

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 4-8: Data-link layer protocol specification – Type 8 elements**

**Réseaux de communication industriels – Spécifications de bus de terrain –
Partie 4-8: Spécification du protocole de couche liaison de données – Éléments
de Type 8**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2007 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 la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

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

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

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you 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It also gives information on projects, replaced and withdrawn publications.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 on-line 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 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

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 la CEI

La Commission Electrotechnique Internationale (CEI) 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 CEI

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

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI 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.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. 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 au monde 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 les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

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.

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 4-8: Data-link layer protocol specification – Type 8 elements**

**Réseaux de communication industriels – Spécifications de bus de terrain –
Partie 4-8: Spécification du protocole de couche liaison de données – Éléments
de Type 8**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XF

ICS 35.100.20; 25.040.40

ISBN 978-2-83220-637-9

**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.....	10
1.1 General.....	10
1.2 Specifications.....	10
1.3 Procedures.....	10
1.4 Applicability.....	10
1.5 Conformance.....	11
2 Normative references	11
3 Terms, definitions, symbols and abbreviations.....	11
3.1 Reference model terms and definitions.....	11
3.2 Service convention terms and definitions.....	12
3.3 Common terms and definitions	13
3.4 Additional Type 8 definitions.....	14
3.5 Symbols and abbreviations.....	15
4 DL-protocol	17
4.1 Overview	17
4.2 DL-service Interface (DLI)	18
4.3 Peripherals data link (PDL).....	22
4.4 Basic Link Layer (BLL)	58
4.5 Medium Access Control (MAC).....	74
4.6 Peripherals network management for layer 2 (PNM2)	108
4.7 Parameters and monitoring times of the DLL	116
Annex A (informative) Implementation possibilities of definite PNM2 functions.....	122
A.1 Acquiring the current configuration	122
A.2 Comparing the acquired and stored configurations prior to a DL-subnetwork error.....	126
Bibliography.....	132
Figure 1 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses.....	13
Figure 2 – Data Link Layer Entity.....	18
Figure 3 – Location of the DLI in the DLL.....	18
Figure 4 – State transition diagram of DLI.....	20
Figure 5 – Location of the PDL in the DLL.....	22
Figure 6 – PDL connection between slave and master	23
Figure 7 – Interface between PDL-user (DLI) and PDL in the layer model.....	24
Figure 8 – Overview of the PDL services	24
Figure 9 – PDL_Data_Ack service between master and only one slave.....	26
Figure 10 – Parallel processing of PDL_Data_Ack services	26
Figure 11 – PSM and GSM service for buffer access	26
Figure 12 – Buffer_Received service to indicate successful data transfer.....	27
Figure 13 – Data flow between PDL-user, PDL and BLL of a PDL_Data_Ack service	30
Figure 14 – Interface between PDL and PNM2 in the layer model	30

Figure 15 – Reset, Set Value and Get Value PDL services	32
Figure 16 – Event PDL service.....	32
Figure 17 – Transmit and receive FCBs on the master and slave sides	35
Figure 18 – Data transmission master → slave with SWA Message	36
Figure 19 – Time sequence of the data transmission master → slave with SWA Message	36
Figure 20 – Data transmission slave → master with SWA/RWA Message.....	37
Figure 21 – Time sequence of the data transmission slave → master with SWA/RWA Message	37
Figure 22 – Allocation of actions of the PDL protocol machines and data cycles	38
Figure 23 – Message transmission: master → slave.....	39
Figure 24 – Message transmission: slave → master.....	39
Figure 25 – Code octet of a PDL PDU	40
Figure 26 – Structure of a message with a size of one word.....	41
Figure 27 – Structure of a SPA Message	41
Figure 28 – Structure of a SVA Message	42
Figure 29 – Structure of a FCB_SET Message	42
Figure 30 – Structure of a RWA Message	42
Figure 31 – Structure of a SWA Message	43
Figure 32 – Structure of a confirmation for SPA or SVA Messages.....	43
Figure 33 – Structure of a FCB_SET as confirmation	43
Figure 34 – Structure of the data octet for FCB_SET as requests and confirmations	43
Figure 35 – Structure of a message with a size of more than one word	44
Figure 36 – PDL base protocol machine.....	45
Figure 37 – Locations of the PDL and the PDL protocol machines in the master and slaves	48
Figure 38 – PDL protocol machine	49
Figure 39 – TRANSMIT protocol machine	52
Figure 40 – RECEIVE protocol machine.....	55
Figure 41 – Location of the BLL in the DLL	58
Figure 42 – Interface between PDL and BLL in the layer model	59
Figure 43 – BLL_Data service.....	60
Figure 44 – Interface between PNM2 and BLL in the layer model.....	62
Figure 45 – Reset, Set Value and Get Value BLL services	64
Figure 46 – Event BLL service	64
Figure 47 – BLL operating protocol machine of the master.....	68
Figure 48 – BLL-BAC protocol machine	70
Figure 49 – BLL operating protocol machine of the slave	73
Figure 50 – Location of the MAC in the DLL	74
Figure 51 – Model details of layers 1 and 2.....	75
Figure 52 – DLPDU cycle of a data sequence without errors	76
Figure 53 – DLPDU cycle of a data sequence with errors.....	76
Figure 54 – Data sequence DLPDU transmitted by the master	77
Figure 55 – Data sequence DLPDU received by the master	77

Figure 56 – Check sequence DLPDU	77
Figure 57 – Loopback word (LBW)	77
Figure 58 – Checksum status generated by the master	80
Figure 59 – Checksum status received by the master	80
Figure 60 – MAC protocol machine of a master: transmission of a message	81
Figure 61 – MAC protocol machine of a master: receipt of a message	84
Figure 62 – MAC sublayer of a master: data sequence identification.....	88
Figure 63 – Data sequence DLPDU received by a slave.....	91
Figure 64 – Data sequence DLPDU transmitted by a slave	91
Figure 65 – Checksum status received by the slave	91
Figure 66 – Checksum status generated by the slave	92
Figure 67 – State transitions of the MAC sublayer of a slave: data sequence.....	93
Figure 68 – State transitions of the MAC sublayer of a slave: check sequence	94
Figure 69 – Interface between MAC-user and MAC in the layer model	99
Figure 70 – Interactions at the MAC-user interface (master)	100
Figure 71 – Interactions at the MAC-user interface (slave).....	101
Figure 72 – Interface between MAC and PNM2 in the layer model	104
Figure 73 – Reset, Set Value and Get Value MAC services.....	106
Figure 74 – Event MAC service.....	106
Figure 75 – Location of the PNM2 in the DLL.....	108
Figure 76 – Interface between PNM2-user and PNM2 in the layer model	109
Figure 77 – Reset, Set Value, Get Value and Get Active Configuration services	111
Figure 78 – Event PNM2 service.....	111
Figure 79 – Set Active Configuration, Get Current Configuration service.....	111
Figure 80 – The active_configuration parameter	115
Figure 81 – Device code structure	118
Figure 82 – Relations between data width, process data channel and parameter channel.....	120
Figure 83 – Structure of the control code	121
Figure A.1 – DL-subnetwork configuration in the form of a tree structure	122
Figure A.2 – State machine for the acquisition of the current configuration	124
Figure A.3 – State machine for comparing two configurations	128
Figure A.4 – State machine for comparing one line of two configuration matrices.....	130
Table 1 – Primitives issued by DLS-/DLMS-user to DLI	19
Table 2 – Primitives issued by DLI to DLS-/DLMS-user.....	19
Table 3 – DLI state table – sender transactions	20
Table 4 – DLI state table – receiver transactions	21
Table 5 – Function GetOffset	22
Table 6 – Function GetLength.....	22
Table 7 – Function GetRemAdd	22
Table 8 – Function GetDIsUserId	22
Table 9 – PDL_Data_Ack.....	27

Table 10 – PDL_Data_Ack L_status values.....	27
Table 11 – PSM.....	28
Table 12 – GSM.....	28
Table 13 – PDL_Reset.....	32
Table 14 – PDL_Set_Value.....	32
Table 15 – PDL variables.....	33
Table 16 – PDL_Get_Value.....	33
Table 17 – PDL_Event.....	34
Table 18 – Events.....	34
Table 19 – Encoding of the L_status.....	40
Table 20 – FCT code (PDL PDU-Types).....	40
Table 21 – State transitions of the PDL base protocol machine.....	46
Table 22 – Counters of the PDL protocol machines.....	48
Table 23 – Meaning of the "connection" flag.....	49
Table 24 – State transitions of the PDL protocol machine.....	50
Table 25 – State transitions of the TRANSMIT protocol machine.....	53
Table 26 – State transitions of the RECEIVE protocol machine.....	55
Table 27 – BLL_Data.....	61
Table 28 – BLL_Data.....	64
Table 29 – BLL_Reset.....	65
Table 30 – BLL_Set_Value.....	65
Table 31 – BLL variables.....	66
Table 32 – BLL_Get_Value.....	66
Table 33 – BLL_Event.....	66
Table 34 – BLL_Event.....	67
Table 35 – State transitions of the BLL operating protocol machine of the master.....	69
Table 36 – State transitions of the BLL-BAC protocol machine.....	71
Table 37 – State transitions of the BLL operating protocol machine of the slave.....	73
Table 38 – FCS length and polynomial.....	78
Table 39 – MAC_Reset.....	106
Table 40 – MAC_Set_Value.....	106
Table 41 – MAC variables.....	107
Table 42 – MAC_Get_Value.....	107
Table 43 – MAC_Event.....	107
Table 44 – MAC_Event.....	108
Table 45 – PNM2_Reset.....	112
Table 46 – M_status values of the PNM2_Reset.....	112
Table 47 – PNM2_Set_Value.....	112
Table 48 – M_status values of the PNM2_Set_Value.....	113
Table 49 – PNM2_Get_Value.....	113
Table 50 – M_status values of the PNM2_Get_Value.....	113
Table 51 – PNM2_Event.....	114
Table 52 – MAC Events.....	114

Table 53 – PNM2_Get_Current_Configuration	114
Table 54 – PNM2_Get_Active_Configuration	115
Table 55 – PNM2_Set_Active_Configuration.....	116
Table 56 – Data direction.....	118
Table 57 – Number of the occupied octets in the parameter channel.....	119
Table 58 – Device class	119
Table 59 – Control data	119
Table 60 – Data width.....	120
Table 61 – Medium control.....	121
Table A.1 – DL-subnetwork configuration in the form of a matrix.....	123
Table A.2 – Acquire_Configuration.....	123
Table A.3 – State transitions of the state machine for the acquisition of the current configuration.....	125
Table A.4 – Check_Configuration.....	126
Table A.5 – Compare_Slave	127
Table A.6 – State transitions of the state machine for comparing two configurations	129
Table A.7 – State transitions of the state machine for comparing one line of two configuration matrixes.....	131

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 4-8: Data-link layer protocol specification –
Type 8 elements**

FOREWORD

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

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

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

Type 8 and possibly other Types:

DE 41 00 629 C1 [PxC] Steuer- und Datenübertragungsanlage

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

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

[PxC]: Phoenix Contact GmbH & Co. KG
Referat Patente / Patent Department
Postfach 1341
D-32819 Blomberg
Germany

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

International Standard IEC 61158-4-8 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

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

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

- a) deletion of the former Type 6 fieldbus, and the placeholder for a Type 5 fieldbus data link layer, for lack of market relevance;
- b) addition of new types of fieldbuses;
- c) division of this part into multiple parts numbered -4-1, -4-2, ..., -4-19.

This bilingual version (2013-02) corresponds to the monolingual English version, published in 2007-12.

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

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

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

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

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

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

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

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

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

INTRODUCTION

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

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

- a) as a guide for implementors and designers;
- b) for use in the testing and procurement of equipment;
- c) as part of an agreement for the admittance of systems into the open systems environment;
- d) as a refinement to the understanding of time-critical communications within OSI.

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-8: Data-link layer protocol specification – Type 8 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 a highly-optimized means of interchanging fixed-length input/output data and variable-length segmented messages between a single master device and a set of slave devices interconnected in a loop (ring) topology. The exchange of input/output data is totally synchronous by configuration, and is unaffected by the messaging traffic.

Devices are addressed implicitly by their position on the loop. The determination of the number, identity and characteristics of each device can be configured, or can be detected automatically at start-up.

1.2 Specifications

This standard specifies

- a) procedures for the timely transfer of data and control information from one data-link user entity to a peer user entity, and among the data-link entities forming the distributed data-link service provider;
- b) the structure of the fieldbus DLPDUs used for the transfer of data and control information by the protocol of this standard, and their representation as physical interface data units.

1.3 Procedures

The procedures are defined in terms of

- a) the interactions between peer DL-entities (DLEs) through the exchange of fieldbus DLPDUs;
- b) the interactions between a DL-service (DLS) provider and a DLS-user in the same system through the exchange of DLS primitives;
- c) the interactions between a DLS-provider and a Ph-service provider in the same system through the exchange of Ph-service primitives.

1.4 Applicability

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

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

1.5 Conformance

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

2 Normative references

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

IEC 61158-2 (Ed.4.0), *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-8, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 3-8: Data link service definition – Type 8 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 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 standard, the following terms, definitions, symbols and abbreviations apply.

3.1 Reference model terms and definitions

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

3.1.1 DL-address	[7498-3]
3.1.2 DL-address-mapping	[7498-1]
3.1.3 DL-connection	[7498-1]
3.1.4 DL-connection-end-point	[7498-1]
3.1.5 DL-connection-end-point-identifier	[7498-1]
3.1.6 DL-data-source	[7498-1]
3.1.7 DL-name	[7498-3]
3.1.8 DL-protocol	[7498-1]
3.1.9 DL-protocol-connection-identifier	[7498-1]
3.1.10 DL-protocol-control-information	[7498-1]
3.1.11 DL-protocol-data-unit	[7498-1]

3.1.12 DL-service-connection-identifier	[7498-1]
3.1.13 DL-service-data-unit	[7498-1]
3.1.14 DL-user-data	[7498-1]
3.1.15 layer-management	[7498-1]
3.1.16 (N)-entity DL-entity Ph-entity	[7498-1]
3.1.17 (N)-interface-data-unit DL-service-data-unit (N=2) Ph-interface-data-unit (N=1)	[7498-1]
3.1.18 (N)-layer DL-layer (N=2) Ph-layer (N=1)	[7498-1]
3.1.19 (N)-service DL-service (N=2) Ph-service (N=1)	[7498-1]
3.1.20 (N)-service-access-point DL-service-access-point (N=2) Ph-service-access-point (N=1)	[7498-1]
3.1.21 (N)-service-access-point-address DL-service-access-point-address (N=2) Ph-service-access-point-address (N=1)	[7498-1]
3.1.22 Ph-interface-control-information	[7498-1]
3.1.23 Ph-interface-data	[7498-1]
3.1.24 primitive name	[7498-3]
3.1.25 reset	[7498-1]
3.1.26 systems-management	[7498-1]

3.2 Service convention terms and definitions

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

- 3.2.1 confirm (primitive);**
requestor.deliver (primitive)
- 3.2.2 DL-service-primitive;**
primitive
- 3.2.3 DL-service-provider**
- 3.2.4 DL-service-user**
- 3.2.5 indication (primitive)**
acceptor.deliver (primitive)

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

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

3.3 Common terms and definitions

NOTE This subclause contains the common terms and definitions used by Type 8.

3.3.1

link, local link

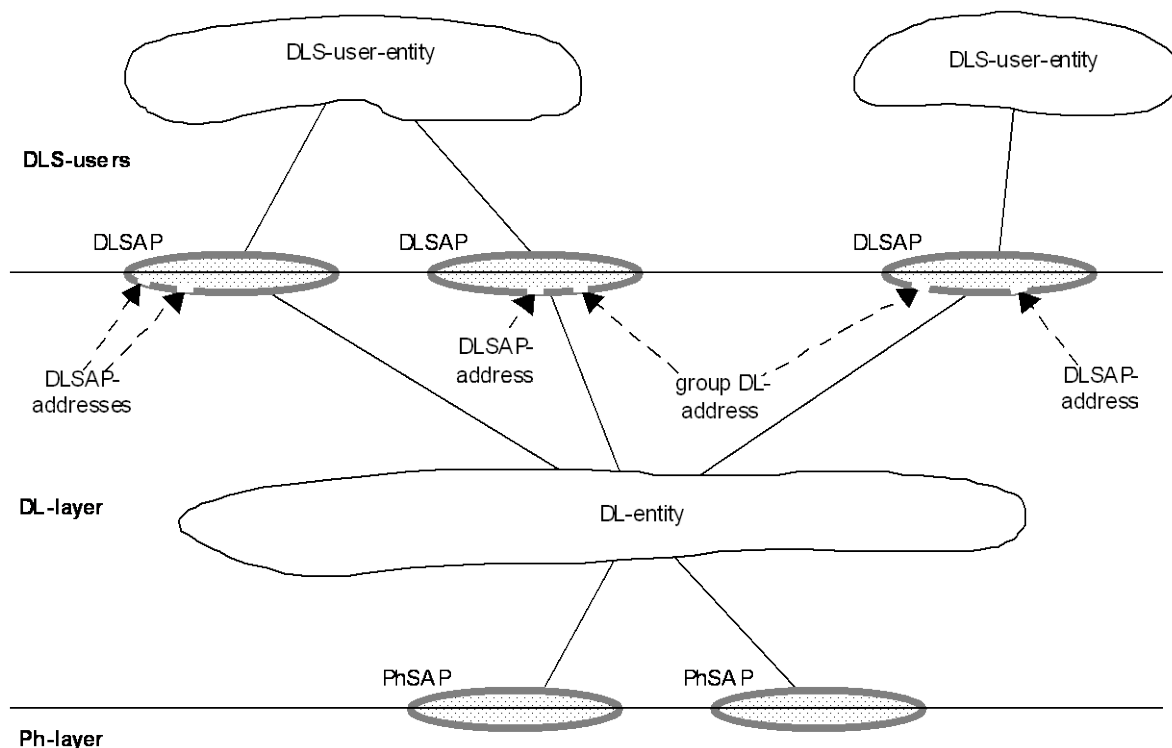
single DL-subnetwork in which any of the connected DLEs may communicate directly, without any intervening DL-relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication

3.3.2

DLSAP

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

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



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

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

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

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

3.3.3

DL(SAP)-address

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

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

3.3.4

extended link

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

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

3.3.5

frame

denigrated synonym for DLPDU

3.3.6

receiving DLS-user

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

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

3.3.7

sending DLS-user

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

3.4 Additional Type 8 definitions

3.4.1 bus coupler

PhL entity which includes or excludes Ph-segments into or from the network

3.4.2 device

slave or master

3.4.3 device code

two octets which characterize the properties of a slave

3.4.4 DLPDU cycle

transaction initiated from the master in which user data or identification/status information is sent to all slaves and – within the same cycle - received from all slaves

3.4.5 IN data

data received by the master and sent by the slaves

3.4.6 master

DL-entity controlling the data transfer on the network and initiating the media access of the slaves by starting the DLPDU cycle

3.4.7 OUT data

data sent by the master and received by the slaves

3.4.8 parameter channel

acyclic transmission path using a client/server communication model

3.4.9 process data channel

conveyance path allowing a very efficient, high-speed and cyclic transmission of process-relevant data, between slaves and master

3.4.10 receive update memory

memory area containing the data, which was received from the network

3.4.11 ring segment

group of slaves in consecutive order

3.4.12 ring segment level

nesting level number of a ring segment

3.4.13 slave

DL-entity accessing the medium only after being initiated by the preceding slave or master

3.4.14 transmit update memory

memory area containing the data to be sent across the network

3.4.15 update time

time which passes between two consecutive starts of DLPDU cycles used for data transfer

3.5 Symbols and abbreviations

3.5.1 Type 8 reference model terms

BLL	basic link layer
BLLSDU	BLL service data unit
BLL_TSDU	BLL transmit service data unit
BLL_RSDU	BLL receive service data unit
MACSDU	MAC service data unit
PDL	peripherals data link
PDLSDU	PDL service data unit
PhMS	Ph-management service

3.5.2 Local variables, timers, counters and queues

add_wait	See Table 15
BLL_access_control	See Table 31
bus_timeout	See Table 31
Ccerr	See Table 22
Cconf	See Table 22
Ccycle	See Table 22
Creq_retry	See Table 22
Cswa	See Table 22
configuration_valid	See Table 31
loopback_word (LBW)	See Table 41
max_dlsdu_size_from_req	See Table 15
max_dlsdu_size_from_res	See Table 15
max_receiving_queue_depth	See Table 15
max_sending_queue_depth	See Table 15
max_spa_retry	See Table 15
max_swa_count	See Table 15
start_bus_cycle	See Table 15
time_timeout	See Table 41
trigger_mode	See Table 15
update_time	See Table 31

3.5.3 DLPDU classes

DATA	data	See Table 20
FCB_SET	frame count bit	See Table 20
IDL	idle	See Table 20
RWA	read word again	See Table 20
SPA	send parameter with acknowledge	See Table 20
SVA	send value with acknowledge	See Table 20
SWA	send word again	See Table 20

3.5.4 Miscellaneous

AT	application triggered
BAC	basic access control

CO	confirmation
CRC	cyclic redundancy check
DL-Ph	Data Link-Physical (interface)
DLI	DL-interface
DSAP	destination service access point
FCB	frame count bit
FCT	function
FMS	fieldbus message specification
GSM	get shared memory
IN	input
L_status	link status
LBW	loopback word
lsb	least significant bit
M, (m)	mandatory
msb	most significant bit
NT	network triggered
O, (o)	optional
OUT	output
PDL	peripherals data Link
PM	protocol machine
PNM1	peripherals network management of Layer 1
PNM2	peripherals network management of Layer 2
PSM	put shared memory
RUM	receive update memory
S	selection
SM	state machine
TUM	transmit update memory

4 DL-protocol

4.1 Overview

The DLL is modelled as a Four-Level model (see Figure 2).

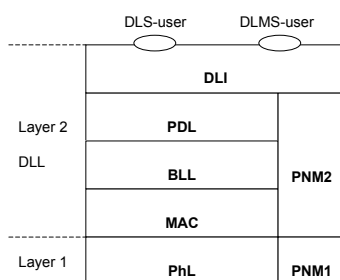


Figure 2 – Data Link Layer Entity

4.2 DL-service Interface (DLI)

4.2.1 General

The Data Link service Interface (DLI) provides service primitives to the DLS-user and DLMS-user (see Figure 3).

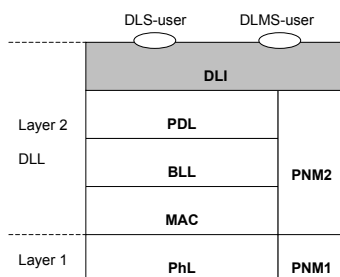


Figure 3 – Location of the DLI in the DLL

The DLI translates and issues the primitives received from the DLS-/DLMS-user to the local PDL and PNM2 interface. It also translates and issues the primitives received from the local PDL or PNM2 interface and delivers it to the DLS-/DLMS-user.

The DLI protocol has only a single state called “ACTIVE”.

4.2.2 Primitive definitions

4.2.2.1 General

Table 1 and Table 2 show the primitives exchanged between DLS-/DLMS-user and DLI.

4.2.2.2 Primitives exchanged between DLS-/DLMS-user and DLI

Table 1 – Primitives issued by DLS-/DLMS-user to DLI

Primitive name	Source	Associated parameters	Functions
DL-PUT request	DLS-user	Buffer DL-identifier, DLS-user-data	Requests the DLE to write a DLSDU into the transmit buffer
DL-GET request	DLS-user	Buffer DL-identifier	Requests the DLE to read a DLSDU from the receive buffer
DL-DATA request	DLS-user	DLCEP DL-identifier, DLS-user-data	Requests the DLE to write a DLSDU into the send queue
DLM-RESET request	DLS-user	(<none>)	Requests the DLE to execute a reset.
DLM-SET-VALUE request	DLMS-user	Variable-name, Desired-value	Requests the DLE to overwrite a local variable
DLM-GET-VALUE request	DLMS-user	Variable-name	This Primitive is issued to request the DLL to read the content of a local variable
DLM-GET-CURRENT-CONFIGURATION request	DLMS-user	Desired-configuration	Requests the DLE to read out the current configuration of the DL-subnetwork.
DLM-GET-ACTIVE-CONFIGURATION request	DLMS-user	(<none>)	Requests the DLE to read out the active configuration of the DL-subnetwork
DLM-SET_ACTIVE_CONFIGURATION request	DLMS-user	Active-configuration	Requests the DLE to execute a certain active configuration of the DL-subnetwork

Table 2 – Primitives issued by DLI to DLS-/DLMS-user

Primitive name	Source	Associated parameters
DL-PUT confirm	DLI	Status
DL-GET confirm	DLI	Status, DLS-user-data
DL-BUFFER-RECEIVED indication	DLI	Status
DL-DATA confirm	DLI	Status
DL-DATA indication	DLI	DLCEP DL-identifier, DLS-user-data
DLM-RESET confirm	DLI	Status
DLM-EVENT indication	DLI	Event-identifier, Additional-information
DLM-SET-VALUE confirm	DLI	Status
DLM-GET-VALUE confirm	DLI	Status, Additional-information
DLM-GET-CURRENT-CONFIGURATION confirm	DLI	Status, Additional-information
DLM-GET-ACTIVE-CONFIGURATION confirm	DLI	Status, Additional-information
DLM-SET-ACTIVE-CONFIGURATION confirm	DLI	Status, Additional-information

4.2.2.3 Parameters of DLS-/DLMS-User and DLI primitives

All parameters used in the primitives exchanged between the DLS-/DLMS-user and the DLI are specified in IEC 61158-3-8.

4.2.3 DLI State Tables

4.2.3.1 General

Figure 4 show a state transition diagram of the DLI.

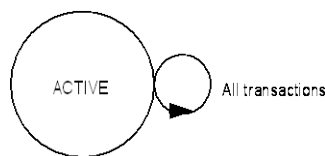


Figure 4 – State transition diagram of DLI

The transitions of the DLI protocol are specified in Table 3 and Table 4. Service primitive names are mixed-case with underscores (“_”) replacing dashes (“-”), and with a dot-separated suffix indicating the underlying type of primitive: request, confirm or indication.

Table 3 – DLI state table – sender transactions

#	Current state	Event Action	Next state
S1	ACTIVE	DL_Put.request PSM.request{ offset := GetOffset(Buffer DL-identifier) length := “length of DLS-user-data” data := DLS-user-data }	ACTIVE
S2	ACTIVE	DL_Get.request GSM.request{ offset := GetOffset(Buffer DL-identifier) length := GetLength(Buffer DL-identifier) }	ACTIVE
S3	ACTIVE	DL_Data.request{ PDL_Data_Ack.request{ rem_add := GetRemAdd (DLCEP DL-identifier) DLSDU := DLS-user-data }	ACTIVE
S4	ACTIVE	DLM_Reset.request PNM2_Reset.request{ }	ACTIVE
S5	ACTIVE	DLM_Set_Value.request PNM2_Set_Value.request { variable_name := Variable-name, desired_value := Desired-value }	ACTIVE
S6	ACTIVE	DLM_Get_Value.request PNM2_Get_Value.request{ variable_name := Variable-name }	ACTIVE
S7	ACTIVE	DLM_Get_Current_Configuration.request PNM2_Get_Current_Configuration.request{ network_configuration := Desired Configuration }	ACTIVE
S8	ACTIVE	DLM_Get_Active_Configuration.request PNM2_Get_Active_Configuration.request{ }	ACTIVE
S9	ACTIVE	DLM_Set_Active_Configuration.request PNM2_Set_Active_Configuration.request{ active_configuration := Active-configuration }	ACTIVE

Table 4 – DLI state table – receiver transactions

#	Current state	Event Action	Next state
R1	ACTIVE	PSM.confirm DL_Put.confirm{ Status := status }	ACTIVE
R2	ACTIVE	GSM.confirm DL_Get.confirm{ Status := status, DLS-user-data := data }	ACTIVE
R3	ACTIVE	Buffer_Received.indication DL_Buffer_Received.indication{ Status := status }	ACTIVE
R4	ACTIVE	PDL_Data_Ack.confirm DL_Data.confirm{ Status := L_status }	ACTIVE
R5	ACTIVE	PDL_Data_Ack.indication DL_Data.indication{ DLCEP DL-identifier := GetDlsUserId(local_add), DLS-user-data := DLSDU }	ACTIVE
R6	ACTIVE	PNM2_Reset.confirm DLM_Reset.confirm{ Status := M_status }	ACTIVE
R7	ACTIVE	PNM2_Event.indication DLM_Event.indication { Event-identifier := event, Additional-information := add_info }	ACTIVE
R8	ACTIVE	PNM2_Set_Value.confirm DLM_Set_Value.confirm { Status := M_status }	ACTIVE
R9	ACTIVE	PNM2_Get_Value.confirm DLM_Get_Value.confirm{ Status := M_status Current-Value := current_value }	ACTIVE
R10	ACTIVE	PNM2_Get_Current_Configuration.confirm DLM_Get_Current_Configuration.confirm{ Status := status, Current-configuration := current_configuration }	ACTIVE
R11	ACTIVE	PNM2_Get_Active_Configuration.confirm DLM_Get_Active_Configuration.confirm{ Status := status, Active-configuration := active_configuration }	ACTIVE
R12	ACTIVE	PNM2_Set_Active_Configuration.confirm DLM_Set_Active_Configuration.confirm{ Status := status, Additional-information := add_info }	ACTIVE

4.2.3.2 Functions used by DLI

The functions used by DLI are given in Table 5 to Table 8. The details of these functions is not specified by of this standard. These functions use information which was stored by the local DL-management when establishing the DLCs.

Table 5 – Function GetOffset

Name	GetOffset	Used in	DLI
Input	Buffer DL-identifier	Output	Offset address
Function	Returns a value that can unambiguously identify the offset address from the transmit buffer		

Table 6 – Function GetLength

Name	GetLength	Used in	DLI
Input	Buffer DL-identifier	Output	Length of data
Function	Returns the size of the DLSDU which can be held by the buffer named by Buffer DL-identifier.		

Table 7 – Function GetRemAdd

Name	GetRemAdd	Used in	DLI
Input	DLCEP DL-identifier	Output	Remote address
Function	Returns a value that can unambiguously identify the remote address from the remote device		

Table 8 – Function GetDisUserId

Name	GetDisUserId	Used in	DLI
Input	Local address	Output	DLCEP DL-identifier
Function	Returns a value that can unambiguously identify the DLCEP DL-identifier from the DLS user		

4.3 Peripherals data link (PDL)

4.3.1 Location of the PDL in the DLL

The Peripherals Data Link (PDL) is part of the Data Link Layer and uses the Basic Link Layer. Figure 5 shows its location.

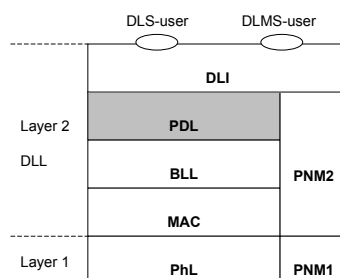


Figure 5 – Location of the PDL in the DLL

By means of the PDL layer each slave can establish a communication link with the master (see Figure 6).

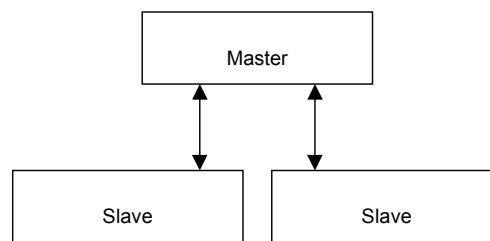


Figure 6 – PDL connection between slave and master

4.3.2 Functionality of the PDL

The PDL performs the following tasks.

- Processing of PDL_Data_Ack service
- Conversion of the non-cyclic PDL_Data_Ack service to cyclic BLL_Data services and vice versa
- Conversion of several DLSDUs of the PDL_Data_Ack.request primitives into a PDLSDU of the BLL_Data.request primitive
- Implementation of two trigger_modes within the PDL (*bus master only*)
- Control of the local PDL protocol machine(s)
- Update of the receive update memory and starting of the PDL protocol machines after a PDLSDU which was received from the BLL has been accepted
- Generation of a PDLSDU from the transmit update memory as well as by means of the PDL protocol machines and transfer of this PDLSDU to be sent to the BLL
- Implementation of a direct access for PDL-user to the PDL receive and transmit update memory.

NOTE A PDLSDU of the master contains all cyclic data via PSM service to be transmitted in a data cycle and PDL message segments. The PDLSDU of a slave is a subset of the PDLSDU of the master and contains only the cyclic data to be transmitted in one data cycle and the PDL message segment of this slave.

The PDL translates these functions by means of the four following protocol machines.

- PDL base protocol machine
- PDL protocol machine
- TRANSMIT protocol machine
- RECEIVE protocol machine.

4.3.3 DLI-PDL interface

4.3.3.1 General

The PDL provides service primitives for the PDL-user (see Figure 7).

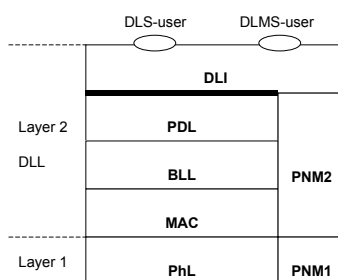


Figure 7 – Interface between PDL-user (DLI) and PDL in the layer model

4.3.3 describes the data transmission services which are available to the PDL-user, together with their service primitives and their associated parameters. These PDL services are mandatory.

4.3.3.2 Overview of the services

4.3.3.2.1 Available services

The following service for data transfer shall be available to the PDL-user:

- Send Parameter with Acknowledge (PDL_Data_Ack).

Furthermore, the PDL-user can use the following services to directly access the update memory.

- **P**ut **S**hared **M**emory (PSM)
- **G**et **S**hared **M**emory (GSM).

Figure 8 shows an overview of the services of the PDL.

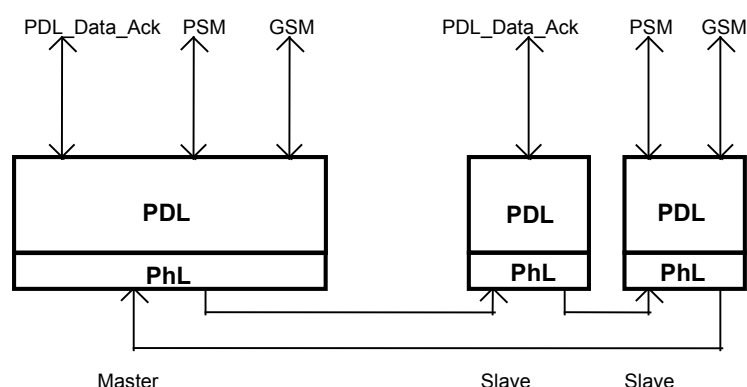


Figure 8 – Overview of the PDL services

4.3.3.2.2 Send parameter with acknowledge (PDL_Data_Ack)

This service allows a local PDL-user to send user data (DLSDU) to a single remote PDL-user. The remote PDL transfers the DLSDU to its PDL-user, provided that the DLSDU was received without errors. The local PDL-user receives a confirmation on the receipt or non-receipt of the DLSDU of the remote PDL.

The PDL_Data_Ack service shall only be used to transfer data from a queue.

Service primitives:

- PDL_Data_Ack.request
- PDL_Data_Ack.indication
- PDL_Data_Ack.confirm

4.3.3.2.3 Put shared memory (PSM)

This service allows a PDL-user to write data of a certain length into the transmit update memory. The BLL shall transmit this data in the next bus cycle.

Service primitives:

- PSM.request
- PSM.confirm

4.3.3.2.4 Get shared memory (GSM)

This service allows a PDL-user to read data of a certain length from the receive update memory.

Service primitives:

- GSM.request
- GSM.confirm

4.3.3.2.5 Buffer received (Buffer_Received)

The PDL uses this service to indicate the local PDL-user, that the contents of

Transmit Update Memory is transmitted, and the contents of
Receive Update Memory is updated with new received data.

Service primitive:

Buffer_Received.indication

4.3.3.3 Overview of the interactions

The services are provided by several service primitives (beginning with PDL_...). In order to request a service, the PDL-user uses a request primitive. A confirmation primitive is returned to the PDL-user after the service has been completed. The arrival of a service request is indicated to the remote PDL-user by means of an indication primitive.

Figure 9, Figure 10, Figure 11 and Figure 12 show the sequences of service primitives to handle the data transfer between master and slave:

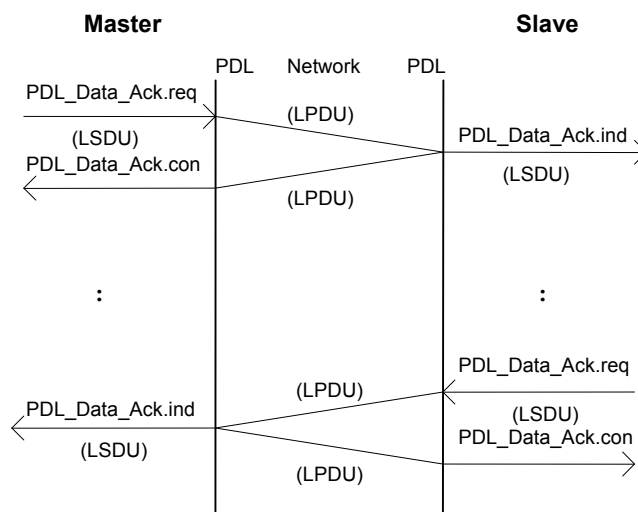


Figure 9 – PDL_Data_Ack service between master and only one slave

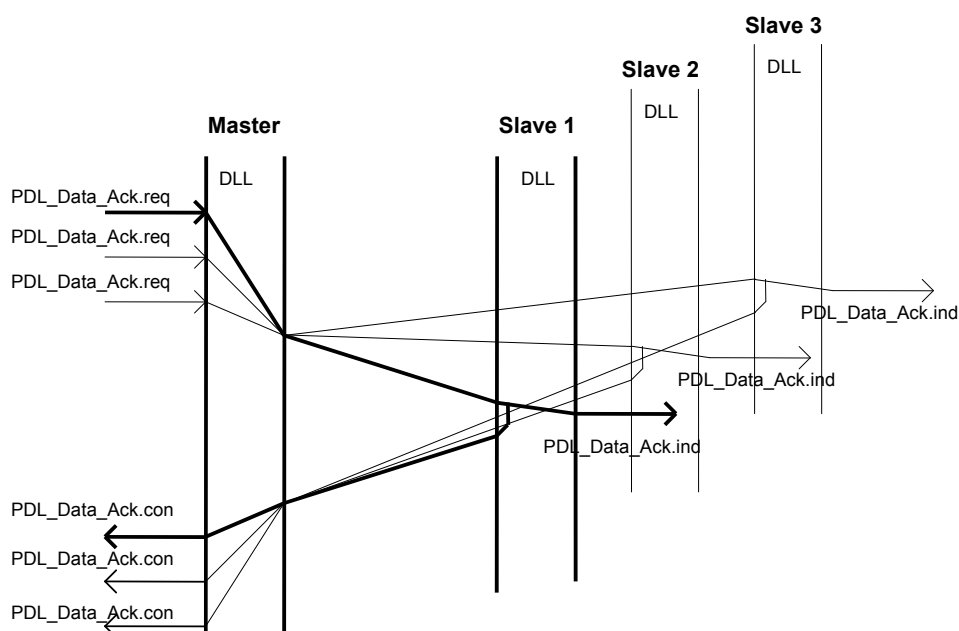


Figure 10 – Parallel processing of PDL_Data_Ack services

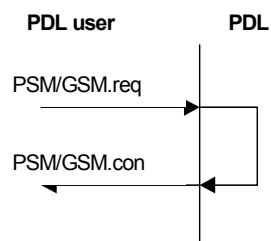


Figure 11 – PSM and GSM service for buffer access

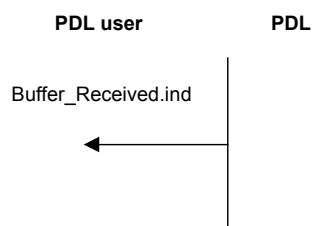


Figure 12 – Buffer_Received service to indicate successful data transfer

4.3.3.4 Formal description of the services and parameters

4.3.3.4.1 PDL_Data_Ack Service

Table 9 shows the parameters of the PDL_Data_Ack service.

Table 9 – PDL_Data_Ack

Parameter name	Request	Indication	Confirm
Argument	M	M	
rem_add	M		
local_add		M	
DLSDU	M	M(=)	
Result			M
rem_add			M
L_status			M

rem_add:

The rem_add parameter defines the PDL address of the remote device. The rem_add corresponds to the physical position of the device in the ring.

local_add:

The local_add parameter conveys the PDL address of the device where the PDL_Data_Ack service was invoked.

DLSDU:

The DLSDU parameter contains the PDL-user data to be transmitted.

L_status:

The L_status parameter indicates the success or failure of the preceding PDL_Data_Ack.request. The following values are defined for this parameter in Table 10:

Table 10 – PDL_Data_Ack L_status values

Value	Meaning
OK	Positive acknowledgement, service executed successfully
RR	Negative acknowledgement, resources of the remote PDL not available or insufficient
LR	Resources of the local PDL not available or insufficient
NA	No or not a plausible response (acknowledge response) from the remote device
DS	PDL layer not synchronized at the moment
IV	Invalid parameter in the request call

4.3.3.4.2 PSM service

Table 11 shows the parameters of the PSM service.

Table 11 – PSM

Parameter name	Request	Confirm
Argument offset length data	M M M M	
Result(+)		S
Result(-) error_type		S M

offset:

This parameter specifies the offset address, beginning from the start address of the PDL transmit update memory, where the data should be written.

length:

This parameter specifies the amount of the data, which should be written into the PDL transmit update memory of layer 2.

data:

This parameter conveys the data, which should be written into the PDL transmit update memory of the layer 2.

error_type:

This parameter indicates the reason, why the service could not be executed successfully.

Possible errors are:

- IV Invalid parameters in the request call
Data to write into the transmit update memory are not allowed, because the given parameter(s) of offset and/or length is/are invalid.

4.3.3.4.3 GSM service

Table 12 shows the parameters of the GSM service.

Table 12 – GSM

Parameter name	Request	Confirm
Argument offset length	M M M	
Result(+) data		S M
Result(-) error_type		S M

offset:

This parameter specifies the offset address, beginning from the start address of the PDL receive update memory, from where the data should be read.

length:

This parameter specifies the amount of the data, which should be read from the PDL receive update memory.

data:

This parameter conveys the data, which was read from the PDL receive update memory.

error_type:

This parameter indicates the reason, why the service could not be executed successfully. Possible error sources:

- IV Invalid parameters in the request call
Data to be read from the receive update memory are not allowed, because the given parameter(s) of offset and/or length is/are invalid.

4.3.3.5 Detailed description of the interactions

4.3.3.5.1 Send parameter with acknowledge (PDL_Data_Ack)

The local PDL-user prepares a DLSDU which is transmitted by a PDL_Data_Ack.request primitive to the local PDL. The PDL accepts this service request and tries to send the DLSDU to the requested remote PDL. The local PDL sends a confirmation to its PDL-user with the PDL_Data_Ack.confirm primitive, which indicates a correct or incorrect data transfer.

Before the local PDL sends a confirmation to its user, a confirmation from the remote PDL is mandatory. If this confirmation is not received within the timeout period $T_{TO_SPA_ACK}$, the local PDL retries to send the DLSDU to the remote PDL. If the confirmation does not come after the Nth repetition (max_retry_count), then the local PDL sends a negative confirmation to its user.

If the data message was received without errors, the remote PDL transfers the DLSDU with a PDL_Data_Ack.indication primitive through the PDL-user interface.

The coding of the DLSDU is described in 4.3.5.3. Figure 13 shows the data flow between PDL-user, PDL and BLL for a PDL_Data_Ack service:

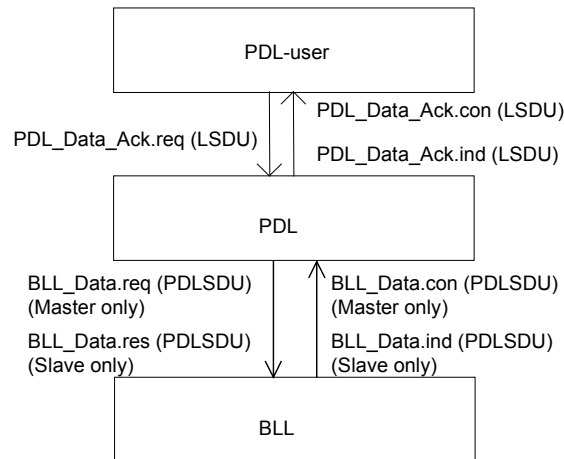


Figure 13 – Data flow between PDL-user, PDL and BLL of a PDL_Data_Ack service

4.3.3.5.2 Put shared memory (PSM)

The PDL-user uses this service to write user data directly to the transmit update memory. The service is locally processed after the PSM.request primitive has arrived. The PDL communicates the successful processing of the service to its PDL-user by means of a PSM.confirm primitive (immediate confirmation).

4.3.3.5.3 Get shared memory (GSM)

The PDL-user uses this service to read user data directly from the PDL receive update memory. The service is locally processed after the GSM.request primitive has arrived. The PDL communicates the successful processing of the service to the PDL-user by means of a GSM.confirm primitive (immediate confirmation).

4.3.4 PDL-PNM2 interface

4.3.4.1 General

This subclause defines the administrative PDL management services which are available to the PNM2, together with their service primitives and the associated parameters.

The PDL management is a part of the PDL that provides the management functions of the PDL requested by the PNM2. The PDL management handles the initialization, monitoring and error recovery in the PDL. Figure 14 shows the interface between PDL and PNM2 in the layer model.

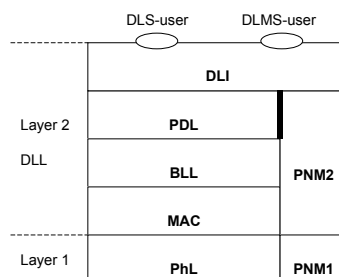


Figure 14 – Interface between PDL and PNM2 in the layer model

The service interface between PDL and PNM2 provides the following functions.

- Reset of the PDL protocol machine.
- Request and change of the current operating parameters of the PDL protocol machine.
- Indication of unexpected events, errors and status changes which occurred or are detected in the PDL.

4.3.4.2 Overview of the services

4.3.4.2.1 Available services

The PDL makes the following services available to the PNM2:

- Reset PDL,
- Set Value PDL or Get Value PDL,
- Event PDL.

The PDL services are described with service primitives (beginning with PDL_...).

4.3.4.2.2 Reset PDL

The PNM2 uses this required service to reset the PDL. Upon execution, the PNM2 receives a confirmation.

Service primitives:

- PDL_Reset.request
- PDL_Reset.confirm

4.3.4.2.3 Set Value PDL

The PNM2 uses this optional service to set new values to the PDL variables. Upon completion, the PNM2 receives a confirmation from the PDL whether the defined variables are assumed with the new value.

Service primitives:

- PDL_Set_Value.request
- PDL_Set_Value.confirm

4.3.4.2.4 Get Value PDL

The PNM2 uses this optional service to read the actual value of the PDL variables. The current value of the defined variable is transmitted with the confirmation from the PDL.

Service primitives:

- PDL_Get_Value.request
- PDL_Get_Value.confirm

4.3.4.2.5 Event PDL

The PDL uses this required service to inform the PNM2 about certain detected events or errors in the PDL.

Service primitive:

- PDL_Event.indication

4.3.4.3 Overview of the interactions

Figure 15 and Figure 16 show the time relations of the service primitives:

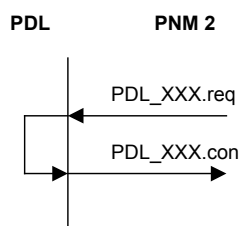


Figure 15 – Reset, Set Value and Get Value PDL services

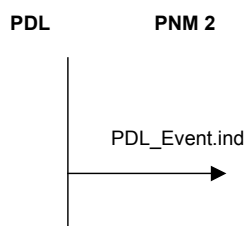


Figure 16 – Event PDL service

4.3.4.4 Detailed definition of the services and interactions

4.3.4.4.1 PDL_Reset

The PDL_Reset service is mandatory. The PNM2 transmits a PDL_Reset.request primitive to reset the PDL protocol machine (see Table 13).

Table 13 – PDL_Reset

Parameter name	Request	Confirm
Argument	M	
Result(+)		M

4.3.4.4.2 PDL_Set_Value

The PDL_Set_Value service is optional. The PNM2 transfers a PDL_Set_Value.request primitive to the PDL to set a defined PDL variable with a desired value. After receipt of this primitive, the PDL tries to select the variable and to set the new value. Upon execution, the PDL transfers a PDL_Set_Value.confirm primitive to the PNM2 (see Table 14).

Table 14 – PDL_Set_Value

Parameter name	Request	Confirm
Argument	M	
variable_name	M	
desired_value	M	
Result(+)		M

variable_name:

This parameter defines the PDL variable which is set to a new value.

desired_value:

This parameter declares the new value for the PDL variable.

Table 15 provides information on which PDL variable may be set to which new value.

Table 15 – PDL variables

Name of PDL variable	Value range	Default
max_spa_retry	0,2,4,6, ... 14	14
max_swa_count	0 ... 255	128
add_wait	1, 2, 3, 4	4
start_bus_cycle	ON, OFF	OFF
trigger_mode	network_triggered (NT), application_triggered (AT)	NT
max_dlsdu_size_from_req	1 ... 256 (see note)	256
max_dlsdu_size_from_res	1 ... 256 (see note)	256
max_receiving_queue_depth	1 ... 256 (see note)	256
max_sending_queue_depth	1 ... 256 (see note)	256
NOTE Only for PDL_Data_Ack services and each link.		

4.3.4.4.3 PDL_Get_Value

The PDL_Get_Value service is optional. The PNM2 transfers a PDL_Get_Value.request primitive to the PDL to read out the current value of a defined PDL variable. After the PDL has received this primitive, it tries to select the defined variable and to transfer its current value to the PNM2 by means of a PDL_Get_Value.confirm primitive (see Table 16).

Table 16 – PDL_Get_Value

Parameter name	Request	Confirm
Argument variable_name	M M	
Result(+) current_value		M M

variable_name:

This parameter defines the PDL variable, whose value should be read.

current_value:

This parameter contains the desired value of this PDL variable.

Only those PDL variables can be read, which can also be written by the service PDL_Set_Value.request.

4.3.4.4.4 PDL_Event

The PDL_Event service is mandatory. The PDL transfers a PDL_Event.indication primitive to the PNM2 to inform it about detected events or errors in the PDL (see Table 17).

Table 17 – PDL_Event

Parameter name	Indication
Argument event	M M

event:

This parameter defines the value of the detected event or error cause in the PDL according to Table 18:

Table 18 – Events

Name	Meaning	Mandatory/optional
PDL_cycle_end	The receive update memory was updated, and the contents of the transmit update memory are transmitted to the BLL	O

4.3.5 Data transfer procedures from a queue

4.3.5.1 Bus access and data transfer mechanism

4.3.5.1.1 Synchronization cycle

Before starting of data transfer between the master and the slave(s), the PDL layers on all devices shall start with a synchronization cycle. In this cycle, a synchronization message resets the frame count bit flags in all devices to a defined value. In addition, the master started with the transmit of configure data to all slaves. After receiving the new configure data, all slaves shall initialize themselves with the new received configure values.

The frame count bits prevent a multiplication of messages at the confirming and/or responding device (responder), as these would cause the loss of positive acknowledgements.

A synchronization cycle only takes place for one communication relationship, that is, between the PDL protocol machine of the master and the PDL protocol machine of a slave. A synchronization cycle is initiated in the following cases:

- after a hardware reset,
- after a reset of the PDL layer by the PDL-user,
- after the detection of protocol errors,
- after a multiple data cycle error (max_swa_count time expired), and
- after a multiple SPA_acknowledge_timeout (the SPA acknowledge timeout occurred max_spa_retry-times).

In the first two cases the buffers and queues of the protocol layer for sending and receiving of messages are cleared from the concerned devices. Thus, all requests, confirmations and indication stored in these buffers are lost. In the remote device, however, no buffers are cleared. After the synchronization cycle this device tries to transmit the interrupted send message again.

In all other cases no buffers are cleared in any device. Upon a successful synchronization, both devices re-try to carry out orders of the application which have not yet been completed.

4.3.5.1.2 SVA message

Upon a successful synchronization and before interrupted messages are sent again, the master sends a SVA Message ("Send Value with Acknowledge") to the slave. The SVA Message transfers variables for the parameterisation of the PDL protocol machine.

The SVA Message transmits `max_swa_count`. The `max_swa_count` variable has a default value of 128 and can be parameterized by means of `PDL_Set_Value`. The slave accepts this value as its own `max_swa_count`.

The `max_swa_count` variable shall be transferred. In addition, other variables may be specified.

4.3.5.1.3 Frame count bit

The frame count bit (FCB) prevents a multiplication of messages at the confirming and/or responding device (responder), as this would cause the loss of positive acknowledgements.

If a positive acknowledgement is lost for whatever reason, the requester tries to send the previous message again. When this message has already been correctly received by the responder, this is indicated by an unchanged FCB. In this case the responder again sends the acknowledgement to the requester, directly after the receipt of the first message segment. Then, the requester stops the repeated sending.

If a new message is to be sent the FCB shall be changed. To ensure that the requester FCB (transmit FCB) and the responder FCB (receive FCB) of the remote device have the same initial value after the initialization of the layer 2 and after protocol errors, there will be a synchronization with `FCB_SET` messages. The FCBs are set to '1' if the synchronization was successful.

There is a FCB pair for both transmission directions (one transmit and one receive FCB for each direction) (see Figure 17).

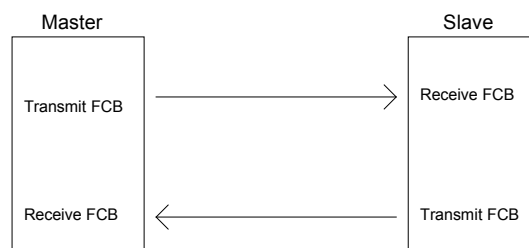


Figure 17 – Transmit and receive FCBs on the master and slave sides

4.3.5.1.4 Data transmission of bus cycles with errors

If a data cycle error occurs during the transmission of a SPA or SVA Message, the queue is not completely transmitted again. The transmission is rather continued with the queue parts that follow the error.

The master responds to cycle errors. Thus, a distinction is to be made between the two transmission directions **master → slave** and **slave → master**. If a cycle error occurs, this does not have any influence on the PDL protocol machine.

1) Data transmission: master → slave

If the master detects a data cycle error while queue is transmitted, the transmission shall be repeated from the error onwards. In this case the master communicates the error to the slave by means of a SWA Message.

Figure 18 and Figure 19 clarify the transmission master → slave with SWA Message. The numbering corresponds to the time sequence:

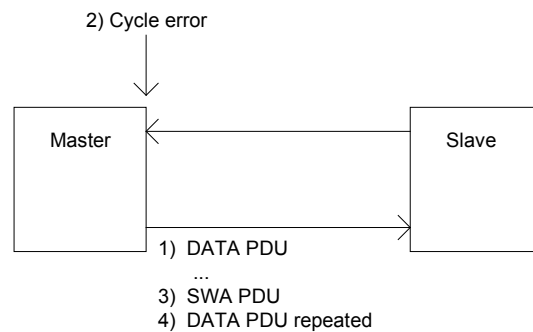


Figure 18 – Data transmission master → slave with SWA Message

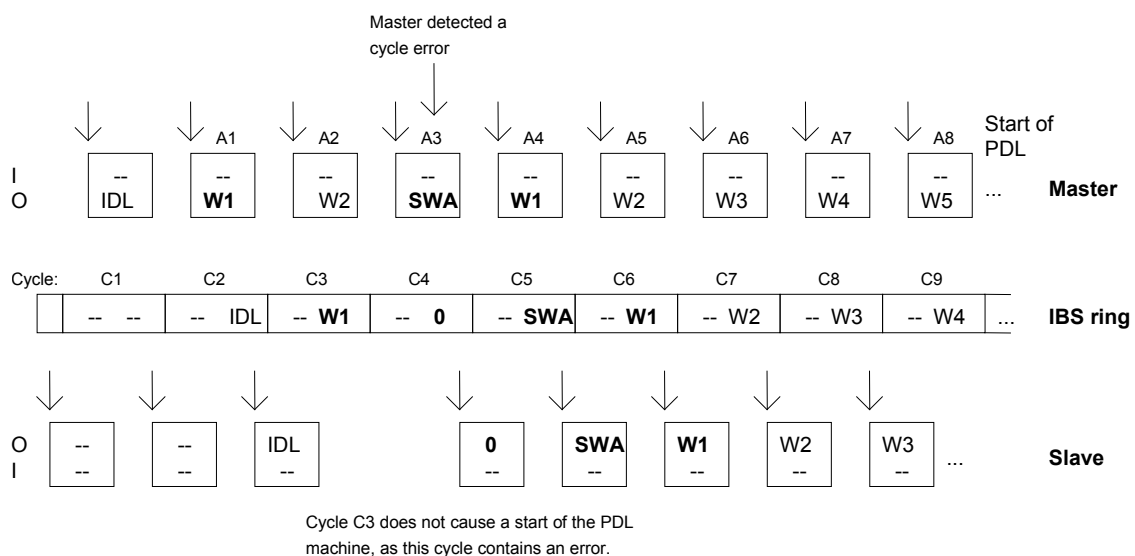


Figure 19 – Time sequence of the data transmission master → slave with SWA Message

2) Data transmission: slave → master:

If the master detects a cycle error when it receives a PDU, the slave will be announced immediately from the master by means of a RWA PDU. The slave shall confirm this RWA PDU with a SWA PDU, before the DATA PDU is sent again. The master uses the SWA PDU to mark the beginning of repeated data transmission.

An outstanding data transmission sequence from master → slave is interrupted during the RWA Message transfer and after the exception handling the data transmission can be continued.

Figure 20 and Figure 21 clarify the transmission slave → master with RWA Message or SWA Message:

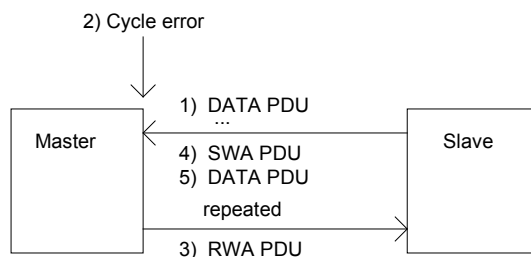


Figure 20 – Data transmission slave → master with SWA/RWA Message

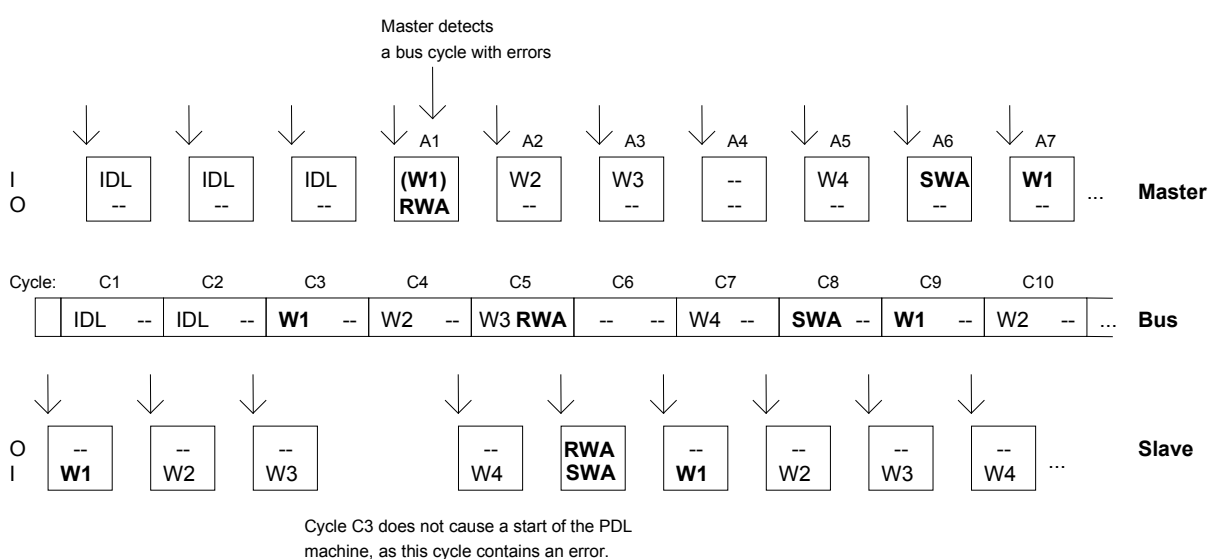


Figure 21 – Time sequence of the data transmission slave → master with SWA/RWA Message

4.3.5.2 Description of the time sequences

Master:

After each data cycle the PDL protocol machine is started once in the master for each slave in the ring having a parameter channel.

Among others, the protocol machine knows the following parameters:

Input parameters:

- The message segment which has been received during the last intact data cycle from the slave.
- The information whether a bus cycle error occurred during the last data cycle.

Output parameters:

- The message segment which is to be sent in the next cycle to the slave.

In Figure 22 these parameters are shown for each PDL protocol call A1...An, where I0...In identify the receive data for the master and the send data for the slave. Accordingly, O0...On are send data of the master and receive data of the slave.

Slave:

After the completion of a data cycle the PDL protocol machine is started on the slave, if the slave determined that the master did not send an IDL PDU and/or there is an outstanding send data request on the slave. IDL PDU are transmitted whenever there is no further user data are to be transmit. The last received message can be read and/or one outstanding send message can be prepared in the PDL protocol machine. The PDL pass the PDL PDU to the BLL for transmitting within the next data cycle to the master. The third line of the representation shows the starts of the PDL protocol machine of the slave (A1...A9) and the associated send and receive messages.

Bus:

In Figure 22, the middle row shows the cycles (C1...C9) and the messages which are transmitted in these cycles from the slave to the master and vice versa.

The transmission of a message from the TRANSMIT protocol machine of the master to the RECEIVE protocol machine of the slave requires two cycles, and the transmission from the TRANSMIT protocol machine of the slave to the RECEIVE protocol machine of the master requires three cycles. The TRANSMIT and RECEIVE protocol machines are components of the PDL protocol machine.

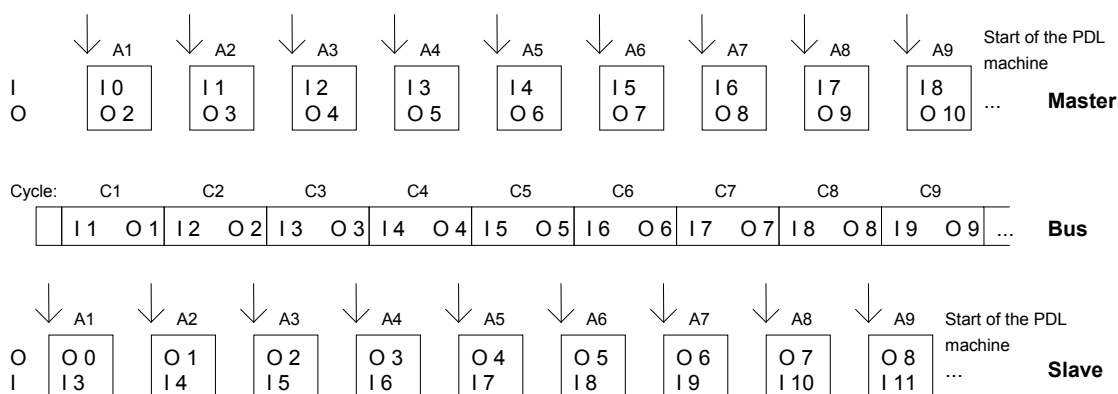


Figure 22 – Allocation of actions of the PDL protocol machines and data cycles

In the slave, the start of the PDL protocol machine for the sending and receiving of PDL PDU shall be completed before a further cycle end was indicated. Otherwise received data can be lost. The DLPDU cycle time depends with the respect from the number of slaves and from the data width of each slave which are connected to the DL-subnetwork.

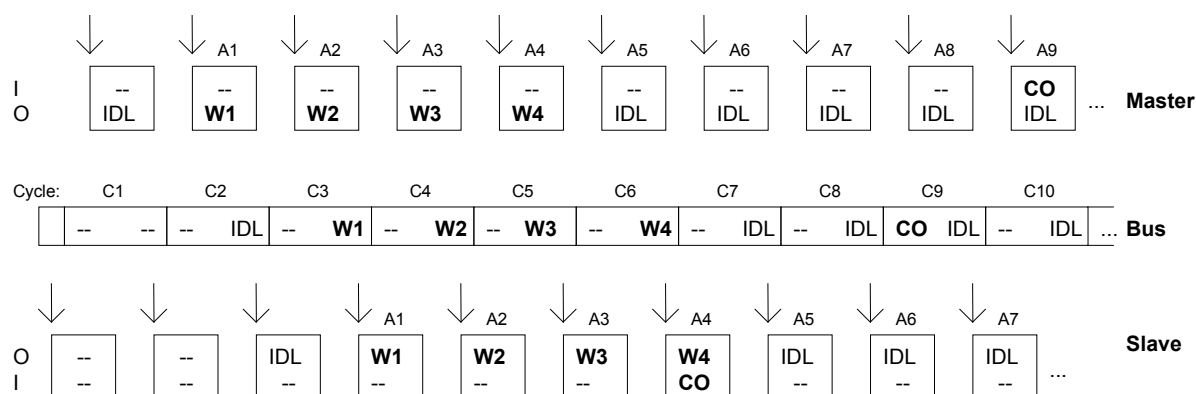


Figure 23 – Message transmission: master → slave

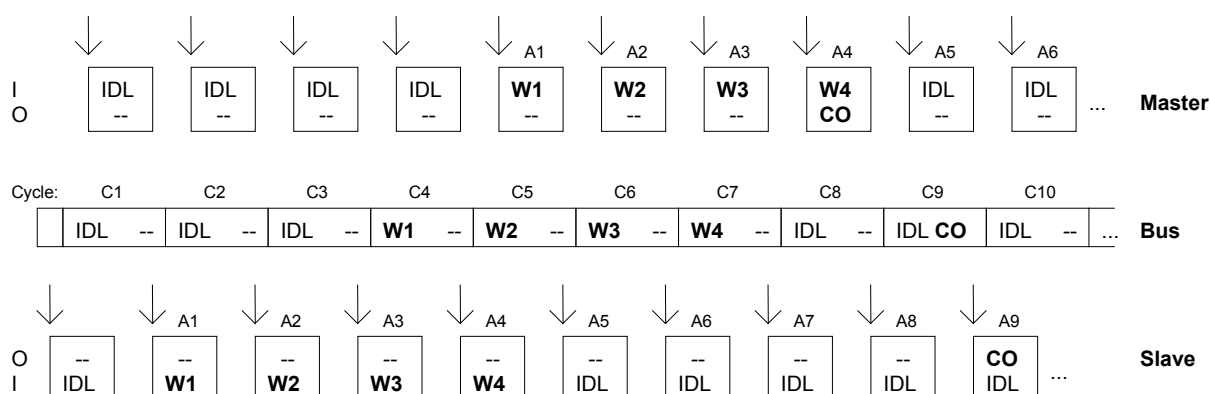


Figure 24 – Message transmission: slave → master

Figure 23 and Figure 24 show that at least $DIST = 5$ data cycles are needed between the sending of the last message, until a confirmation (CO) for this message is received.

Delays while confirmations may be also taken into the account, so the protocol need additional cycles for waiting of a confirmation. This number of additional cycles is stored in the variable "add_wait". On the master side the variable add_wait can be parameterized with the PDL_Set_Value (value range: 1 to 4). The contents of the variable add_wait is transmitted from the master to the slave within the FCB_SET PDU while the PDL protocol machines are synchronized.

4.3.5.3 Coding of the messages

4.3.5.3.1 Overview

The message length for a slave with the parameter channel consists at least one octet for FCT and additional the length of the data 1, 3 or 7 octets, depending from the size of the message for transmission.

4.3.5.3.2 Structure of the code octet

4.3.5.3.2.1 General

Figure 25 shows the structure of a code octet, Table 19 shows the L_status encoding and Table 20 the FCT code.

B16	B15	B14	B13	B12	B11	B10	B9
L_status			FCT code			FCB	IDL

Figure 25 – Code octet of a PDL PDU

Table 19 – Encoding of the L_status

L_status (B16 B15 B14)	Generation (Local/Remote)	Meaning
0 0 0	–	No confirmation
0 0 1	L / R	Acknowledgement positive
0 1 0	R	Acknowledgement negative, no resource available (buffer full)
0 1 1	L / R	Acknowledgement negative, multiple data cycle error (transmission via the bus only from the master to the slave)
1 0 0	R	Acknowledgement positive (repeated)
1 0 1	R	Acknowledgement negative, no resource available (repeated)
1 1 0	L	Acknowledgement negative, acknowledge_timeout
1 1 1	R	FCB_SET confirmation, the FCT code equals 0; FCB = 1!

Table 20 – FCT code (PDL PDU-Types)

Function code	PDL-PDU Types	Meaning
0 0 0	IDL PDU	The IDL PDU do not contain any information (except for a FCB_SET confirmation).
0 0 1	DATA PDU	The DATA PDU contains segments of user data or PDL variables.
0 1 0	SPA PDU	The SPA PDU defines the start of a new queue data transmission and contains at least the length information of DLSDU in octets-1 and segments of user data.
0 1 1	SVA PDU	The SVA PDU defines the start of a new variable data transmission and contains at least the length information of variable data in octets-1 and possibly PDL variables.
1 0 0	Don't spare	Reserved
1 0 1	RWA PDU	RWA PDU (Receive Word Again): The data octets of the message contain at least the amount of successfully received data octets+1 and possibly DLSDU data or PDL variables.
1 1 0	SWA PDU	SWA PDU (Send Word Again): The data octets of the message contain at least the amount of successfully sent data octets + 1 and possibly DLSDU data or PDL variables.
1 1 1	FCB_SET PDU	The FCB_SET PDU contains L_status = 0 and FCB = 1, for an FCB_SET request from the master to the slave. The first data octet of the message contains user data for the PDL protocol machine of the slave.

4.3.5.3.2.2 FCB

Frame count bit : 0/1, alternating bit.

The FCB is only relevant within the start segment of SPA Message and SVA Messages.

4.3.5.3.2.3 Idle message

If the IDL-Bit (see Figure 25; Bit 9) equals 0, the message does not contain any information and is called IDLE message.

4.3.5.3.3 Messages with a size of one word

4.3.5.3.3.1 General

Figure 26 shows the structure of a message with a size of one word.

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X

Figure 26 – Structure of a message with a size of one word

4.3.5.3.3.2 Call messages

1) **SPA Message** (PDL PDU-Type for transmission of queued user data):

In the L_status of the code octets, a confirmation for a SPA Message of the remote device can be transmitted at the same time (see Figure 27).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	0	1	0	FC B	1	n-1: DLSDU length in octets-1: $1 \leq n \leq 256$							
X	X	X	0	0	1	X	1	1st DLSDU octet							
X	X	X	0	0	1	X	1	2nd DLSDU octet							
X	X	X	0	0	1	X	1	Nth DLSDU octet							

Figure 27 – Structure of a SPA Message

2) **SVA Message** (DL PDU-Type for transmission of configuration data):

After a confirmed FCB_SET, further variables are transmitted from the master to the slave with the 'Send Value with Acknowledge' message. On the master side these variables can be set with the PDL_Set_Value.request and are also valid for the slave after the transmission of the SVA Message. The number of variable octets-1 is transmitted in the data octet of the start segment.

The swa_count variable shall be transferred. In addition, further variables may be specified (see Figure 28).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	0	1	1	FC B	1	n-1: number of variable octets-1							
X	X	X	0	0	1	X	1	Swa_count (mandatory)							
X	X	X	0	0	1	X	1	2 nd variable octet (optional)							
...															
X	X	X	0	0	1	X	1	Nth variable octet (optional)							

Figure 28 – Structure of a SVA Message

3) FCB_SET Message:

This message is used to synchronize the PDL protocol machines of a parameter channel on the master and on the slave, that is, to set the FCBs to a defined value (see 4.3.5.1.3).

In the first data octet, the master also transmits the protocol specific parameters which have been set with the PDL_Set_Value service (see Figure 29).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
0	0	0	1	1	1	1	1	Coding see below							

Figure 29 – Structure of a FCB_SET Message

4.3.5.3.3 Control segments

1) RWA Message:

With the RWA (Read Word Again) message the master indicates to the slave during a SPA Message that the slave has received one message without errors. In the first data octet the RWA Message contains the number of DLSDU octets which have been received without errors (see Figure 30).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	1	0	1	X	1	k Number of correctly received DLSDU octets during the current SPA Messages							

Figure 30 – Structure of a RWA Message

2) SWA Message:

For the SWA Message a difference between master and slave has to be observed, as only the master can immediately detect a bus cycle with errors (see Figure 31).

- When a SPA Message is sent from the slave to the master, its identifies the beginning of repeated data transmission. Here, the SPA Message is sent after a RWA Message.
- If the master detects a data cycle with errors when a SPA or SVA Message is sent to the slave, the transmission is repeated from the error onwards. This error-1 = number of PDU sent without errors is communicated to the slave by means of a SWA Message.

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	1	1	0	X	1	k Number of data octets sent without errors							
X	X	X	0	0	1	X	1	(k+1)th data octet							
X	X	X	0	0	1	X	1	(k+2)th data octet							
...															
X	X	X	0	0	1	X	1	N-th data octet							

Figure 31 – Structure of a SWA Message

4.3.5.3.3.4 Response messages

1) Data confirmation (confirmation for SPA or SVA Messages):

The higher three bits of the code octet contain the confirmation for the call messages described above. This confirmation can also be transmitted together with a message start segment of SPA or SVA, control or user data segment (exception: FCB_SET PDU) (see Figure 32).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
L_status			X	X	X	X	1	X	X	X	X	X	X	X	X

Figure 32 – Structure of a confirmation for SPA or SVA Messages

2) FCB_SET confirmation:

The FCB_SET confirmation acknowledges a FCB_SET request (see Figure 33).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
1	1	1	0	0	0	1	1	Coding see below							

Figure 33 – Structure of a FCB_SET as confirmation

Coding of the data octet for FCB_SET Message as request and confirmation

In the data octet of the FCB_SET as request or the FCB_SET as confirmation, the master transmits the protocol specific parameters which have been set with the PDL_Set_Value service (see Figure 34):

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	X	X	X	X	X	a)			b)			c)	

Figure 34 – Structure of the data octet for FCB_SET as requests and confirmations

a) B8-B6: err_max

The contents of the variable err_max should be multiplied with 2, accordingly to get the number of attempts for sending a same message

b) B5-B3: Reserved

c) **B2-B1: add_wait**

The variable add_wait-1 gives the number of cycles that is additionally waited for a confirmation.

4.3.5.3.4 Queue data with a segment size of more than one word

Example: Queue data size = 2 words, data transmission with SWA Message after a cycle error (see Figure 35).

Code octet			1st data octet		2nd data octet	3rd data octet	
X	X	X	1	1	0	X	1
			k number of message octets sent without an error		(k+1)th message octet	(k+2)th message octet	
X	X	X	0	0	1	X	1
			(k+3)th message octet		(k+4)th message octet	(k+5)th message octet	
X	X	X	1	1	0	X	1
			(N-1)th message octet		N-th message octet	X	X
						X	X
						X	X
						X	X
						X	X
						X	X
						X	X

Figure 35 – Structure of a message with a size of more than one word

Number of segments to be transmitted for a message (DLSDU):

Calculation: $G_m(N) = (N-1)/m + 1$

where

N is the user data quantity (including the number of octets sent/received without errors or the data length)

M is the number of data octets per segment (PDU)

G_m(N) is the number of PDU.

NOTE (N-1) / m is a not rounding integer division.

Example:

A queue data with 13 octets is to be transmitted. N = 14, including the data length. The segment size is two words, that is, three data octets are available in the segment (m = 3):

$$G_3(14) = (14-1) / 3 + 1 = 4 + 1 = 5$$

4.3.6 PDL protocol machines

4.3.6.1 PDL base protocol machine

4.3.6.1.1 Description of the states

The PDL base protocol machine provides the functionality of the PDL and has the following five states:

4.3.6.1.2 PDL_INIT

State after power on. This state is only left when all required PDL protocol machines have been generated by means of the communication relationship list entries and the PDL receive and transmit update memories are initialized.

4.3.6.1.3 PDL_RECEIVE_UPDATE

In this state the PDL takes the user data out of the PDLSDU which was received from the BLL and updates the PDL receive update memory of layer 2.

4.3.6.1.4 PDL_PM_ACTIVE

With the transition into this state all protocol machines are initiated at the same time. This state shall only be left after all protocol machines have been processed.

4.3.6.1.5 PDL_TRANSMIT_UPDATE

In this state the PDL takes the user data out of the transmit update memory of layer 2, generates the user data part of the PDLSDU to be sent to the BLL and transmits the PDLSDU with a BLL_Data.request (master side) or BLL_Data.response (slave side) primitive to the BLL.

4.3.6.1.6 PDL_WAIT_FOR_PDLSDU

In this state the PDL base protocol machine (see Figure 36) waits for a BLL_Data.confirm(master side) or BLL_Data.indication (slave side) primitive of the BLL.

In all states (except PDL_INIT) the PDL_Data_Ack.request primitives, which are passed from the PDL-user to the PDL, are processed as follows:

First, the service request is locally confirmed by means of a PDL_Data_Ack.confirm primitive. Then at the receiving side, the PDL generates a PDL_Data_Ack.indication primitive that is passed on to the PDL-user, which can be identified by the remote address.

A reset generally causes a transition to the PDL_INIT state.

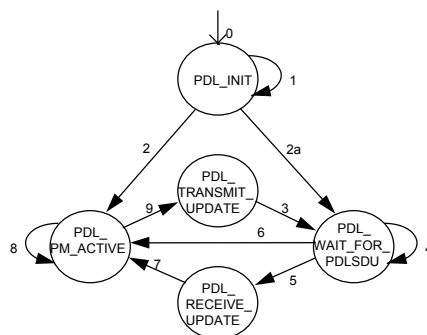


Figure 36 – PDL base protocol machine

The state transitions from every state (except PDL_INIT) to the same state when a PDL_Data_Ack.request primitive arrives from the PDL-user are not shown to simplify matters. However, they are listed in the state transition table.

4.3.6.1.7 Description of the transitions

Table 21 shows the state transitions of the PDL base protocol machine.

Table 21 – State transitions of the PDL base protocol machine

Initial state event \ condition ⇒ action	Transition	Follow-up state
After Power on	0	PDL_INIT
PDL_INIT \ initialization of the PDL protocol machines and the update memories are not yet completed	1	PDL_INIT
PDL_INIT (<i>master side only</i>) \ initialization completed ⇒ start of all PDL protocol machines	2	PDL_PM_ACTIVE
PDL_INIT (<i>slave side only</i>) \ initialization completed	2a	PDL_WAIT_FOR_PDLSDU
PDL_TRANSMIT_UPDATE ⇒ copy of user data from TUM to the PDLSDU AND send PDLSDU to BLL with BLL_Data.request/response	3	PDL_WAIT_FOR_PDLSDU
PDL_WAIT_FOR_PDLSDU \ no BLL_Data.confirm/indication received	4	PDL_WAIT_FOR_PDLSDU
PDL_WAIT_FOR_PDLSDU BLL_Data.confirm/indication received \ update_info == OK	5	PDL_RECEIVE_UPDATE
PDL_WAIT_FOR_PDLSDU BLL_Data.confirm received \ update_info == NOK ⇒ start of all PDL protocol machines ⇒ if trigger_mode == AT, then start_bus_cycle = OFF	6	PDL_PM_ACTIVE
PDL_RECEIVE_UPDATE ⇒ copy user data from PDLSDU to RUM ⇒ start all PDL protocol machines ⇒ if trigger_mode == AT, then start_bus_cycle = OFF	7	PDL_PM_ACTIVE
PDL_PM_ACTIVE \ not all PDL protocol machines are stopped OR (<i>master side only</i>) start_bus_cycle == OFF	8	PDL_PM_ACTIVE
PDL_PM_ACTIVE \ all PDL protocol machines are stopped AND (<i>master side only</i>) start_bus_cycle == ON	9	PDL_TRANSMIT_UPDATE
Any_state PDL_Reset.request received ⇒ PDL_Reset.confirm		PDL_INIT
Any_state PDL_Data_Ack.request (...rem_add,...) ⇒ PDL_Data_Ack.confirm ⇒ PDL_Data_Ack.indication to PDL-user with rem_add		same_state
Any_state all PDL management services ⇒ process and confirm the services		same_state

4.3.6.2 PDL protocol machine

4.3.6.2.1 Overview

For each slave with a parameter channel in the ring, the PDL base protocol machine of the master manages a PDL protocol machine. The PDL base protocol machine of a slave with a parameter channel, however, has only one PDL protocol machine for this parameter channel.

A PDL protocol machine processes all PDL_Data_Ack services which are transmitted via the parameter channel to the PDL and has, in particular, the following tasks:

Connection:

- Establishing the PDL connection to the remote device.

Sending side:

- Testing the send requests (PDL_Data_Ack.request) for plausibility
- Process the send requests
- Confirming the send requests (PDL_Data_Ack.confirm)
- Segmentation of the DLSDU to data segments
- Error detection while data transfer
- Passing the Message segments and confirmations with the BLL_Data.request/response to the Basic Link Layer (BLL).

Receiving side:

- Receiving confirmations and data segments with the BLL_Data.confirm
- Re-assembling the data segments to DLSDU
- Transferring received DLSDU to the PDL-user with PDL_Data_Ack.indication.

Figure 37 explains the PDL protocol machine in general for a master and a slave. The differences between master and slave are given. Each PDL protocol machine controls and uses a TRANSMIT and a RECEIVE protocol machine.

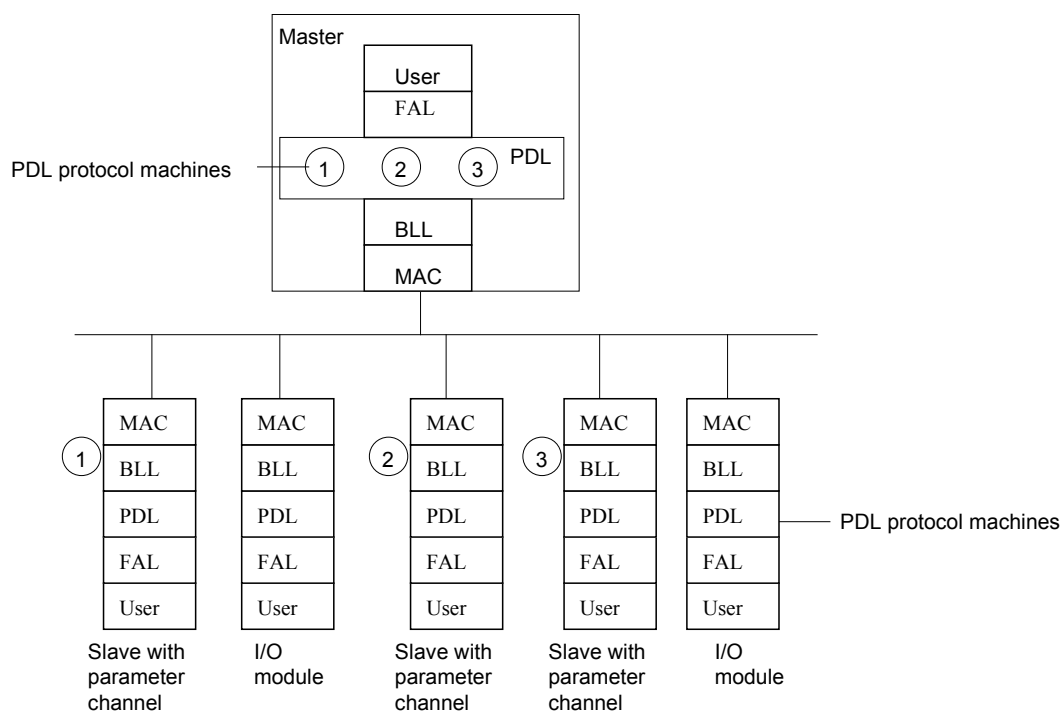


Figure 37 – Locations of the PDL and the PDL protocol machines in the master and slaves

4.3.6.2.2 Counters and flags in the protocol machines

4.3.6.2.2.1 Counters

The counters described in the following are used in the PDL protocol machines for the parameter channel (see Table 22):

Table 22 – Counters of the PDL protocol machines

Counter	Description	Used in protocol machine
Ccycle	Counter for the number of data cycles since the start signal of a SPA or SVA Message has been sent. This counter checks the correctness of repeated confirmations.	TRANSMIT
Ccerr	Counter for the number of defective data cycles after another while a message is sent.	TRANSMIT
Cconf	Counter for the number of cycles during which a confirmation is awaited.	PDL protocol and TRANSMIT
Creq_retry	Counter for the attempts to send messages when confirmations were lost due to defective data cycles.	TRANSMIT
CSWA	Counter for the number of cycles during which a SWA Message is awaited.	RECEIVE

4.3.6.2.2.2 Flags

connection

The PDL protocol machine manages a flag called "connection". This flag is the same as DISCONN, when the PDL protocol machine of the master is not synchronized with that of a