Set-publisher-configuration.conf { Status:= "Success" }	READY
9	READY	DLM-Set-publisher-configuration.req { Desired-speed-class, Desired-configuration } / CHECK_PC (Desired-speed-class, Desired-configuration) <> "Valid" => Status:= "Failure" DLM-Set-publisher-configuration.conf { Status }	READY
10	READY	DLM-Get-publisher-configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) = "Valid" => Current-configuration:= GET_PC (Desired-speed-class, Desired-configuration) DLM-Get-Publisher-Configuration.conf { Status:= "Success", Current-configuration:= Current-configuration }	READY
11	READY	DLM-Get-Publisher-Configuration.req { Desired-speed-class } / CHECK_SPEED (Desired-speed-class) <> "Valid" => DLM-Get-Publisher-Configuration.conf { Status:= "Failure", Current-configuration:= NIL }	READY
12	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) = "Valid" => START_TCC (Desired-speed-class) DLM-Activate-TCC.conf { Status:= "Success" }	READY
13	READY	DLM-Activate-TCC.req { Desired-speed-class } / CHECK_ATCC (Desired-speed-class) <> "Valid" => DLM-Activate-TCC.conf { Status:= "Failure" }	READY
14	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) = "Valid" => STOP_TCC (Desired-speed-class) DLM-Deactivate-TCC.conf { Status:= "Success" }	READY
15	READY	DLM-Deactivate-TCC.req { Desired-speed-class } / CHECK_DTCC (Desired-speed-class) <> "Valid" => DLM-Deactivate-TCC.conf { Status:= "Failure" }	READY

7.4.2.4 Fonction

Les fonctions DLM sont récapitulées dans le Tableau 95.

Tableau 95 – Table de fonctions DLM

Nom de la fonction	Description et fonctionnement
CHECK_PC (Desired-speed-class, Desired-configuration)	Vérifier la validité des Desired-speed-class et Desired-configuration demandées, spécifiquement pour la valeur de DLCEP-address et le nombre total de Desired-configuration. La valeur potentielle de la Desired-speed-class est "High-speed" ou "Medium-speed". Lorsque la transmission de données TCC de la Desired-speed-class demandée est active, la valeur "valide" est renvoyée, dans le cas contraire la valeur "invalid" est renvoyée
CHECK_SPEED (Desired-speed-class)	Vérifier la validité de la Desired-speed-class demandée. La valeur potentielle de la Desired-speed-class est "High-speed" ou "Medium-speed"
CHECK_VALUE (Variable-name, Desired-value)	Vérifier la validité des variables demandées avec la valeur souhaitée. Les variables potentielles avec la valeur sont définies en 4.6
CHECK_VAR (Variable-name)	Vérifier la validité des Variables demandées. Les variables potentielles sont définies en 4.6
CHECK_ATCC (Desired-speed-class)	Vérifier la validité de la Desired-speed-class demandée. La valeur potentielle de la Desired-speed-class est "High-speed" ou "Medium-speed". Lorsque la transmission de données TCC de la Desired-speed-class demandée est active, la valeur "valide" est renvoyée, dans le cas contraire la valeur "invalid" est renvoyée
CHECK_DTCC (Desired-speed-class)	Vérifier la validité de la Desired-speed-class demandée. La valeur potentielle de la Desired-speed-class est "High-speed" ou "Medium-speed". Lorsque la transmission de données TCC de la Desired-speed-class demandée est active, la valeur "valide" est renvoyée, dans le cas contraire la valeur "invalid" est renvoyée
SET_PC (Desired-speed-class, Desired-configuration)	Définir la Desired-configuration spécifiée par la Desired-speed-class
SET_VALUE (Variable-name, Desired-value)	Définir la Desired-value selon la Variable spécifiée par le Variable-name
GET_CURRENT_VAL (Variable-name)	Obtenir la valeur de la variable demandée
GET_PC (Desired-speed-class, Desired-configuration)	Obtenir la Desired-configuration spécifiée par la Desired-speed-class
RESET_VAR ()	Initialiser toutes les variables et l'ensemble de Publisher-configuration (Configuration de l'éditeur) pour chaque Speed-class (Classe de vitesse). Le service de transfert de données TCC est désactivé
ACM_RESET ()	Initialiser l'ACM. La valeur "Vrai" est renvoyée après initialisation de l'ACM
RMC_RESET ()	Initialiser la commande RMC. La valeur "Vrai" est renvoyée après initialisation de la commande RMC
START_TCC (Desired-speed)	Activer la transmission de données TCC de la Desired-speed-class demandée
STOP_TCC (Desired-speed)	Désactiver la transmission de données TCC de la Desired-speed-class demandée

Bibliographie

CEI 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

CEI 61784-2, *Réseaux de communication industriels – Profils – Partie 2: Profils de bus de terrain supplémentaires pour les réseaux en temps réel basés sur l'ISO/CEI 8802-3*

ISO/CEI 3309:1993², *Technologies de l'information – Télécommunications et échange d'informations entre systèmes – Procédures de commande de liaison de données à haut niveau (HDLC) – Structure de trame*

ISO/IEC 8802 (toutes les parties), *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements* (disponible en anglais seulement)

ISO/IEC 9314-2:1989, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 2: Token Ring Media Access Control (MAC)* (disponible en anglais seulement)

ISO/TR 13283:1998, *Automatisation industrielle – Architectures des communications en temps réel – Prescriptions des utilisateurs et gestion du réseau pour les systèmes de communications en temps réel*

² Cette publication a été annulée.

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

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