IEC 61158-4-7

Edition 1.0 2007-12

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 4-7: Data-link layer protocol specification – Type 7 elements

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: 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.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61158-4-7

Edition 1.0 2007-12

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 4-7: Data-link layer protocol specification – Type 7 elements

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XE

ICS 35.100.20; 25.040.40

ISBN 2-8318-9434-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-7: Data-link layer protocol specification – Type 7 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.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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.

International Standard IEC 61158-4-7 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.

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.

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 4-7: Data-link layer protocol specification – Type 7 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) 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 part of 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 document. 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 61158-3-7, *Industrial communication networks – Fieldbus specifications – Part 3-7: Data link service definition – Type 7 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 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 correspondent (N)-entities [7498-1]
correspondent DL-entities (N=2)
correspondent Ph-entities (N=1)

3.1.2 DL-address [7498-3]

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-name [7498-3]

3.1.7 DL-protocol [7498-1]

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

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

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

3.1.11 DL-relay [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 flow control [7498-1]

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

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

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

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

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

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

3.1.22 peer-entities [7498-1]

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

3.1.24 Ph-interface-data [7498-1]

3.1.25 primitive name [7498-3]

3.1.26 reset [7498-1]

3.1.27 responding-DL-address [7498-3]

3.1.28 routing [7498-1]

3.1.29 segmenting [7498-1]

3.1.30 sequencing [7498-1]

3.1.31 system management [7498-1]
systems-management

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 deliver (primitive)

3.2.3 DL-service-primitive;
primitive

3.2.4 DL-service-provider

3.2.5 DL-service-user

3.2.6 DL-user-optional-facility

3.2.7 indication (primitive)
acceptor.deliver (primitive)

3.2.8 multi-peer

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

3.2.10 requestor

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

3.2.12 submit (primitive)

3.3 Other terms and definitions

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

For the purpose of this part of IEC 61158, the following definitions also apply:

3.3.1
acknowledgement response DLPDU
information that the recipient of an acknowledged message emits in order to signal either the proper reception of the message or the lack of available resources to store the message, received by the DLE on the local link that emitted the message which requested the acknowledgement

3.3.2
basic cycle
sequence of scanning by the bus-arbitrator of:

a) a set of DLCEP-identifiers for variables, requests, and cyclical application messages,

b) plus the window provided for aperiodic exchanges,

c) plus the window provided for message services,

d) plus the window provided for synchronization

3.3.3
basic transaction
succession of DLPDUs related to a single DL-service instance

3.3.4
bus-arbitrator (BA)
DLE that controls each data producer's right to access the medium

NOTE At any given instant one and only one bus-arbitrator is active in each DL-segment of a DL-subnetwork.

3.3.5
consumed identified variable
identified variable that corresponds to a DLCEP-identifier for which the entity in question receives data

3.3.6
control field
portion of an emitted or received DLPDU that gives the nature of the data exchanged and the type of exchange

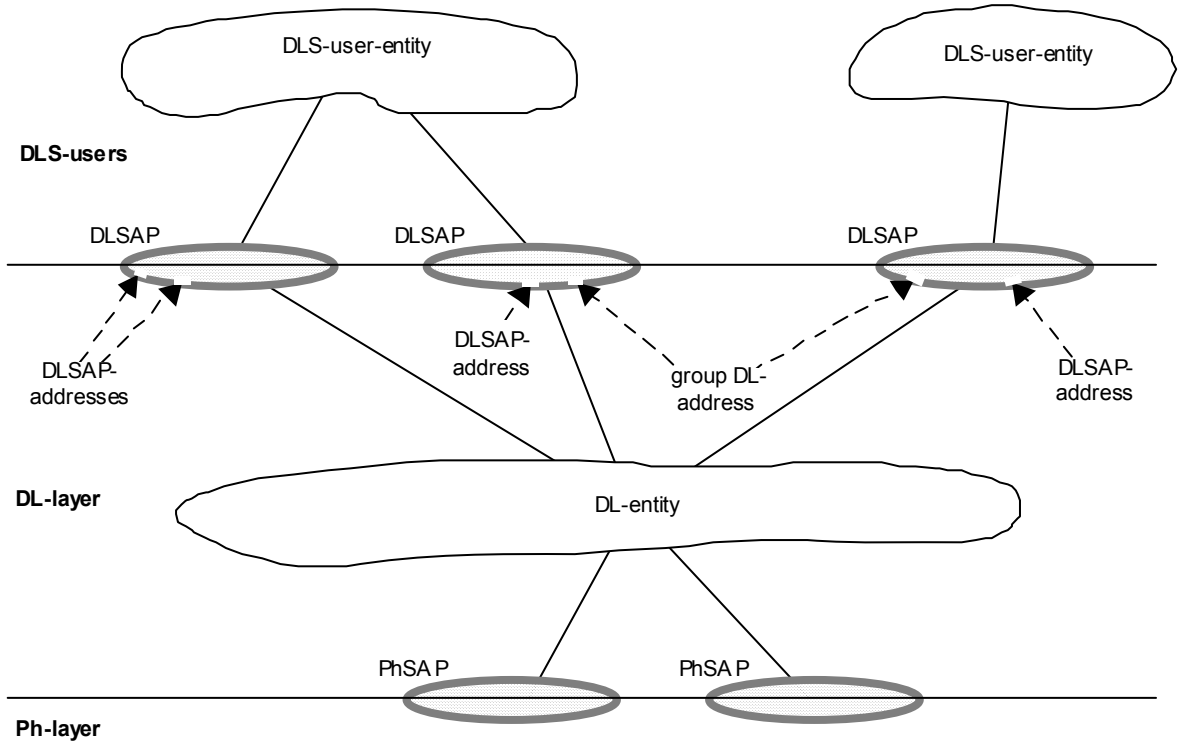
3.3.7
destination address
three octets specifying the DL-segment of the DLE to whom the message is sent, and the destination DLSAP's sub-address within the local link DL-segment

3.3.8
DLCEP-address
information that the bus-arbitrator emits to allocate the medium to a data producer for the purpose of exchanging a variable

3.3.9
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

3.3.10
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

3.3.11
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

3.3.12
(individual) DLSAP-address
DL-address that designates only one DLSAP within the extended link

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

3.3.13
end of message transaction indication DLPDU
information that the source entity of a message emits in order to return link access control to the bus-arbitrator at the end of a message transaction

3.3.14
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.

3.3.15
frame
denigrated synonym for DLPDU

3.3.16
group DL-address
DL-address that potentially designates more than one DLSAP within the extended link. A single DL-entity may have multiple group DL-addresses associated with a single DLSAP. A single DL-entity also may have a single group DL-address associated with more than one DLSAP

3.3.17
identified variable (or simply "variable")
DLL variable (buffer) for which an associated DLCEP-identifier has been defined

3.3.18
identifier
16-bit word associated with a system variable. A DLCEP-identifier uniquely designates a single variable within the DL-subnetwork

3.3.19
invalid DLCEP-identifier
identifier not recognized locally

3.3.20
local link
set of devices that respect the DL-protocol and that are interconnected through a medium . Only one bus-arbitrator is active on a single local link

3.3.21
macrocycle
set of basic cycles needed for all cyclical DLCEP-identifiers to be scanned

3.3.22
message DL-address
information that the bus-arbitrator emits to allocate the medium to a source entity for a message transfer

3.3.23
message response DLPDU
information that a data producer emits in response to a message DLCEP-identifier DLPDU

NOTE The desired destination entity or entities pick up this information.

3.3.24
node
single DL-entity as it appears on one local link

3.3.25
periodic scanning of variables
action by the bus-arbitrator that guarantees the cyclical exchange of variables

NOTE This is the basic principle of the Type 7 DL-service and protocol.

3.3.26
produced identified variable
identified variable that corresponds to a DLCEP-identifier for which the DLE emits data

3.3.27
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.

3.3.28
request DLCEP-address
information that the bus-arbitrator emits to allocate the medium to the initiator of an explicit request for a variable exchange

3.3.29
request response DLPDU
information that the initiator of an explicit request for a variable exchange emits in response to a request DLCEP-identifier DLPDU, and to whose transmittal the bus-arbitrator also responds

3.3.30
sending DLS-user
DL-service user that acts as a source of DL-user-data

3.3.31
source address
24-bit word including the DL-segment number of the entity sending the message, and the entity's sub-address within the DL-segment

3.3.32
timers
see 5.6

3.3.33
triggered message scanning
function of a bus-arbitrator that makes it possible to transfer messages

3.3.34
triggered scanning of variables
function of a bus-arbitrator that makes possible the non-cyclical exchange of variables

3.3.35
triggered periodic scanning of messages
function of a bus-arbitrator that makes it possible to request triggered message exchanges cyclically

3.3.36
triggered periodic scanning of variables
function of a bus-arbitrator that makes it possible to request triggered variable transfers cyclically

3.3.37
turnaround time
time interval between reception or emission of the last MAC symbol of a DLPDU, signaled by a SILENCE indication from the PhL, and the reception or emission of the first MAC symbol of the subsequent DLPDU, signaled by an ACTIVITY indication from the PhL, both as measured in a given station

3.3.38
variable response DLPDU
information that a data producer emits in response to a DLCEP-identifier DLPDU, which also alerts data consumers to the relevance of the immediately time-proximate DLPDU

3.4 Symbols and abbreviations

3.4.1 BA	bus-arbitrator
3.4.2 B_Dat_Cons	buffer which contains the value of the data consumed
3.4.3 B_Dat_Prod	buffer which contains the value of the data produced
3.4.4 B_Req1/2	buffer containing the list of DLCEP-identifiers that are the objet of a specified explicit request for a transfer at the priority 1 (urgent) or 2 (normal)
3.4.5 ID_DAT	DLPDU used to allocate the medium to a buffer transfer
3.4.6 ID_MSG	DLPDU used to allocate the medium to a message exchange
3.4.7 ID_RQ1/2	DLPDU used to allocate the medium to a request for a buffer transfer
3.4.8 PRT	physical reaction time
3.4.9 Q_IDMSG	queue of requested DLCEP-identifiers received by the BA for message transfer
3.4.10 Q_IDRQ1/2	queue of requested DLCEP-identifiers received by the BA at the priority 1 (urgent) or 2 (normal)
3.4.11 Q_Msg_Cyc	queue which contains messages to be emitted that are associated with cyclical exchanges
3.4.12 Q_Msg_Aper	queue which contains messages to be emitted that are associated with aperiodic exchanges
3.4.13 Q_Req1/2	queue containing the list of DLCEP-identifiers that are the objet of a free explicit request for a transfer at the priority 1 (urgent) or 2 (normal)
3.4.14 Q_RPRQ	queue for aperiodic transfers in progress
3.4.15 RQ_Inhibit	Indicator used to manage explicit request for variable

exchanges

3.4.16 RP_ACK DLPDU used to transfer an acknowledgement of message exchange, Transfer OK

3.4.17 RP_DAT DLPDU used to carry the value of the identified variable previously requested.

3.4.18 RP_DAT_MSG DLPDU used to carry the value of the identified variable previously requested with a request for a message exchange.

3.4.19 RP_DAT_RQ1/2 DLPDU used to carry the value of the identified variable previously requested with a request for an explicit transfer of variables.

3.4.20 RP_DAT_RQ1/2_MSG DLPDU used to carry the value of the identified variable previously requested with a request for an explicit transfer of variables and a request for a message transfer.

3.4.21 RP_MSG_ACK DLPDU used to carry the message to be exchanged with a request of acknowledgement of this exchange.

3.4.22 RP_MSG_NOACK DLPDU used to carry the message to be exchanged without a request of acknowledgement of this exchange.

3.4.23 RP_NAK DLPDU used to transfer an acknowledgement of message exchange, that the message was received correctly, but was not stored by the DLE.

3.4.24 RP_END DLPDU used to terminate a message exchange.

3.4.25 STT station turnaround time

3.4.26 TI latency time

4 Overview of the DL-protocol

4.1 Overall description of medium allocation

An element known as the **bus-arbitrator (BA)** controls the right of each data producer to access the medium by emitting a DLPDU containing a DLCEP-identifier. At any given instant there should be only one active bus-arbitrator in each local link.

NOTE The term "data producer" designates the sole station connected to the local link that is recognized as having the responsibility of emitting the data associated with the identifier DLPDU circulating on the medium.

Each transaction belongs to one of the three medium allocation classes defined below:

— cyclical exchange of variables, requests, or messages,

— explicit request for variable exchange,

— explicit request for message transfer.

For cyclical variable exchanges a basic transaction is made up of the following phases. The bus-arbitrator broadcasts a variable identifier DLPDU. The sole producer of the information required then broadcasts a variable response DLPDU. During this phase consumers take the information from the local link. Figure 2 shows the various phases of a variable exchange transaction. When one transaction has been completed the bus-arbitrator begins the following transaction according to guidelines defined when the system is configured.

During a cyclical variable exchange the information producer can, using the response DLPDU, transmit to the BA an explicit request for the exchange of variables or messages.

For an explicit request for a variable exchange, a basic transaction is made up of the following phases: The bus-arbitrator broadcasts a request identifier DLPDU. Then the initiator of the request broadcasts a request response DLPDU. One or more transactions identical to the cyclical variables exchange transaction then follow.

For message transfers a basic transaction consists of the following phases: The bus-arbitrator broadcasts a message identifier DLPDU. Then the message response DLPDU is exchanged between the communicating entities. This DLPDU may or may not be followed by an acknowledgement DLPDU. The source entity then transmits an end of transaction indication DLPDU to the bus-arbitrator.

The time interval separating the reception or emission of the end of one DLPDU and the emission or reception of the following DLPDU is known as the station's turnaround time, whether the station's function be that of a producer/consumer or bus-arbitrator.

A more detailed definition of turnaround time is given in 3.3.37. The impact of turnaround time on data-link timers is described in 5.6.2.

The role of the bus-arbitrator is to "give the floor" to each data producer, taking into account the services required for the type of data (according to the three medium allocation classes defined above).

The bus-arbitrator thus has three types of functions:

— periodic scanning of variables or periodic triggering of variable and message exchanges,

— triggered scanning of variables,

— triggered scanning of messages.

In addition, the bus-arbitrator can provide a synchronization function in order to guarantee the constant length of scanning cycles.

Each type of scanning takes place in a specific window, that is, respectively in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window, as shown in Figure 2. These four windows constitute a basic scanning cycle.

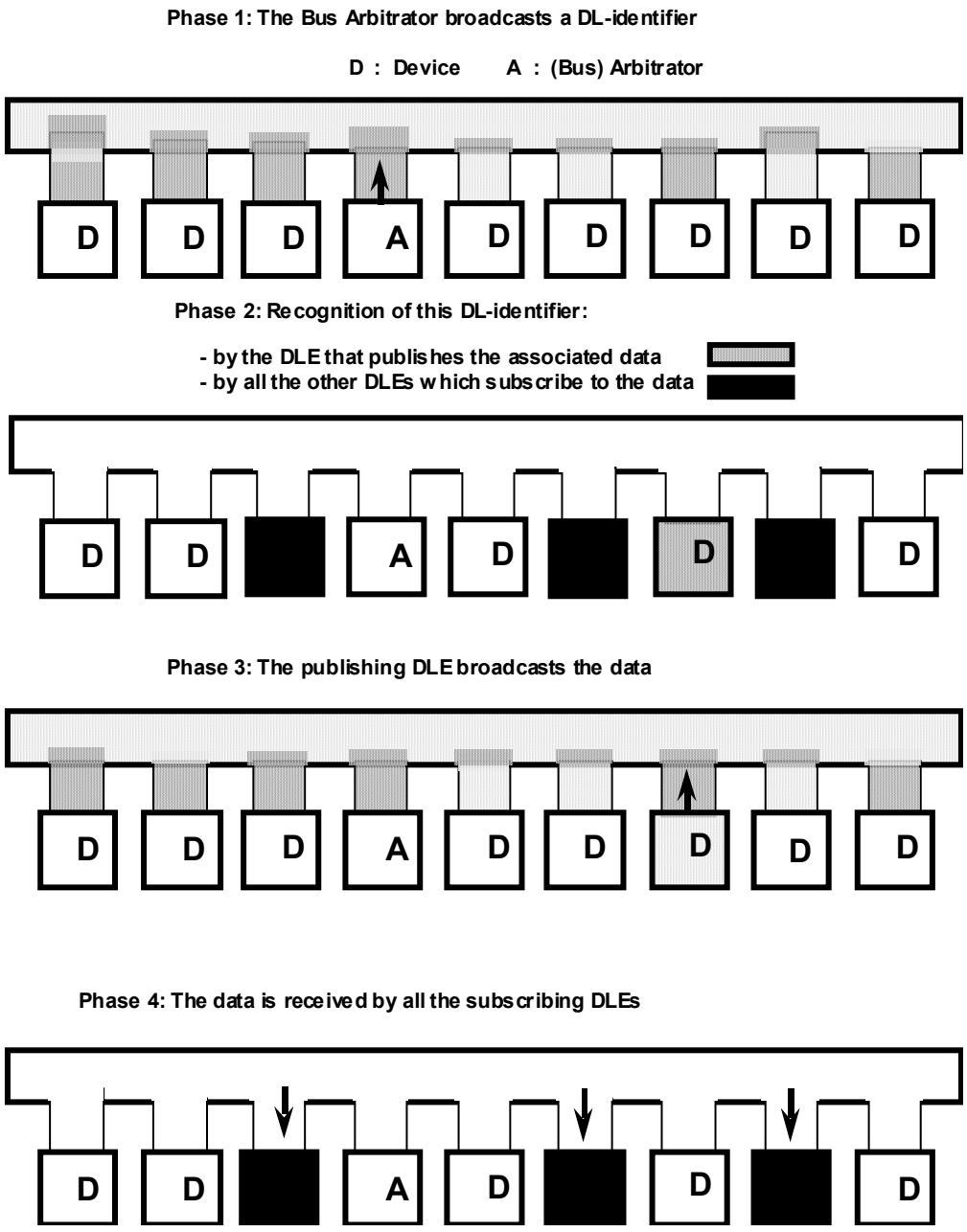


Figure 2 – General description of medium allocation

The medium access technique has the following characteristics:

— broadcasting of identified variables,

— increased efficiency in cyclical variable exchanges,

— parameters for medium sharing can be set by the user when the system is configured,

— guaranteed access time for cyclical variable exchanges, under all circumstances and regardless of the number of requests for triggered variable and/or message exchanges.

— possibility of triggering a transaction in accordance with a global clock, that is a clock that indicates the same time for all stations.

In addition, the medium access technique makes it possible to:

— give cyclical exchanges highest priority,

— respect the scanning period associated with each variable,

— give different priorities to triggered messages transfers and variable exchanges. These transactions are triggered in adjustable windows: the lengths of the "aperiodic variable" and "aperiodic message" windows are defined in terms of maximum limits set by the user when the system is configured.

— change the priority of aperiodic transactions by inserting them in the periodic window.

4.2 Types of entities

4.2.1 Producer/consumer entity

DLL services use various types of DLPDUs. Each DLPDU type is defined in the control field of the DLPDU. Interactions between a specific service and DLPDU types will be described in the detailed specifications of each service.

NOTE 5.5 describes all types of DLPDUs and also explains the use of various fields.

The functioning of an entity belonging to the highest conformance class requires:

— a mechanism for analyzing DLPDU type (use of the control field),

— a table of identifiers recognized upon emission,

— a table of identifiers recognized upon reception (both tables are defined at the interface between the DLL and system management),

a mechanism that provides read/write access to variables and messages.

The conceptual model of a data-link producer/consumer entity includes various buffers: queues needed to provide the services offered by the DLL, as shown in .Figure 3.

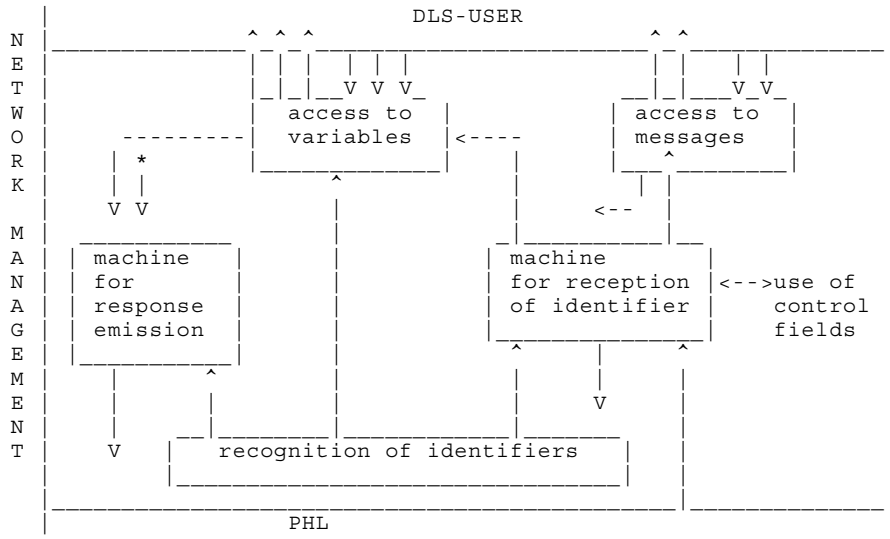


Figure 3 – Internal structure of a producer/consumer entity

The following are associated with each identifier produced:

— a buffer called B_DATprod, which contains the value of the variable produced,

— a buffer called B_REQ, containing the list of identifiers that are the object of an explicit request for a buffer transfer, if the identifier has been reserved for the explicit request for buffer transfer service,

— a queue called Q_MSGcyc, which contains messages to be emitted that are associated with cyclical message transfer, if the identifier has been reserved for cyclical message transfer,

— an indicator of the validity of the identifier produced (system management),

— for each of the B_DATprod and B_REQ buffers, an availability indicator,

— an indicator associated with Q_MSGcyc: queue filled or not,

— indicators used to manage explicit requests for variable exchange and message transfer requests:

— explicit request for buffer transfer in progress (RQ),

— priority associated with an explicit request (PR),

— scope of the explicit request (RQ_INHIBIT)

— message transfer request in progress (MSG) if the identifier has been reserved for aperiodic message transfer,

— reference to the queue of messages to be emitted associated with aperiodic message transfer.

The following are associated with each identifier consumed:

— a buffer called B_DATcons, which contains the value of the variable consumed,

— an indicator of the validity of the identifier consumed (system management),

— an indicator linked to the management of B_DATcons: availability of the buffer (access conflict),

NOTE The buffer availability indicator provides information concerning the integrity of data manipulated by service primitives.

Each entity that supports a message transfer request service also uses:

— a queue called Q_MSGaper, which is associated with aperiodic message transfer and contains messages to be emitted,

— a queue called Q_MSGrec that contains messages received.

— an indicator of queue status (full or not) is associated with the queue reserved for aperiodic message transfer.

In addition, each entity that supports acknowledged message service has the following characteristics:

— value of the source entity's even/odd bit,

— maximum value of the restart counter,

— current value of the restart counter.

The following are associated with the queue Q_MSGrec:

— an indicator of queue status (full or not),

— value of the destination entity's even/odd bit,

— value of the stored source address.

If the entity also supports the free explicit request for buffer transfer service, there are two global queues for holding free explicit requests for buffer transfer:

— Q_REQ1 is associated with urgent requests,

Q_REQ2 is associated with normal requests.

A status indicator (full or not) is associated with each queue.

Figure 4 shows many of these associated buffers and queues.

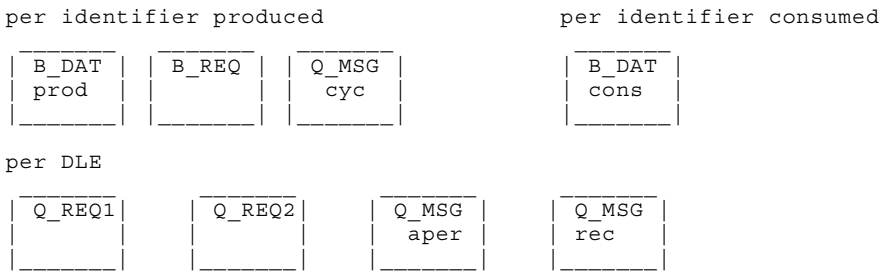


Figure 4 – Associated buffers and queues

Timers associated with the data-link protocol are also managed by the producer/consumer entity (see 5.6).

Annex B presents an object model that defines the attributes of a DLCEP-identifier.

4.2.2 Bus arbitrator entity

The function provided by the bus-arbitrator consists of "giving permission to speak" to each data producer. This permission is granted for three types of scanning:

— periodic or triggered periodic scanning of variables and messages,

— triggered scanning of variables,

— triggered scanning of messages.

The bus-arbitrator also provides the following functions:

— chaining of basic transactions,

— chaining of the various types of scanning,

— analysis of DLPDU control fields (the control field carries aperiodic variable and message requests),

— filling of aperiodic windows in accordance with these requests.

NOTE A basic transaction is made up of the succession of DLPDUs related to a single service.

In addition, the bus-arbitrator can provide a synchronization function to guarantee the constant length of a basic scanning cycle.

Each type of scanning takes place in a special window: respectively in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window.

The four windows constitute a basic scanning cycle.

Figure 5 provides an overview of the bus-arbitrator structure. A more complete description of the bus-arbitrator is given in 7.4.

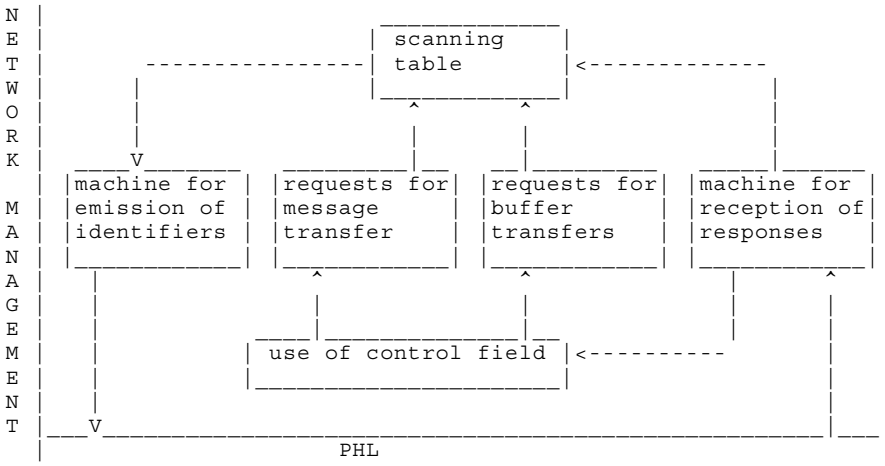


Figure 5 – Internal structure of a bus arbitrator

The conceptual model of the bus-arbitrator details the various tables and queues needed to provide the services offered by the DLL.

The bus-arbitrator manages according to system configuration, as shown in Figure 6:

— a scanning table with static portions and portions that are modified dynamically during scanning,

— a recovery buffer for the triggered periodic scanning of variables,

— a queue called Q_IDRQ1 for urgent explicit requests for buffer transfer,

— a queue called Q_IDRQ2 for normal explicit requests for buffer transfer,

— a queue for aperiodic transfers in progress (Q_RPRQ),

— a queue called Q_IDMSG for requests for message transfer,

— a basic padding sequence used to fill the synchronization window.

NOTE The management algorithm and/or calculation of these tables should provide equal access rights for the various entities, that is, starving certain entities should be avoided.

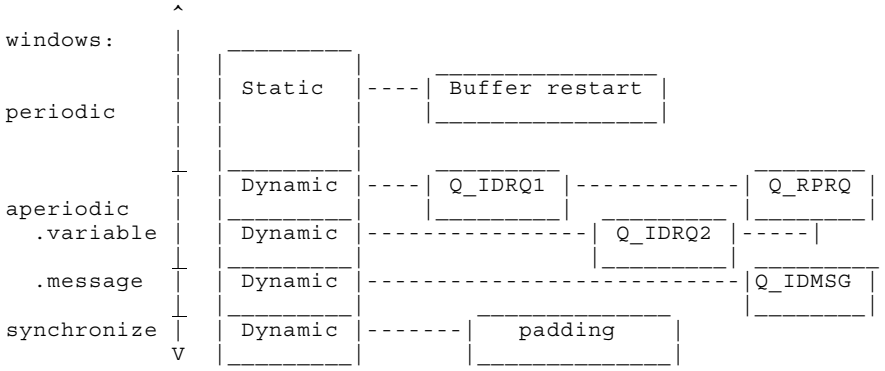


Figure 6 – Polling BA Table

4.3 Addressing

4.3.1 DLSAP-user relationship

The role of the DLL is to transmit information reliably and in accordance with a transmission protocol.

Within a single physical station several user entities have access to DLL services. Each of these entities uses a service access point (SAP) belonging to the DLL, in conformance with the ISO's static model in which each entity (N+1) is recognized by the address of the service access point (N) to which the entity is statically attached.

Several associations between user entities are possible on the level of access points to DLL services.

Thus, in accordance with the ISO model, a user entity (N+1) requests the establishment of a connection (N) in order to communicate with another user entity (N+1). The (N) entities, each of which is linked to one of the two (N+1) entities, create and manage this connection after negotiating it.

All exchanges between the entities (N+1) thus associated take place using this connection. The connection is identified locally at each SAP by a specific connection end point identifier (CEPi).

The number of DLSAPs corresponds to the number of entities in a station that are potential users of DLL services.

The number of CEPi per DLSAP corresponds to the number of potential communication links allocated to the (N+1) entity linked to the DLSAP.

4.3.2 OSI model relationship

This DLL follows the same rules as ISO/IEC 7498. Each potential user entity calls upon the services of the DLL through a DLSAP.

The DLL places two distinct addressing spaces as the disposal of the DLS-user:

— the addressing space concerning buffer transfer, each identifier being coded in 16 bits,

— the addressing space concerning message exchanges, which enables addressing messaging DLS-users, each address being coded in 24 bits.

The identifiers, encoded in 16 bits, allow addressing buffers within a local link, whereas the messaging addresses, encoded in 24 bits are meant to identify all the messaging DLS-users of the DL-subnetwork, that is those of all the DL-segments of the DL-subnetwork. Consequently, a communication system composed of n local links will have a single messaging address space but will have n identifier spaces to address the buffers.

Figure 7 illustrates this usage.

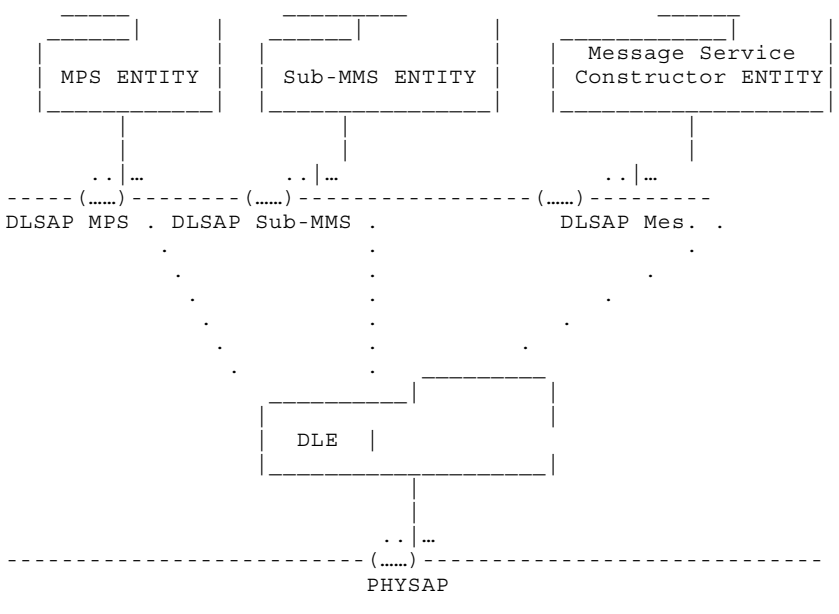


Figure 7 – Addressing scheme

4.3.2.1 Buffer transfer addressing

For reasons relating to performance and determinism, all associations between the various DLL user entities are known and defined when the system is configured.

Thus upon configuration the number of DLSAP is known, as well as the number of connections within the DLSAP.

This DLL uses broadcasting as its mode of transmission over the physical support. There is one producer and one or more consumers of each piece of information transmitted.

The connections within each DLSAP are multipoint connections providing unilateral communication (connections have more than two extremities, and over these connections data communication occurs in a single direction set in advance: from the producer to the consumers).

Connection end point identifiers (CEPi) that are known as identifiers and are encoded using 16 bits identify associations between user entities.

Thus CEPis are not recognized locally within a DLSAP, but globally throughout the DL-subnetwork, and the role of the DLSAP address is less important in the addressing model.

Thus the concept of an DLSAP is not the same as in the general ISO model. This DLL represents a group of between 1 and n identifiers employed by a user entity attached to the DLL. On the other hand, a DLSAP does not address the Application entity directly, but through the use of identifiers.

Each connection (identifier) can be used differently (according to the configuration and periodic or aperiodic services chosen) to communicate with other stations (variable transfers) or with the bus-arbitrator (requests for aperiodic transfers).

4.3.2.2 Message exchange addressing

4.3.2.2.1 Addressing requirements

The addressing of the messaging DLS-users must allow meeting the usual communication requirement such as indicated in the ISO and which are as follows:

— point-to-point communication between two DLS-users, whether or not located in the same local link,

— multipoint communication between an entity and all the entities of a group, the latter being located in the same equipment or the local link or the extended link,

— communication by distribution between a DLS-user and all the other DLS-users of a DLE, a local link or the extended link.

NOTE This corresponds to the reservation of a particular group containing all the entities of a device, a local link or the extended link. The addressing of the messaging DLS-users must distinguish between two types of addresses:

— the physical address, in order to identify an application entity with respect to its physical location, by means of the DL-segment number and the number of the station where it is located,

— the logic address, in order to identify an application entity whose physical location will not be required.

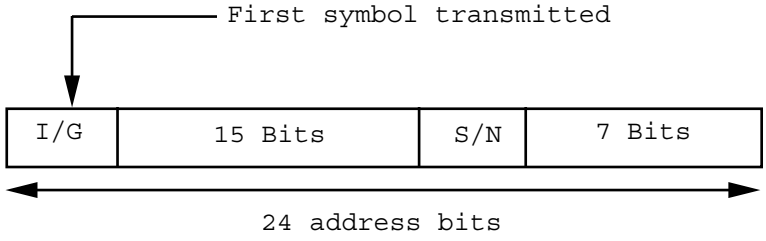
The notions of restricted groups on the DL-subnetwork with respect to groups of DLS-users in a device or in a DL-segment, have been defined with an aim to optimize the flow on the bus. Thus when addressing a group, it is thus necessary to have the possibility of a filter on the level of each bridge. The function of this filter is to prevent distribution on all the DL-segments of the extended link when this is not necessary.

The addressing of the buffers limits the number of devices on a DL-segment to 256. This must be taken into account in the messaging addressing mode.

4.3.2.2.2 Encoding of the addresses

The source and addressee DLS-users exchanging messages are identified in a DL-subnetwork with the help of addresses which are found in the link protocol data units (RP_MSG_NOACK and RP_MSG_ACK).

In order to meet the individual addressing requirements of the DLS-users in a device, in a group of devices of a local link or a group of devices of the extended link, the addressing space offered by the 24 bits is divided as shown in Figure 8.



I/G designates an individual DLS-user (I)or a group of DLS-users (G)

S/N designates a group of DLS-users on a DL-segment (S) or on the DL-subnetwork (N)

Figure 8 – Address partitioning

The encoding of the individual addresses and the groups of addresses is shown in Table 1.

Table 1 – Individual and group address encoding

I/G	S/N	Type of addressing
0	0	Individual address
1	0	Address of a group in a DL-segment
not significant	1	Address of a group in the DL-subnetwork

The link layer thus offers an addressing space which is divided into two sub-spaces:

— one containing link addresses meant to identify DLS-users individually,

— the other meant to identify DLS-user groups.

The group addresses are themselves divided into two sub-spaces:

— the sub-space defining addresses of restricted groups, thus enabling identification of the DLS-users which belong to the same local link,

— the other space defining the addresses of generalized groups, thus enabling identification of the application identities which all belong globally to the DL-subnetwork.

Each of these previously defined sub-spaces (individual addresses and addresses of restricted groups) are divided into:

— physical addresses,

— logical addresses.

4.3.2.2.2.1 Individual addressing

The individual addresses enabling a DLS-user to correspond with one and only one other DLS-user are structured as shown in Figure 9 and Figure 10:

a) Individual physical addresses:

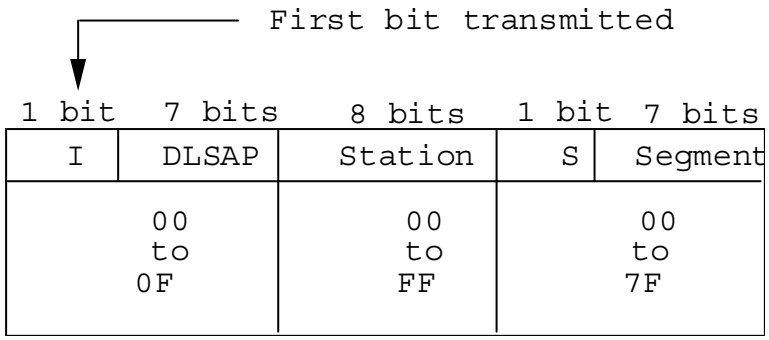


Figure 9 – Structure of an individual physical address

This structure allows identifying 16 physical DLSAPs per device.

b) Individual logical addresses:

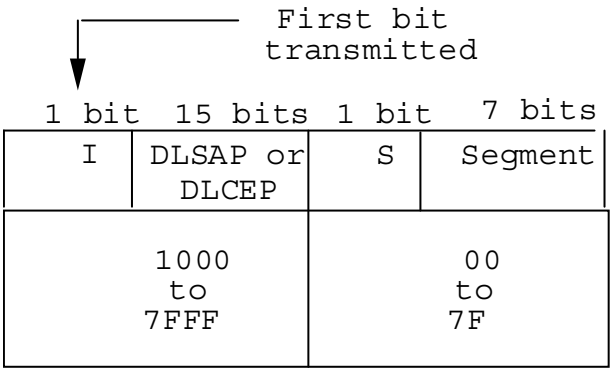


Figure 10 – Structure of an individual logical address

The addressing capacity is a total of 28K logical DLSAPs and DLCEPs per DL-segment.

4.3.2.2.2.2 Addressing of restricted groups

The restricted group addresses which enable a DLS-user to correspond with a group of DLS-users of the same device or of the same DL-segment are represented by their codes as shown in Figure 11 and Figure 12:

a) Addresses of restricted physical groups:

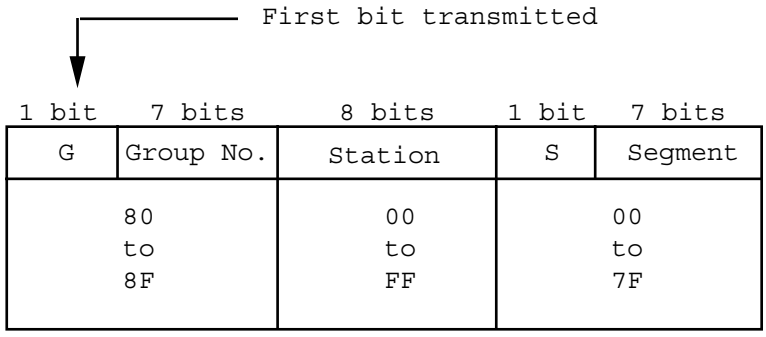


Figure 11 – Structure of restricted physical group address

The addressing capacity is then 16 numbers of physical restricted groups in a DLE. The distribution in a device is ensured by the specific group number, 8F.

b) Addresses of restricted logical groups:

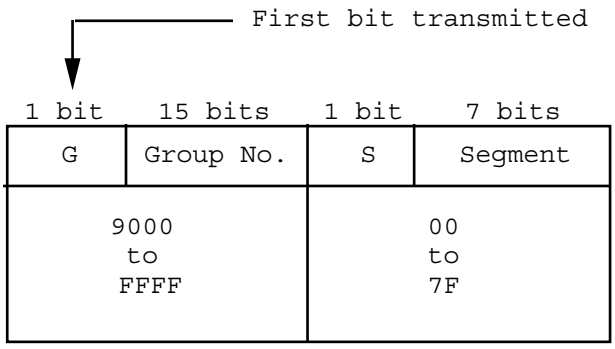


Figure 12 – Structure of a restricted logical group address

This structure allows addressing 28K numbers of restricted logical groups in a local link. FFFF corresponds to the distribution group in a DL-segment.

4.3.2.2.2.3 Addressing of generalized groups

The addresses of generalized groups allow a DLS-user to correspond with a group of DLS-users on the extended link. These addresses are divided into three sub-areas:

— an addressing sub-area which can be used by all the equipment of the extended link,

— a reserved sub-area,

— a vendor addressing sub-area.

The encoding structure of these addresses is as shown in Figure 13:

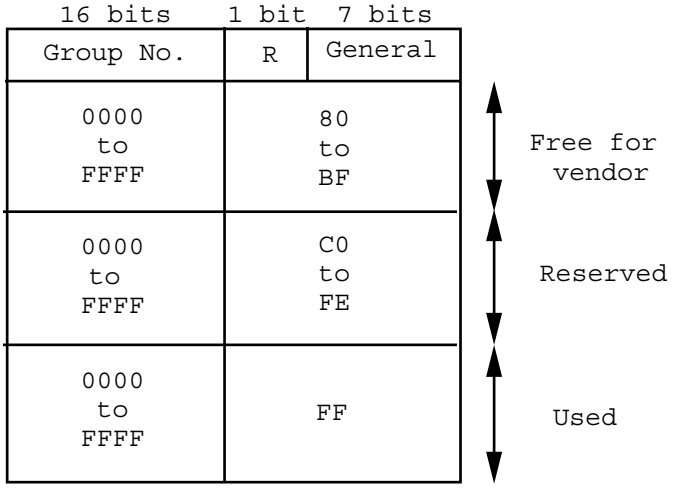


Figure 13 – Structure of a generalized group address

This addressing area allows identifying generalized 64K groups by sub-area element.

The distributionto the entire extended link is ensured by means of the particular group number FFFF.

4.3.2.3 Addressing capacity

The encoding rules offer the following possibilities for DL-subnetwork in terms of maximum capacity:

— 126 DL-segments in a DL-subnetwork;
— the "00" segment number is the DL-segment number by default;
— 256 DLEs per DL-segment in a DL-subnetwork;
— 16 physical DLSAPs in a DLE;
— 16 physical restricted group DL-addresses in a DLE;
— 28K logical DLSAPs in a local link;
— 28K logical restricted group DL-addresses in a local link;
— 64K generalized groups DL-addresses in the DL-subnetwork in the used sub-area.

The exact structure of the encoding of the beginning of an RP_MSG_XX message response DLPDU is as shown in Figure 14:

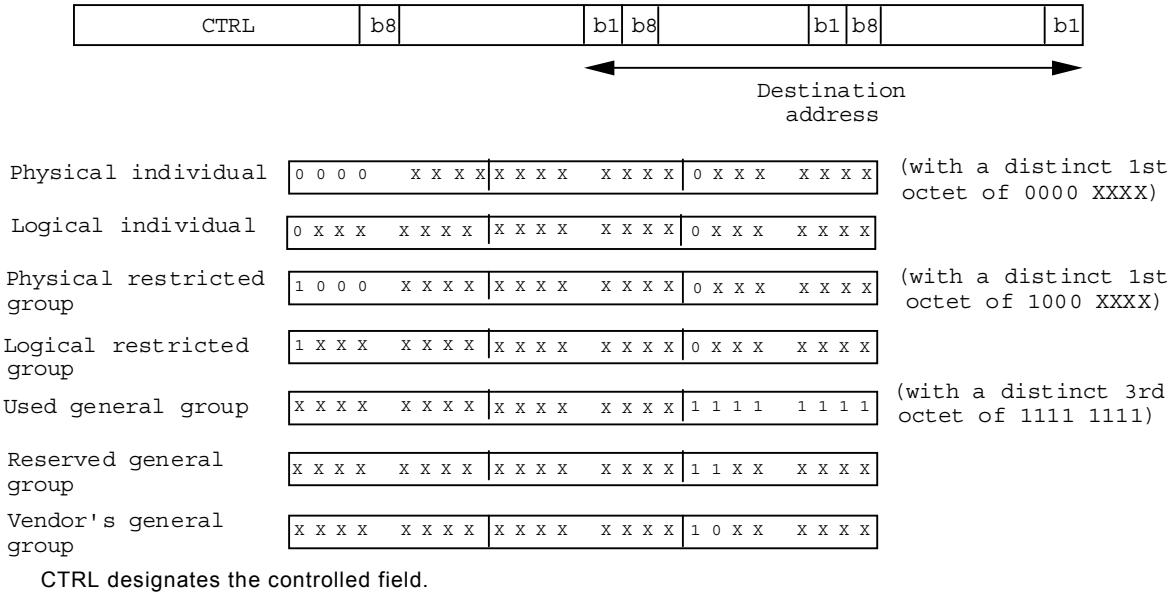


Figure 14 – Summary of address structure

The source address follows the destination address according to the same encoding.

4.4 Flow control

4.4.1 Variable exchanges

For the cyclical transfer of identified variables the principle of periodic scanning implies the replacement of an old variable value with a new variable independent of any reading access by the data consumer.

Management of flow control for these exchanges is definitively settled upon configuration:

— the bus-arbitrator manages flow control by respecting the scanning period associated with the variable,
— stations manage flow control by associating a buffer with each identified variable.

This principle also applies to triggered variable transfers since the user is only interested in the most recent validated value, and since the user associates a buffer with each identified variable. In addition, upon configuration of a DLL system a time delimiter is defined for the aperiodic window for triggered variable transfers for each of the bus-arbitrator's basic scanning cycles.

4.4.2 Message transfers

Flow control for message transfers is handled by the acknowledgement mechanism and by the queues for messages to be emitted and messages received.

When a DLL system is configured a time delimiter is defined for the aperiodic window for triggered message transfers for each of the bus-arbitrator's basic scanning cycles.

For acknowledged message transfers a portion of flow control is provided upon emission since the DLL only accepts requests for acknowledged message transfers if there is available space in the queue for messages to be emitted that is specified in the request. With this type of transfer there is also flow control upon reception by the queue of messages received and through the transmission of an acknowledgement DLPDU.

4.4.3 Detection of DLPDU duplication/loss

Detection mechanisms provided apply to errors caused by communication problems (cut-off DLPDUs, FCS error) or by out-of-service stations.

These errors include:

— loss of a DLCEP-identifier DLPDU,

— lack of a response DLPDU (DLPDU lost or station absent),

— loss of a request response DLPDU,

— loss of a message response DLPDU,

— loss of a message acknowledgement DLPDU,

— loss of a DLPDU indicating the end of a message transaction.

and can cause either the loss or the duplication of data.

DLPDU losses are taken into account in the graphs of final state machines defining data-link protocols.

Loss or duplication of DLPDUs can have the following impact on services:

— loss of a DLCEP-identifier DLPDU or a response concerning periodically exchanged variables is not usually detrimental since the value of the variable will be broadcast again during the next local link cycle;

— loss of an unacknowledged message DLPDU or of a DLPDU associated with a triggered variable exchange will only be handled by the upper layers;

— loss of an acknowledgement DLPDU results in a negative confirmation being sent to the initiating DLS-user;

— the concept of DLPDU duplication does not apply to periodically exchanged variables since the principle of cyclical variable refreshing implies that the same variable will be sent repeatedly;

— acknowledged message DLPDUs are protected from duplication by an even/odd numbering mechanism for messages;

— the duplication of a DLPDU associated with an explicit request for a buffer transfer will result in an additional overriding of the variable. This duplication is not detrimental since the concept of exchanging variables cyclically or upon explicit request is based on an asynchronous refreshing that is handled by the local link, and that the consumer cannot control. The semantics of data transmitted must be compatible with this working principle, that is, the variables exchange service is designed for the transmission of statuses, and not for the transmission of sequences of events.

4.5 Graphical representation

To be helpful for the reader some mechanisms are explained by a graphical representation. These representations are based on evaluation network built on an oriented graph with three types of nodes:

— the states, represented by a circle,

— the requests represented by a rectangle,

— the transitions, represented by a horizontal line.

The evaluation network is built according to the following principles, as illustrated in Figure 15:

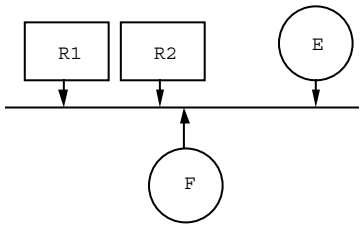


Figure 15 – Example of an evaluation net

When the described system is in state E, the arrival of request R1 or R2 results in a transition which switches it state F.

On the other hand, crossing a transition takes place, by convention, in zero time.

4.5.1 Simple actions

Simple actions can be associated with each transition, corresponding either to the transmission of requests or to the execution of local actions.

They are materially represented by a triangle, labeled with the name of the request or of the description of the executed action.

4.5.2 Conditions

The crossing of a transition can concern a condition. In this case, the oriented graph associated with the representation, shows a condition associated with the arc which precedes the transition.

A corollary of this representation allows defining choices for crossing a transition from the same initial state.

In the same way as for a simple transition, the transmission of an action can be associated with the crossing of a conditional transition.

5 General structure and encoding of PhIDUs and DLPDUs and related elements of procedure

5.1 DLPDU formats and components

This subclause describes the structure of DLPDUs and the use of the various control fields within the DLPDU. The term "DLPDU" refers to the protocol data units exchanged by data-link entities.

The basic structure of a DLPDU is as shown in Figure 16. The transmission and reception order is shown in Figure 17:

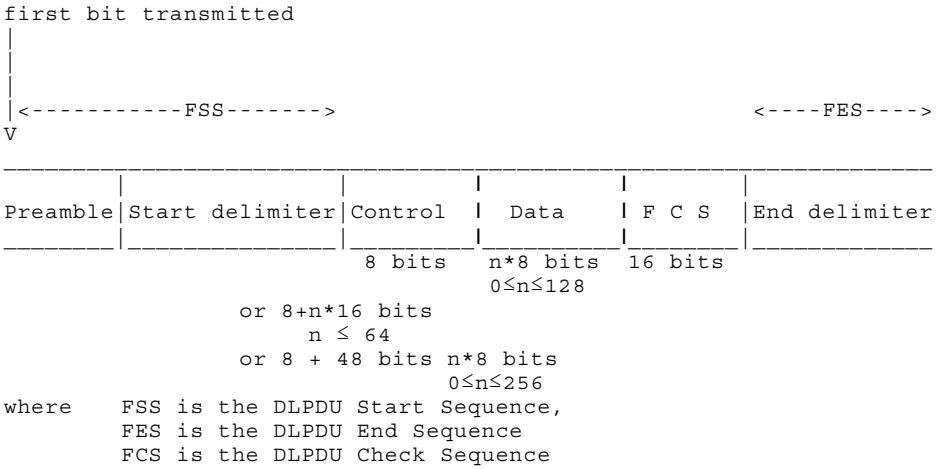


Figure 16 – Basic DLPDU structure

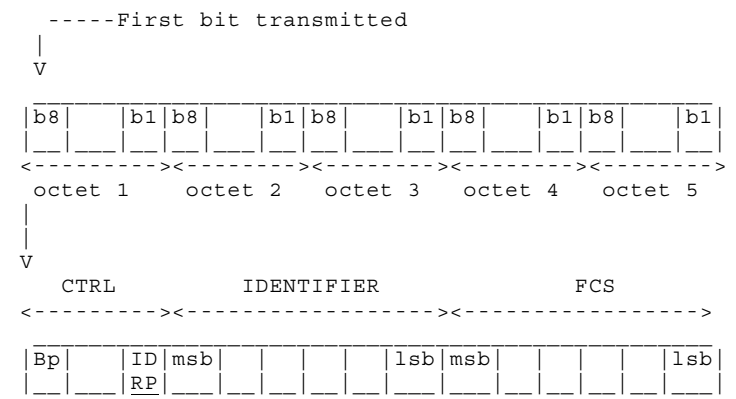


Figure 17 – DLPDU transmission / reception order

Octets are transmitted in the same order in which they are received (from the DLS-user or the PhL).

5.2 Description of each DLPDU component

5.2.1 Preamble

(See IEC 61158-2)

5.2.2 DLPDU delimiters

(See IEC 61158-2)

5.2.3 Control and addressing field

The control and addressing field specifies the type of DLPDU being exchanged:

— bit 1 of the control field indicates whether the DLPDU is a DLCEP-identifier DLPDU (bit 1 = 1) or a response DLPDU (bit 1 = 0)

— bits 2 through 7 of the control field, where each bit has a specific function, as shown in Table 2:

  – if bit 2 is set at 1 the DLPDU is related to a buffer transfer

  – if bit 3 is set at 1 the DLPDU is related to a message transfer

  – if bit 4 is set at 1 the DLPDU is related to an explicit request for a buffer transfer

  – if bit 5 is set at 1 the DLPDU is related to an acknowledgement

  – bit 6 indicates priority or the result of the acknowledgement

  – bit 7 set at 1 is used only to indicate an end of message transaction DLPDU;

— bit 8 of the control field (the first bit transmitted) is used only for a message response DLPDU or an acknowledgement response DLPDU. Bit 8 provides for even/odd numbering of messages to support duplicate message detection.

For a DLCEP-identifier DLPDU these bits specify the type of exchange in progress. For a response DLPDU these bits indicate the type of exchange in progress as well as requests for message transfer and explicit requests for variable exchanges, with their priorities.

Table 2 – DLPDU control-field coding

Bits 2 3 4 5 6 7	Identifier DLPDU
1 0 0 0 0 x	allocation for identified variable
0 1 0 0 0 x	allocation for message
0 0 1 0 1 x	allocation for urgent explicit request
0 0 1 0 0 x	allocation for normal explicit request
	Response DLPDU
1 0 0 0 0 x	identified variable response
1 1 0 0 0 x	identified variable response + message request
1 0 1 0 1 x	identified variable response + explicit urgent request
1 0 1 0 0 x	identified variable response + explicit normal request
1 1 1 0 1 x	identified variable response + explicit urgent request + message request
1 1 1 0 0 x	identified variable response + explicit normal request + message request
0 1 0 1 0 x	acknowledged message
0 1 0 0 0 x	unacknowledged message
0 0 0 1 1 x	positive acknowledgement
0 0 0 1 0 x	negative acknowledgement
0 0 1 0 1 x	list of identifiers for an urgent request
0 0 1 0 0 x	list of identifiers for a normal request
0 0 0 0 1 x	end of message transaction

For identifier DLPDUs the control and addressing field is extended to 8 + 16 bits. In this case it includes a single DL-identifier that represents a local-link DL-address encoded in 16 bits.

For request response DLPDUs the control and addressing field is extended to 8 + × times 16 bits. In this case it includes the list of DL-identifiers associated with the variables requested.

For message response DLPDUs the control and addressing field is extended to 8 + 24 + 24 bits. In this case it includes two extended-link DL-addresses — the destination and source, respectively, of the message.

Table 3 gives the correspondence between the name and coding of the control field's bits.

Table 3 – Correspondence between name and coding of 8 bits in the control field

Bits	Associated name
1 2 3 4 5 6 7 8	
1 1 0 0 0 0 × x	ID_DAT
1 0 1 0 0 0 × x	ID_MSG
1 0 0 1 0 1 × x	ID_RQ1
1 0 0 1 0 0 × x	ID_RQ2
0 1 0 0 0 0 × x	RP_DAT
0 1 1 0 0 0 × x	RP_DAT_MSG
0 1 0 1 0 1 × x	RP_DAT_RQ1
0 1 0 1 0 0 × x	RP_DAT_RQ2
0 1 1 1 0 1 × x	RP_DAT_RQ1_MSG
0 1 1 1 0 0 × x	RP_DAT_RQ2_MSG
0 0 1 0 1 0 × N	RP_MSG_ACK
0 0 1 0 0 0 × x	RP_MSG_NOACK
0 0 0 0 1 1 × N	RP_ACK+
0 0 0 0 1 0 × N	RP_ACK-
0 0 0 1 0 1 × x	RP_RQ1
0 0 0 1 0 0 × x	RP_RQ2
0 0 0 0 0 0 1 x	RP_END
NOTE N takes the value 0 or 1; x values are ignored but should be 0	

5.2.4 DLS-user-data field

Only response DLPDUs include a DLS-user-data field. The DLS-user-data field contains:

— either the value of a variable,

— or a message.

5.2.5 DLPDU frame check sequence (FCS) field

Within this subclause, any reference to bit *K* of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^K.

NOTE This is sometimes referred to as "little endian" bit numbering.

For the protocol Type of this standard, as in other International Standards (for example, ISO/IEC 3309, ISO/IEC 8802 and ISO/IEC 9314-2), DLPDU-level error detection is provided by calculating and appending a multi-bit frame check sequence (FCS) to the other DLPDU fields during transmission to form a "systematic code word"[1)] of length ***n*** consisting of ***k*** DLPDU message bits followed by ***n - k*** (equal to 16) redundant bits, and by calculating during

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

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 equation (6) and the polynomial for the receiver's expected residue is specified in equation (11). The specific polynomials for this standard are specified in Table 4. An exemplary implementation is discussed in Annex A.

Table 4 – FCS length, polynomial and expected residual

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

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

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

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

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

5.2.5.1 At the sending DLE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

That is,

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

NOTE 1 The L(X) terms are included in the computation to detect initial or terminal message truncation or extension by adding a length-dependent factor to the FCS.

NOTE 2 As a typical implementation when n-k = 16, the initial remainder of the division is preset to all ones. The transmitted message bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (7). 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.

5.2.5.2 At the receiving DLE

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

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

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

A remainder R(X) shall be computed for V(X), the received DLPDU and FCS, by a method similar to that used by the sending DLE (see 5.2.5.1) in computing F(X)

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

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

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

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

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

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

NOTE 2 As a typical implementation, the initial remainder of the division is preset to all ones. The received bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (7).

5.3 PhIDU structure and encoding

Each PhIDU consists of Ph-interface-control-information (PhICI) and in some cases one octet of Ph-interface-data. When the DLE transmits a DLPDU, it computes a DLPDU check sequence for the DLPDU as specified in 5.2.5, concatenates the DLPDU and DLPDU check sequence, and transmits the concatenated pair as a sequence of PhIDUs as follows:

a) The DLE issues a single Ph-DATA request primitive with PhICI specifying start-of-activity, and awaits the consequent Ph-DATA confirm primitive.

b) The DLE issues a sequence of Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the DLPDU as Ph-interface-data, from first to last octet of the DLPDU, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.

c) The DLE issues a sequence of two Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the FCS as Ph-interface-data, from first to last octet of the FCS, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.

d) The DLE issues a single Ph-DATA request primitive with PhICI specifying END-OF-DATA-AND-ACTIVITY, and awaits the consequent Ph-DATA confirm primitive.

The DLE forms a received DLPDU by concatenating the sequence of octets received as Ph-interface-control-information of consecutive Ph-DATA indications, computing a DLPDU check sequence for those received octets as specified in 5.2.5, and checks the syndrome of the computed FCS for correctness as follows:

e) The DLE receives a single Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, and initializes its computation of an FCS for the received DLPDU.

f) The DLE receives a sequence of Ph-DATA indication primitives with PhICI specifying DATA, each accompanied by one octet of the received DLPDU as Ph-interface-data, incrementally computes an FCS on the received octet, and concatenates all but the last two of those received octets to form the received DLPDU.

g) The DLE receives a single Ph-DATA indication primitive with PhICI specifying either END-OF-DATA, END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY, and checks the syndrome of the computed FCS for correctness:

   1) If the PhICI specified END-OF-DATA or END-OF-DATA-AND-ACTIVITY, and the computed FCS syndrome was correct, then the DLE reports the reconstructed DLPDU and the two octets of received FCS as a correctly-received DLPDU suitable for further analysis.

   2) Otherwise the DLE increments its management statistics to reflect the erroneously received DLPDU.

5.4 Common DLPDU structure, encoding and elements of procedure

Each DLPDU consists of a DLPDU control field which specifies the type of DLPDU and conveys small size (fractional octet) parameters of the DLPDU; zero to two explicit address fields, each containing a DL-address, all of the same length; and for most DLPDUs, a user data field conveying all or part of a DLSDU. To this is appended before transmission, and removed after reception, an FCS field used to check the integrity of the received DLPDU.

5.5 Valid DLPDU types

For easier reading, DLPDUs are described without their DLPDU start and DLPDU end sequences.

For each type of DLPDU a name is given for the control field. This name is shown in parenthesis.

The correspondence between this name and the 8-bit control field code is given in the table below.

For each type of DLPDU we have indicated the separation between fields that contain data-link protocol control information (DLPCI) and fields containing data (DLSDU).

5.5.1 Identifier DLPDU

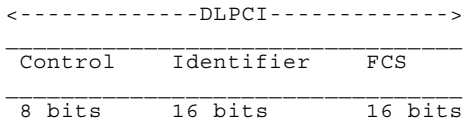


Figure 18 – Identifier DLPDU

There are four types of identifier DLPDU, whose basic structure is shown in Figure 18:

— medium allocation for identified variables (ID_DAT)

— medium allocation for message (ID_MSG)

— medium allocation for urgent explicit request (ID_RQ1)

— medium allocation for normal explicit request (ID_RQ2).

5.5.2 Variable response DLPDU

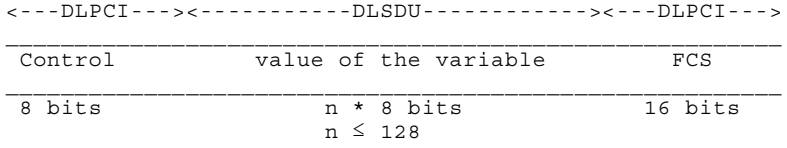


Figure 19 – Variable response DLPDU

There are six types of variable value response DLPDU, whose basic structure is shown in Figure 19:

— identified variables (RP_DAT)

— identified variables + message request (RP_DAT_MSG)

— identified variables + urgent explicit request (RP_DAT_RQ1)

— identified variables + normal explicit request (RP_DAT_RQ2)

— identified variables + urgent explicit request + message request (RP_DAT_RQ1_MSG)

— identified variables + normal explicit request + message request (RP_DAT_RQ2_MSG).

5.5.3 Request response DLPDU

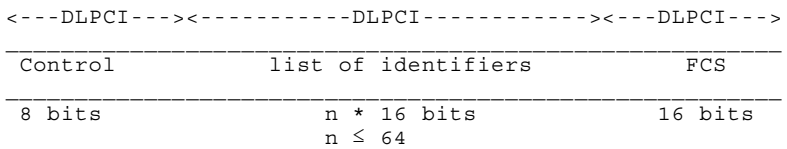


Figure 20 – Request response DLPDU

There are two types of request response DLPDU, whose basic structure is shown in Figure 20:

— list of urgent identifiers (RP_RQ1)

— list of normal identifiers (RP_RQ2).

5.5.4 Message response DLPDU

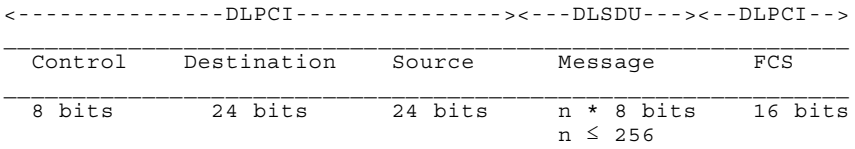


Figure 21 – Message response DLPDU

There are two types of message response DLPDU, whose basic structure is shown in Figure 21:

— acknowledged message (RP_MSG_ACK)

— unacknowledged message (RP_MSG_NOACK).

5.5.5 Acknowledgement response DLPDU

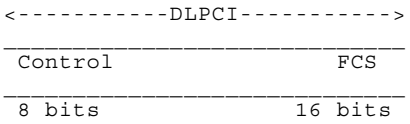


Figure 22 – Acknowledgement response DLPDU

There are two types of acknowledgement response DLPDU, whose basic structure is shown in Figure 22:

— positive acknowledgement (RP_ACK+) indicates that the message has been correctly received by the remote DLE,

— negative acknowledgement (RP_ACK-) indicates that the distant entity's queue for received message is filled.

5.5.6 End of message transaction response DLPDU

Figure 23 – End of message transaction response DLPDU

There is one type of end of message transaction response DLPDU, whose basic structure is shown in Figure 23:

NOTE The chaining of basic DLPDUs is indicated in the specifications for each service.

The RP_END DLPDU involves 2 entities: the source entity and the bus-arbitrator. The source entity returns control of the local link to the bus-arbitrator at the end of a message transaction.

5.6 DLL timers

This subclause briefly describes the timers involved in the description of data-link final state devices, proposes relationships between the various timers, and describes the importance of a station's turnaround time in the definition of its timers.

5.6.1 Common criteria

The criterion common to all stations for launching timers is the absence of activity on the local link.

Two signals are used to evaluate this criterion: "Absence of activity" (SILENCE) and "Signal present" (BUSY).

A reference timer T0 is defined as corresponding to the maximum silent time permitted on a local link.

The T0 timer is a global systems parameter.

T0 is activated upon reception of an indication of the absence of activity. This indication is emitted by the Type 2 PhL (the primitive PhL-DATA indication (SILENCE)).

T0 is deactivated by any correct functioning of the local link, upon reception of the indication of signal presence. This indication is emitted by the Type 2 PhL (the primitive PhL-DATA indication (BUSY)).

The maximum silent time permitted on the local link, that is, T0 is linked to the turnaround time of stations in the system.

5.6.2 Turnaround time

The reception or emission of the last MAC symbol is signaled by a SILENCE indication from the PhL. Likewise, the reception or emission of the first MAC symbol is signaled by an BUSY indication from the PhL.

The time lapsed between these two signals within a single station defines the station turnaround time (STT), whether the station's function be that of producer/consumer or bus-arbitrator.

Turnaround time takes into account parameters that are local to the station and parameters that are associated with the system to which the station belongs. These parameters are:

— physical reaction time (PRT): the time the PhL needs to detect the end or beginning of activity. These times depend on physical local link parameters:

  station: time needed to detect the absence or presence of a signal

  support: time needed to traverse the media and active copper or optical stars

— latency time (TI) resulting from processing time in the station.

TI takes on various values depending on the protocol situation in which the station finds itself:

a) the station has emitted a DLPDU and will emit the next DLPDU,

b) the station has received a DLPDU and will emit the next DLPDU,

c) the station has emitted a DLPDU and will receive the next DLPDU,

d) the station has received a DLPDU and will receive the next DLPDU.

All the parameters combine to form a minimum station turnaround time (under the most favorable conditions) and a maximum station turnaround time (under the least favorable conditions).

For the system to function correctly the following criteria *must* be respected:

Rule 1) a station that receives a DLPDU must always have a shorter turnaround time than the station that emitted the DLPDU, regardless of the two stations (receiving station, emitting station) involved.

This condition is expressed by:

Max(i) STT receiving station i ≤ Min(j) STT emitting station j

Rule 2) minimum turnaround time corresponds to the maximum value of physical reaction time PRT.

Rule 3) given the impact of a station's turnaround time on transmission yield (use of the bandwidth), it is important that maximum STT take yield criteria into account.

Rules 1 and 2 are interworking criteria. Rule 3 is a performance criterion that can be verified on the level of system management through access to the maximum turnaround time value and the resulting T0 timer.

NOTE The choice of minimum and maximum STT values can result in different interoperability and performance classes.

5.6.3 Timer functions

Five timers are defined using T0 as a basis.

The T1, T4, and T6 timers correspond to the use of T0 in the specific context of the status of a DLE (Station or bus-arbitrator). Activation and deactivation conditions are thus identical (SILENCE or BUSY), and the action after expiration of the timers differs according to the status of the DLE.

The T3 and T5 timers also refer to T0: their lengths (which are different from T0) are calculated using T0 length as a basis. In addition, they are activated and deactivated under the same conditions (SILENCE or BUSY).

An additional T2 timer is also used. T2 is not based on T0. This timer determines the length of a basic synchronized cycle.

All of these timers are listed in Table 5

Table 5 – DL-Timers

Name of timer	Location	Function
T1	B.A.	monitors the absence of RPxx after IDxx
T2	B.A.	monitors the filling of the synchronization window by identifiers from the basic padding sequence
T3	B.A.	monitors the absence of IDxx after RPxx switching of bus-arbitrators
T4	consuming station	monitors the absence of RP_DATxx after ID_DAT
T5	B.A.	monitors the lack of RP_END after ID_MSG
T6	message source station	monitors the absence of RP_ACK after RP_MSG_ACK

Timer definitions refer to names of statuses mentioned in the protocol descriptions presented in Clause 7.

5.6.3.1 Timer T1

— Is located in the active bus-arbitrator

— Its purpose is to monitor the emission of a variable response DLPDU or a request response DLPDU within a given time.

— Is active when the bus-arbitrator is waiting for a variable response DLPDU or a request response DLPDU (BA_WAITING_DAT or BA_WAITING_RQ status), after its emission of a variable identifier or request identifier DLPDU.

— The length of timer T1 is equal to the length of timer T0.

— Expiration of the timer gives the bus-arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

5.6.3.2 Timer T2

— Is located in the active bus-arbitrator

— Its purpose is to monitor the length of the synchronization window of the basic cycle. The active bus-arbitrator emits the identifier DLPDUs of the basic padding sequence as long as the T2 timer has not expired.

— Is activated at the beginning of each basic synchronized cycle, when the bus-arbitrator takes on the status of emitting identifier DLPDUs (NEXT status after BA_WAITING_SYNC status).

— Is not deactivated by any event.

— The length of the synchronization window is a function of the basic cycle's aperiodic exchanges or message services;

— Expiration of the timer gives the bus-arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

5.6.3.3 Timer T3

— Is located in potential bus-arbitrators

— Its purpose is to monitor the active bus-arbitrator's emission of identifier DLPDUs within a given time.

— Is active as long as the bus-arbitrator has potentially active status.

— The length of the T3 timer should be superior to the length of the T0 timer. It should have a different value for each of the potential bus-arbitrators. The note below suggests a value for this length.

— Expiration of the timer triggers the activation of a bus-arbitrator.

NOTE T3 timer dimensions are set as follows:

$T3 = k \times n \times T0$ with
n>2 a function of the geographic address of the bus-arbitrator to which T3 is attached and
k a coefficient that allows the bus-arbitrator's turnaround time to be covered.

5.6.3.4 Timer T4

— Is located in stations

— Its purpose is to monitor the emission of a variable response DLPDU on the bus within a given time, in order for example to correctly associate a DLCEP-identifier DLPDU received with the corresponding response DLPDU.

— Is active when the station is waiting for a variable response DLPDU (WAITING_DAT status).

— The length of T4 is equal to the length of T0

— Expiration of the timer causes the station to be placed in the status of waiting for the next identifier DLPDU (WAITING_ID status).

5.6.3.5 Timer T5

— Is located in the active bus-arbitrator

— Its purpose is to monitor a message source station's emission of a response DLPDU indicating the end of a message transaction within a given time.

— Is active when the bus-arbitrator is waiting for an end of message response DLPDU (BA_WAITING_END status), after its emission of a message identifier DLPDU.

— The length of timer T5 should be greater than the length of timer T0. The note below suggests a value for T5.

— Expiration of the timer gives the bus-arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

NOTE The length of timer T5 is equal to two times the length of timer T0.

With such a choice of lengths collision between two operators is avoided. For example, when an acknowledgement DLPDU RP_ACK is lost the bus-arbitrator cannot emit a second ID_MSG at the same time as the source station answers with the emission of RP_MSG_ACK (see T6).

5.6.3.6 Timer T6

— Is located in stations

— Its purpose is to monitor the emission of an acknowledgement DLPDU by the destination station within a given time.

— Is active when the source station is waiting for an acknowledgement DLPDU (WAITING_ACK status) after the end of its emission of an acknowledged message response DLPDU.

— The length of T6 is equal to the length of T0

— Expiration of the timer causes the station to be placed in the status of re-emitting the message if possible (RESTART_SOURCE status) or else in the status of waiting for the next identifier DLPDU (WAITING_ID status) after emitting the end of message transaction DLPDU.

The definition of these timers is necessary to guarantee the ability to interoperate of the various devices connected to a DLL local link.

All the timers within a single local link should have the same value. A default value is defined. These values can be modified by a station's system management as long as the equivalence of values is respected. Values will be agreed upon later in accordance with implementation needs.

6 DLPDU-specific structure, encoding and element of procedure

6.1 General

This clause describes the structure, contents of the PDU and specifies elements of procedure related to the invoked service.

Behavior of particular cases is described in detail in the relative subclause by the use of dedicated state machines.

6.2 Buffer read

Buffer reading service is local to an entity since the interactions related to this service take place only through the DLS-user/DLL interface, as shown in Figure 24. Correspondence

between DLSDU service data units and interactions between the DLL and the communication medium are provided by the buffer transfer service.

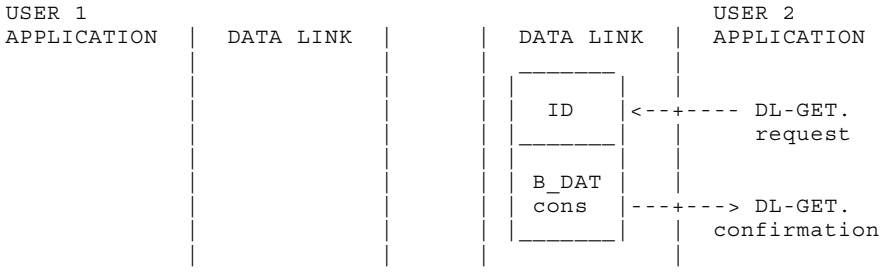


Figure 24 – Buffer reading service interactions

6.3 Buffer write

Buffer writing service is local to an entity since the interactions related to this service take place only through the DLS-user/DLL interface, as shown in Figure 25. Correspondence between DLSDU service data units and interactions between the DLL and the communication medium are provided by the buffer transfer service

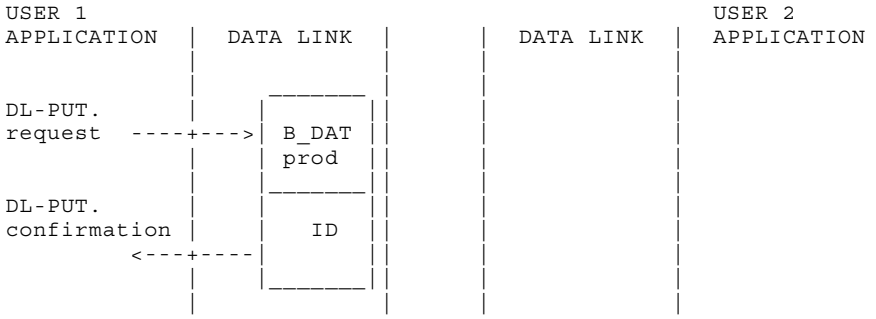


Figure 25 – Buffer writing service interactions

6.4 Buffer transfer

There are two phases to buffer transfer for all variables, as shown in Figure 26:

— the bus-arbitrator emits the identifier DLPDU associated with the variable,

— the producing entity emits the corresponding response DLPDU and generates a DL-SENT indication primitive for the DLS-user. The value of the variable is picked up by all consuming entities and a DL-RECEIVED indication primitive is generated and passed on to each of the DLS-users concerned.

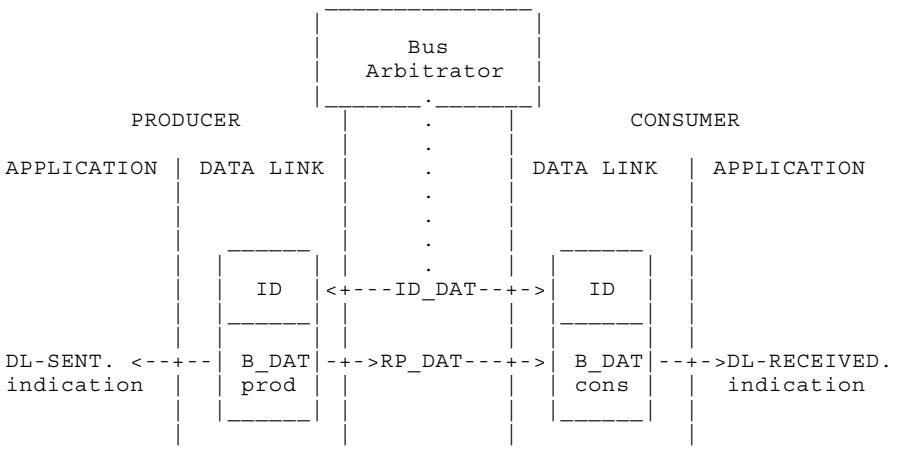


Figure 26 – Buffer transfer service interactions

6.4.1 Associated DLPDUs

A buffer transfer involves the circulation of two DLPDUs.

The bus-arbitrator broadcasts a DLCEP-identifier DLPDU (ID_DAT) that allocates the medium to the entity that produces the associated variable. The producing entity then broadcasts the variable in a response DLPDU (RP_DAT) that is picked up by the consuming entity or entities. The phases of this sequence are shown in Figure 27.

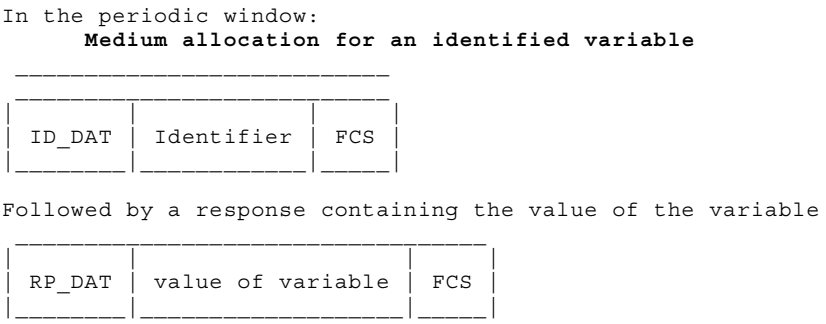


Figure 27 – Buffer transfer DLPDU sequence

NOTE A description of all DLPDUs, and the use of the various fields, can be found in 4.5.

6.5 Specified explicit request

A specified explicit request for a variable exchange can be initiated by any entity. The request triggers exchanges whose form depends on the value of the indicator RQ_INHIBIT associated with SPECIFIED_IDENTIFIER. This value is set upon configuration.

If RQ_INHIBIT = FALSE, there are three phases to the exchange:

phase 1:

Following a DL-SPEC-UPDATE request primitive, the initiating entity uses the control field of the variable response DLPDU (corresponding to the identifier specified when the service was requested) to ask the bus-arbitrator to circulate an urgent request identifier.

phase 2:

The bus-arbitrator emits the corresponding request identifier in the window allocated to triggered variable exchanges (aperiodic window),

phase 3:

The initiating entity emits a request response DLPDU whose "data" field contains the list of identifiers that the entity would like broadcast. The initiating entity also transmits a DL-SPEC-UPDATE confirm primitive to the DLS-user. One or more buffer transfers then take place within the aperiodic traffic cycle.

— the bus-arbitrator emits the identifiers of the variables requested,

— the producer of each variable requested emits a variable response DLPDU.

If RQ_INHIBIT = TRUE, there are two phases to the exchange:

phase 1:

Regardless of requests, for all identifiers set aside for specified service whose RQ_INHIBIT indicator is "true" the bus-arbitrator emits a request identifier in the periodic window.

phase 2:

The initiating entity emits a request response DLPDU whose "data" field contains the list of variable identifiers that the entity would like broadcast. The initiating entity also transmits a DL-SPEC-UPDATE confirmation primitive to the DLS-user. One or more buffer transfers then take place within the periodic traffic cycle immediately following the request.

— the bus-arbitrator emits the identifiers of the variables requested,

— the producer of each variable requested emits a variable response DLPDU.

The interaction sequences for the two classes of non-erroneous operation are shown in Figure 28 and Figure 29.

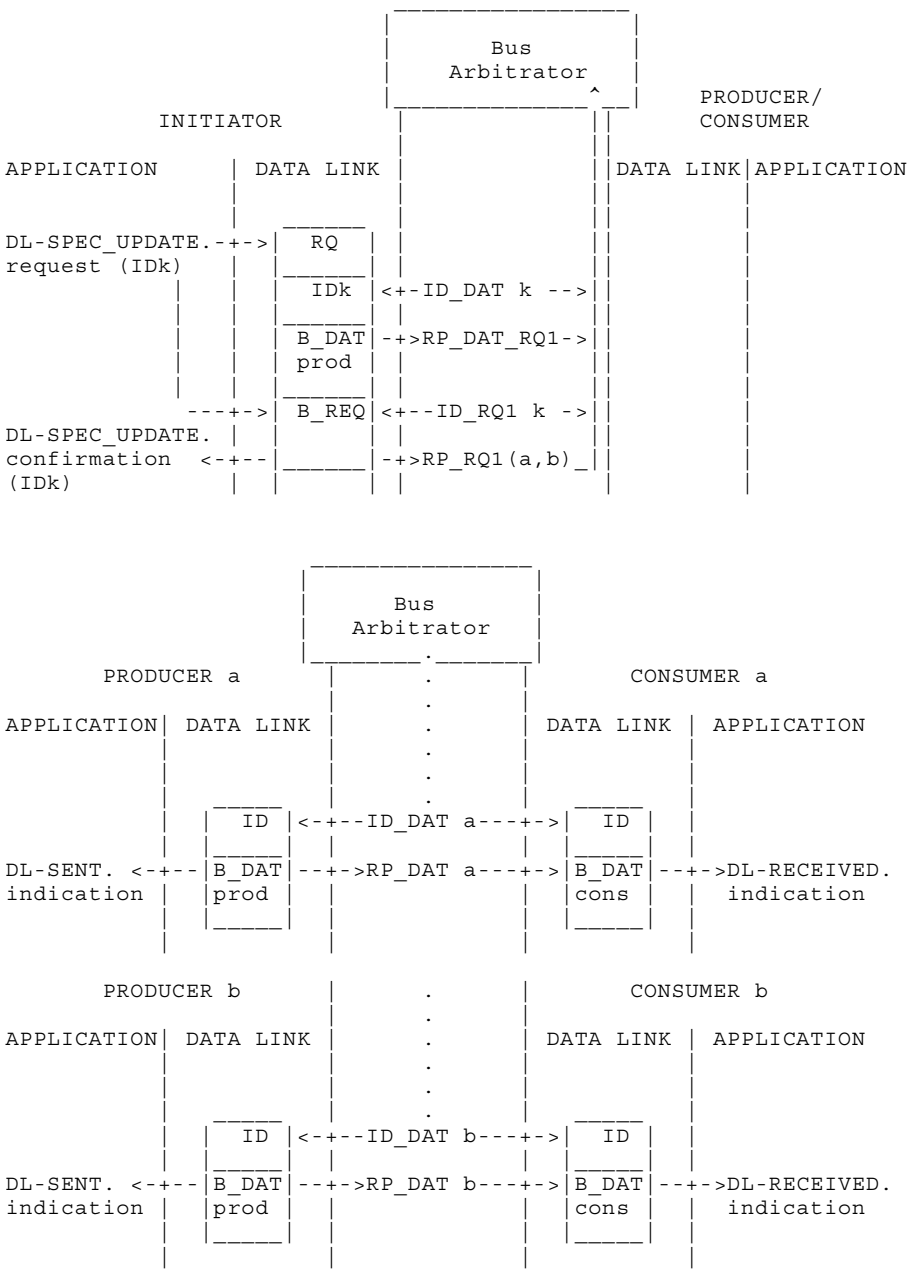


Figure 28 – Interactions within the specified explicit request for buffer transfer service in the aperiodic window

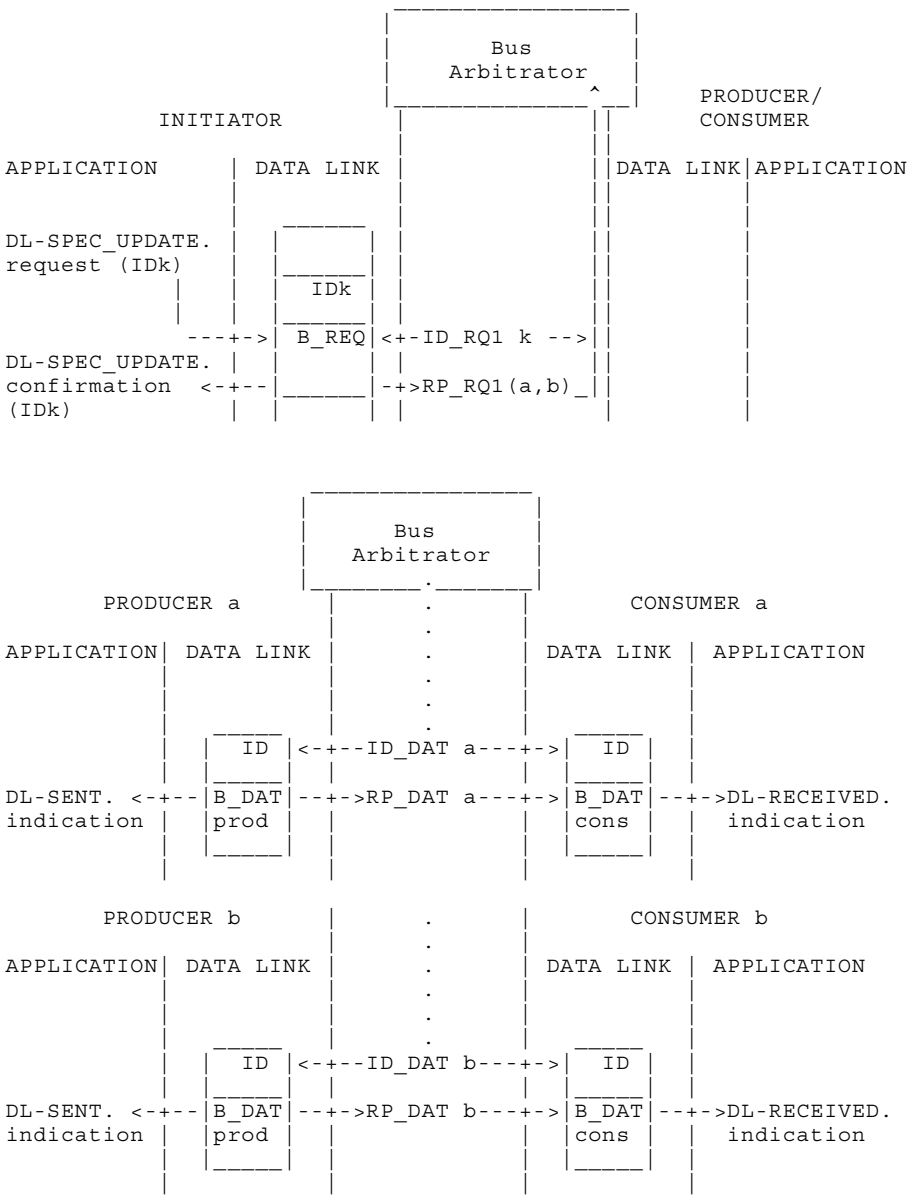


Figure 29 – Interactions within the specified explicit request for buffer transfer service in the periodic window

6.5.1 Associated DLPDUs

An explicit request for a buffer transfer corresponds to the circulation of three DLPDUs. The first of these DLPDUs is part of normal periodic scanning.

The initiating entity broadcasts a variable exchange response DLPDU whose control field includes the request to the bus-arbitrator to circulate a request identifier. The initiating entity states the priority level (RP_DAT_RQi, or RP_DAT_RQi_MSG if an aperiodic message service is also requested with this identifier).

The bus-arbitrator then broadcasts a request identifier DLPDU (ID_RQi) during an aperiodic window. The bus-arbitrator uses the identifier which carried the request.

The list of variable identifiers to be broadcast is sent to the bus-arbitrator (RP_RQi).

The bus-arbitrator responds with one or more buffer transfers: (ID_DAT, RP_DAT) transaction.

If the specified explicit request is part of the bus-arbitrator's periodic scanning cycle, the circulation of the first DLPDU is null and the other DLPDUs are exchanged during aperiodic scanning.

Figure 30 shows the DLPDU sequence for an explicit request for a transfer:

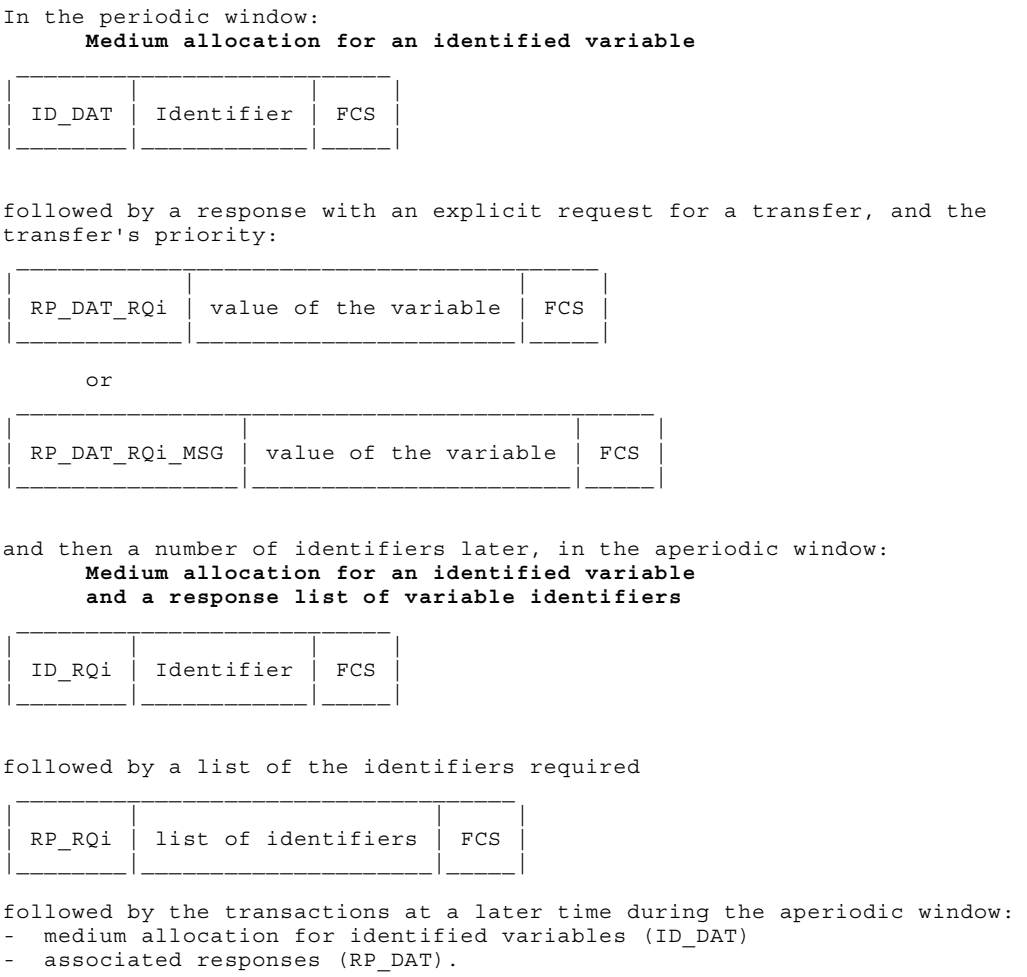


Figure 30 – DLPDU sequence for an explicit request for a transfer

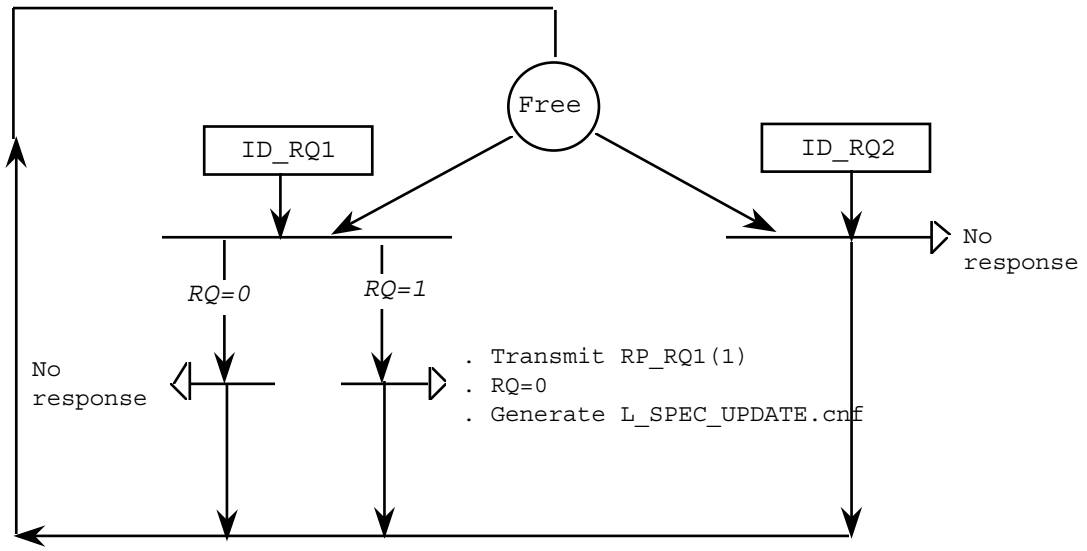
6.5.2 Behavior with mismatched parameters

6.5.2.1 Buffer transfer specified explicit request (RQ_INHIBITED = False)

The evaluation network of Figure 31 allows specifying the behavior of a producer/consumer entity in the following cases:

— reception of an ID_RQ2 DLPDU containing a DLCEP-identifier bearing a buffer transfer exchange specified explicit request (RQ=1),

— reception of an ID_RQ1 DLPDU containing a DLCEP-identifier not bearing a buffer transfer specified explicit request (RQ=0).



NOTE 1 When RQ=1 and B_REQ contains an empty list, RP_RQ1 is transmitted with an empty list.

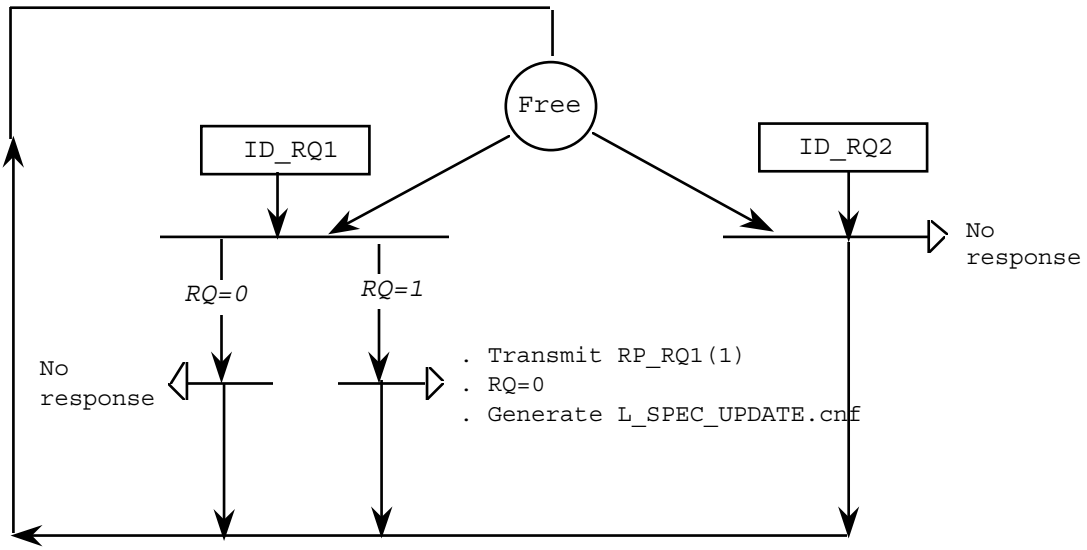
Figure 31 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = False)

6.5.2.2 Buffer transfer specified explicit request (RQ_INHIBITED = True)

The evaluation network of Figure 32 allows specifying the behavior of a producer/consumer entity in the following cases:

— reception of an ID_RQ2 DLPDU containing a DLCEP-identifier configured for the service of the buffer transfer specified explicit request such as RQ_INHIBITED=TRUE,

— reception of an ID_RQ1 DLPDU containing a DLCEP-identifier configured for the service of the buffer transfer specified explicit request such as RQ_INHIBITED=TRUE and whose content has already been transmitted on the bus.



NOTE 1 When RQ=1 and B_REQ contains an empty list, RP_RQ1 is transmitted with an empty list.

Figure 32 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = True)

6.6 Free explicit request

A free explicit request for the exchange of an identified variable can be initiated by any entity and takes place in three phases.

phase 1:

Following a DL-FREE-UPDATE request primitive, the initiating entity uses the control field of the variable response DLPDU to ask the bus-arbitrator to circulate a request identifier. The entity indicates priority.

phase 2:

The bus-arbitrator emits the corresponding request identifier in the window allocated to triggered variable exchanges (aperiodic window),

phase 3:

The initiating entity emits a request response DLPDU whose "data" field contains the list of variable identifiers that the entity would like broadcast. The initiating entity also transmits a DL-FREE-UPDATE confirmation primitive to the DLS-user.

One or more buffer transfers then take place within the aperiodic traffic cycle.

The interaction sequences for non-erroneous operation is shown in Figure 33.

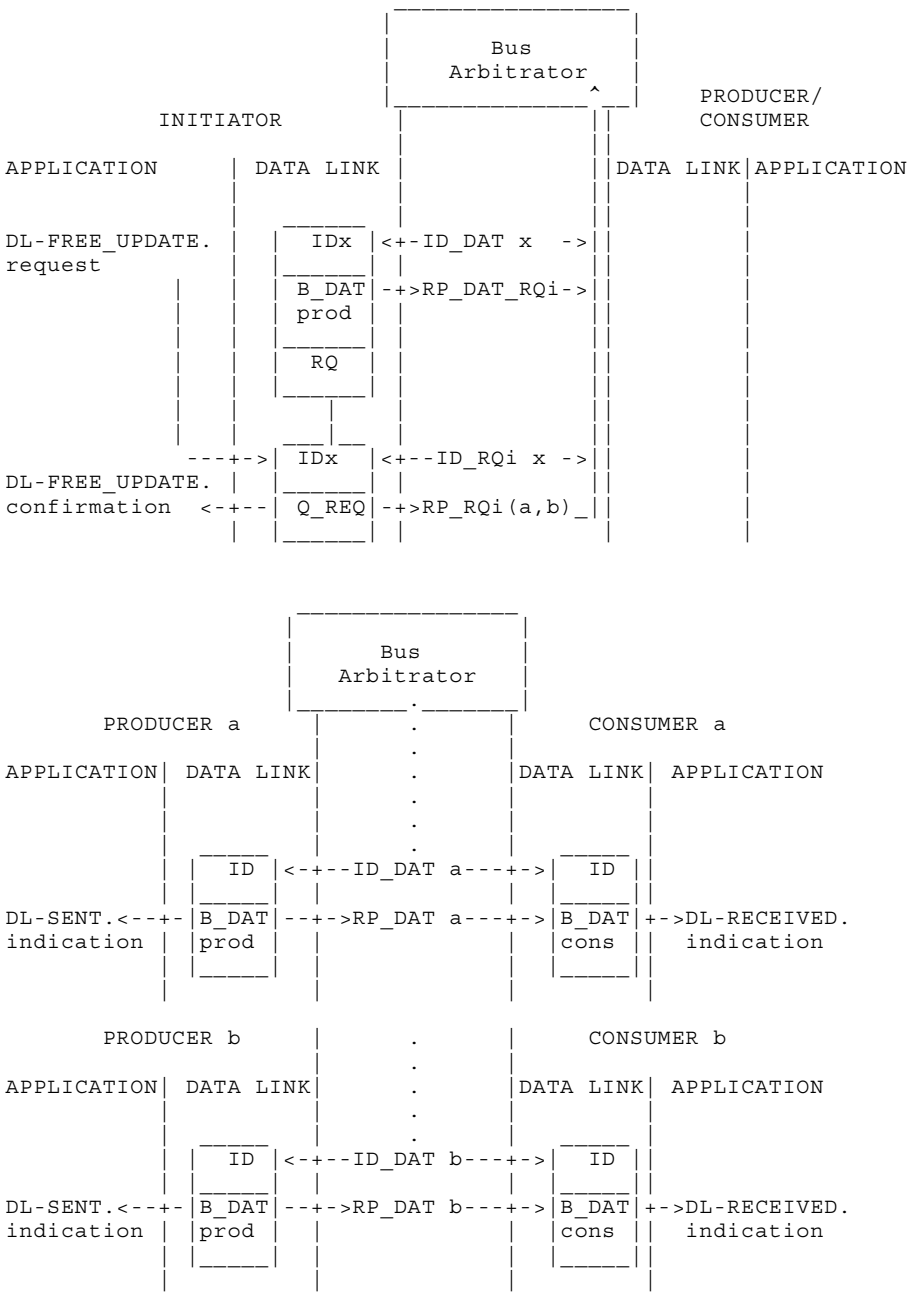


Figure 33 – Diagram of interactions within the free explicit request for buffer transfer service

6.6.1 Associated DLPDUs

A free explicit request for a buffer transfer corresponds to the circulation of three DLPDUs, as shown in 6.5.1. The first of these DLPDUs is part of normal periodic scanning.

The initiating entity broadcasts a variable exchange response DLPDU whose control field includes the request to the bus-arbitrator to circulate a request identifier. The initiating entity states the priority level (RP_DAT_RQi, or RP_DAT_RQi_MSG if an aperiodic message service is also requested with this identifier).

The bus-arbitrator then broadcasts a request identifier DLPDU (ID_RQi) during an aperiodic window. The bus-arbitrator uses the identifier, which carried the request.

The list of variable identifiers to be broadcast is sent to the bus-arbitrator (RP_RQi).

The bus-arbitrator responds with one or more buffer transfers: (ID_DAT, RP_DAT) transaction.

If the specified explicit request is part of the bus-arbitrator's periodic scanning cycle, the circulation of the first DLPDU is null and the other DLPDUs are exchanged during period scanning.

6.6.2 Behavior with mismatched parameters

The evaluation network of Figure 34 allows specifying the management of a producer/ consumer entity in the following cases.

— Reception of an ID_RQ1 or ID_RQ2 DLPDU containing a DLCEP-identifier configured for service of the buffer transfer free explicit request and not associated with Q_REQ1 or Q_REQ2 (RQ of the identifier has a value of 0).

— Reception of an ID_RQ1 or ID_RQ2 DLPDU containing a DLCEP-identifier associated with an empty Q_REQ1 or Q_REQ2 queue.

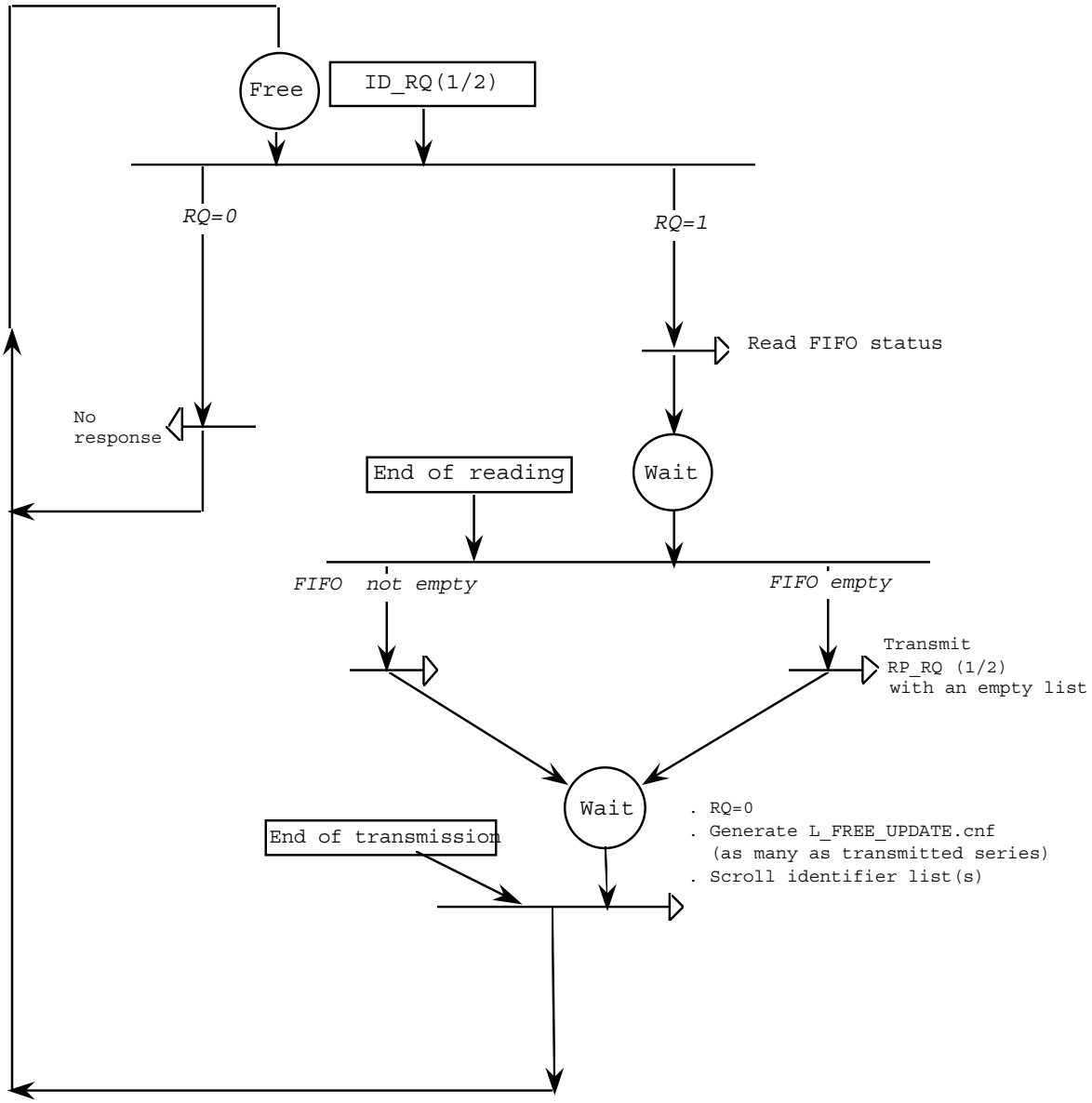


Figure 34 – Evaluation network for a free explicit request

After transmission of the RP_RQ, the request is detached from the separator concerned (RQ set to 0); If the FIFO is not empty, the request will be attached during the next reception of an ID_DAT concerning a DLCEP-identifier configured for this service. The RQ bit of this identifier will be set to 1.

6.7 Messaging

6.7.1 General introduction to message services

Message services make it possible to exchange acknowledged or unacknowledged messages.

The initiator of the message exchange request is the DLS-user that is the source of the message. The request is filled either during aperiodic scanning or during the bus-arbitrator's periodic scanning cycle, depending on configuration.

If the request is filled during aperiodic scanning the message is deemed aperiodic. A single resource is set aside for this type of message: a queue that is dynamically linked to one or more identifiers configured for this mode of message transfer. The variable response DLPDUs associated with these identifiers carry the message transfer requests. Requests are filled in the aperiodic window.

If the request is filled during periodic scanning it is deemed cyclical, and the message transfer request takes place in the periodic window by configuration. Resources are allocated upon configuration and do not change. The resource set aside is a queue attached to a DLCEP-identifier configured for this mode of transfer. The message identifier DLPDU associated with this identifier is transmitted in the periodic window.

A single identifier cannot be configured for both message transfer modes.

Service primitives for message transfer requests are identical for both aperiodic and cyclical modes. Only one parameter of the request primitive indicates the message transfer mode requested and makes it possible to call upon the proper mechanism within the source DLE.

To put the communicating entities into contact two addresses are given during the message transaction:

— a destination address (access point for a message service) made up of 24 bits that encode the local link number of the entity and the entity's sub-address within that DL-segment;

— a source address (access point for a message service) made up of 24 bits that encode the local link number of the entity and the entity's sub-address within that DL-segment.

An access point for a message service addresses an Application message service entity for emission (source) and for reception (destination). Each address is unique within the system.

The format of the destination address and the source address is described in detail in an appendix.

Each entity detects the message response DLPDU (RP_MSG_NOACK or RP_MSG_ACK) and verifies its destination address to determine whether or not it is the destination of the message.

A recovery mechanism intervenes upon detection of the loss of an acknowledgement DLPDU linked to an acknowledged message transfer.

This mechanism for recovery upon loss of acknowledgement, and the restart counters, are found within the source entity. The number of authorized restarts is the same for all source entities within the system, and this number can be set upon configuration. The recovery mechanism can lead to the duplication of messages. An algorithm for numbering messages between the source entity and the destination entity is used to detect message duplication caused by the recovery mechanism (see 4.4.3).

6.7.2 Diagram of interactions

Calling upon the unacknowledged message transfer request service causes the addition of a message to the queue for messages to be emitted. This queue is designated by the parameter SPECIFIED_IDENTIFIER. How the exchange is carried out depends on this queue's type.

6.7.2.1 Aperiodic transfers

The queue is set aside for aperiodic transfer. The transfer includes four phases:

phase 1:

Upon reception of a DLCEP-identifier that can support an aperiodic request, and that is not already in use, the source entity indicates in the control field of the variable response DLPDU its request that the bus-arbitrator circulate a message transfer DLPDU, provided that the number of occupied places in the queue is inferior to the number of identifiers associated with this resource.

In addition, for all identifiers already in use the source entity maintains the indication of a message transfer request in the variable response DLPDU.

phase 2:

The bus-arbitrator gives control of the local link to the source entity by emitting a message DLSAP-address in the window allocated to messages.

phase 3:

The source entity emits the first message stored in the queue of messages to be emitted. The entity emits the message in a message response DLPDU whose extended control DLPDU contains the destination address and source address of the message.

phase 4:

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter has the value NIL).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns control of the local link to the bus-arbitrator immediately, and phase 3 is eliminated.

These phases are shown diagramatically in Figure 35.

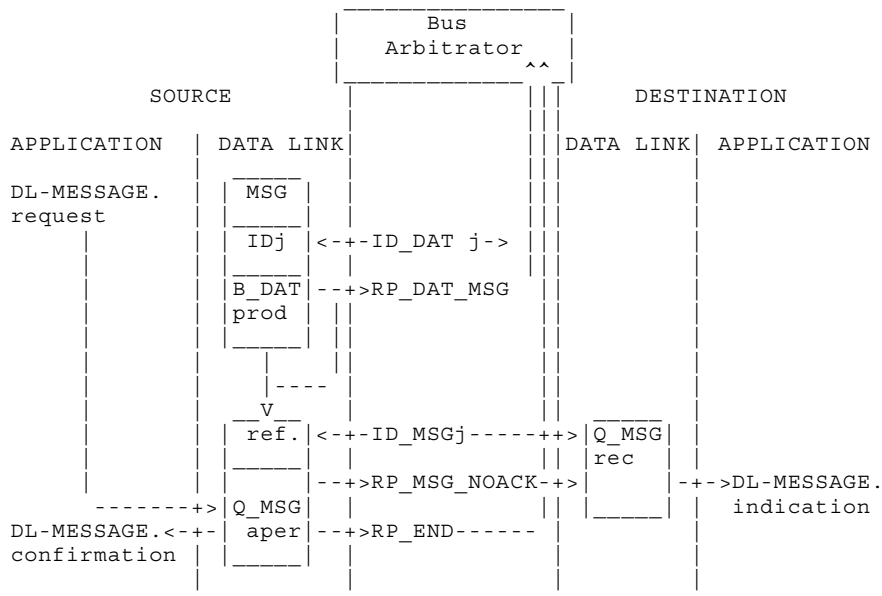


Figure 35 – Diagram of interactions within the unacknowledged message transfer request service for an aperiodic transfer

6.7.2.2 Cyclical transfers

The resource available for reception is a queue known as Q_MSGrec. The queue is set aside for cyclical transfers.

There are three phases to transfers:

phase 1:

The bus-arbitrator gives the source entity control of the local link by emitting a message identifier DLPDU in the periodic window,

phase 2:

The source entity emits the message response DLPDU whose extended control field contains the message's destination and source addresses.

phase 3

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter corresponds to the identifier in the message identifier DLPDU).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the bus-arbitrator immediately, and phase 2 is eliminated.

The resource available for reception is a queue known as Q_MSGrec.

These phases are shown diagramatically in Figure 36.

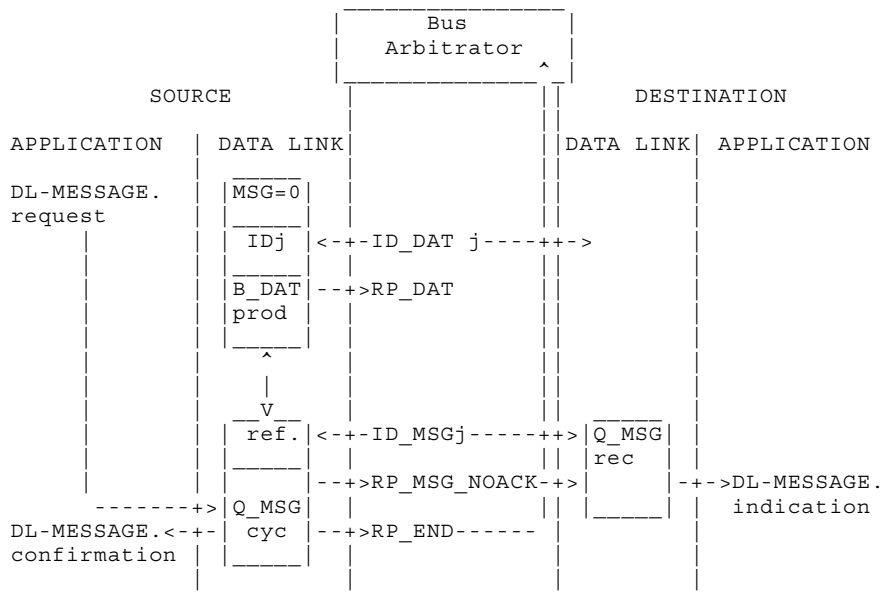


Figure 36 – Diagram of interactions within the unacknowledged message transfer request service for a cyclical transfer

6.7.3 Associated DLPDUs

An unacknowledged message transfer request involves the circulation of four DLPDUs in the case of an aperiodic message transfer. In the case of a cyclical message transfer, the first DLPDU is not broadcast.

6.7.3.1 Aperiodic message transfer

In the case of an aperiodic message transfer the source entity broadcasts a variable response DLPDU whose control DLPDU contains the request that the bus-arbitrator circulate a message identifier DLPDU (RP_DAT_MSG or RP_DAT_RQi_MSG). The first of these DLPDUs is part of normal cyclical traffic.

The identifier refers to the queue for messages to be emitted that is set aside for aperiodic message transfer.

The bus-arbitrator gives local link control to the source entity during an aperiodic message window by broadcasting a message identifier DLPDU (ID_MSG) with a DLCEP-identifier that has carried a request.

The message is emitted by the source entity (RP_MSG_NOACK) which states in the extended control field the message destination and source addresses.

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

On reception of a PDU containing a destination address which is homogeneous with a group address, layer 2 consults the local directory (by specific functions) in order to have the cross-reference between this group address and one or more local individual addresses to which is sent the DL-MESSAGE indication(Ad,As,m) indication primitive with the group address contained in the PDU as destination address.

Figure 37 shows the DLPDU sequence as it occurs in the periodic and aperiodic windows:

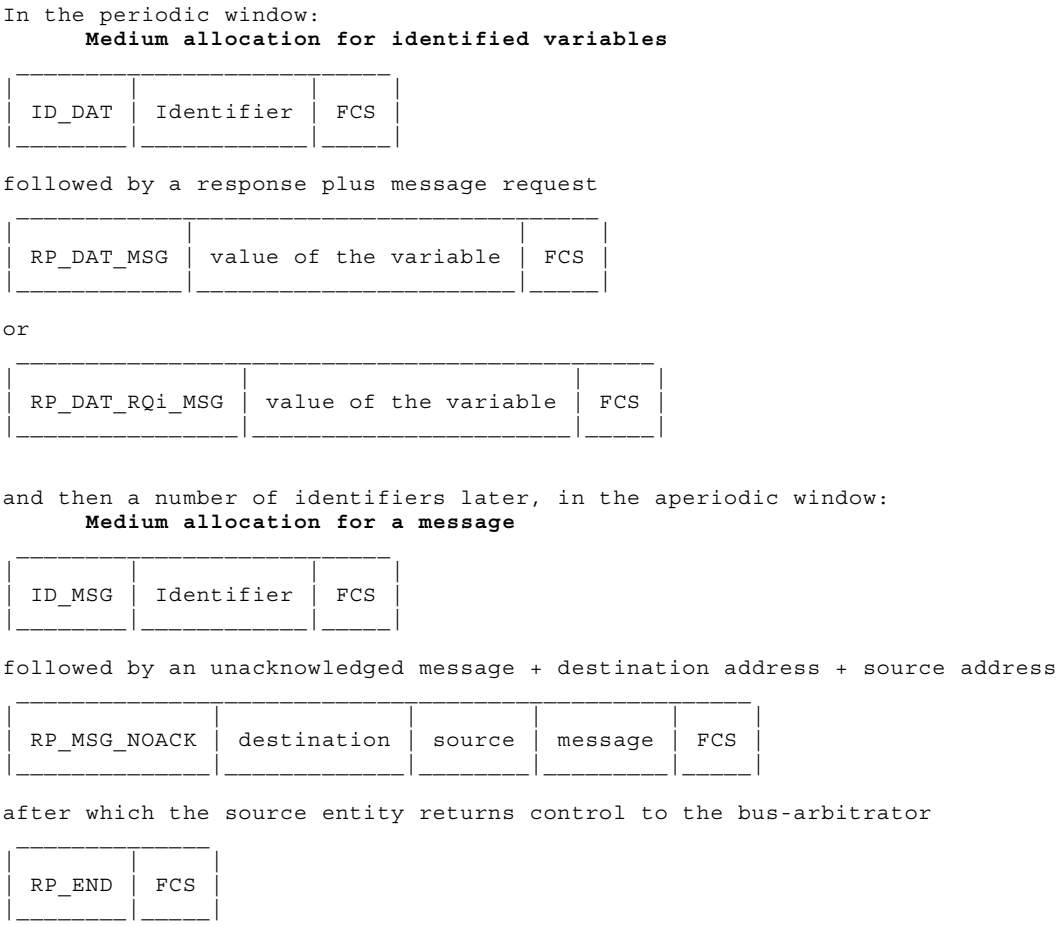


Figure 37 – DLPDU sequence for an aperiodic message transfer

6.7.3.2 Cyclical message transfer

In the case of a cyclical message transfer, the bus-arbitrator gives control to the source entity by broadcasting a message identifier DLPDU (ID_MSG) in the periodic window.

The message is emitted by the source entity (RP_MSG_NOACK) which notes in the extended control field the destination and source addresses.

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

On reception of a PDU containing a destination address which is homogeneous with a group address, layer 2 consults the local directory (by specific functions) in order to have the cross-reference between this group address and one or more local individual addresses to which is sent the DL-MESSAGE indication(Ad,As,m) indication primitive with the group address contained in the PDU as destination address.

Medium allocation for an identified variable is shown in Figure 38.

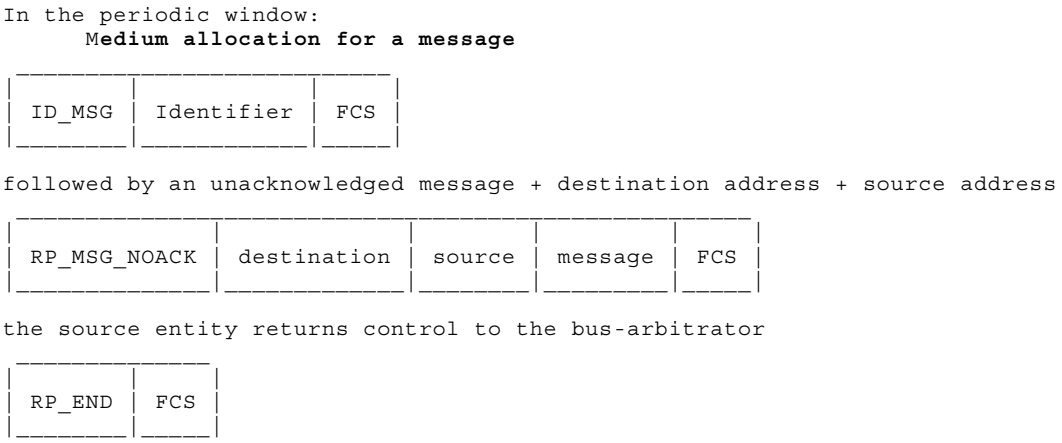


Figure 38 – DLPDU sequence for a cyclical message transfer

6.8 Acknowledged messaging

6.8.1 Diagram of interactions

Calling upon the acknowledged message transfer request service (DL-MESSAGE-ACK request) causes the addition of a message to the queue of messages to be emitted. This queue is designated by the parameter SPECIFIED_IDENTIFIER. The manner in which exchanges are carried out depends on the type of queue.

6.8.1.1 Queue dedicated to aperiodic transfer

If the queue is set aside for aperiodic transfer the exchange includes five phases:

phase 1:

Upon reception of a DLCEP-identifier that can support an aperiodic request, and that is not already in use, the source entity indicates in the control field of the variable response DLPDU its request that the bus-arbitrator circulate a message transfer DLPDU, provided that the number of occupied places in the queue is inferior to the number of identifiers associated with this resource.

In addition, for all identifiers already in use the source entity maintains the indication of a message transfer request in the variable response DLPDU.

phase 2:

The bus-arbitrator gives control to the source entity by emitting a message DLSAP-address in the window allocated to messages.

phase 3:

The source entity emits the first message stored in the queue of messages to be emitted. The entity emits the message in a message response DLPDU whose extended control DLPDU contains the destination address and source address of the message.

phase 4:

The destination entity emits the acknowledgement DLPDU immediately, except if the address destination is a group address. In this case, the entity ignores the DLPDU.

phase 5:

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE-ACK confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter has the value NIL).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the bus-arbitrator immediately, and phases 3 and 4 are eliminated.

The resource available for reception is a queue known as Q_MSGrec.

These phases are shown diagramatically in Figure 39.

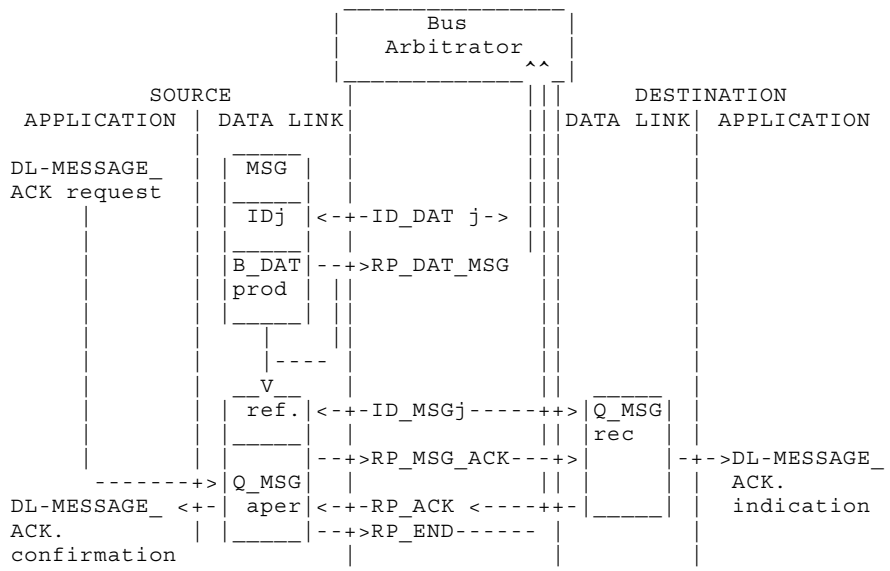


Figure 39 – Diagram of interactions within the acknowledged message transfer request service for an aperiodic transfer

6.8.1.2 Queue dedicated to cyclical transfers

The queue is set aside for cyclical transfers. There are four phases to transfers:

phase 1:

The bus-arbitrator gives the source entity control of the local link by emitting a message identifier DLPDU in the periodic window**.**

phase 2:

The source entity emits the message response DLPDU whose extended control field contains the message's destination and source addresses.

phase 3:

The destination entity emits the acknowledgement DLPDU immediately, except if the address destination is a group address. In this case, the entity ignores the DLPDU.

phase 4:

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE-ACK confirm primitive is sent to the DLS-user. (This primitive's SPECIFIED_IDENTIFIER parameter corresponds to the identifier in the message identifier DLPDU).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the bus-arbitrator immediately, and phases 2 and 3 are eliminated.

The resource available for reception is a queue known as Q_MSGrec.

These phases are shown diagramatically in Figure 40.

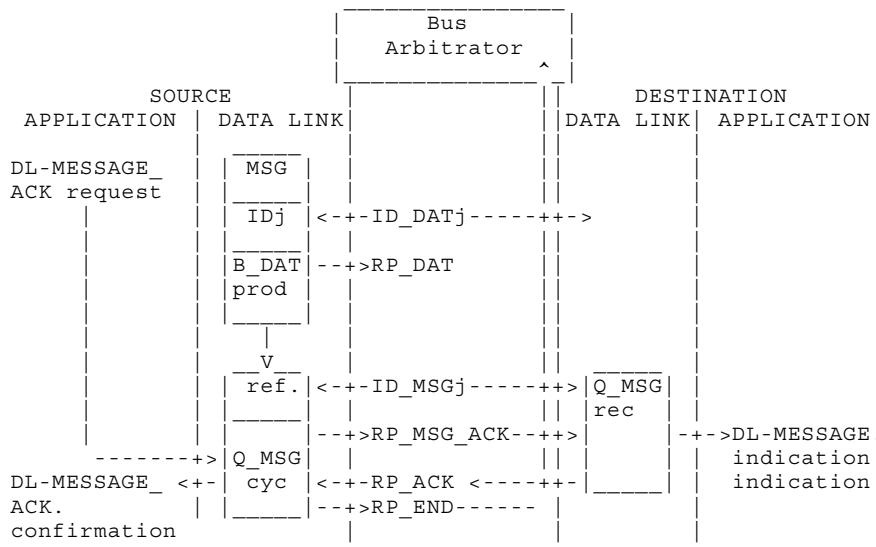


Figure 40 – Diagram of interactions within the acknowledged message transfer request service for a cyclical transfer

6.8.2 Associated DLPDUs

An acknowledged message transfer request involves the circulation of five DLPDUs in the case of an aperiodic message transfer. In the case of a cyclical message transfer, the first DLPDU is not broadcast.

6.8.2.1 Aperiodic message transfer

In the case of an aperiodic message transfer the source entity broadcasts a variable response DLPDU whose control DLPDU contains the request that the bus-arbitrator circulate a message identifier DLPDU (RP_DAT_MSG or RP_DAT_RQi_MSG). The first of these DLPDUs is part of normal cyclical traffic.

The identifier refers to the queue for messages to be emitted that is set aside for aperiodic message transfer.

The bus-arbitrator gives local link control to the source entity during an aperiodic message window by broadcasting a message identifier DLPDU (ID_MSG) with the identifier that has carried the request.

The message is emitted by the source entity (RP_MSG_ACK) which states in the extended control field the message destination and source addresses.

The destination entity immediately emits the acknowledgement DLPDU.

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

Figure 41 shows the DLPDU sequence as it occurs in the periodic and aperiodic windows:

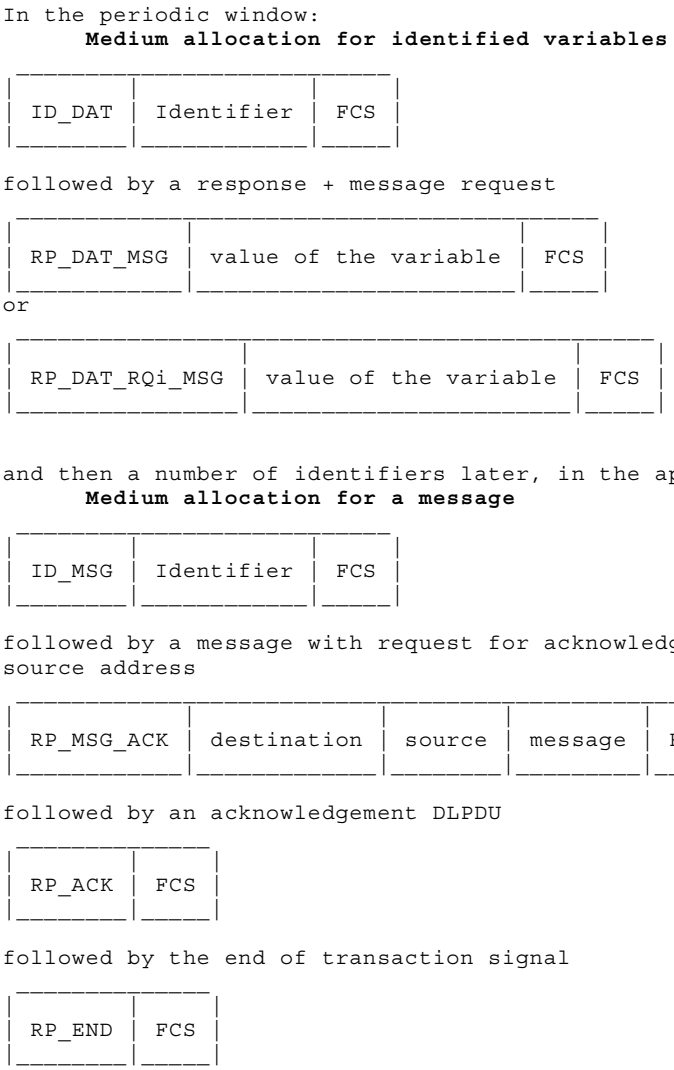


Figure 41 – DLPDU sequence for an aperiodic message transfer

6.8.2.2 Cyclical message transfer

In the case of a cyclical message transfer, the bus-arbitrator gives control to the source entity by broadcasting a message identifier DLPDU (ID_MSG) in the periodic window.

The message is emitted by the source entity (RP_MSG_ACK) which notes in the extended control field the destination and source addresses.

The destination entity emits the acknowledgement DLPDU (RP_ACK) immediately.

The source entity returns control to the bus-arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

Figure 42 shows the DLPDU sequence as it occurs in the periodic window:

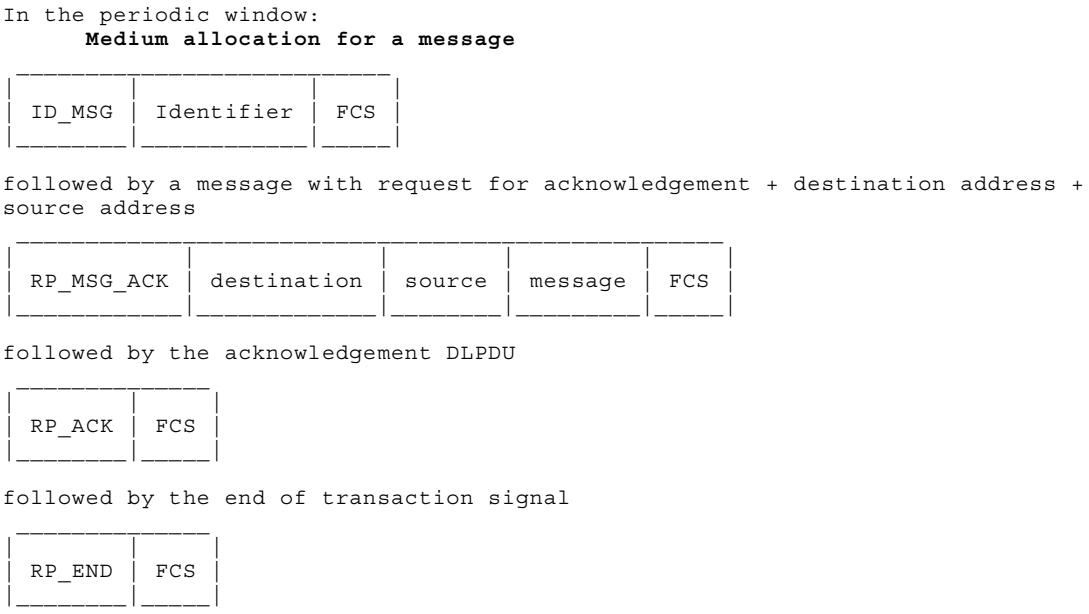


Figure 42 – DLPDU sequence for a cyclical message transfer

The RP_MSG_ACK and RP_ACK control fields containing the numbering bit Bp given by the algorithm defined in 6.9.

NOTE This DL-protocol provides SDA and SDN services as specified in IEC 60955 (PROWAY C). The DLL primitives also directly correspond to primitives specified in the documents ISO/IEC 8802-2 and ISO TC 97 SC6 WG1 N169. The DL-MESSAGE primitive corresponds to the DLSDU primitive. The DL-MESSAGE-ACK primitive corresponds to the DLSDU_ACK primitive.

6.9 Numbering of acknowledged messages

The algorithm describing the message numbering protocol uses the following data:

— each destination entity records the source address (ADRsource_stoc) of the last message response DLPDU received, and the associated even/odd bit (Bc),

— upon reception of any DLPDU other than RP_MSG_ACK, entities give the stored source address the value "empty",

— recovery is immediate (no other intermediate transaction). Recovery is provided by the source entity, which manages a waiting period for the acknowledgement DLPDU and a restart counter,

— the source entity sends the message with the indication Bp, and changes the value of this bit upon transmission of RP_END.

When the system is initialized each entity has the following information:

— ADRsource_stoc = “empty”,

— Bp has no significance,

— Bc has no significance.

The algorithm has three parts. One part is implemented by the message source entity, another by the destination entity, and the third by the bus-arbitrator.

6.9.1 Destination algorithm:

```
Upon reception of RP_MSG_ACK,
compare ADRsource received with ADR source_stoc
if equal compare Bp received with Bc
  if equal
     if there is room to store
        then send RP_ACK+Bp
             store message
             Bc=1-Bp
     else send RP_ACK-Bp
  else send RP_ACK+Bp
       ignore message because duplicate
else
  if there is room to store
     send RP_ACK+Bp
     Bc=1-Bp
     store the message
     ADRsource_stoc=ADR source received
  else send RP_ACK-Bp

if reception of DLPDU other than RP_MSG_ACK
     ADRsource_stoc=empty
```

6.9.2 Source algorithm:

```
Upon reception of ID_MSG(IDk) and IDk recognized by the source entity
send message with Bp indication
wait for RP_ACK

if reception RP_ACK
             set MSG of IDk at 0
             send confirmation with status (ack ±)
             reset restart counter to zero
             send RP_END and Bp=1-Bp

if expiration of waiting time for RP_ACK,
  test restart counter

  if counter > maximum number of restarts authorized
     set MSG of IDk at 0
     reset restart counter to zero
     send confirmation with negative status
     send RP_END and Bp=1-Bp

  if counter ≤ maximum number of restarts authorized
     retransmit same RP_MSG_ACK
     increment restart counter
     wait for RP_ACK
```

6.9.3 bus-arbitrator algorithm:

After sending ID_MSG

The bus-arbitrator is waiting for RP_END:

if the timer for waiting for RP_END has expired

  go on to the next identifier

if reception of RP_END

  go on to next identifier

if reception of DLPDU other than RP_END, or of an invalid DLPDU

  continue to wait for RP_END

6.10 Behavior with mismatched parameters

6.10.1 Message aperiodic transfer (acknowledged or not)

The evaluation network of Figure 43 allows specifying the behavior of a producer/consumer entity in the following cases.

— Reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for aperiodic transfer of messages and not associated with the Q_MSG.aper queue (the identifier MSG has a value of 0).

— Reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for aperiodic transfer of messages and associated with the Q_MSG.aper queue (the identifier MSG has a value of 1), the latter being empty.

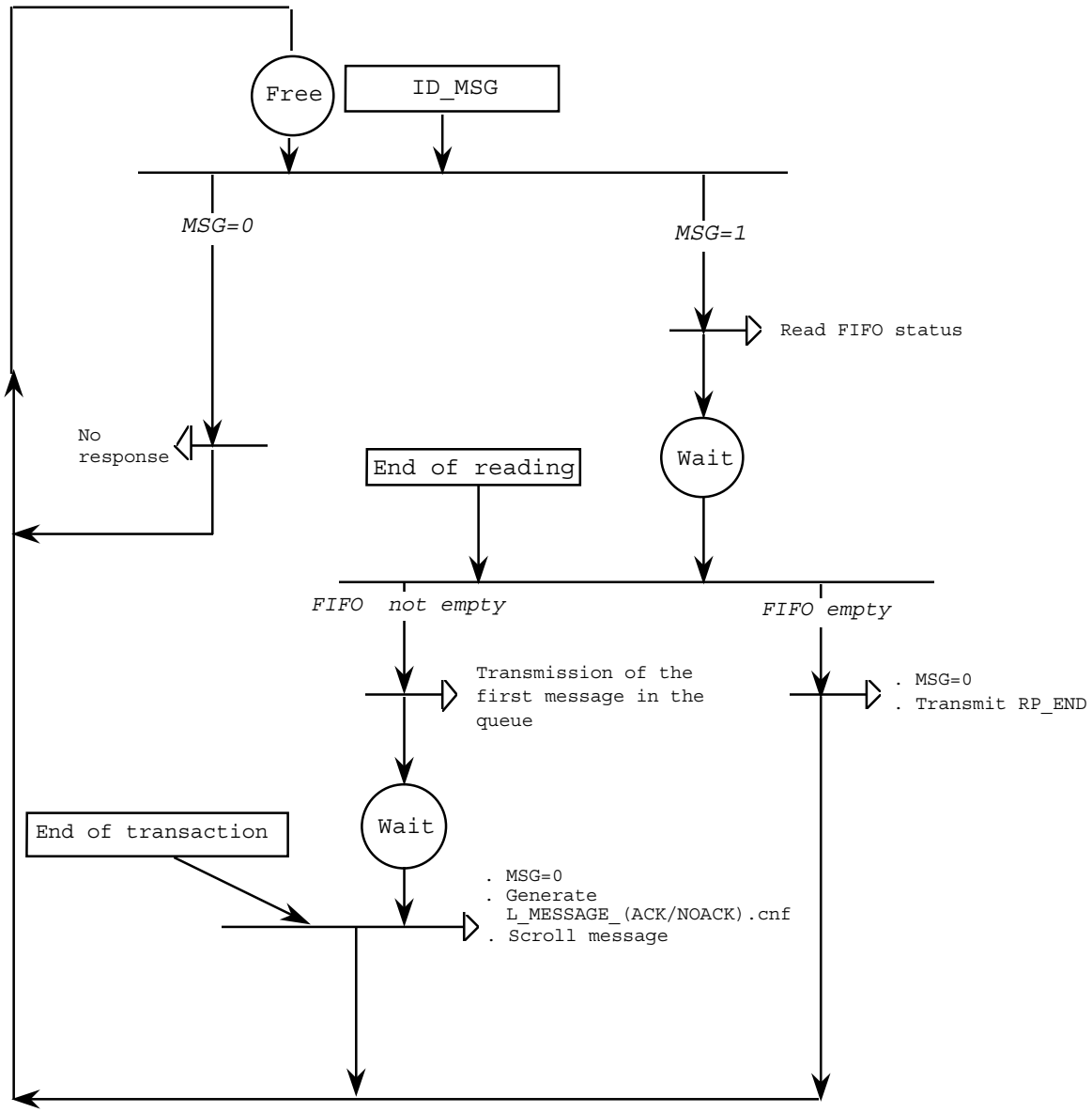


Figure 43 – Evaluation network for message aperiodic transfer

6.10.2 Message cyclic transfer (acknowledged or not)

The evaluation network of Figure 44 allows specifying the behavior of a producer/consumer entity, on reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for the cyclic transfer of messages and whose Q_MSG.cyc queue is empt.

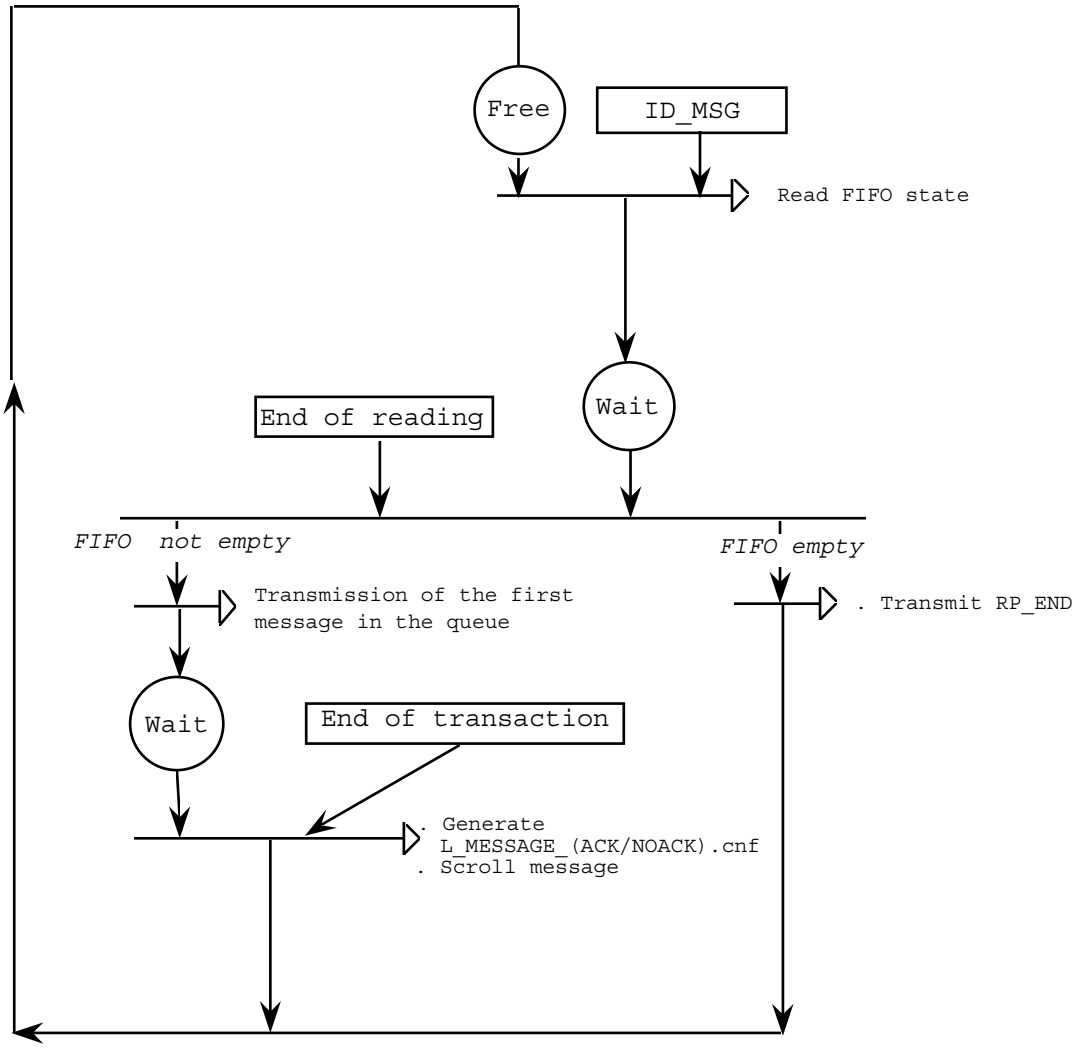


Figure 44 – Evaluation network for message cyclic transfer

7 DL-service elements of procedure, interfaces and conformance

7.1 General

These finished status machines describe the overall functioning of a DLE as a function of the DLPDUs circulating on the medium, and they use the time-fillers defined above.

Transition tables complete the description of the functioning of a DLE by stating actions carried out upon reception of a DLPDU.

The protocols associated with each service provided by the DLL are described for a producer/consumer entity and for the bus-arbitrator.

The interfaces to the DLS-user, DL- Management and the PhL are summarized and the Type 7 conformance classes are shown in Figure 183.

7.2 Producer/consumer entity

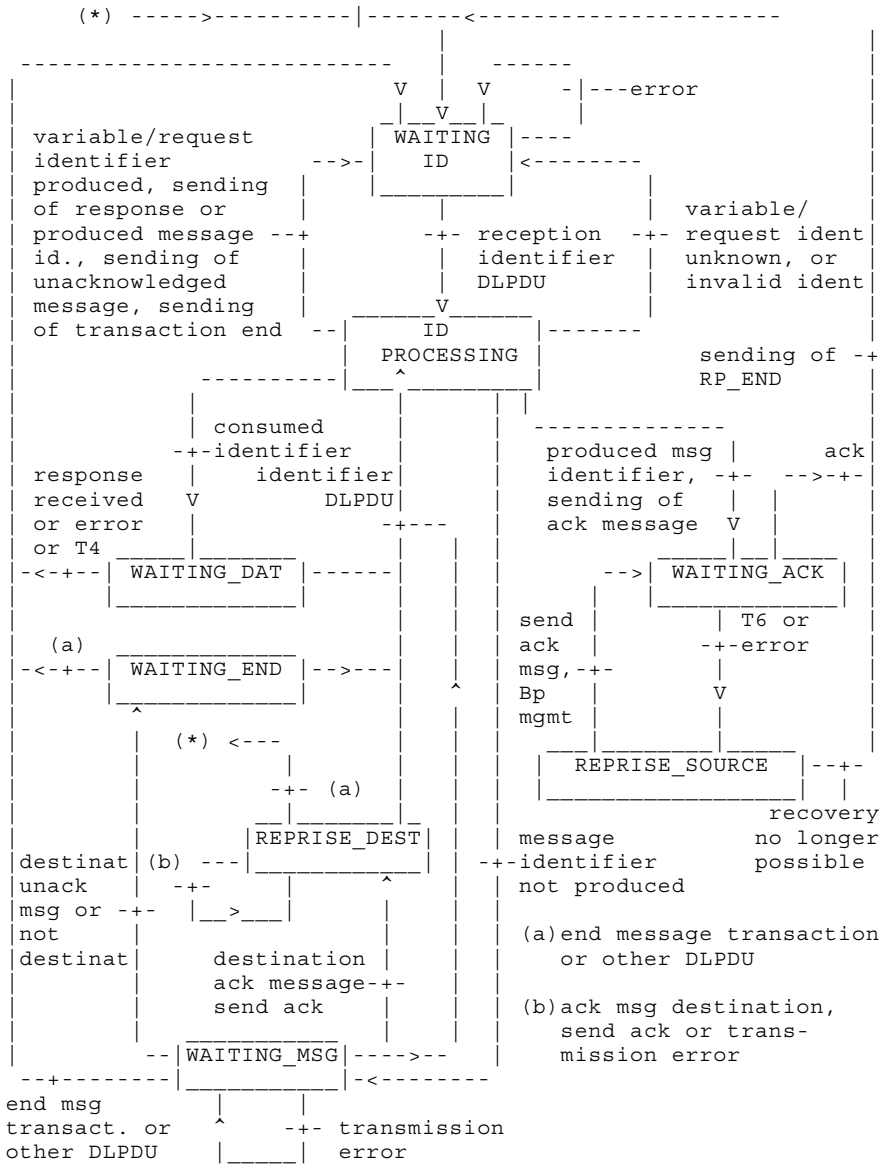


Figure 45 – Simplified states machine for a producer/consumer entity

The status machine indicates the behavior of a station during reception and transmission. This machine takes into account in its various statuses:

— reception of various types of link DLPDU,

— time-outs specific to the station (they are not those of the BA).

It is important to note that these various link DLPDUs taken into account in the various statuses of this machine can only be taken into account in so far as the latter at least observe the temporal characteristics of the protocol.

a) Temporal characteristics of the protocol without interruption.

In the case of normal sequencing of the protocol, the temporal characteristics of the latter are fixed by the turn-around time value during production of the various stations. Each DLPDU received by a device which must provide a response is separated from the latter on the bus by a time interval equal to its own turn-around time during production.

NOTE Knowing that the device which has the maximum turn-around time during production conditions the choice of the time-out T0.

b) Temporal characteristics of the protocol in case of interruption

If the DLPDU sequencing is degraded for whatever reason (disturbance on the bus, absence of production) the temporal characteristics of the protocol are fixed by the bus arbitrator.

Thus for example, an ID_XX DLPDU shall be taken into account in the various reception statuses only if the BA observes the following temporal constraints.

— In the WAIT_DAT status, the BA must wait at least T1=T0 before proceeding in the transmission of an ID_XX. Knowing that time-out T1 on the BA level is set by the line silence following transmission of a n ID_DAT DLPDU and that it can elapse only if silence is maintained all through its life.

— In the MSG_WAIT or END_WAIT or DEST_RESTART, the BA must attain at least T5=2T0, in case of non-reception of RP_END before proceeding with transmission of an ID_XX DLPDU. Knowing that time-out T5 at the BA level is set by each line silence following transmission of an ID_MSG DLPDU and that it can only elapse if the silence is maintained all through its life.

NOTE In the case of an arbitrator which has identified the cause of this interruption (for example, producer absent) and which would like to turn around immediately to pass to transmission of the following identifier, operation of the protocol is not guaranteed.

7.2.1 Transitions table for a producer/consumer entity

```
WAITING_ID (initial status)
   ID_xx:                                                --> PROCESSING_ID

   other DLPDU:                                             --> WAITING_ID
   transmission error:                                      --> WAITING_ID


PROCESSING_ID
   ID_DAT,ID produced:
      if the identifier is invalid, no response
                                                            --> WAITING_ID

      if indicator of urgent explicit request emit RP_DAT_RQ1
      if indicator of normal explicit request emit RP_DAT_RQ2
      if indicator of message request emit RP_DAT_MSG
      if 2 indicators emit RP_DAT_RQi_MSG
      else emit RP_DAT

      generate a DL-SENT indication (ID produced)           --> WAITING_ID

   ID_DAT,ID consumed: activate T4                         --> WAITING_DAT
   ID_DAT, new ID:                                          --> WAITING_ID

   ID_MSG,ID produced:
      if msg exists and is unack, emit RP_MSG_NOACK
                                  emit RP_END               --> WAITING_ID

      if msg exists and is ack, emit RP_MSG_ACK
                                activate T6                 --> WAITING_ACK

      if msg does not exist, emit RP_END                    --> WAITING_ID

   ID_MSG, new ID:                                         --> WAITING_MSG

   ID_RQ1,ID produced:
      emit RP_RQ1
      generate DL-SPEC_UPDATE confirm (ID produced, OK)
      or DL-FREE_UPDATE confirm (OK)                        --> WAITING_ID

   ID_RQ1, new ID:                                          --> WAITING_ID

   ID_RQ2, ID produced:
      emit RP_RQ2
      generate DL-FREE_UPDATE confirm (OK)                  --> WAITING_ID

   ID_RQ2, new ID:                                          --> WAITING_ID
```

```
WAITING_DAT
   RP_DAT or RP_DAT_MSG or RP_DAT_RQi or RP_DAT_RQi_MSG:
      store value of the variable
      generate a DL-RECEIVED indication (ID consumed)
                                                          --> WAITING_ID

   ID_xx:                                               --> PROCESSING_ID

   other DLPDU:                                           --> WAITING_ID
   transmission error:                                    --> WAITING_ID
   T4 timer: indicate transaction aborted                 --> WAITING_ID


WAITING_MSG
   RP_MSG_NOACK, my address is destination:
      receive message
      generate a DL-MESSAGE indication (ADRdestination, ADRsource,
      message)                                              --> WAITING_END

   RP_MSG_NOACK,other destination address:                --> WAITING_END

   RP_MSG_ACK, my address is destination:
      generate message numbering bit
      receive msg, send RP_ACK
      generate a DL-MESSAGE_ACK indication (ADRdestination, ADRsource,
      message)                                           --> RECOVERY_DEST
   RP_MSG_ACK, other destination address                   --> WAITING_END

   RP_END: end of message transaction                      --> WAITING_ID

   ID_xx:                                               --> PROCESSING_ID

   other DLPDU:                                            --> WAITING_ID
   transmission error:                                     --> WAITING_MSG


WAITING_ACK
   RP_ACK:
      generate the message numbering bit (see 6.9)
      generate a DL-MESSAGE_ACK confirm
      (ADRdestination, ADRsource, status)
   emit RP_END                                             --> WAITING_ID

   T6 timer:                                          --> RECOVERY_SOURCE

   other DLPDU:                                       --> RECOVERY_SOURCE
   transmission error:                                --> RECOVERY_SOURCE


WAITING_END
   RP_END:                                                 --> WAITING_ID

   ID_xx                                                --> PROCESSING_ID

   other DLPDU:                                            --> WAITING_ID
   transmission error:                                     --> WAITING_ID


RECOVERY_DESTINATION
   RP_END:                                                 --> WAITING_ID

   RP_MSG_ACK: management of Bp (see 6.9)
               send ack                                   --> RECOVERY_DEST

   ID_xx                                                --> PROCESSING_ID

   other DLPDU:                                            --> WAITING_ID
   transmission error:                                --> RECOVERY_DEST


RECOVERY_SOURCE
   if recovery possible
      emit same RP_MSG_ACK
      activate T6                                          --> WAITING_ACK
   else send RP_END
      indicate transaction aborted                         --> WAITING_ID
```

NOTE Types of errors are defined in 4.6.22 (DLL-counter class description) of EN 50170-7.

7.3 Protocol elements by service

NOTE The protocols associated with the services provided by a producer/consumer entity are specified by describing the entity's behavior for each primitive exchanged with the DLS-user.

7.3.1 Reception of a DL-PUT request

Upon reception of a DL-PUT request primitive (IDk, DLSDU) a producer triggers the following actions:

* semantic testing (unknown identifier or incorrect length)

— semantics incorrect

immediate transmission of DL-PUT confirm (IDk, unknown identifier or length of DLSDU incorrect)

— semantics correct

* test validity of identifier

— identifier invalid

immediate transmission of DL-PUT confirm (IDk, invalid identifier)

— identifier valid

* test availability of buffer

— buffer unavailable,

immediate transmission of DL-PUT confirm (IDk, buffer unavailable)

— buffer available,

write the new value

immediate transmission of DL-PUT confirm (IDk, OK)

The various tests carried out are based on data concerning:

— recognition of identifier parameter,

— the identifier is known if it has been declared as a produced identifier by the producer/consumer entity,

— compatibility of length of DLSDU with the buffer B_DATprod (length less than 128 octets),

— validity of the identifier,

— a DLCEP-identifier is valid if the data-link configuration of the associated variable is valid,

— availability of buffer,

— buffer unavailable if the same B_DATprod buffer is already being accessed by the local link. To manage this conflict the standard recommends giving the local link highest priority in all cases.

7.3.2 Reception of a DL-GET request

Upon reception of a DL-GET request primitive (IDk) a producer triggers the following actions:

* test identifier IDk

— identifier unknown

immediate transmission of DL-GET confirm (IDk, identifier unknown)

— identifier known

* test availability of buffer

— buffer unavailable,

immediate transmission of DL-GET confirm (IDk, buffer unavailable)

— buffer available

* test validity of identifier

— identifier invalid

immediate transmission of DL-GET confirm (IDk, invalid identifier)

— identifier valid

immediate transmission of DL-GET confirm (IDk, DLSDU, reading OK).

The various tests carried out are based on data concerning:

— recognition of identifier parameter

the identifier is known if it has been declared as a consumed identifier by the producer/consumer entity,

— availability of buffer

buffer unavailable if the same B_DATcons buffer is already being accessed by the local link. To manage this conflict the standard recommends giving the local link highest priority in all cases,

— validity of the identifier,

— a DLCEP-identifier is valid if the data-link configuration of the associated variable is valid.

7.3.3 Reception of a DL-SPEC-UPDATE request

Upon reception of a DL-SPEC-UPDATE request primitive (IDk, LIST) a producer triggers the following actions:

* test IDk

— IDk unknown or not configured for this service

immediate transmission of DL-SPEC-UPDATE confirm (IDk, identifier unknown)

— IDk known and configured for this service

* test availability of B_REQ (IDk) buffer

— buffer unavailable

immediate transmission of DL-SPEC-UPDATE confirm (IDk, buffer unavailable)

— buffer available

* test transmission of preceding request

— request not transmitted (RQ of IDk=1)

. immediate transmission of DL-SPEC-UPDATE confirm (IDk, override) of the preceding request

. store list of identifiers (this list can be empty)

— request transmitted (RQ of IDk=0)

. set request indicator (RQ of IDk=1)

. store list of identifiers

* reception of ID_DAT (IDk) produced

* test RQ of IDk

— RQ of IDk inhibited by configuration

emit only RP_DAT

— RQ of IDk not inhibited by configuration

emit RP_DAT_RQ

* reception of ID_RQ (IDk)

. emit RP_RQ with list of identifiers requested

. generate DL-SPEC-UPDATE confirm (IDk, OK)

. set RQ of IDk to 0

The various tests carried out are based on data concerning:

— availability of buffer

buffer unavailable if B_REQ is already being accessed by the local link,

— waiting for the request to be taken into account by the bus-arbitrator

RQ of IDk is set at 1 by DL-SPEC-UPDATE request

RQ of IDk is set at 0 upon transmission of RP_RQ.

A specific transfer request that contains an empty variables list makes it possible to clear a buffer of specific transfer requests.

7.3.4 Reception of a DL-FREE-UPDATE

Upon reception of a DL-FREE-UPDATE request primitive (LIST, PRIORITY) a producer triggers the following actions:

* test queue

— sufficient space not available,

immediate transmission of DL-FREE-UPDATE confirm (priority, saturation)

— space available and queue not empty

store list of identifiers

— queue empty

. store list of identifiers

. wait for next produced ID_DAT

* reception of produced ID_DAT (IDp)

* test IDp

— IDp not authorized

wait for next produced ID_DAT

— IDp authorized

. associate IDp with the queue concerned

. set RQ of IDp at 1

. send RP_DAT_RQ

* reception of ID_RQ (IDp)

* test status of queue

— queue empty no answer

— queue not empty

. emit RP_RQ with all stored lists of identifiers (up to a limit of 64 identifiers)

. generate as many DL-FREE-UPDATE confirm (OK) primitives as there are lists stored

. run through the lists of identifiers

. set RQ of IDp at 0

* test queue status

— queue not empty

wait for next produced ID_DAT

The various tests carried out are based on data concerning:

— status of the queue Q_REQi: queue filled or not, queue empty or not

wait for an authorized produced ID_DAT for DL-FREE-UPDATE. A mechanism makes it possible to prohibit the use of certain identifiers for DL-FREE-UPDATE.

7.3.5 Reception of a DL-MESSAGE request

Upon reception of a DL-MESSAGE request primitive (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message) a source entity triggers the following actions:

* test SPECIFIED_IDENTIFIER

— incorrect parameter (identifier not configured for cyclical message transfer, or value of parameter different from NIL)

immediate transmission of DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, incorrect parameter)

— parameter correct

* test status of queue

— queue full

. immediate transmission of DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, queue full)

— queue not full

. store message

. wait for ID_MSG (IDp) or ID_DAT (IDp)

* reception of ID_DAT (IDp)

* test IDp

— IDp configured for cyclical message transfer

. emit RP_DAT

. wait for next produced ID_DAT

— IDp configured for aperiodic message transfer

* test MSG of IDp

- MSG of IDp = 1

. emit RP_DAT_MSG

. wait for next ID_DAT

— MSG of IDp = 0

* test status of the resource Q_MSGaper set aside for aperiodic message transfer

- queue empty

. emit RP_DAT

- queue not empty and number of identifiers associated with the resource equal to the number of places occupied in said resource

. emit RP_DAT

- queue not empty and number of identifiers associated with the resource less than number of places occupied in the resource

. establish reference between IDp and the queue for messages waiting to be emitted set aside for aperiodic message transfer

. set MSG of IDp at 1

. emit RP_DAT_MSG

. wait for next ID_DAT

* reception of ID_MSG (IDp)

* test IDp

— IDp shows no reference to a resource

. no response

— IDp shows reference to a resource

* test status of resource given by IDp

- if resource is empty, send RP_END

— else

. emit RP_MSG_NOACK with the first message of the queue given by IDp

. send RP_END

. send DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message sent)

. free up place in queue

. if the resource is the queue set aside for aperiodic message transfer:

. set MSG of IDp at 0

. cancel reference between IDp and the queue.

The various tests carried out are based on data concerning:

— status of the queue used: empty, full, or space available,

— indicator showing whether the bus-arbitrator has taken the message request into account in the case of aperiodic message transfer

MSG is set at 1 through attribution of IDp to the queue set aside for aperiodic message transfer.

MSG is set at 0 upon emission of RP_END.

Number of identifiers associated with the queue set aside for aperiodic message transfer, and number of places occupied in this queue.

7.3.6 Receipt of a DL-MESSAGE-ACK request

Upon reception of a DL-MESSAGE-ACK request primitive (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message) a source entity triggers the following actions:

* test SPECIFIED_IDENTIFIER

—incorrect parameter (identifier not configured for cyclical message transfer, or value of parameter different from NIL) immediate transmission of DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, incorrect parameter)

parameter correct

*test ADR Destination

ADR destination is a group address

. ignore the message

. immediate transmission of DL-MESSAGE-ACK confirm(Spec Id, ADR Dest, ADR Source, Group Address)

* test status of queue

— queue full

. immediate transmission of DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, queue full)

— queue not full

. store message

. wait for ID_MSG (IDp) or ID_DAT (IDp)

* reception of ID_DAT (IDp)

* test IDp

— IDp configured for cyclical message transfer

. emit RP_DAT

. wait for next produced ID_DAT

— IDp configured for aperiodic message transfer

* test MSG of IDp

— MSG of IDp = 1

. emit RP_DAT_MSG

. wait for next ID_DAT

— MSG of IDp = 0

* test status of the resource Q_MSGaper set aside for aperiodic message transfer

- queue empty

. emit RP_DAT

- queue not empty and number of identifiers associated with the resource equal to the number of places occupied in said resource

. emit RP_DAT

- queue not empty and number of identifiers associated with the resource less than number of places occupied in the resource

. establish reference between IDp and the queue for messages waiting to be emitted set aside for aperiodic message transfer

. set MSG of IDp at 1

. emit RP_DAT_MSG

. wait for next ID_DAT

* reception of ID_MSG (IDp)

* test IDp

— IDp shows no reference to a resource

. no response

— IDp shows reference to a resource

* test status of resource given by IDp

— if resource is empty, send RP_END

— else

. emit RP_MSG_ACK with the first message of the queue referenced by IDp

. launch timer for waiting for RP_ACK

. wait for acknowledgement DLPDU

* reception of RP_ACK

* test acknowledgement

- acknowledgement positive

. send RP_END

. send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message sent)

. free up place in queue

. if the resource is the queue set aside for aperiodic message transfer:

. set MSG of IDp at 0

. cancel reference between IDp and the queue.

— acknowledgement negative

. send RP_END

. send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, acknowledgement negative)

. free up place in queue

. if the resource is the queue set aside for aperiodic message transfer:

. set MSG of IDp at 0

. cancel reference between IDp and the queue.

* expiration of timer for acknowledgement reception

* test restart counter

—- the counter has not reached its maximum value

— emit RP_MSG_ACK with the first message in the queue referenced by IDp

— launch the timer for waiting for RP_ACK

— wait for acknowledgement DLPDU

—- the counter has reached its maximum value

— send RP_END

— send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, maximum number of restarts attained)

— free up place in queue

— if the resource is the queue set aside for aperiodic message transfer:

— set MSG of IDp at 0

— cancel reference between IDp and the queue.

The various tests carried out are based on data concerning:

— status of the queue used: empty, full, or space available,

— indicator showing whether the bus-arbitrator has taken the message request into account in the case of aperiodic message transfer MSG is set at 1 through attribution of IDp to the queue set aside for aperiodic message transfer MSG is set at 0 upon emission of RP_END,

— current value of the restart counter

number of identifiers associated with the queue set aside for aperiodic message transfer, and number of places occupied in this queue.

7.3.7 Invalid identifier

This part of standard define the actions to be performed at reception of an ID_XX DLPDU concerning an invalid identifier.

The invalidation of a DLCEP-identifier concerns at least one of the four services: cyclic messaging, specified explicit periodic request for transfer of identified objects, consumption, production. In all cases, a producer/consumer entity does not answer the demands from the medium which concern the invalidated service(s) of a DLCEP-identifier.

Besides, at invalidation of a DLCEP-identifier in production (Product_ID_Validity attribute set to FALSE), the possible associations with Q_REQ1 or Q_REQ2 as well as with Q_MSG.aper are interrupted. The attachment attribute relative to Q_REQ1 or Q_REQ2 is set to FALSE if the identifier was associated with one of these files before being invalidated; the Attachment_number attribute relative to Q_MSG.aper is decreased by one if the identifier was associated with this file before being invalidated.

7.4 Bus arbitrator operation

7.4.1 General description of the bus arbitrator

The role of the bus-arbitrator is to "give permission to speak" to each information producer, while taking into account the services needed by the user application.

The bus-arbitrator thus has three functions:

1) periodic scanning of variables or triggered periodic scanning of variables and messages,
2) triggered scanning of variables,
3) triggered scanning of messages.

In addition, the bus-arbitrator can provide a synchronization function to guarantee the constant length of a scanning cycle.

Each type of scanning takes place in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window respectively.

The four windows together make up a basic scanning cycle.

The functions of the bus-arbitrator are broken down into three layers, as are other DL-services. For each of these layers there is a set of arbitrating services and arbitrating data. The complete specification of the bus-arbitrator will be found in 5.1 (bus-arbitrator) of EN 50170-3.

The remainder of this subclause describes the data-link functions provided by the bus-arbitrator. The data-link level of a bus-arbitrator entity offers the following functions:

a) emission of identifier DLPDU sequences,

b) reception of DLPDUs containing identifiers,

c) detection of requests for aperiodic variable and message exchanges,

d) emission of identifier DLPDUs used for synchronization.

The resources used to provide these functions are also described: they include the identifier sequences, queues for requests for aperiodic variable and message exchanges, the restart buffer.

The bus-arbitrator's data-link level oversees the proper outcome of the various basic transactions, and executes requests from the DLS-user for services, in order to properly carry out Basic Cycles and Macro Cycles.

7.4.2 Management of basic transactions

7.4.2.1 General

A basic transaction is made up of the successive DLPDUs related to a single service.

Managing basic transactions means controlling the succession of DLPDUs that make up a transaction. Valid basic transactions are:

ID_DAT + RP_DAT

ID_DAT + RP_DAT_MSG

ID_DAT + RP_DAT_RQi

ID_DAT + RP_DAT_RQi_MSG

ID_MSG + RP_MSG_NOACK + RP_END

ID_MSG + RP_MSG_ACK + RP_ACK + (repetitions) + RP_END

ID_RQi + RP_RQi

Subclause 7.4.6 specifies the bus-arbitrator's simplified states machine as well as the associated transitions table.

7.4.2.2 Emitting a variable identifier

After emitting a variable identifier DLPDU (ID_DAT) the bus-arbitrator activates the timer T1 and then waits for a variable response DLPDU.

a) If the DLPDU received is a simple response (RP_DAT) the bus-arbitrator goes on to the next identifier to be scanned.

b) If the DLPDU received is a variable response accompanied by a message transfer request (RP_DAT_MSG), and if the bus-arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for message transfer requests (Q_IDMSG) and goes on to the next identifier to be scanned.

c) If the DLPDU received is a variable response accompanied by an explicit request for a buffer transfer (RP_DAT_RQi), and if the bus-arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for explicit buffer transfer requests that has the proper priority (Q_IDRQi) and goes on to the next identifier to be scanned.

d) If the DLPDU received is a variable response accompanied by a request for a message transfer and an explicit request for a buffer transfer (RP_DAT_RQi_MSG) and if the bus-arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for message requests (Q_IDMSG) and in the queue for explicit buffer transfer requests that has the proper priority (Q_IDRQi) and goes on to the next identifier to be scanned.

e) If the DLPDU received is of a different type the bus-arbitrator detects an error and goes on to the next identifier to be scanned.

f) If the DLPDU received has a flawed FCS the bus-arbitrator detects a transmission error and goes on to the next identifier to be scanned.

g) If the T1 timer expires the bus-arbitrator detects the loss of a response DLPDU and goes on to the next identifier to be scanned.

7.4.2.3 Emitting a message identifier

After emitting a message identifier DLPDU (ID_MSG), the bus-arbitrator activates the T5 timer and waits for the response DLPDU indicating the end of the message transaction.

a) If the DLPDU received is an end of message transaction the bus-arbitrator goes on to the next identifier to be scanned.

b) If the DLPDU received is an unacknowledged message (RP_MSG_NOACK), an acknowledged message (RP_MSG_ACK) or an acknowledgement response DLPDU (RP_ACK), the bus-arbitrator activates T5 and continues to wait for a response DLPDU indicating the end of the message transaction.

c) If the DLPDU received is of a different type the bus-arbitrator detects an error and goes on to the next identifier to be scanned.

d) If the DLPDU received has an incorrect residual FCS (see 5.2.5.2) the bus-arbitrator detects a transmission error and continues to wait for the end of message transaction DLPDU.

e) If the timer expires the bus-arbitrator detects the loss of an end of message transaction DLPDU and goes on to the next identifier to be scanned.

7.4.2.4 Emitting a request identifier

After emitting a request identifier DLPDU (ID_RQi), the bus-arbitrator activates the T1 timer and waits for a request response DLPDU.

a) If the DLPDU received is an urgent request response DLPDU (RP_RQ1) and if the bus-arbitrator is in its periodical scanning phase, it records the list of identifiers in the recovery buffer and immediately scans all the variable identifiers stored.

b) If the DLPDU received is an urgent request response DLPDU and if the bus-arbitrator is in its aperiodic scanning phase, it records the list of identifiers in the aperiodic queue being processed and goes on to the next identifier to be scanned.

c) If the DLPDU received is a normal request response DLPDU (RP_RQ2) the bus-arbitrator records the list of identifiers in the aperiodic queue being processed and goes on to the next identifier to be scanned.

d) If the DLPDU received is of a different type the bus-arbitrator detects an error and goes on to the next identifier to be scanned.

e) If the DLPDU received has a flawed FCS the bus-arbitrator detects a transmission error and continues to wait for the end of message transaction DLPDU.

f) If the timer T1 expires the bus-arbitrator detects the loss of a response DLPDU and goes on to the next identifier to be scanned.

7.4.2.5 Synchronization window

During the synchronization window the bus-arbitrator goes through a loop composed of the following actions.

a) It emits a synchronization identifier DLPDU (this DLPDU corresponds, for example, to a DLCEP-identifier that is neither produced nor consumed) and activation of the T1 timer (used for waiting for a response DLPDU).

b) If the bus-arbitrator receives a response DLPDU or detects the expiration of the T1 timer, it restarts loop operations.

c) If the bus-arbitrator detects the expiration of the T2 timer signifying the end of the Basic Cycle, it abandons the loop and goes on to the next Basic Cycle.

This function guarantees the constant length of a Basic Cycle. The precise timing of the beginning of a cycle is thus at best equal to the length of the emission of a DLCEP-identifier DLPDU followed by a timer used to wait for a response DLPDU.

7.4.3 Resources used

The DLL uses two categories of resources to implement the data-link level functions of the bus-arbitrator:

— the basic sequences of identifier numbers that constitute the cyclical portion of the scanning table and make it possible to generate the bus's periodic flow,

— the queues used to store the numbers of identifiers that have carried a normal (Q_IDRQ2) or urgent (Q_IDRQ1) variable request,

— the queues used to store the numbers of identifiers that are the object of aperiodic variable exchange requests (Q_RPRQ),

(These four queues make it possible to generate the bus's aperiodic flow.)

— a recovery buffer for the immediate emission of identifiers involved in a triggered periodic request for a variable exchange,

— a basic padding sequence used to generate synchronization window flow.

For further details, see 4.2.2.

7.4.4 Chaining of basic transactions

7.4.4.1 Bus arbitrator cycles

The basic function of the bus-arbitrator is to grant access to the bus by emitting identifier DLPDUs.

Several types of identifier DLPDUs have been defined, and a hierarchy in the order of emission of these identifiers can be set up to accommodate specific user needs.

The bus-arbitrator's highest priority function is to allocate the medium so that the cyclical transfer of variables, requests, and messages is guaranteed.

Allocation of the medium for explicit requests for variable or message transfers has a lower priority.

Screening time requirements are not identical for all the variables scanned cyclically, but are always multiples of a basic period.

A Basic Cycle includes the bus-arbitrator's scanning of:

— a set of variable, request, and cyclical message identifiers based on the application. This set of identifiers characterizes the individual Basic Cycle,

— plus the aperiodic exchanges window,

— plus the message services window,

— plus the synchronization window.

A Macro Cycle is the group of Basic Cycles that are juxtaposed to obtain the scanning of all the application's cyclical identifiers.

The Macro Cycle covers all the application's cyclical communications needs, and provides an available bandwidth for triggered transactions.

A Basic Cycle thus has a maximum of 4 phases:

P1: periodic scanning of variables (cyclical ID_DAT), triggered periodic scanning of variables (cyclical ID_RQi) or triggered periodic scanning of messages (cyclical ID_MSG)

P2: triggered message scanning

P3: triggered scanning of variables with management of the two priority levels

P4: wait for synchronization signal if a synchronized scanning cycle is requested.

Restart is possible in P1 and P2.

P3 can precede P2.

A Macro Cycle includes at least one Basic Cycle.

All Basic Cycles include a P1 (periodic scanning) phase.

Chaining of cycles

A DLL system incorporates several variable-scanning periods, but bus-arbitrator operations include only two levels of chaining:

— chaining of sequences to make up a Basic Cycle,

— chaining of Basic Cycles to make up a Macro Cycle.

In addition, chaining of basic cycles can be

— with synchronization,

— without synchronization.

Consequently, the DLS-user sends the bus-arbitrator a set of service request primitives that describe the execution of the Basic Cycles (the flow unit on the bus) and the Macro Cycles (the repetitious unit on the bus, made up of the chaining of one or several Basic Cycles).

These primitives are used to express user needs both in terms of the information to be exchanged and in terms of the screening time constraints to be respected.

If the synchronized scanning function has not been requested, Phase P4 of the Basic Cycle is eliminated. The maximum length of the Basic Cycle can still be guaranteed, however. In this operating mode if the Basic Cycle continues beyond its maximum length the next Basic Cycle is automatically begun.

7.4.4.2 Cycle configuration parameters

Certain configuration parameters are connected with each phase of the Basic Cycle.

Parameters linked to phase P1 of a Basic Cycle are

— the list of basic sequences that characterize the basic cycle. These identifiers will be scanned using a variable, request, or cyclical message identifier (ID_DAT, ID_RQ1 or ID_MSG).

In the case of a cyclical request transaction (ID_RQ1, RP_RQ1) the number of transactions (ID_DAT, RP_DAT) that immediately follow is limited by the entity making the request and is known in advance. Thus a maximum length for the transaction is guaranteed.

The parameter linked to phase P2 of a Basic Cycle is

— the time limit on the window allocated to triggered message transfers. This limit is defined with respect to the beginning of the Basic Cycle. When this window exists, it corresponds at minimum to the transmission of an ID_MSG DLPDU, followed by an RP_MSG_NOACK DLPDU and a RP_END DLPDU. If acknowledged message transfers and a restart mechanism are supported the window corresponds to the transmission of an ID_MSG DLPDU, followed by a RP_MSG_ACK DLPDU, followed by a RP_ACK DLPDU and enough time for as many repetitions of RP_MSG_ACK, RP_ACK as the number of authorized restarts allows. All these DLPDUs are followed, of course, by an RP_END DLPDU.

The parameter linked to phase P3 of a Basic Cycle is

— the time limit on the window allocated to triggered aperiodic variable exchanges. This limit is defined with respect to the beginning of the Basic Cycle. When this window exists, it corresponds at minimum to the transmission of an ID_RQi DLPDU, followed by an RP_RQi DLPDU.

The parameter linked to phase P4 of a Basic Cycle is

— the total length of a Basic Cycle and the basic padding sequence emitted while waiting for the indication signaling the end of the Basic Cycle.

7.4.5 Multiple bus arbitrators

The information presented in this subclause is further developed in the document C46 605 Network Management.

The overall results given below are based on the hypothesis that the multiple bus-arbitrators are connected to a single physical support:

— at a given instant ONLY ONE BA is active,

— a procedure for choosing the active BA is defined,

— the other BA listen to what is occurring on the physical support and manage their own scanning tables, but do not emit any identifier DLPDUs,

— a timer value is specified in the final states machine of each producer/consumer entity in order to detect any prolonged silence on the part of the active BA (timer T3),

— a protocol for the exchange of synchronization data can be used to synchronize Macro Cycles.

7.4.6 Bus arbitrator

A simplified state machine for the active bus-arbitrator is shown in Figure 46 and Table 6.

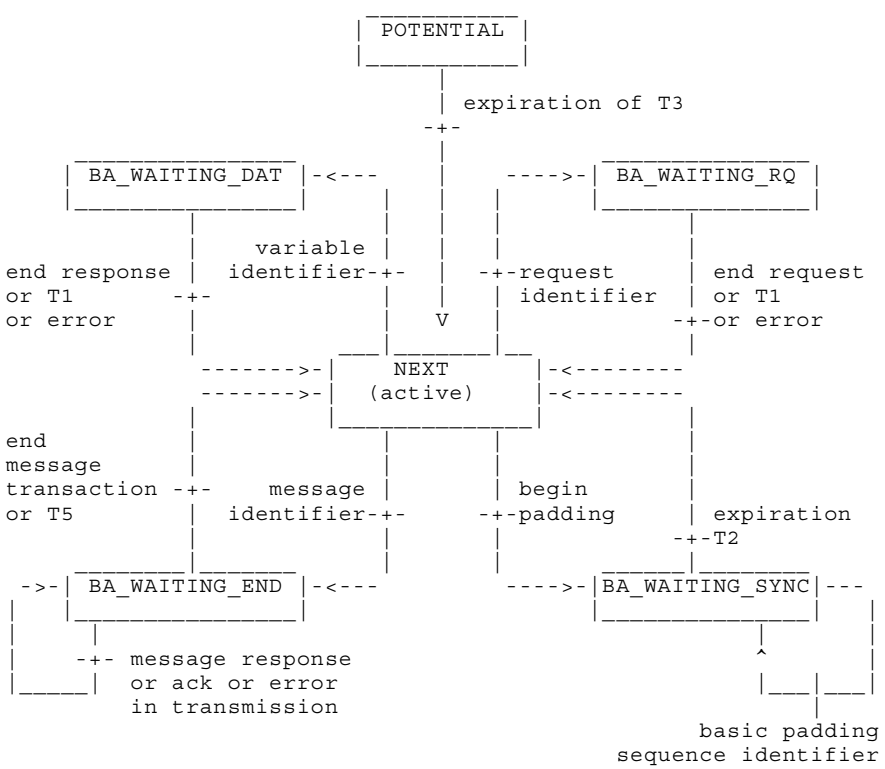


Figure 46 – Active bus arbitrator's simplified state machine

Table 6 – Bus arbitrator state transition table

Current state Event	Action	Next state
(initial)		
expiration T3		-→ NEXT
NEXT		
emission ID_DAT	activate T1	-→ BA_WAITING_DAT
emission ID_RQ1	activate T1	-→ BA_WAITING_RQ
emission ID_RQ2	activate T1	-→ BA_WAITING_RQ
emission ID_MSG	activate T5	-→ BA_WAITING_END
emission basic padding sequence		-→ BA_WAITING_SYNC
BA_WAITING_DAT		
RP_DAT		-→ NEXT
RP_DAT_MSG	record ID message emitter	-→ NEXT
RP_DAT_RQ1	record ID request emitter	-→ NEXT
RP_DAT_RQ2	record ID request emitter	-→ NEXT
RP_DAT_RQ1_MSG	record ID request emitter and ID message emitter	-→ NEXT
P_DAT_RQ2_MSG	record ID request emitter and ID message emitter	-→ NEXT
other DLPDU		-→ NEXT
transmission error		-→ NEXT
T1 timer	transaction aborted	-→ NEXT
BA_WAITING_END		
RP_END		-→ NEXT
RP_MSG_NOACK, RP_MSG_ACK, RP_ACK	activate T5	-→ BA_WAITING_END
other DLPDU		-→ NEXT
transmission error		-→ BA_WAITING_END
T5 timer	transaction aborted	-→ NEXT
BA_WAITING_RQ		
RP_RQ1	record list of identifiers in urgent queue	-→ NEXT
RP_RQ2	record list of identifiers in normal queue	-→ NEXT
other DLPDU		-→ NEXT
transmission error		-→ NEXT
T1 timer	transaction aborted	-→ NEXT
BA_WAITING_SYNC		
T2 not expired	emit identifier for the basic padding sequence	-→BA_WAITING_SYNC
other DLPDU		-→ NEXT
T2 timer		-→ NEXT

7.5 Bridges

7.5.1 Introduction

This subclause describes additional functions to allow the organization of a DL-subnetwork into several local links interconnected by bridges, as typified in Figure 47.

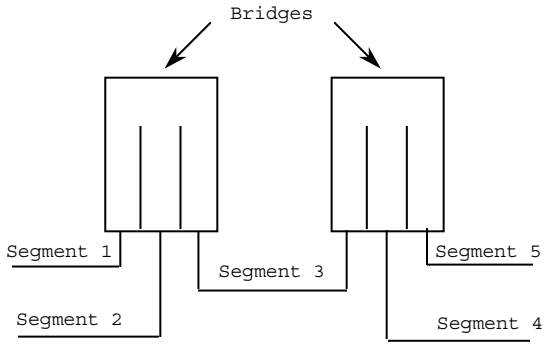


Figure 47 – Typical bridge usage

The specification of a bridge thus requires definition of the principle and the various possible modes for switching the information from one DL-segment to another.

7.5.2 Bridges

This subclause allows the reader to locate the bridge with respect to the DLL communication architecture functions.

7.5.2.1 Situation of the bridge in the DLL communication architecture

The bridge has a certain number of functions specific to the DLL (DLL) and particularly to the Logic Control (LC) level in order to allow a DL-subnetwork organization into several local links.

Each of the logical DL-segments of such a DL-subnetwork has a medium access control (MAC) independent from that of the other DL-segments. Consequently, there will be an active bus-arbitrator on each of the local links.

Figure 48shows the placement of bridges in the OSI Basic Reference Model (ISO/IEC 7498).

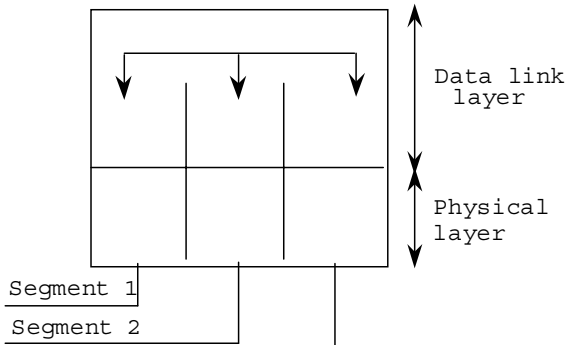


Figure 48 – Architectural placement of bridges in OSI Basic Reference Model (ISO/IEC 7498)

Because a bridge can store a received DLPDU for retransmission, the PhLs of the various local links can be of different speeds. They also can have differing media and modulation.

NOTE 1 Change of media or coding/modulation can also be achieved through Ph-relay devices, commonly called repeaters.

It is quite possible to envisage linking devices which are generalizations of bridges, and which could interconnect different types of DL-protocol. Usually there would be some loss of DL-service when transitioning the different protocol types.

All these possibilities of diversity pose specific problems, which are not dealt with in this addendum. However, a MAC address of the same size is required in all the local links, otherwise it is necessary to include address cross-reference tables.

NOTE 2 Having different access control on two interconnected DL-segments corresponds to considering, for example, a DL-subnetwork with a predominantly centralized administration such as the Type 7 DL-protocol, and another with a predominantly decentralized administration such as found in ISO/IEC 8802-4 or the Type 2 or Type 3 DL-protocol.

7.5.2.2 Communication through a bridge

Bridges are DL-relay devices which allow users of various local links to communicate without the user applications having to know or take into consideration the existence of these bridges, as in Figure 49.

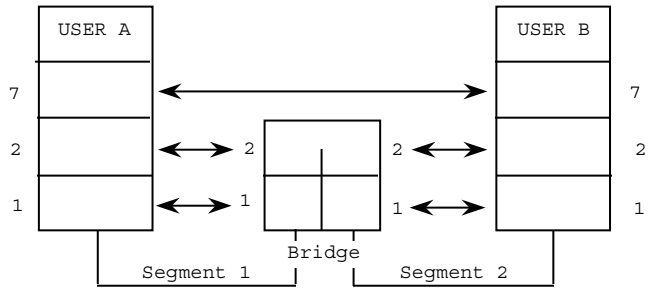


Figure 49 – Representation of an extended link communication

There are thus direct exchanges between the final devices belonging to various DL-segments at the level of the DLS-user and the users, whereas exchanges at the medium access control (MAC) level and the physical (Ph) level are performed from DL-segment to DL-segment.

The functions of such a bridge exclusively concern the messaging services.

NOTE The buffer transfer services can only be executed in a single-DL-segment environment.

7.5.3 Bridge function

7.5.3.1 General

A bridge is formed of several channels each of which has a PhL and a link layer allowing connection between various DL-segments.

The DLL functions relative to the cyclic messaging (outside the service primitives) are minimum functions, which must be supported to accept the bridges.

7.5.3.2 Link functions required

The link functions supported by each bridge channel can contain all the link layer services in case the bridge is also a station. However the minimum link layer resources and functions necessary for operation of the bridge are those of the cyclic messaging, with or without acknowledgement. These include:

— a queue, containing the messages to be transmitted, meant for cyclic transfer of messages called Q_MSGcyc,

— a queue full/not full indicator, associated with Q_MSG.cyc,

— a DLCEP-identifier configured for messaging,

— the attributes associated with a link entity supporting the messaging service with acknowledgement:

— . value of the even/odd bit, as source entity,

— . maximum value of the restart counter,

— . current value of the restart counter.

— a mechanism supporting the cyclic messaging service, whether acknowledged or not.

7.5.3.3 Properties

The extended link must have the properties of connectivity and non-meshing.

Connectivity consists in ensuring that there is always a path from any given DL-segment to any other DL-segment of the extended link .

Non-meshing consists in ensuring that the transmission of a message between the transmitter DL-segment and the destination DL-segment can only be executed along a single path (to prevent messages being received more than once).

These two rules allow calculating the local data base specific to each bridge in order to ensure its operation.

7.5.4 Object model

7.5.4.1 Bridge object description

A bridge is modeled on the link level with the help of the object described in Table 7:

Table 7 – Bridge object description

Class:	BRIDGE
Key Attribute:	Bridge identification
Attribute:	Operating status [ON; OFF]
Attribute:	*List of*
Attribute:	Channel *reference*

7.5.4.1.1 Identification of the bridge

This attribute corresponds to a number allowing unique identification of a bridge in the DL-subnetwork.

7.5.4.1.2 Operating mode

A bridge connected to a DL-subnetwork has two operating modes:

OFF: in this state a bridge does not ensure any of its functions. The passage from this state to another can only be obtained by a local action at the device level.

ON: in this state a bridge ensures its functions. The services offered by each channel are thus usable; the channels, considered individually, are not necessarily in operation.

7.5.4.1.3 List of channel references

This attribute allows designating all the bridge channels. Each of these channels has minimum functions allowing acceptance of the bridge retransmission mechanism.

7.5.4.2 Channel object description

Each of the bridge channels is modeled with the help of the object described in Table 8:

Table 8 – Channel object description

Class:	CHANNEL
Key Attribute:	Channel number
Attribute:	Segment number
Attribute:	Channel validity [Valid; Invalid]
Attribute:	Segment directory *reference*
Attribute:	Network directory *reference*

7.5.4.2.1 Channel number

This attribute is a number identifying a channel relative to a bridge.

7.5.4.2.2 Segment number

This attribute designates the segment connected to the bridge channel identified by the "Channel number" attribute.

7.5.4.2.3 Channel validity

A bridge channel has two states:

— VALID: This state corresponds to normal operation of a bridge channel.

  It is able to:

  — retransmit messages from other channels,

  — redirect received messages towards other channels.

— INVALID: In this state a bridge channel does not ensure any of its functions.

The passage from one state to another is ensured by the system management. These two states are necessary for the configuration and addition/deletion of bridges in the DL-subnetwork.

When the bridge is in the OFF state, each of its bridge channels is in the INVALID state.

7.5.4.2.4 Segment directory reference

The segment directory has a bridge channel giving the rules for retransmission of a message received by this channel on another channel of the same bridge, when the message destination address has a segment number (S/R=1).

The set of segment directories must be built to respect the properties of connectivity and non-meshing.

7.5.4.2.5 Network directory reference

The network directory which has a bridge channel gives the rules for retransmission of a message received by this channel on one or more channels of the same bridge when the message destination address does not have a segment number (S/R=0).

The set of network directories must be built in a manner observing the properties of connectivity and non-meshing.

7.5.4.3 Segment directory object description

The segment directory is modeled with the help of the object described in Table 9:

Table 9 – Segment directory object description

Class:	SEGMENT DIRECTORY
Key Attribute:	Directory number
Attribute:	*List of*
Attribute:	Recognized segment number
Attribute:	Retransmission channel number

7.5.4.3.1 Directory number

This attribute corresponds to a number allowing identification of the individual directory associated with a bridge channel.

7.5.4.3.2 Recognized segment number and retransmission channel number

The received messages with a destination address containing this segment number will be retransmitted on the associated retransmission channel.

7.5.4.4 Network directory object description

The network directory is modeled with the help of the object described in Table 10:

Table 10 – Network directory object description

Class:	NETWORK DIRECTORY
Key Attribute:	Directory number
Attribute:	*List of*
Attribute:	Retransmission channel number

7.5.4.4.1 Directory number

This attribute is a number allowing identification of the network directory associated with a bridge channel.

7.5.4.4.2 Retransmission channel number list

The messages received with a destination address not containing a segment number are retransmitted on each of the retransmission channels.

7.5.5 Mechanisms

The inter-DL-segment messaging can be acknowledged or not. In the first case, the acknowledgement is partial; each of the transactions performed on each of the DL-segments is acknowledged, but there is no acknowledgement from end to end.

The restart mechanism defined in this document is used in the case of transmission of acknowledged messages.

7.5.5.1 Retransmission rule

On reception of a message on one channel:

— Either the destination address contains a DL-segment number belonging to the channel segment directory: the bridge then retransmits the message on the corresponding retransmission channel.

— Or the destination address does not contain a DL-segment number: the bridge then retransmits the message on all the retransmission channels belonging to the receive channel network directory.

7.5.5.2 Retransmission triggering

On reception of an RP_MSG_ACK or RP_MSG_NOACK DLPDU containing a destination address observing the retransmission rule, a bridge channel stores the message:

— in the Q_MSG cycle of the retransmission channel if the destination address contains a DL-segment number,

— in the Q_MSG.cyc file of each of the retransmission channels if the destination address does not contain a DL-segment number.

On reception of a messaging DLPDU, a bridge channel does not call the indication primitives. The acknowledgement status (positive or negative) is determined by the state (full or not) of the retransmission channel Q_MSG.cyc queue.

7.5.5.3 Retransmission mechanism

The retransmission mechanism is triggered on reception of an RP_MSG_NOACK or RP_MSG_ACK DLPDU. The evaluation networks of Figure 50 and Figure 51 describe this mechanism.

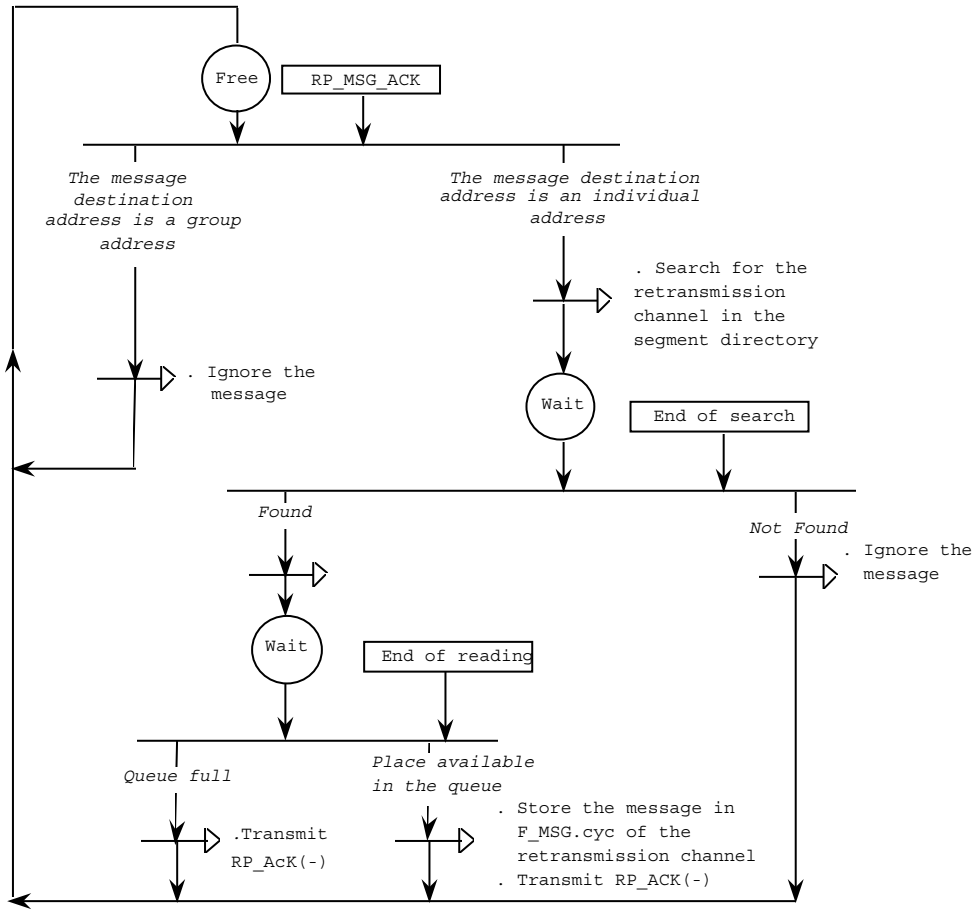


Figure 50 – Evaluation network for reception of an RP_MSG_ACK DLPDU

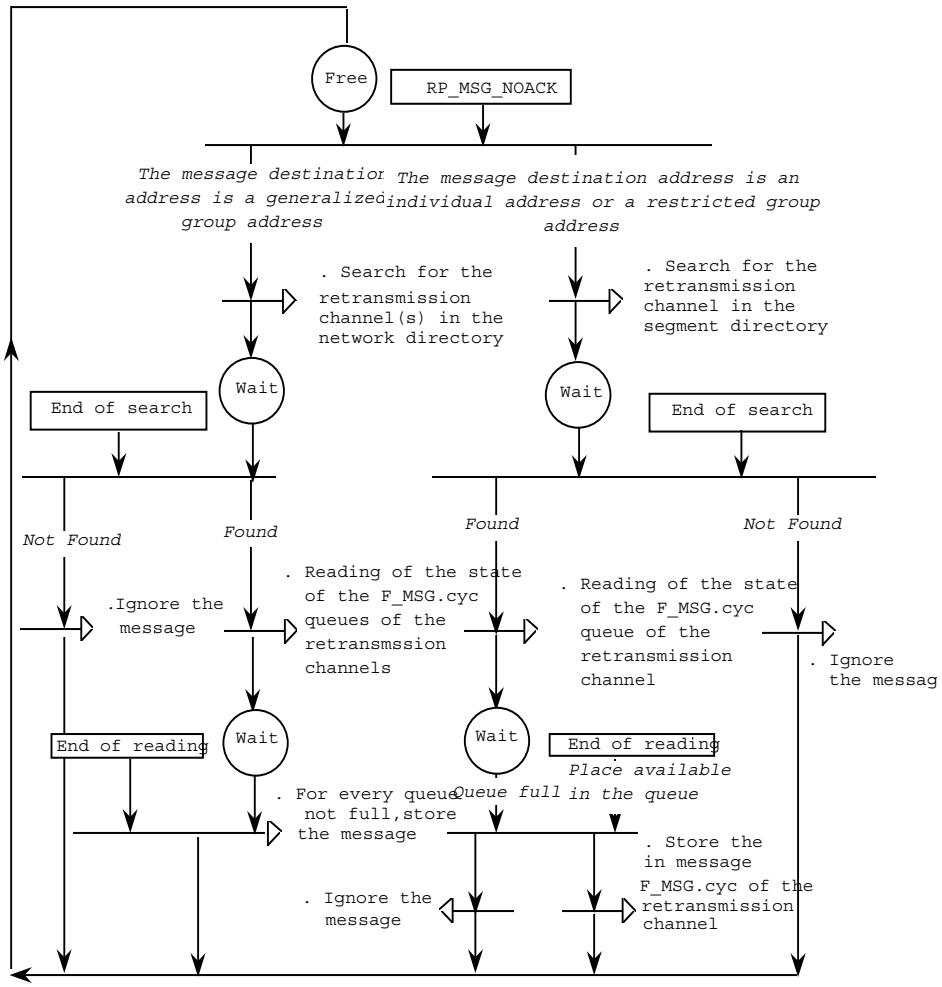


Figure 51 – Evaluation network for reception of an RP_MSG_NOACK DLPDU

7.6 Interfaces

7.6.1 DLS-user interface

The DLL provides the DLS-user with the following set of services:

— buffer writing,
— buffer reading,
— buffer transfer,
— specified explicit request for a buffer transfer,
— free explicit request for a buffer transfer,
— unacknowledged message transfer request,
— acknowledged message transfer request.

The primitives associated with each service completely describe the interface between the data-link and DLS-users. Detailed specification of primitives can be found in Part 3 of this standard.

The primitives associated with each service are given in Table 11.

Table 11 – Service primitives by type

Type of service	Primitive
Buffer writing:	DL-PUT request
	DL-PUT confirm
Buffer reading:	DL-GET request
	DL-GET confirm
Buffer transfer:	DL-SENT indication
	DL-RECEIVED indication
Specified explicit request for a buffer transfer:	DL-SPEC-UPDATE request
	DL-SPEC-UPDATE confirm
Free explicit request for a buffer transfer:	DL-FREE-UPDATE request
	DL-FREE-UPDATE confirm
Unacknowledged message transfer request:	DL-MESSAGE request
	DL-MESSAGE confirm
	DL-MESSAGE indication
Acknowledged message transfer request	DL-MESSAGE-ACK request
	DL-MESSAGE-ACK indication
	DL-MESSAGE-ACK confirm

7.6.2 DL- Management interface

DL-Management offers a number of services that will be specified in the Management document. These services make possible:

Control of DLE functioning modes such as:

— initialization of the DLE operating function,

— control of bus arbitrating,

— changing the bus-arbitrator functioning mode.

Reports on events concerning changes within the DLE such as:

— expiration of a timer,

— queue full,

— bus-arbitrator function change from PASSIVE to ACTIVE.

Parameter setting for DLE functions such as:

— installation of the various data-link objects, that is, the resources associated with identifiers for cyclical and aperiodic services, queues for messages, the bus-arbitrator's scanning table,

— turnaround time,

— the basic timer T0,

— the maximum value of the restart counter,

— the assigning of a physical address to the device.

Access to DLE error and performance counters such as:

— DLPDU fragment error,

— FCS error,

— DLPDU length error,

— pierced DLPDU error,

— coding error,
— overrun error,
— DLPDU sequencing error,
— DLPDU type not identified,
— message reception file saturated,
— message acknowledgement lost,
— number of cyclical transactions,
— number of aperiodic transactions,
— number of message service transactions,
— number of messages refused,
— total number of restarts.

This set of device-level locally available services is used locally by a specific application dedicated to system management.

7.6.3 PhL interface

See IEC 61158-2.

7.7 Conformance

The DLL provides the DLS-user with the following set of services:

a) buffer writing,
b) buffer reading,
c) buffer transfer,
d) specified explicit request for buffer transfer,
e) free explicit request for buffer transfer,
f) unacknowledged message transfer request,
g) acknowledged message transfer request.

A DLE provides the buffer transfer service and the buffer writing service in all cases, and, according to conformance classes, at least one of the following services:

1) buffer reading,
2) specified or free explicit request for a buffer transfer,
3) unacknowledged message transfer request,
4) acknowledged message transfer request.

The reading service makes it possible to extract data from the DLL and route it to the DLS-user. The service is initiated by the DLS-user for the purpose of reading the value of an identified variable.

The writing service makes it possible to deposit data from the DLS-user in the DLL. The service is initiated by the DLS-user for the purpose of writing the value of an identified variable.

The buffer transfer service participates in variable exchanges. In this service the producer of a variable emits a variable response DLPDU for the variable concerned. This DLPDU is then received by the consumer or consumers of the variable. The service generates a report (indication) for each variable emitted or received via the medium and sends this report to the DLS-user.

The specified explicit request for buffer transfer makes it possible to give the bus-arbitrator a request for the circulation of one or more identifier DLPDUs. The transfer of the associated identified variable or variables is thus triggered. This service is initiated by the DLS-user for the purpose of placing one or more identifiers in circulation.

When the service is requested it is linked to a specific identifier, since the request is transmitted to the bus-arbitrator during the cyclical variable exchange concerning this identifier.

The request is fulfilled during the aperiodic variable window (triggered scanning of variables) or during the bus-arbitrator's periodic window (periodic triggered scanning of variables) depending on configuration.

The free explicit request for buffer transfer makes it possible to give the bus-arbitrator a request for the circulation of one or more identifier DLPDUs. Transfer of the associated identified variable or variables is thus triggered. This service is initiated by the DLS-user for the purpose of placing one or more identifiers in circulation.

No identifier is specified when this service is requested. The request is transmitted to the bus-arbitrator during the first exchange of a variable produced by the initiating entity.

The request is fulfilled only during the bus-arbitrator's aperiodic variable window (triggered scanning of variables).

When the bus-arbitrator receives an explicit (specified or free) request for a buffer transfer it begins a transaction that conforms to the buffer transfer service previously described.

For each free request for a buffer transfer, two priority levels are provided: the bus-arbitrator processes the request in the urgent service or normal service mode.

Only urgent service is provided for a specified explicit request for a buffer transfer.

These two types of service make it possible to manage the priority levels of triggered variable exchanges without negative impact on cyclical variable exchanges or message transfers.

The unacknowledged message transfer request service causes the bus-arbitrator to put a message identifier DLPDU in circulation. Circulation of this DLPDU triggers the transfer of an unacknowledged message.

The request is fulfilled either during aperiodic scanning or during the bus-arbitrator's periodical scanning, depending on configuration.

If the request is filled during aperiodic scanning the unacknowledged message transfer is deemed aperiodic. The message transfer request is transmitted to the bus-arbitrator during the first exchange of a variable produced by the source entity. The message identifier DLPDU is then put in circulation during the bus-arbitrator's aperiodic messages window (triggered message scanning).

If the request is filled during the bus-arbitrator's periodic scanning, the unacknowledged message transfer is deemed cyclical. In this case the message transfer request is in the periodical window because of configuration. Thus the message identifier DLPDU is put in circulation without prior request.

In both cases the message exchange between entities then makes use of an unacknowledged message response DLPDU.

When the transaction has been completed the source entity sends the bus-arbitrator a DLPDU indicating the end of the message transaction.

The acknowledged message transfer request service causes the bus-arbitrator to put a message identifier DLPDU in circulation. Circulation of this DLPDU triggers the transfer of an acknowledged message.

The request is fulfilled either during aperiodic scanning or during the bus-arbitrator's periodical scanning, depending on configuration.

If the request is filled during aperiodic scanning the acknowledged message transfer is deemed aperiodic. The message transfer request is transmitted to the bus-arbitrator during the first exchange of a variable produced by the source entity. The message identifier DLPDU is then put in circulation during the bus-arbitrator's aperiodic messages window (triggered message scanning).

If the request is filled during the bus-arbitrator's periodic scanning, the acknowledged message transfer is deemed cyclical. In this case the message transfer request is in the periodical window because of configuration. Thus the message identifier DLPDU is put in circulation without prior request.

In both cases the message exchange between entities then makes use of an acknowledged message response DLPDU.

Upon reception of the message, the receiving entity immediately sends the source entity an acknowledgement DLPDU.

When the transaction has been completed the source entity sends the bus-arbitrator a DLPDU indicating the end of the message transaction.

NOTE Specifications for the DLL support several Application message service entities of various types.

A DLE offers the conformance classes enumerated in Table 12:

Table 12 – Conformance classes

Class	1	2	3	4	5	6	7	8	9
Service									
Buffer transfer	X	X	X	X	X	X	X	X	X
Buffer write	X	X	X	X	X	X	X	X	X
Buffer read		X	X	X	X	X	X	X	X
Free explicit request				X	X			X	X
Specified explicit request			X		X				X
Unacknowledged message						X	X	X	X
Unacknowledged message							X	X	X

Annex A
(informative)

Exemplary FCS implementation

This annex provides an example implementation of FCS generation and FCS syndrome checking for Type 7.

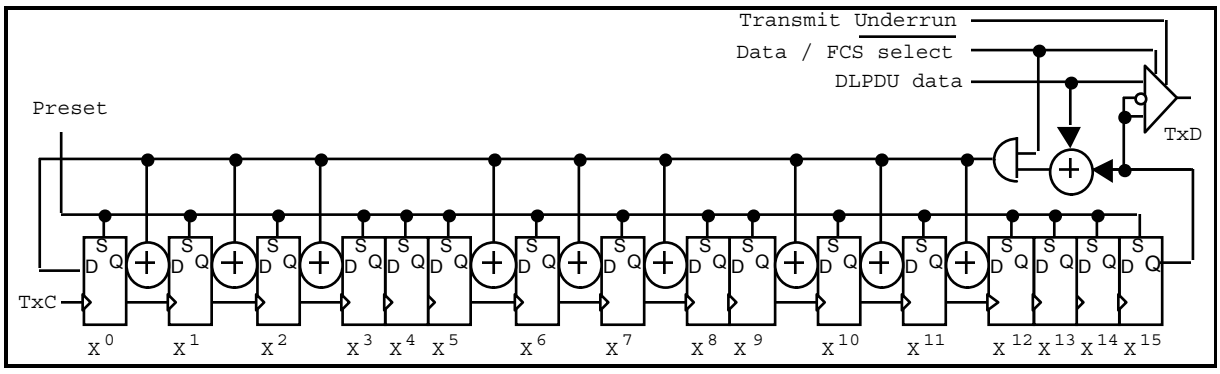


Figure A.1 – Example of FCS generation

In this example, shown in Figure A.1, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial preset of the register before transmission serves to include the initial L(X) term in the FCS computation. Feedback is disabled during transmission of the FCS itself, and the FCS is transmitted complemented to provide the final L(X) inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

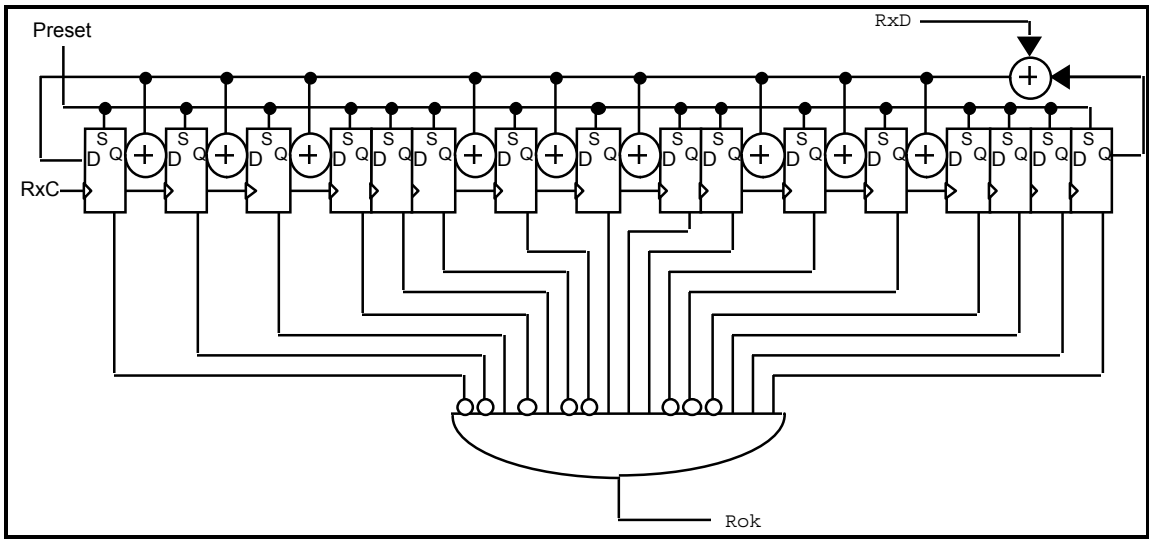


Figure A.2 – Example of FCS syndrome checking on reception

In this example, shown in Figure A.2, the residual FCS is computed in a similar register. The Q outputs of the 16 flip-flops are compared to the expected residual value by the 16-input “and” gate, half of whose inputs are complemented.

In this example, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial pre-set of the register before transmission serves to include the initial L(X) term in the FCS computation. Feedback is disabled during transmission of the FCS it self, and the FCS is transmitted complemented to provide the final L(X) inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

Annex B
(informative)

Object modeling

B.1 Modeling of the IDENTIFIER object

Class:	IDENTIFIER
Key Attribute:	I_Number
Attribute:	Type_APER.CYC (TRUE, FALSE)
Constraint:	Type_APER.cyc=TRUE
Attribute:	*Inherit from* BUFFER
Attribute:	Validity_ID_APER.cyc (TRUE, FALSE)
Attribute:	RQ (IN_PROGRESS, VACANT)
Attribute:	Type_MSG.cyc (TRUE, FALSE)
Constraint:	Type_MSG.cyc=TRUE
Attribute:	*Inherit from* QUEUE
	(Q_MSG_CYC)
Attribute:	Validity_ID_Consumed (TRUE. FALSE)
Attribute:	Type_Prod (TRUE.FALSE)
Constraint:	Type_Prod=TRUE
Attribute:	*Inherit from* BUFFER
	(B_DAT.prod)
Attribute:	Validity_ID_Produced (TRUE. FALSE)
Attribute:	Transmission (PERIODIC, APERIODIC)
Constraint:	Transmission=PERIODIC
Constraint:	Type_APER.cyc=FALSE
Attribute:	Aperiodic_Request (NON_SPEC_RQ_INHIBITED, FREE, NONE)
Constraint:	Aperiodic_Request=NON_SPEC_RQ_INHIBITED
Attribute:	*Inherit from* BUFFER
	(B_REQ)
Attribute:	RQ (IN_PROGRESS, VACANT)
Constraint:	Aperiodic_Request=FREE
Attribute:	RQ (IN_PROGRESS, VACANT)
Attribute:	PR (URGENT, NORMAL)
Attribute:	Reference QUEUE
	(Q_REQ1 OR Q_REQ2)
Constraint:	Type_MSG.cyc=FALSE
Attribute:	Type_MSG.aper (TRUE, FALSE)
Constraint:	Type_MSG.aper=TRUE
Attribute:	MSG (IN_PROGRESS, VACANT)
Attribute:	Reference FILE
	(Q_MSG.aper)

B.2 Description of the IDENTIFIER object attributes

B.2.1 I_Number

This attribute corresponds to a 16-bit word associated with a DL-subnetwork variable and which characterizes the variable uniquely in the DL-subnetwork.

B.2.2 Type_APER.cyc

This attribute characterizes the range of the buffer transfer specified requests.

When a local DLS-user performs a buffer transfer specified explicit request, the associated identifiers are deposited in the B_REQ buffer associated with the identifier indicated at the request.

This request can either remain local in case the B_REQ buffer is scanned during the cyclic part of the scanning table or be transmitted to the bus arbitrator in the other case.

This attribute takes the value TRUE when the buffer transfer specified explicit request service is supported by this identifier and that the scanning of the B_REQ buffer is included in the cyclic part of the scanning table (no RP_DAT_RQ to be transmitted). This service can be offered by produced, consumed, produced and consumed identifiers or by identifiers which are neither produced nor consumed.

This attribute takes the value FALSE:

— when no buffer transfer specified explicit request service is supported by this identifier,

— when the buffer transfer specified explicit request service is supported by this identifier but the request must be transmitted to the bus arbitrator (Scanning of the B_REQ buffer included in the cyclic part of the scanning table). This service can be offered only by the produced identifiers.

When the identifier supports the specified explicit request service, the Type_APER.cyc attribute corresponds to RQ_INHIBITED.

B.2.3 Inherit from BUFFER

The Identifier class inherits from the BUFFER class, thus giving it the attributes relative to B_REQ.

B.2.4 ID_APER.cyc validity

This binary attribute concerns any identifier supporting the buffer transfer specified explicit request periodic service. When this attribute takes the value TRUE, the list of identifiers contained in B_REQ transits on the bus after passage of the identifier request concerned. When this attribute takes the value FALSE, the producer/consumer entity does not answer the ID_RQ requests on this identifier.

B.2.5 RQ

This attribute characterizes the status of a buffer transfer request.

When the buffer transfer explicit requests are transmitted to the bus arbitrator, the requester has information representing the request status (bit RQ).

In the case of a specified request, this attribute has the value IN_PROGRESS, between the instant at which the request was placed in B_REQ and the end of transmission of the RP_RQ following reception of the corresponding ID_RQ type PDU.

In the case of a free request, this attribute has the value IN_PROGRESS between the instant at which a DLCEP-identifier is assigned to the queue in which the request has been placed and the end of transmission of the RP_RQ following reception of the corresponding ID_RQ type PDU.

Outside these intervals, the status of the request is VACANT to indicate that the identifier does not bear any buffer transfer explicit request.

B.2.6 Type MSG.cyc

Two message transfer modes are defined: one (aperiodic) uses a queue Q_MSG.aper reserved for this transfer mode and linked dynamically to identifiers, the other (cyclic) uses a set of Q_MSG.cyc queues of which each one is linked statically to a specific identifier.

This attribute takes the value TRUE when the scanning of the message identifier DLPDU is included in the bus arbitrator cyclic scanning (no RP_DAT_MSG to be transmitted). This service can be offered by produced, consumed, produced and consumed identifiers or by identifiers which are neither produced nor consumed.

It takes the value FALSE:

— when no messaging service is supported by this identifier,

— when the messaging service supported by this identifier requires that the request be transmitted to the bus arbitrator so that the message identifier DLPDU circulates in the messaging periodic window. This service can only be offered by produced identifiers (see Type_MSG.aper).

Inherit from FILE

The Identifier class inherits from the Queue class which contains all the attributes concerning the Q_MSG.cyc resources. The latter corresponds to the queue associated with each identifier supporting the cyclic messaging.

B.2.7 Type_Cons

This attribute defines if a DLCEP-identifier is consumed or not.

Inherit from BUFFER

The Identifier class inherits from the BUFFER class thus giving it the attributes concerning the B_DATcons.

B.2.8 ID_Consumed_Validity

This binary attribute concerns all consumed identifiers. When this attribute takes the value TRUE, a DLCEP-identifier consumes the data meant for it. When this attribute takes the value FALSE, a DLCEP-identifier does not answer in consumption to the requests of the bus arbitrator on this identifier.

B.2.9 Type_Prod

This defines if a DLCEP-identifier is produced or not.

Inherit from BUFFER

The identifier class inherits from the BUFFER class thus giving it the attributes concerning the B_DAT.prod.

B.2.10 Produced ID validity

This binary attribute concerns all the produced identifiers. When this attribute takes the value TRUE, the value contained in B_DAT.prod transits on the bus after passage of the variable identifier.

When this attribute takes the value FALSE, a DLCEP-identifier does not reply in production to the requests of the bus arbitrator on this identifier. Besides, the possible associations between this identifier and Q_RQEQ1, Q_REQ2 or Q_MSG.aper are interrupted.

B.2.11 Transmission

A DLCEP-identifier is exchanged periodically if it appears in the cyclic part of the bus arbitrator scanning table. Otherwise the exchange is triggered on explicit request of a DLS-user.

This attribute conditions the possibility of having some services on this identifier; buffer transfer specified explicit request with RQ_INBIBITED false, buffer transfer free explicit request and periodic messaging.

It thus allows restricting the latter to cyclic identifiers only.

Aperiodic request.

Three buffer transfer explicit request services are defined.

— The identifier which will support the request is specified. This request is not transmitted to the bus arbitrator. The B_REQ is scanned cyclically (RQ_INHIBITED true). In this case the attribute takes the value TRUE.

— The identifier which will support the request is specified. This request is transmitted to the bus arbitrator, the B_REQ buffer is not scanned cyclically. In this case the RQ_INHIBITED attribute takes the value FALSE and the Aperiodic_Request attribute the value SPEC_NON_RQ_INHIBITED value.

— The identifier which will support the request is not specified. The Aperiodic_Request takes the value FREE.

These three types of service are exclusive. If the Aperiodic_Request attribute has the value NOTHING, the identifier will not support any.

B.2.12 Inherit from BUFFER

The identifier class inherits from the BUFFER class, thus giving it the attributes relative to B_REQ.

B.2.13 RQ

See above.

B.2.14 PR

This attribute indicates if the buffer transfer free explicit request must be processed urgently or at the normal speed.

For a specified explicit request, only the urgent priority exists.

The existence of two priority levels for the free explicit requests implying the corresponding resources in the station (Q_REQi) and in the bus arbitrator (Q_IDRQi).

B.2.15 Reference_QUEUE

This attribute allows designating the Q_REQ1 or Q_REQ2 resource, objects resulting from the instance of the queue class. These queues are available to all the identifiers supporting the buffer transfer free explicit request service.

Q_REQ1 contains list(s) of identifiers from one or several normal free explicit requests which have not yet been transmitted to the bus arbitrator.

B.2.16 MSG.aper type

Two message transfer modes are defined: one (aperiodic) uses an queue reserved for this transfer mode and dynamically linked to identifiers, the other (cyclic) uses a set of queues each of which is statically linked to a specific identifier.

This attribute takes the value TRUE when the scanning of the message identifier DLPDU is not included in the bus arbitrator cyclic scanning, that is it is necessary to transmit a bus arbitrator request so that consequently the message identifier DLPDU circulates in the course of the aperiodic window reserved for the messaging. This service can only be offered by produced identifiers.

It takes the value FALSE:

— When no messaging service is supported by this identifier,

— When the scanning of the message identifier DLPDU is included in the bus arbitrator cyclic scanning (see Type_MSG.cyc).

B.2.17 MSG

This attribute characterizes the state of a message aperiodic transfer request.

It has the value IN_PROGRESS between the instant when the identifier has been associated with the queue reserved for transfer of messages and the reception of the message transaction linked to the corresponding ID_MSG DLPDU. Outside this interval, the status of the request is VACANT to indicate that the identifier bears no message transfer request.

B.2.18 QUEUE_Reference

This attribute allows designating the Q_MSG.aper object resource resulting from the instance of the Queue class. This queue is available to all the identifiers supporting the aperiodic messaging service.

It contains the message(s) resulting from one or several message transfer requests which have not yet been satisfied.

B.3 Modeling of the QUEUE object

Class:	QUEUE
Key Attribute:	Q_Number
Attribute:	Queue_status (EMPTY, FULL, PLACE AVAILABLE)
Attribute:	Type (Q_REQ, Q_MSG)
Constraint:	Type=Q_REQ
Attribute:	Content Q_REQ (identifier lists queue)
Attribute:	Connection (TRUE, FALSE)
Constraint:	Type=Q_MSG
Attribute:	Content Q_MSG (message queue)
Attribute:	Num_Attachment (integer)

B.4 Description of the QUEUE object attributes

B.4.1 Q_Number

This attribute associated with a queue allows characterizing the latter uniquely in the DLE.

B.4.2 Queue_status

This attribute describes the actual use of the queue. The possible values are:

An Q_MSG queue is empty when it contains no message. An F_MSG queue can contain an empty message; this will consist only of the source address and the destination address. In this case the Queue_Status attribute takes the value PLACE AVAILABLE or FULL.

B.4.3 Q_REQ content

This attribute corresponds to the content of the Q_REQ1 and Q_REQ2 queues. It is the set of identifier lists which have not yet been transmitted to the bus arbitrator. Each list has been covered locally by a free explicit request and is transmitted on the local link automatically. At an RP_RQ reply several lists can be transmitted in so far as all these lists are not made of more than 64 identifiers.

B.4.4 Attachment

This attribute indicates if the queue is associated with a DLCEP-identifier or not.

B.4.5 Q_MSG content

This attribute corresponds to the content of the Q_MSG.received, Q_MSG.Cyc and Q_MSG.aper. files. It is a messaging file.

In the case of Q_MSG.cyc and Q_MSG.aper. files, it includes all the messages covered by a local message transfer request which has not been satisfied. In the case of Q_MSG.received, it is the set of messages for which the station has been recognized as destination.

B.4.6 Attachment_Number

This attribute indicates how many associations are missing between the queue and its identifiers. It is a whole number greater than 0 and lower than the number of places in the queue. It thus corresponds to the number of messages present in the queue less the number of identifiers associated with the latter.

B.5 Modeling of the BUFFER object

Class:	BUFFER
Key Attribute:	B_Number
Attribute:	Type (B_DAT, B_REQ)
Attribute:	Availability (OCCUPIED, FREE)
Constraint:	Type=B_DAT
Attribute:	Content Prod_Cons (variable value)
Constraint:	Type=B_REQ
Attribute:	Content B_REQ (list of identifiers)

B.6 Description of the BUFFER object attributes

B.6.1 B_Number

This attribute associated with a buffer allows characterizing the latter uniquely in a DLE.

B.6.2 Type

This attribute allows determining if the buffer is of the B_DAT type (produced or consumed) or B_REQ.

B.6.3 Availability

This attribute is meant to solve the problems of access conflict which could take place.

B.6.4 Prod_Cons content

This attribute corresponds to the content of a B_DAT or B_DATcons buffer. This is the value of a produced or consumed variable.

B.6.5 B_REQ content

This attribute corresponds to the content of a B_REQ buffer. It is a list of identifiers coming from a buffer transfer specified explicit request.

Annex C
(informative)

Topology of multi-segment DL-subnetwork

C.1 Introduction

This annex describes how to specify the topology of a multi-segment DL-subnetwork. The aim is to propose a data structure, which could be minimal while allowing correct operation of the bridge retransmission function.

The topology of a DL-subnetwork can first be specified globally, in order to verify a certain number of properties (connectivity, non-meshing, etc.); then on the basis of this specification the local data base specific to each bridge must be calculated in order to ensure it operates correctly.

Although this appendix proposes a method to achieve this goal, only the specifications of the data structures, global or local to each bridge, which define the DL-subnetwork topology, as well as the properties which it should fulfil, must be taken into account in the standard. The suggested method shows how to obtain a solution to the problem by taking into account certain optimization problems.

C.2 Global specification

The topology of a multi-segment DL-subnetwork can be defined by the following elements:

— the set S of its segments: $S = \left\{ S_i \mid i \in [1, n] \right\}$

— the set B of its bridges: $B = \left\{ b_i \mid k \in [1, m] \right\}$

— and for each bridge of B, the data of a matrix B^k of dimension n × n. whose coefficients b^k_{ij} are defined by:

— b^k_{ij}= 0 if i = j.

— b^k_{ij}= • if the bridge b^k does not allow transfer of messages from segment S_i to segment S_j.

— b^k_{ij}= a with a ∈ **R**+*, if the bridge b^k allows the transfer of messages from segment S_i towards segment S_j, with a as load coefficient which allows taking into account of a different efficiency rate according to the transfers.

A load coefficient b^k_{ij} can represent the load, as a rate of occupation of the medium, of the retransmission segment s_j. In reality, either the destination is directly s_j., or there are several paths possible, passing through intermediate segments, to reach s_j. and in this case the choice shall be to pass by the least loaded path.

It is of course possible to take as coefficients of the same value (1 for example).

If a bridge allows two-way retransmission with the same load coefficient for the two directions, its matrix is symmetrical.

The matrix B^k of a bridge also allows knowing all the segments to which it is connected:

— either in reception, S_r^k = { segments whose corresponding line in the matrix includes at least one non-null finite coefficient} note n_r^k = card (S_r^k),

— or in transmission, S_e^k = {segments whose corresponding column in the matrix includes at least one non-null finished coefficient}; note n_e^k = card (S_e^k).

C.3 Local specification

The information which a bridge must have locally allows it to answer the following question: when I receive a message on such a segment $sr_i \in S_r^k$ destined for such another segment s_j, must I do nothing or must I retransmit on segment $se_i \in S_e^k$).

To fulfil this purpose, it is enough to allocate to each bridge b^k a transfer matrix T^k with dimensions $n_r^k \times$ n, whose elements r_{ij}^k are defined by:

— the line index $i \in$ [I, n_r^k] references segments s_{r_i} connected in reception ($\in S_r^k$),

— the column index j ∈ [I, n] references the segments s_j of the DL-subnetwork ($\in S$),

— $r_{ij}^k = 0$ if on reception of a message on segment $s_{r_i} \in S_r^k$ addressed to segment s_j, the bridge must retransmit to segment se_i,

— $r_{ij}^k = s_{e_i}$, with $s_{e_i} \in$ S_e^k; if on reception of a message on segment $s_{r_i} \in S_r^k$ addressed to segment s_j, the bridge must retransmit to segment s_{e_i}.

NOTE Indexes i and j correspond to channel numbers whereas sr_i is the segment connected in reception to channel i and s_{e_i} is the segment connected in transmission to channel I.

C.4 Properties

The properties which should satisfy the DL-subnetwork are connectivity and non-meshing.

Connectivity consists in ensuring that there is always a path from any given segment of S to any other segment of S.

Non-meshing consists in ensuring that the transmission of a message from a transmitter located on segment s_e and addressed to a receiver located on segment s_r can be routed by only one path (thus preventing the message from being received more than once).

In fact, it is the definition of the local specification of each bridge and the calculation of its transfer matrix which ensure this property: by definition, on reception of a message on segment sr_i addressed to segment s_j, either the bridge does not retransmit it, in particular if segment s_{r_i} is equal to segment s_j, or the bridge retransmits it on a single segment s_{e_i}, whereas by calculation of the matrix, it is necessary to make sure that one and only one bridge, connected in reception to segment s_{r_i}, retransmits the message.

C.5 Methods

The method consists in calculating the matrix C of the minimum loads of the paths between any two segments. By this way we check the connectivity since none of these coefficients is infinite.

The second stage consists in calculating the transfer matrix of each bridge so that this gives the global DL-subnetwork the property of non-meshing while preserving the property of connectivity.

C.5.1 Minimum load matrix

a) P row load matrix

Definition: the load matrix of rank P, C^P, with dimensions n × n, is the matrix whose coefficients C_{ij}^{P} give the minimum load to travel from segment *i* towards segment *j* by passing via not more than P bridges.

We have:

— C_{ij}^{P} = 0,

— C_{ij}^{P} = •, if there is no path from segment i to segment j by passing via not more than P bridges,

— if C_{ij}^{P} is finite, the optimal corresponding path includes not more than P bridges (it can include less than P).

Obtaining by recurrence:

— The coefficients C_{ij}^{1}, of the load matrix of row 1, C^1, are given by:

— $C_{ij}^{1} = \underset{k}{min} \in [1,\ m] \left(b_{ij}^{k} \right)$

We have:

— C_{ij}^{1} = 0,

— C_{ij}^{1} = •, if there is no permanent bridge allowing transfer from segment i towards segment j,

— C_{ij}^{1} finite, if there exists one or more bridges allowing the transfer from segment i towards segment j.

— the coefficient C_{ij}^{P}, for P > 1, of the load matrix of row P, C^P, are given by:

— $C_{ij}^{P} = \underset{k}{min} \in [l,m] \left(C_{ik}^{1} + C_{lg}^{P\text{-}1} \right)$

In reality, the minimum load between two segments i and j passing via P bridges corresponds to a path composed of:

— a bridge allowing the passage from segment i to segment k, with a minimum load c_1,

— and a path with minimum load c_2 between this segment k and segment j, passing via P-1 bridges.

The intermediate bridge is most often used so that $c_1 + c_2$ are minimal.

b) Minimum load matrix

Definition: the minimum load matrix *C*, dimension n × n, is the matrix whose coefficients c_{ij} give the minimum load to go from segment i to segment j. We thus have:

— c_{ii} = 0

— c_{ij} = •, if there is no path from segment i to segment j,

— if *cij* is finite, there is a path s_1 = i, s_2 .... s_L. = j with length L, passing via bridges bk*,

— h ∈ [1, *L* - 1], whose load is cij with cij = $c_{ij} = \prod_{h=1}^{h=L-1?} b^{k*}_{Sn\ Sn+1}$

By definition the connectivity is well ensured by the fact that all the minimum load matrix coefficients *C* are finite.

Property: the minimum load matrix *C* is the limit of the load matrixes of row P, when P tends to infinity: $C = \lim_{P\to\infty} C^P$.

In reality, the series C^P is stationary, at least from row m, where m is the number of bridges. A path which passes via more than m bridges passes at least twice via the same bridge and cannot thus have a minimum load.

Suppose Q the row from which the C^P series is stationary ($C^{Q+1} = C^Q$).

C.5.2 Calculation of the transfer matrices

Suppose now that the DL-subnetwork is connected.

The transfer matrices T^k are calculated by iteration according to the number P of bridges, from 1 to Q, requiring a minimum load path from a source segment s and a destination segment d.

At the start, $T^k = 0$ for every k.

The transfer matrix coefficients are referenced in the same manner as in L.3, that is:

— the line index i ∈ [1, n_r^k] references the s_{r_i} segments connected in reception (∈ S_r^k),

— the column index j ∈ [1, *n*] references the s_j segments of the DL-subnetwork (∈ *S*).

a) Passage of a segment s to a segment d via 1 bridge

— For all pairs of segments s and d such that c^1_{sd} is finite.

— for one and only one k (as selected) such that $c^1_{sd} = b^k_{sd}$,

— the following assignment is performed for the transfer matrix T^k:

— for i ∈ [1, n_r^k] such that $s_{r_i} = s_i$ for j ∈ [1, *n*] such that $s_j = d, r^k_{ij} = s_j$

b) Passage of a segment *s* to a segment d via P bridges

— For every pair of segments *s* and d such that c^p_{sd} is finite whereas c^{p-1}_{sd} is infinite.

— For one and only one k (as selected) such that $\exists s' \mid C_{sd}^{P} = b_{ss'}^{k} + c_{sd}^{P\text{-}1}$,

— the following assignment is thus performed for the T^k transfer matrix:

— for $i \in [1, n_r^k]$ such that $s_{r_i} = s_i$ for $j \in [1, n]$ such that $s_j = d... \; r_{ij}^{k} = s_j$ with $s_i = s'$

NOTE In the last two paragraphs, the bridge k which verifies the necessary property is not necessarily unique, but an assignment must be made for a single bridge to ensure the property of non-meshing.

Annex D
(informative)

Management of transmission errors

D.1 Transmission of RP_DAT_XX

If a transmission error is detected during the transmission of a given response DLPDU, the transmission is interrupted, and:

— the DL-SEND primitive is not generated;

— in the case of a configured identifier for the service of the free explicit request service, the RQ indicator is not modified;

— in the case of a DLCEP-identifier configured for message aperiodic transfer, the MSG indicator value is unchanged.

The transmission of DL-SEND_ind and the management of the RQ and MSG indicators are performed only at the end of the RP_DAT_XX DLPDU transmission and if the variable producer has not detected an error during transmission. The evaluation DL-subnetwork of Figure D.1 describes this operation.

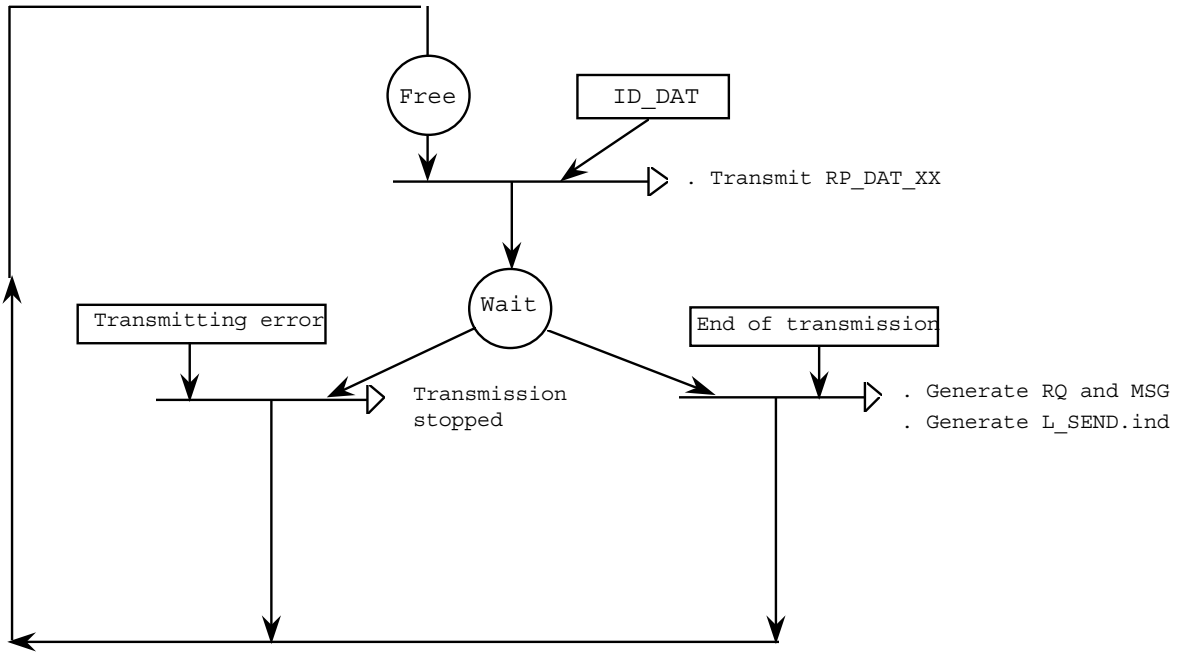


Figure D.1 – Evaluation DL-subnetwork for transmission of RP_DAT_XX

D.2 Transmission of a free RP_RQ(1/2)

On detection of an error during transmission of RP_RQ1 or RP_RQ2, forming continuation of a buffer transfer free explicit request, the producing entity does not generate any DL-FREE-UPDATE confirm primitive and does not scroll the list of identifiers. The evaluation DL-subnetwork of Figure D.2 describes this operation.

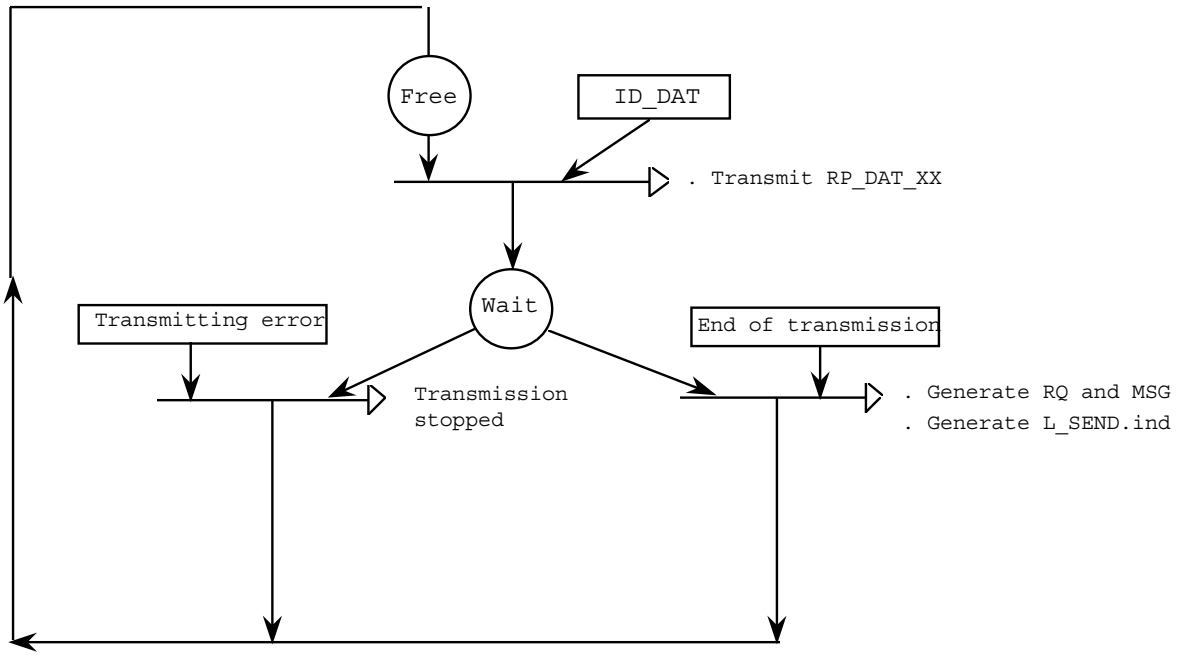


Figure D.2 – Evaluation DL-subnetwork for transmission of a free RP_RQ(1/2)

After transmission of the RP_RQ without error, the request is detached from the identifier concerned (RQ set to 0); If the FIFO is not empty, the request will be attached at the next reception of an ID_DAT concerning a DLCEP-identifier configured for this service. The RQ bit of this identifier shall be set to 1.

D.3 Transmission of the specified RP_RQ1

On detection of an error during transmission of RP_RQ1, following a buffer transfer specified explicit request, the producing entity interrupts the transmission of the DLPDU and does not generate a DL-SPEC-UPDATE confirm primitive.

Besides, the RQ indicator remains set to 1: the buffer transfer request shall be transmitted to the BA at the next transaction in case the RQ_INHIBITED indicator of the identifier has the FALSE value. The evaluation DL-subnetwork of Figure D.3 describes this operation.

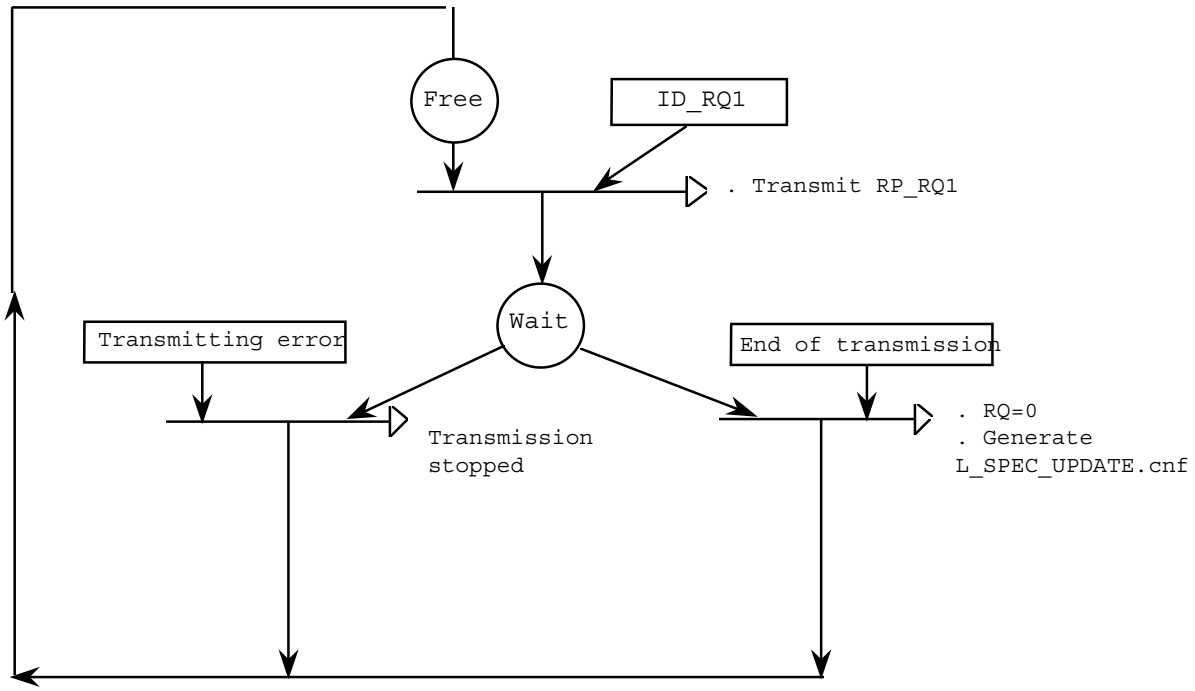


Figure D.3 – Evaluation DL-subnetwork for transmission of the specified RP_RQ1

D.4 Transmission of RP_MSG_NOACK

Several behaviors are possible during an error at transmission of an RP_MSG_NOACK:

— The producer/consumer entity interrupts the transmission and restores in the initial context, keeping the message. The transmission of the RP_MST_NOACK is then postponed to the next request from the bus arbitrator.

— The producer/consumer entity interrupts the transmission keeping only the message. The link layer indicates the loss of the message to the DLS-user with the help of the appropriate confirmation.

D.4.1 First behavior

On detection of an error during transmission of RP_MSG_NOACK, the producing entity interrupts the transmission of the DLPDU.

Besides:

— It does not generate the primitive of the DL-MESSAGE-ACK confirm primitive.

— The place in the message queue is not freed.

— If the identifier associated with the request is not included in the BA periodical scanning table, the MSG indicator remains positioned to 1; the message transfer request will be transmitted to the BA at the next transaction.

The evaluation DL-subnetwork of Figure D.4 describes this operation.

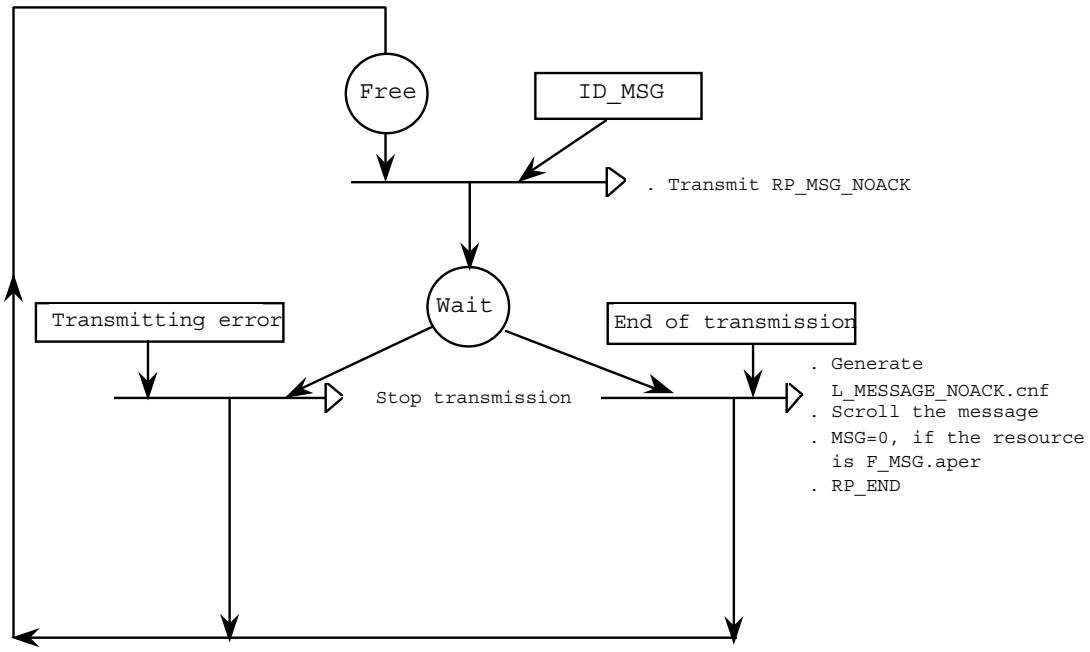


Figure D.4 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, first behavior

D.4.2 Second behavior

On detection of an error during the transmission of RP_MSG_NOACK, the producing entity interrupts the transmission of the DLPDU.

Besides:

— It generates a DL-MESSAGE-ACK confirm primitive whose status signifies Transmission error (1).

— The place in the queue is freed.

The evaluation DL-subnetwork of Figure D.5 describes this operation.

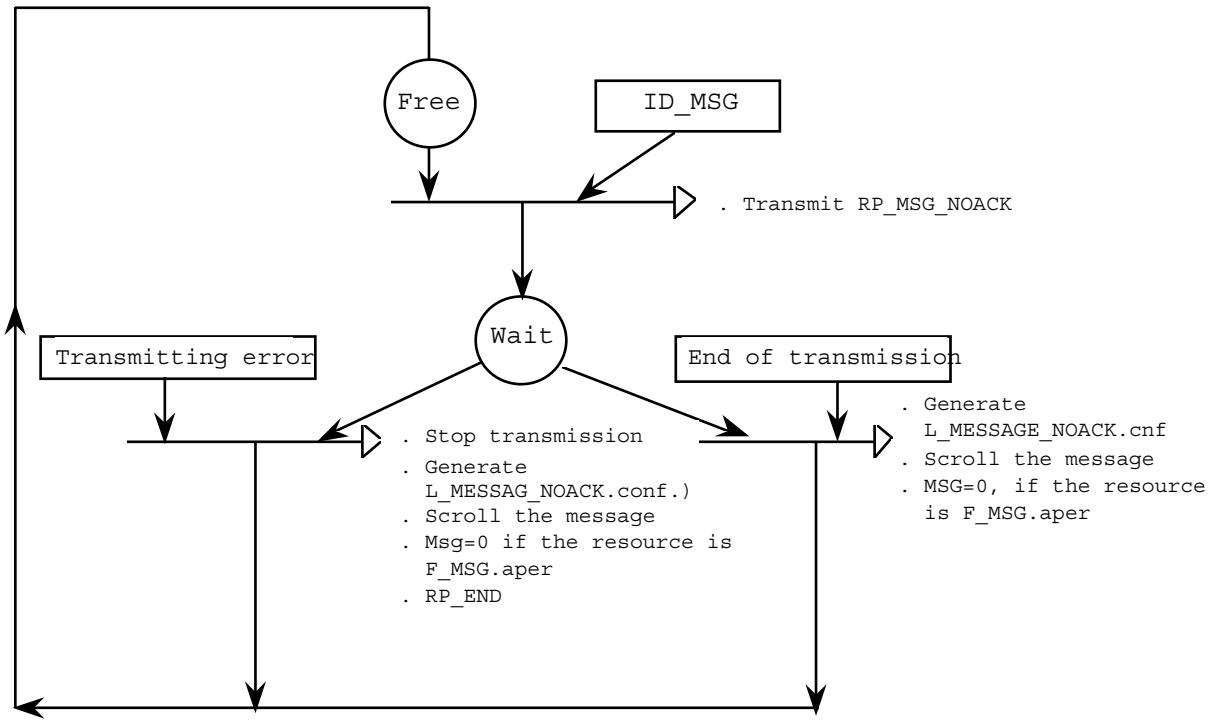


Figure D.5 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, second behavior

NOTE This status, not provided for by this standard, is to be added in the case of choice of this behavior at error during transmission.

D.5 Transmission of RP_MSG_ACK

Several behaviors are possible during an error at transmission of RP_MSG_ACK:

— The producer/consumer entity interrupts the transaction and is restored in the initial context, keeping the message. The transmission of the RP_MSG_ACK is then postponed to the next request coming from the bus arbitrator.

— The producer consumer entity interrupts the transmission and if the restart counter permits, retransmits the RP_MSG_ACK DLPDU immediately.

D.5.1 First behavior

On detection of an error during transmission of RP_MSG_ACK, the producing entity interrupts the transmission of the DLPDU.

Besides:

— It does not generate the DL-MESSAGE-ACK confirm primitive.

— The place in the message queue is not freed.

— If the identifier associated with the request does not form part of the BA periodic scanning table, the MSG indication shall be positioned at 1; the message transfer request will be transmitted to the BA at the next transaction.

The evaluation DL-subnetwork of Figure D.6 describes this operation.

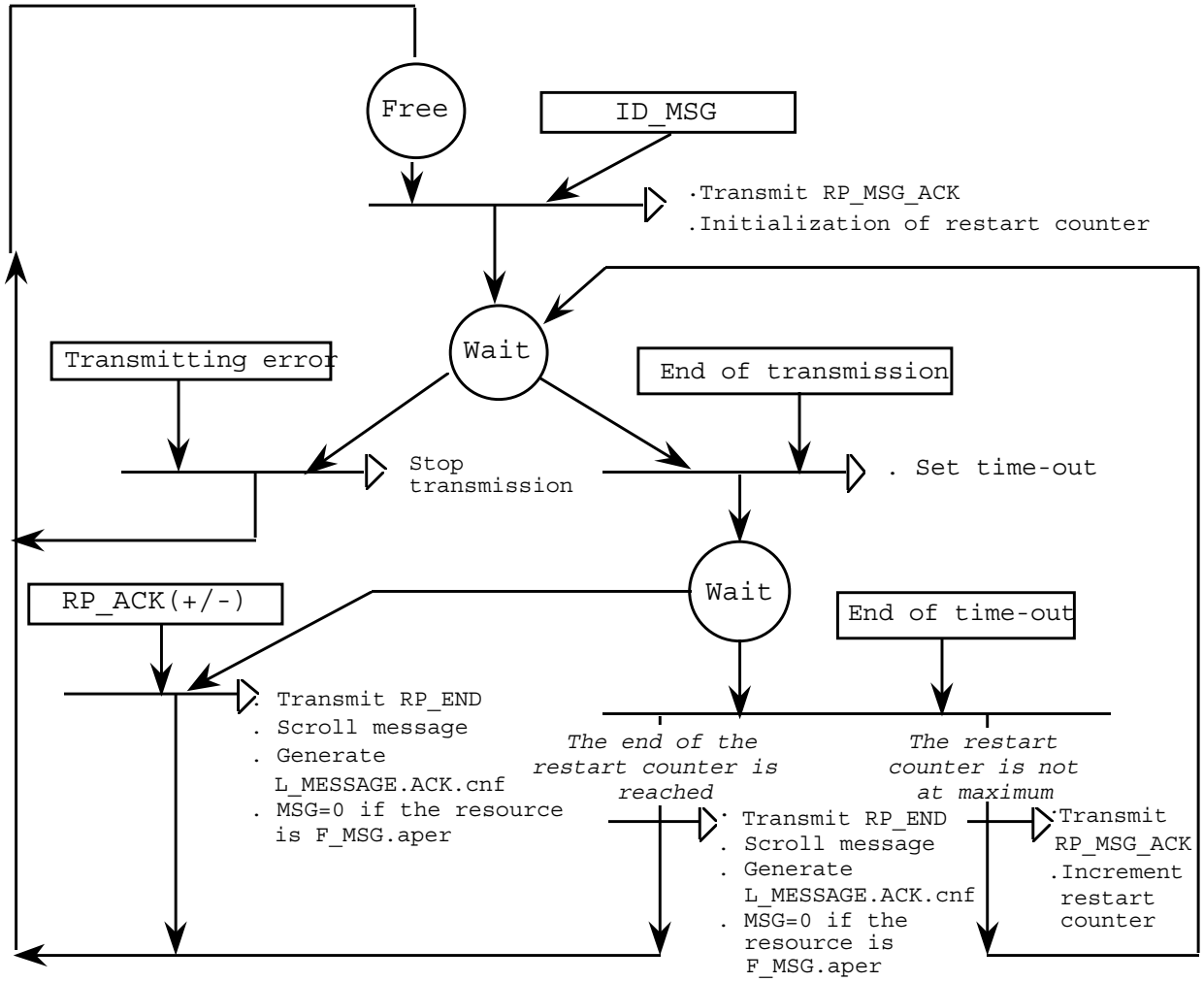


Figure D.6 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, first behavior

D.5.2 Second behavior

On detection of an error during transmission of the RP_MSG_ACK DLPDU, the actions by the producing entity are the following.

— Interruption of the DLPDU transmission.

— If the restart counter is not reached the producing entity performs a restart immediately without waiting for the expiration of time-out T6.

The evaluation DL-subnetwork of Figure D.7 describes this operation.

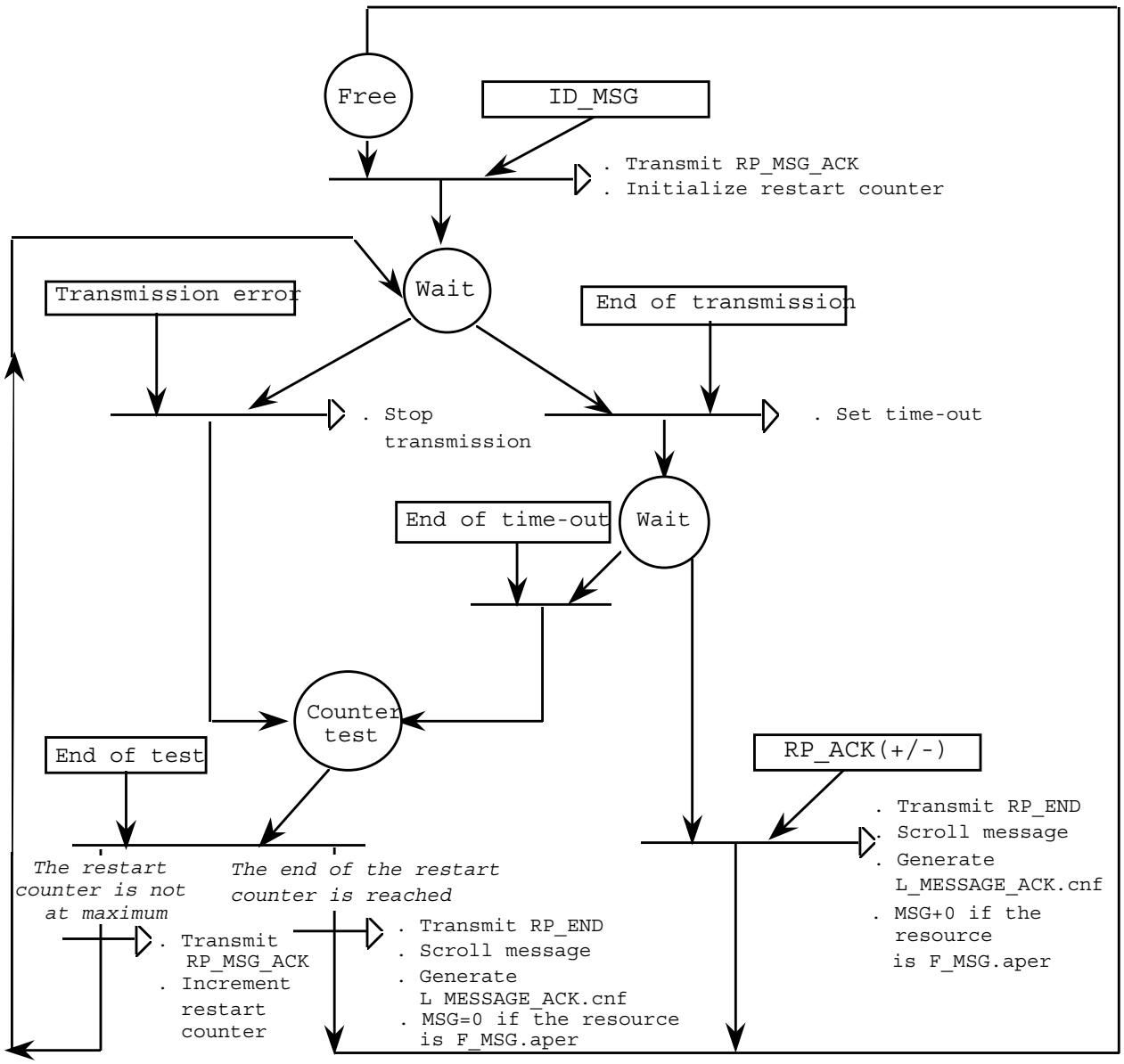


Figure D.7 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, second behavior

Bibliography

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

IEC 60847, *Characteristics of LANs*

IEC 60870-5-1, *Telecontrol equipment and systems – Part 5: Transmission protocols – Section one: Transmission frame formats*

IEC 60955, *Process data highway, Type C (PROWAY C), for distributed process control systems*

IEC 61131-2, *Programmable controllers – Part 2: Equipment requirements and tests*

IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

IEC/TR 61158-1 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-5-7, *Industrial communication networks – Fieldbus specifications – Part 5-7: Application layer service definition – Type 7 elements*

IEC 61158-6-7, *Industrial communication networks – Fieldbus specifications – Part 6-7: Application layer protocol specification – Type 7 elements*

IEC 61784-1 (Ed.2.0), *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

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

ISO/IEC 8802 (all parts), *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks*

- ISO/IEC TR 8802-1, *Specific requirements – Part 1: Overview of Local Area Network Standards*
- ISO/IEC 8802-2, *Specific requirements – Part 2: Logical link control*
- ISO/IEC 8802-3, *Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*
- ISO/IEC 8802-4, *Information processing systems – Local area networks – Part 4: Token-passing bus access method and physical layer specifications*
- ISO/IEC 8802-5, *Specific requirements – Part 5: Token ring access method and physical layer specifications*

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

ISO/IEC 9646-1, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts*

ISO/IEC 9646-2, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 2: Abstract test suite specification*

ISO/IEC 10646-1, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO/IEC 15802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common specifications – Part 1: Medium Access Control (MAC) service definition*

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

ISO 3309, *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure.*

ISO 8509, *Information processing systems – Open System Interconnection – Service Conventions*

ITU-T V.41, *Code-independent error-control system.*

EN 50170:1996, *General purpose field communication system*
Amendment 3-4:2002, *Data link layer definitions*
Amendment 7-4:2002, *Network management*

ANSI X3.66 (R1990), *Advanced data communication control procedures (ADCCP)*

ANSI X3.159, *Information Systems – Programming Language C*

ANSI X3J16 / ISO WG21 *committee draft working paper for a C++ standard*

Internet Engineering Task Force (IETF), *Request for Comments (RFC)*:

RFC 791, *Internet Protocol*

RFC 793, *Transmission Control Protocol*

RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets : MIB-II*

RFC 1643, *Definitions of Managed Objects for the Ethernet-like Interface Types*

IETF <draft-ietf-dhc-fqdn-option-01.txt>: March 2, 2001,*The DHCP Client FQDN Option*

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
P.O. 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