IEC 62056-3-1

Edition 1.0 2013-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling

Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse

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

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

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

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It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

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

Customer Service Centre - webstore.iec.ch/csc

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

A propos de la CEI

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

A propos des publications CEI

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

Liens utiles:

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

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,…).
Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

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

IEC 62056-3-1

Edition 1.0 2013-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling

Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XE

ICS 17.220; 35.110; 91.140.50

ISBN 978-2-8322-1046-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE –

Part 3-1: Use of local area networks on twisted pair with carrier signalling

FOREWORD

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

International Standard IEC 62056-3-1 has been prepared by IEC technical committee 13: Electrical energy measurement, tariff- and load control.

This first edition cancels and replaces the first edition of IEC 62056-31, issued in 1999, and constitutes a technical revision.

The main technical changes with regard to the previous edition are as follows:

- addition of a profile which makes use of the IEC 62056 DLMS/COSEM Application layer and COSEM object model,
- review of the data link layer which is split into two parts:
  - a pure Data Link layer;
  - a "Support Manager" entity managing the communication media;
- ability to negotiate the communication speed, bringing baud rate up to 9 600 bauds.

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

FDIS	Report on voting
13/1546/FDIS	13/1552/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.

A list of all parts of IEC 62056 series, published under the general title *Electricity metering data exchange – The DLMS/COSEM suite*, can be found on the IEC website.

Future standards in this series will carry the new general title as cited above. Titles of existing standards in this series will be updated at the time of the next eidition.

The numbering scheme has changes from IEC 62056-XY to IEC 62056-X-Y. For example, IEC 62056-31 becomes IEC 62056-3-1.

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

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

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

ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE –

Part 3-1: Use of local area networks on twisted pair with carrier signalling

1 Scope

This part of IEC 62056 describes three profiles for local bus data exchange with stations either energized or not. For non-energized stations, the bus supplies energy for data exchange.

Three different profiles are supported:

- base profile: this three-layer profile provides remote communication services;

  NOTE This first profile has been published in IEC 61142:1993 and became known as the Euridis standard.

- profile with DLMS: this profile allows using DLMS services as specified in IEC 61334-4-41;

  NOTE This second profile has been published in IEC 62056-31 Ed. 1.0:1999;

- profile with DLMS/COSEM: this profile allows using the DLMS/COSEM Application layer and the COSEM object model as specified in IEC 62056-5-3 Ed. 1.0:— and in IEC 62056-6-2 Ed. 1.0:— respectively.

The three profiles use the same physical layer and they are fully compatible, meaning that devices implementing any of these profiles can be operated on the same bus.

The transmission medium is twisted pair using carrier signalling and it is known as the Euridis Bus.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61334-4-41:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocol – Section 41: Application protocols – Distribution line message specification*

IEC 62056-51:1998 *Electricity Metering – Data exchange for meter reading, tariff and load control – Part 51: Application Layer Protocols*

IEC 62056-5-3 Ed. 1.0:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer*

ISO/IEC 8482:1993, *Information technology – Telecommunications and information exchange between systems – Twisted pair multipoint interconnections*

EIA 485 – *Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*

3 Abbreviations

Abbreviation	English rendering
ADP	Primary Station Address
ADG	General Secondary Address. Broadcast Address
ADS	Secondary Station Address
AGN	Normal Wakeup
AGT	General call for a General Energized Station
APDU	Application Protocol Data Unit
APG	General Primary Address
ARJ	COM field value: Rejection of authentication in remote programming exchange
ASDU	Application Service Data Unit
ASO	COM field value: Call to Forgotten Stations
AUT	COM field value: Authentication command
COM	Control field of the Data Link layer
COSEM	Companion Specification for Energy Metering
DAT	COM field value: Response of remote reading exchange
DES	Data Encryption Standard
DLMS	Distribution Line Message Specification (IEC 61334-4-41) Device Language Message Specification (IEC 62056-5-3)
DSDU	Data link Service Data Unit
DRJ	COM field value: Data Rejected Value of COM notifying the rejection of remote programming exchange data
Dsap	Transport data unit label. Coded over 3 bits. Its value is 6.
DTSAP	Destination of Transport Service Access Point
ECH	COM field value: Echo of remote programming exchange data
ENQ	Remote reading exchange request
EOS	COM field value: End of remote programming exchange
IB	Initialisation of the bus
MaxRetry	Maximum number retransmissions. Limited to 2.
MaxRSO	Maximum number of RSO listening windows. Fixed at 3.
PDU	Protocol Data Unit
PRE	COM field value: Pre-selection of energised stations
REC	COM field value: Remote programming exchange request
RSO	COM field value: Response to a call to forgotten stations
SEL	COM field value: Acknowledgement of the pre-selection of energized stations
STSAP	Source Transport Service Access Point
TAB	In the case of the Euridis profiles without DLMS and without DLMS/COSEM: data code. In the case of profiles using DLMS or DLMS/COSEM: value at which the equipment is programmed for Discovery
TABi	List of TAB field
TASB	Duration of an Alarm Signal on the Bus
TOAG	Maximum wait time for an energized station once selected, to recognise a general call AGN
TOALR	Wait before sending an AGN after reception of an AGN or AGT
TOL	Maximum waiting time for a request from the upper layer

Abbreviation	English rendering
TOPRE	Maximum waiting time for a response to a pre-selection.
TPDU	Transport Protocol Data Unit
TSDU	Transport Protocol Service Unit
TRA	COM field value: Acknowledgement of point to point transfer
TRB	COM field value: Broadcast remote transfer frame not acknowledged
TRF	COM field value: Point to point remote transfer exchange
T1	Time out to wait for a response according to a request
XBA	COM field value: Response to a change of speed request
XBR	COM field value: Change of speed request
ZA1	Field reserved for bidirectional programming authentication
ZA2	Field reserved for bidirectional programming authentication

4 General description

4.1 Basic vocabulary

All communication calls upon two systems called Primary Station and Secondary Station. The Primary Station is the system that decides to initialize a communication with a remote system called Secondary Station; these designations remain valid throughout the duration of the communication.

A communication is broken down into a certain number of transactions. Each transaction consists of a transmission from the Transmitter to the Receiver. During the sequence of transactions, the Primary Station and Secondary Station systems take turns to act as Transmitter and Receiver.

For the local bus data exchange profile with DLMS or DLMS/COSEM, the terms Client and Server have the same meaning as for the DLMS model (refer to IEC 61334-4-41 or IEC 62056-5-3 Ed. 1.0:—). The Server (which is a Secondary Station) receives and processes all submissions of specific service requests. The Client (which is a Primary Station) is the system that uses the Server for a specific purpose by means of one or more service requests.

4.2 Profiles, layers and protocols

4.2.1 Overview

This standard specifies three profiles as shown in Figure 1.

- the base profile (without DLMS), see 4.2.2;
- the profile with DLMS, see 4.2.3;
- the profile with DLMS/COSEM; see 4.2.4.

The physical layer in the three profiles is the same except that in the DLMS/COSEM profile speed negotiation is available. This common physical layer allows stations using different profiles to be installed on the same bus.

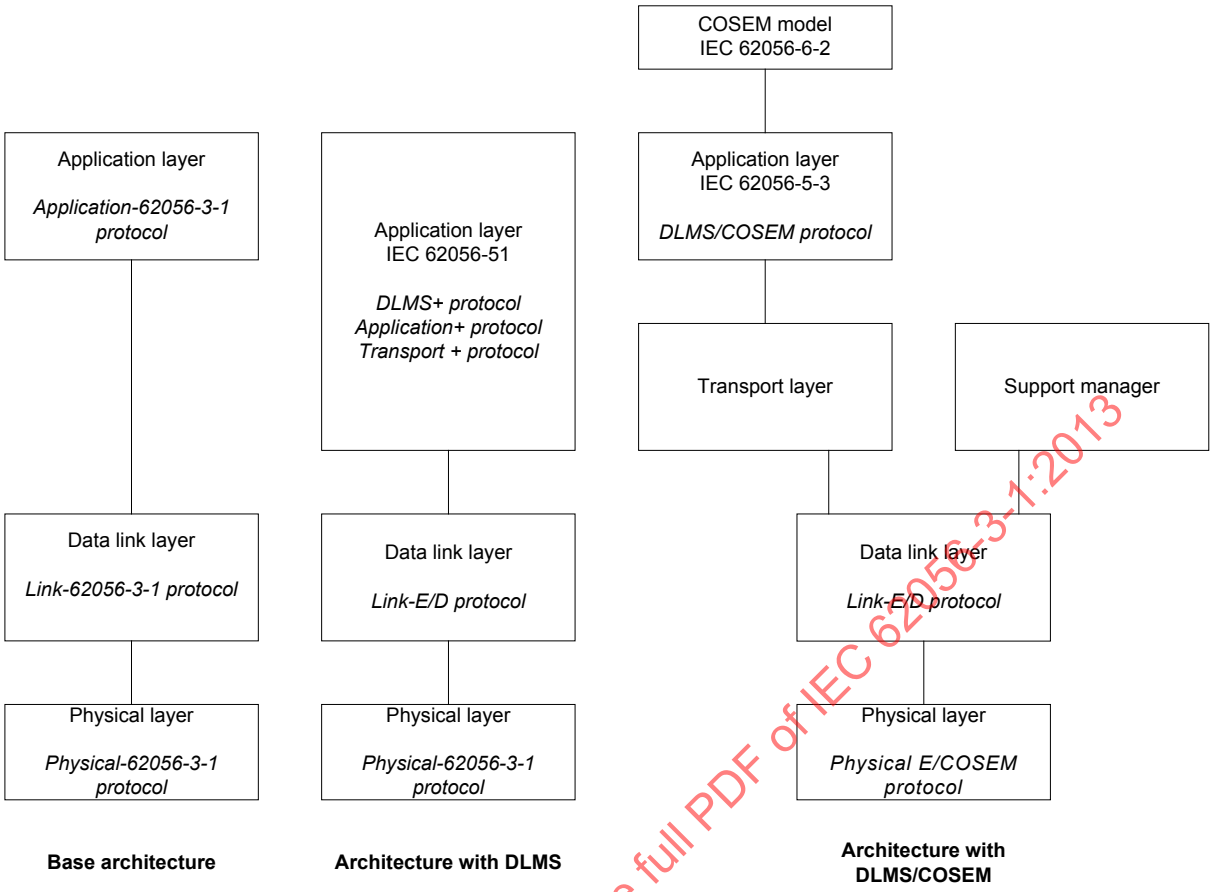


Figure 1 – IEC 62056-3-1 communication profiles

4.2.2 Base profile (without DLMS)

The base profile (without DLMS) uses three protocol layers:

- the physical layer with the *Physical-62056-3-1* protocol specified in 5.1;
- the data link layer with the *Link-62056-3-1* protocol, specified in 5.2, and
- the application layer with the *Application-62056-3-1* protocol specified in 5.3.

This profile allows remote reading, remote programming, point-to-point remote transfer – which is a simplified remote programming service – broadcast remote transfer, remote supply of secondary stations, detecting forgotten stations and alarm functions. The related communication services are specified in 4.4.

4.2.3 Profile with DLMS

The profile with DLMS uses three protocol layers:

- the same physical layer as the base profile, specified in 5.1;
- the data link layer using the *Link-E/D* protocol, specified in 6.2; and
- the application layer specified in IEC 62056-51, using the *Transport+, Application+* and *DLMS+* protocols, see 6.3.

This profile also allows using DLMS as specified in IEC 61334-4-41. The related communication services are specified in 4.5.

4.2.4 Profile with DLMS/COSEM

The profile with DLMS/COSEM uses four protocol layers:

- the physical layer, similar to the one used in the base profile and the profile with DLMS, specified in 5.1, but with speed negotiation, see 7.2;
- the data link layer using the *Link-E/D* protocol. This is the same as the data link layer of the profile with DLMS, except that it interfaces with the support manager layer and the transport layer. See 7.3;
- the support manager layer supports some specific process for the management of the bus, see 0;
- the transport layer provides segmentation and reassembly of APDUs, see 0;
- the application layer as specified in IEC 62056-5-3 Ed. 1.0:—taking into account some restrictions of the Euridis bus, see 0.

The profile with DLMS/COSEM allows using the COSEM object model and the DLMS services accessing the COSEM objects over the Euridis bus.

4.3 Specification language

In this standard, the protocol of each layer is described by state transitions represented in the form of tables. The syntax used in making up these tables is defined by a specification language described in Annex A.

In the event of a difference in interpretation between part of the text and a state transition table, the table is always taken as the reference.

4.4 Communication services for local bus data exchange without DLMS

4.4.1 Overview

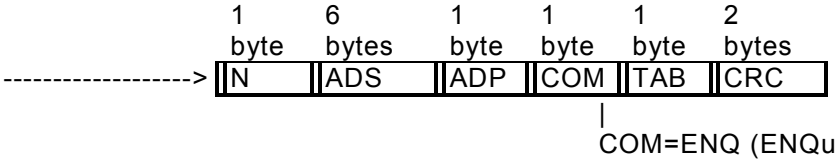
The list of available services at the Application level layer is:

a) remote reading of data, see 0;
b) remote programming of data, see 4.4.3;
c) point to point remote transfer, which is a simplified remote programming service, see 4.4.4;
d) broadcast remote transfer, 4.4.5;
e) bus initialization, 4.4.6;
f) forgotten station call, 4.4.7.

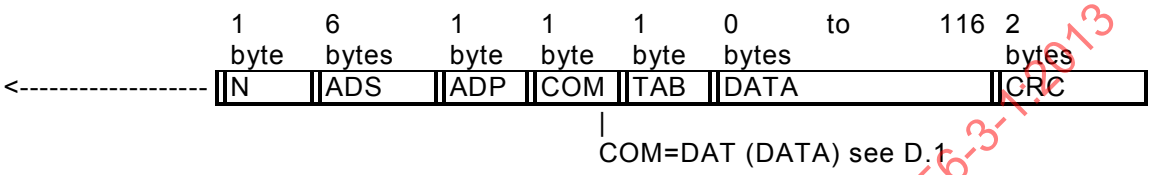
4.4.2 Remote reading exchange

The ENQ exchange consists of two frames arranged in one sequence:

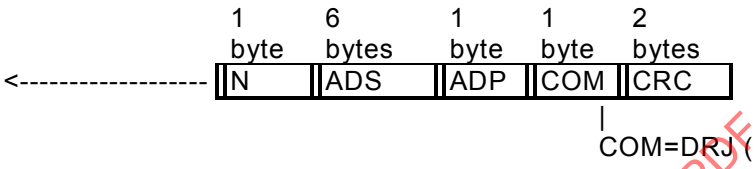
- remote reading frame containing the type of data to select in the TAB field



- positive acknowledgement frame with the selected data in the DATA field



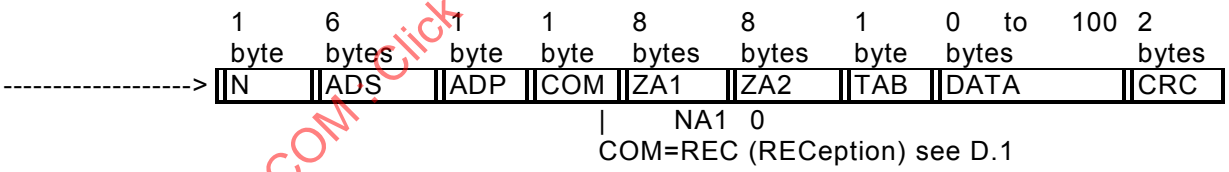
- negative acknowledgement frame (TAB identifier unknown)



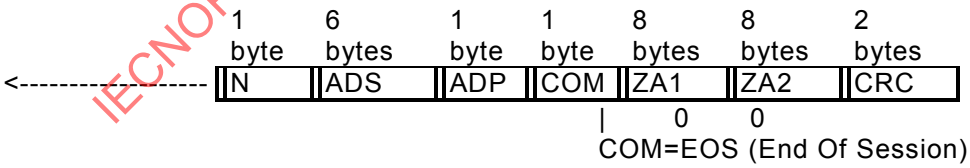
4.4.3 Remote programming exchange

The REC exchange consists of four frames arranged in two sequences. Since there is an internal sequence for authentication purpose, from the application point of view, it seems to be only one sequence with two frames:

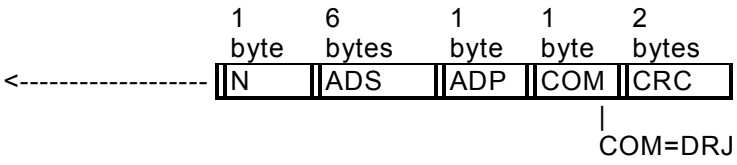
- remote programming frame containing data in the DATA field and their type in the TAB field



- positive acknowledgement frame (no authentication trouble)



- negative acknowledgement frame (no authentication trouble but remote programming data not validated)



Authentication is carried out by an exchange of random numbers ciphered using a secret key specific to each Secondary Station. The random numbers are defined in 8 bytes and they are ciphered with the DES algorithm using an 8-byte ciphering key Ki known both to the Primary and the Secondary station.

A random number NA1 is first generated by the Primary Station and transmitted into the ZA1 field of the remote programming frame while field ZA2 is set to zero.

On arrival at the Secondary Station, field ZA1 is ciphered by the DES algorithm with key Ki to get the ciphered random number NA1K. Then occurs the internal sequence for authentication which consists of two frames.

The first frame (from Secondary to Primary Station) contains this random number NA1K in field ZA1 and a random number NA2 generated by the Secondary station in field ZA2.

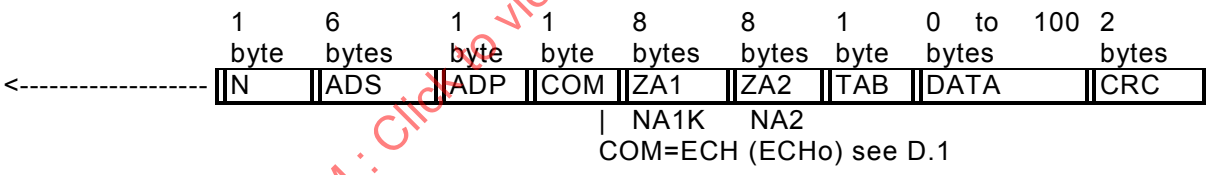
On reception of this frame, the Primary Station compares the ZA1 field to an NA1´ number obtained by ciphering the transmitted NA1 number using the DES algorithm with key Ki. If NA1´ = ZA1, then the Primary Station considers the called Secondary Station as authenticated. Otherwise, it considers the Secondary Station has not been authenticated and aborts the communication session.

After correct authentication of the Secondary Station, the Primary Station first ciphers the random number NA2 by the DES algorithm with key Ki to get the ciphered random number NA2K and then transmits it into field ZA2 while field ZA1 is set to zero.

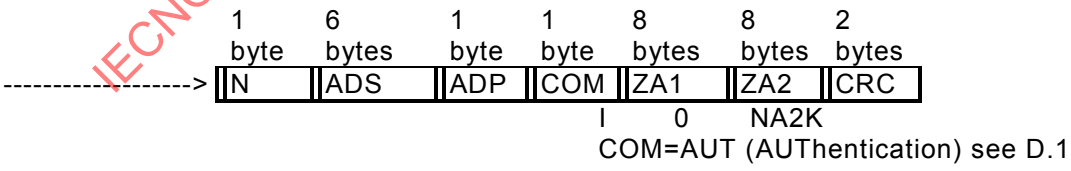
On reception of this response frame, the Secondary Station compares the ZA2 field to an NA2´ number obtained by ciphering the transmitted NA2 number using the DES algorithm with key Ki. If NA2´ = ZA2, then the Secondary Station considers the Primary Station as authenticated. Otherwise, it considers the Primary Station has not been authenticated and sends a negative acknowledgment frame.

The internal authentication exchange is the following:

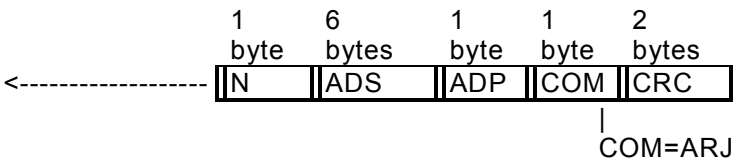
- internal authentication frame containing the ciphered random number NA1K in field ZA1 and the random number NA2 in field ZA2



- positive response frame containing the ciphered random number NA2K in field ZA2 (if the Secondary Station is considered as authenticated)



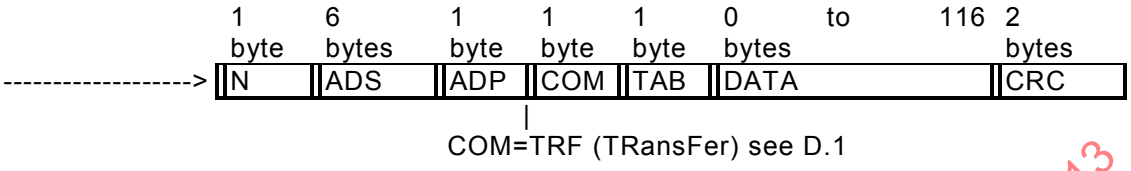
- an authentication rejection frame replaces the normal EOS or DRJ frame when the Primary Station is not considered authenticated



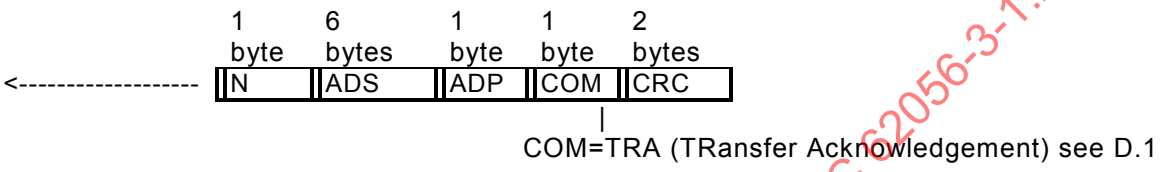
4.4.4 Point to point remote transfer exchange

This TRF exchange consists of two frames arranged in one sequence. From the application point of view, it seems to be a remote programming exchange in a single sequence with no authentication:

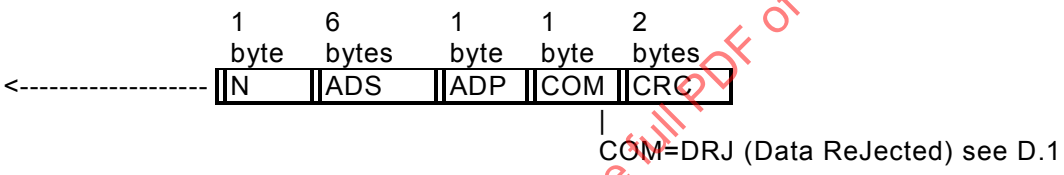
- point to point remote transfer frame containing data in the DATA field and their type in the TAB field



- positive acknowledgement frame



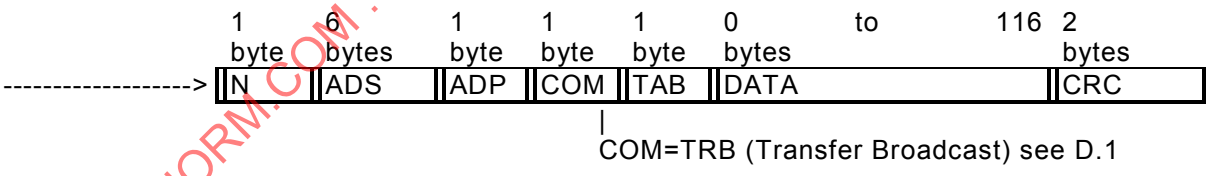
- negative acknowledgement frame (remote transfer data not validated)



4.4.5 Broadcast remote transfer frame

This TRB frame does not involve any frame answer. From the application point of view, it seems to be a point to point remote transfer, but without acknowledgement since it is a broadcast

- broadcast remote transfer frame containing data in the DATA field and their type in the TAB field

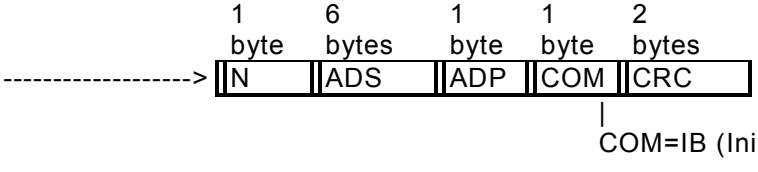


The secondary address (which defines the receiving Secondary Stations) shall be a broadcast address.

4.4.6 Bus initialization frame

This IB frame does not involve any frame answer. From the application point of view, it seems to be a broadcast remote transfer, but without data since its purpose is only to reset a special flag (called forgotten station flag) to TRUE for all Secondary Stations that have been programmed with the ADP address:

- bus initialization frame



The secondary address (which defines the receiving Secondary Stations) shall be a broadcast address.

After the bus initialization frame, any Secondary Station receiving a correct ENQ frame containing a known TAB identifier will then no longer be considered as a “forgotten station”.

4.4.7 Forgotten station call exchange

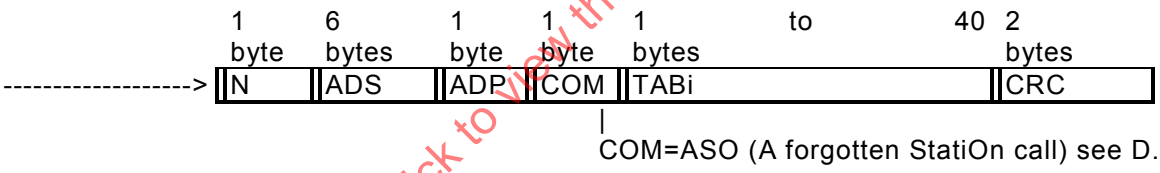
This ASO exchange consists of two frames arranged in one sequence. At the end of a remote reading sequence, the Primary Station can search for stations whose forgotten station flag is TRUE (maximum 5 in 100).

As a correct remote reading exchange sets the forgotten station flag of the corresponding station to FALSE, the ASO exchange normally occurs after the completion of a remote reading sequence that is one or several remote reading exchanges preceded by a bus initialization frame.

The Primary Station manages several time slots. When detecting a collision, it has to retry an ASO exchange. Nevertheless, each time a correct Secondary Station answer is received, the Primary Station shall eliminate it from the list of forgotten stations by operating a correct remote reading exchange with this station.

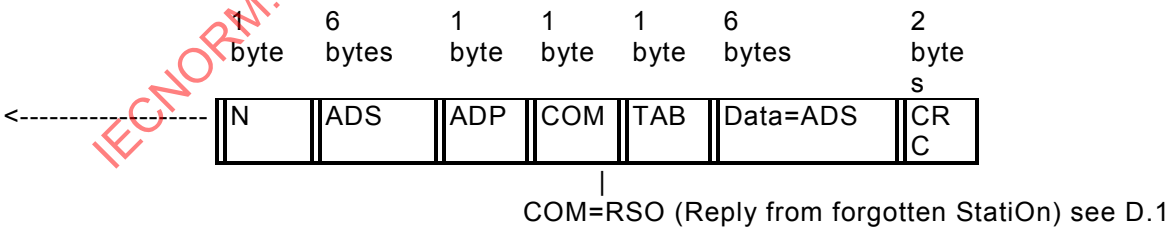
In order to ensure the selection constraints (described in 4.4.9), the non-energized stations shall answer in the first time slot of the first ASO exchange. Then, only the forgotten stations are selected and the usual principle can be used for the following ASO exchanges.

- forgotten station call frame containing selection criteria in the TABi field (1 to 40 TAB identifiers)



The secondary address (which defines the receiving Secondary Stations) should be a broadcast address.

- acknowledgement frame containing the first TAB recognized by the unit and the ADS of the station



- The data field containing the ADS of the secondary station responds to the call to forgotten stations.

4.4.8 Frame fields

N total number of bytes in the frame, including N.

ADS absolute physical address of the Secondary Station coded as a 48-bit string. There is only one broadcast physical address which is the general broadcast ADG coded as "000000000000"in hexadecimal [1].

The ADS also corresponds exactly to the System Title of the Secondary Station.

ADP physical address of the Primary Station coded as an 8-bit string. The value "00"H is reserved for the coding of the physical address of the general primary APG [2]. Any Secondary Station solicited by a Primary Station whose physical address is APG, replies with the first primary physical address with which it has been programmed

COM command code depending on the exchange and the frame direction (see Annex D)

ZA1, ZA2 fields reserved for authentication operated during the remote programming exchange.

TAB type of data selected associated with some command codes (ENQ, DAT, REC, TRF, TRB or RSO). The value "00"H is reserved for systems management, the value "FF"H for alarm management.

DATA information packet from the host application. This field can be eventually empty depending on the command code.

CRC Cyclic Redundancy Check field corresponds to the 16 redundant bits of the CRC whose principle is described in Annex E.

The frame fields are transmitted in an ascending order (from N to CRC). When a field contains data over several bytes, the transmission begins with the least significant byte and ends with the most significant one. However, the DATA field is considered as a byte string and is transmitted in an ascending order.

4.4.9 Principle of the energy remote supply

The general principle of the data exchanges is preserved for the non-energized stations. The notion of energy remote supply is only added for communication between a Primary Station and one or more Secondary Stations.

To begin a communication session, the Primary Station shall send a “Wakeup Call” designed to alert the communications system of every Secondary Station connected to the bus. This call is a continuous carrier for a nominal time depending on the energy remote supply mechanism:

- the “Wakeup Call” signal duration is AGT to wake up non-energized stations;
- the “Wakeup Call” signal duration is AGN to wake up energized stations.

Remark: A Secondary Station can be configured in Alarm mode. It is then remote supplied continuously and so can transmit the alarm to the Primary Station (see 4.4.11).

Then, whatever type of remote station is selected (energized or not), an intermediate AGN “Wakeup Call” signal shall also be required at the Primary Station side in the following circumstances:

- before the first ENQ or TRF exchange;
- before the sixth consecutive and successful ENQ or TRF exchange with the same Secondary Station;

---

[1] Other broadcast addresses could be defined depending on the naming rules adopted in companion standards for the semantics of the System Titles which are often based on a manufacturer code, a manufacture year and an equipment type.

[2] Other general addresses could be defined depending on the naming rules adopted in companion standards for the semantics of operator identifiers which are often based on a utility code.

- before the first ENQ or TRF exchange with a different Secondary Station to the one previously selected in the preceding ENQ or TRF exchange;
- before any REC exchange;
- before any TRB frame;
- before any IB frame;
- before any ASO exchange.

For non-energized stations, it means that the Primary Station can avoid to wake up all the remote stations when not necessary, and then save its energy.

A Primary Station can use a specific modem ensuring both the energy remote supply as well as the modulation and demodulation functions. The communication time and the number of non-energized stations shall be optimized in order to save the battery of the Primary Station.

As another possibility, the Primary Station might only focus on the modulation and demodulation functions. In this case, an auxiliary station continuously supplies the bus with energy.

A Secondary Station generally contains only one logical application referenced by its ADS. Such a station may or may not be energized.

A multiple Secondary Station (containing several logical applications corresponding to several ADSs) should be a non-energized station. This feature is described more fully in Clause 8.

4.4.10 Non-energized station preselection exchange

To optimize the bus consumption, a preselection exchange enables the Primary Station to select a non-energized Secondary Station.

The preselection exchange occurs after an AGT “Wakeup Call” signal addressed to all non-energized stations of the bus. To limit the bus consumption, the first frame sent by the Primary Station should be short enough and the addressed Secondary Station should answer before the triggering of the TOPRE wakeup. Not seeing an answer in time, the modem of the Secondary Station goes back in a low consumption state.

During the preselection exchange, all the non-energized stations consume energy. The bus voltage and the energy storage capacitors decrease until the non-selected stations goes back in a low consumption state. Then the continuously sent energy charges the energy storage capacitors and the bus voltage increases.

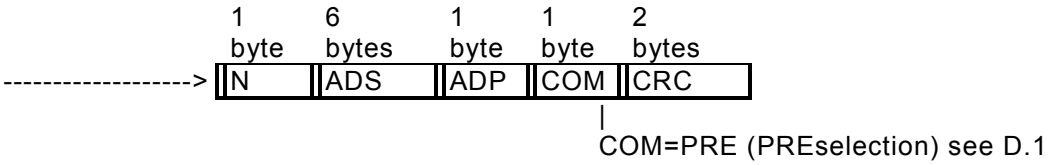
The modem of the Primary Station should store sufficient energy before the first preselection. This step is guaranteed by a wait-time controlled thanks to the TICB wakeup. At the end of a preselection, the energy storage capacitors are empty and the Primary Station shall wait for the bus voltage increase before a second preselection.

As the preselection frame shall not be more than 18 bytes long, it can be

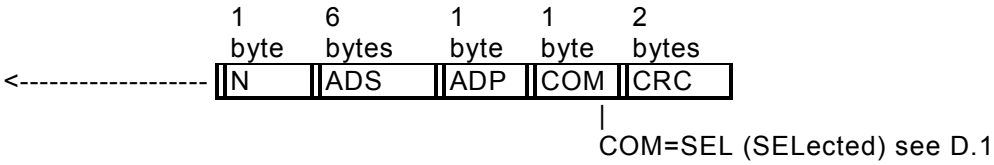
- an ENQ frame;
- a TRB or TRF frame, if and only the data field is less than or equal to 6 bytes long;
- an IB frame;
- an ASO frame, if and only the number of TABi fields is less than or equal to 7.

As the first frame of REC and TRF exchanges may be too long, an additional service is provided for preselection. This fully transparent preselection exchange consists of two frames arranged in one sequence.

- non-energized station preselection frame



- acknowledgement frame



To save the energy of the Primary Station, there is no retry during a preselection exchange. If an addressed non-energized station does not answer correctly, it is not selected and the Primary Station shall send a new AGT “Wakeup Call” signal.

4.4.11 Communication exchange after preselection

After preselection, the modem of a non-energized station can stay awake for the continuing communication and delays are not critical since the number of connected devices is limited. The Primary Station supplies the selected station and charges the capacitive reservoirs of the non-selected stations.

The normal end of the communication session occurs differently depending on the energy remote supply mechanism:

- after a short period of inactivity during the communication session when no intermediate AGN “Wakeup Call” signal is required for energized stations. This period is checked by the wakeup TOL;
- after a longer period of inactivity during the communication session for non-energized stations. This period is checked by the wakeup TOAG.

Note that for a non-energized station, as far as there is no timeout of TOAG wakeup, an intermediate AGN “Wakeup Call” signal is enough to go on the current communication session.

4.4.12 Alarm function

A device integrated in a simple or multiple Secondary Station (see 8.2.3) can transmit alarms to the primary station, providing it can integrate functions of interface as described hereinafter.

An alarm shall be fetched from the Secondary Station in 10 s maximum.

A programmable configuration on the Interface and one on each device selects the status of Alarm mode: Active or Inactive.

When Alarm mode is active, the device can generate an alarm, inside the secondary station. The Alarm function is effective only if the supply is present and permanent on the bus.

The device sends the alarm during TASB. TASB is long enough to force an “0” state on the secondary bus and to be detected by the Interface, even if a communication is in progress.

Alarm mechanism is described in Figure 2.

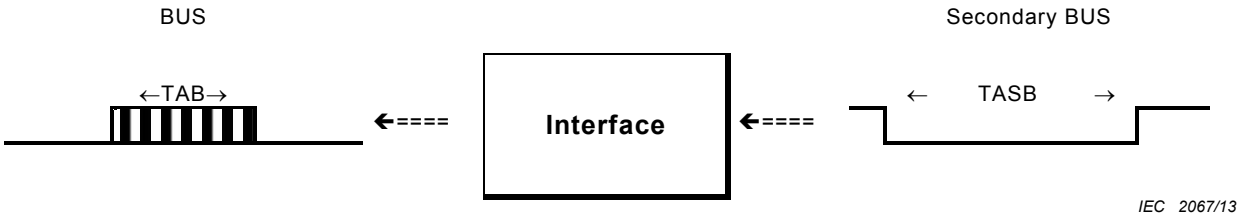


Figure 2 – Alarm mechanism

The alarm is not directly transmitted towards the primary station. The interface receives the alarm and transmits it by sending a “0” (50 kHz carrier) during TAB on the bus when it is possible:

a) No communication on the bus.

When the interface receives the alarm on the secondary bus, it transmits it on the bus.

b) On synchronization of AGN or AGT when a communication is in progress. When a communication is in progress on the bus, the Interface memorises the alarm received. It transmits it to the bus after one of the following events:

- TOALR after the end of AGN or AGT reception;
- when the normal end of the communication session occurs.

In this way, the interface can filter the alarm to avoid conflict on the bus.

After the alarm generation, the Secondary Station will be considered as a “forgotten station” with a selection criterion equal to FF.

The primary station configured in Alarm mode listens to the bus when there is no communication on the bus and after transmission of an AGN or AGT in order to detect an alarm. When the primary station receives an alarm, it enters in Forgotten Station Call procedure with a selection criteria in the TABi field equal to FF (see 4.4.7).

Timing diagrams explain Alarm management in 0.

4.5 Communication services for local bus data exchange with DLMS

DLMS does not offer services to operate the bus initialization and forgotten station call mechanisms. Nevertheless, the IB frame and the ASO exchange are supported and managed as they are with the local bus data exchange profile without DLMS except that the forgotten station flag is considered as a global variable shared with the Application Programming Interface.

Remote reading of data and point to point remote transfer are directly foreseen by DLMS. But the (redundant) remote programming of data is not supported since authentication is reserved for the *Application* layer.

As data semantics is managed by DLMS, the frame format is very simple and only unmarked frames are required. To ensure compatibility with the profile without DLMS, this frame format is defined by the following nine fields:

1 byte	6 bytes	1 byte	3 bits	1 bit	2 bits	2 bits	0 to 117 bytes	2 bytes
Size	ADS	ADP	DATA+	Priority	Send	Confirm	Text	CRC

Size total number of bytes in the frame, including Size. If its value is not 11, the Receiver knows that the frame contains data in the Text field

ADS same rules as for local bus profile without DLMS

ADP same rules as for local bus profile without DLMS

DATA+ always coded "111"B

Priority transmission priority level of the current frame. The *Application* layer sets this priority according to the requested service

Send number of the last frame sent

Confirm number of the last frame received without error

Text DSDU (Data link Service Data Unit) from the higher level. A frame does not necessarily contain text. If data from the *Application* layer is available when the frame is sent, then the *Text* field will contain data, otherwise it will be empty. This mechanism provides the conditions for balanced bi-directional data transmission. In order not to confuse DATA+ frame with frames from the profile without DLMS, the DATA+, Priority, Send and Confirm fields make up a special command code COM whose values are different from the already reserved COM values (see Annex D)

CRC same rules as for local bus profile without DLMS

The frame fields are transmitted in ascending order (from Size to CRC). When a field is coded on several bytes, the transmission begins with the least significant byte and ends with the most significant one. However, the Text field is considered as a byte string and transmitted in ascending order.

4.6 Systems management

The purpose of Systems management is to allow an enrolment. This enrolment includes an identification of Secondary Stations on a bus. The Discover service is provided for this purpose.

The enrolment consists of a sequence of Discover requests issued by the active initiator located inside the Primary Station. Each Discover service is provided to inform the remaining new stations that they will have a chance to respond in the next time slots.

A Discover request conveys a specific response-probability argument as an integer in the range [0, 100]. It expresses the probability, in per cent, that a new station responds. When it is set to 100, all the new stations on the bus shall respond.

On reception of a Discover indication, each Secondary Station tests the value of its flag Discovered. If it is set to TRUE, the indication is discarded; otherwise it draws a random number between 1 and 100. If this number is smaller than or equal to the response-probability argument, the new station will issue a Discover response and set its flag Discovered to TRUE.

The flag Discovered is always reset on a receiving of an IB frame.

To ensure a maximum compatibility (for stations including DLMS/COSEM, DLMS or otherwise), it is proposed to implement the systems management as indicated in Annex H.

5 Local bus data exchange without DLMS

5.1 Physical layer

5.1.1 Physical-62056-3-1 protocol

The *Physical-62056-3-1* protocol of the *Physical* layer of the local bus data exchange profile without DLMS behaves asymmetrically. The state machine of the Primary Station is therefore different from that of the Secondary Station.

The *Physical-62056-3-1* protocol supports the Secondary Stations whether or not they are energized. As already stated in the general description, the remote stations are woken up either by an AGN or an AGT "Wakeup Call" signal and a communication session ends after expiry of TOL or TOAG wakeup.

After a "Wakeup Call" signal, a communication session is then made asynchronously and by half-duplex at 1 200 bits/s on the bus.

5.1.2 Physical parameters

The value of the maximum size of a frame being received, MaxIndex, is set to 128.

The value of the maximum number of RSO time slots for the processing of a "Forgotten Stations Call", MaxRSO, is set to 3.

The AGN duration of an AGN "Wakeup Call" signal shall be in the range [50, 150[ms, while the AGT duration of an AGT "Wakeup Call" signal shall be in the range [200, 300[ms.

Timing type and characteristics are described in Annex B.

The values of Table 1 are defined for a Primary Station.

Table 1 – Primary Station timing

	Min. ms	Nominal ms	Max. ms	Type, see Clause B.1	Definition
TA10	–	–	120	TSL1	Maximum waiting period of the first byte of a frame being received
TAB		100		TC	Duration of an alarm signal on the bus
TAGN	–	100	–	TPDF	Duration of an AGN "Wakeup Call" signal
TAO	–	–	40	TC	Maximum waiting period of one byte of a frame being received whose expiry indicates the end of a frame
TARSO	–	500	–	TC	Length of an RSO time slot
TASB		1 200		TC	Waiting period after the beginning of an alarm signal
TEMPO	–	40	–	TC	Safety delay at the end of transmission of a Wake up signal or a frame
TOE	–	–	2 500	TL	Safety delay for transmitting to protect against defective hardware
TOL	–	–	100	TSL2	Maximum waiting time for a request coming from the upper layer
T1	–	10 000	–	TL	Maximum waiting time for a response from the secondary station
Non-energized station specific (Supply)					
TAGT	–	250	–	TPDF	Duration of an AGT "Wakeup Call" signal
TICB	8 000	–	–	Ta	Initial bus charge period
TOAG	–	–	3 000	TPFD	Maximum delay for a selected non-energized station to recognize an AGN "Wakeup Call" signal

The values of Table 2 are defined for a Secondary Station.

Table 2 – Secondary Station timing

	Min. ms	Nominal ms	Max. ms	Type[a]	Definition
TA1O	30[b]	–	160	TSL1	Maximum waiting period of the first byte of a frame being received
TAB		100	–	TC	Duration of an alarm signal on the bus
TAGN	50	100	150	TPDF	Duration of an AGN “Wakeup Call” signal
TAO	–	–	40	TC	Maximum waiting period of one byte of a frame being received whose expiry indicates the end of a frame
TARSO	–	–	500	TC	Length of an RSO time slot
TOALR	20	–	–	TL	Waiting to send an AGN after an AGN or AGT reception
TOE	–	–	2 500	TL	Safety delay for transmitting to protect against defective hardware
TOL	–	–	100	TSL2	Maximum waiting time for a request coming from the upper layer
Non-energized station specific (Supply)					
TAGT	200	250	300	TPDF	Duration of an AGT “Wakeup Call” signal
TASB	–	1 200	–	TL	Duration of a secondary-bus alarm signal
TICB	8 000	–	–	Ta	Initial bus charge period
TOAG	–	–	3 000	TPFD	Maximum delay for a selected non-energized station to recognize an AGN “Wakeup Call” signal
TOAGN	–	–	300	Tc	Maximum delay of inactivity to recognize the end of a communication session with an energized station
TOAPPEL	–	–	180	TPFD	Maximum waiting period of the first byte of a preselection frame being received
TOBAVARD	–	–	260	TPDF	Safety delay to protect against defective length of preselection frame
TOPRE	–	–	130	TPFD	Maximum waiting for a preselection answer
TOSEUIL	–	150	–	TC	Duration of a “Wakeup Call” signal which wakes up a non-energized station
TVASB	40	–	–	TL	Minimum duration of an alarm signal on the secondary bus

[a] For the definition of different timing types, see Clause B.1.

[b] After a “Wakeup Call”, a minimum duration of 30 ms is necessary.

5.1.3 Timing diagrams

Figures 3, 4 and 5 can be used to show different types of session of the protocol for non-energized secondary stations.

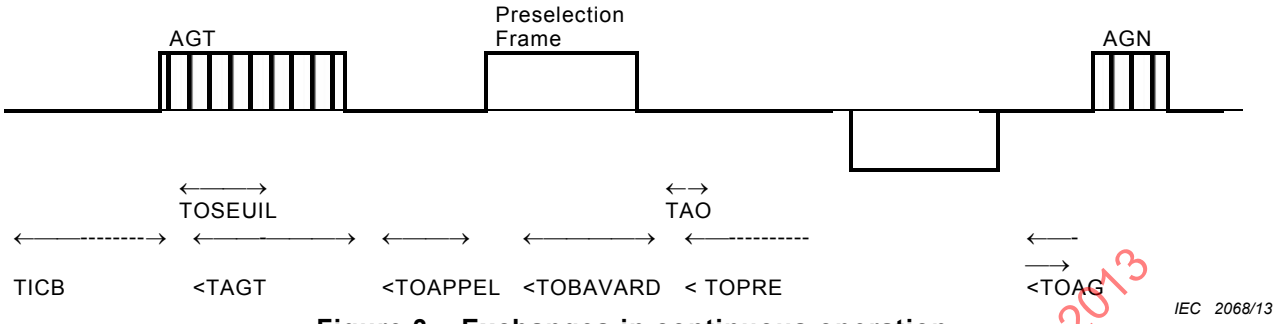


Figure 3 – Exchanges in continuous operation

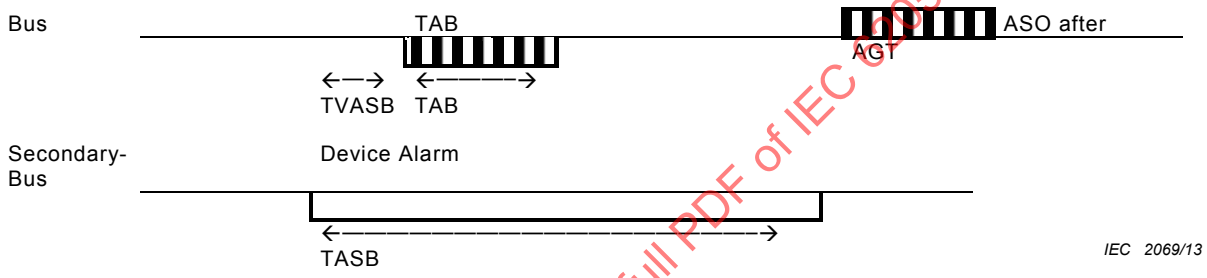


Figure 4 – Alarm event without any communication in progress

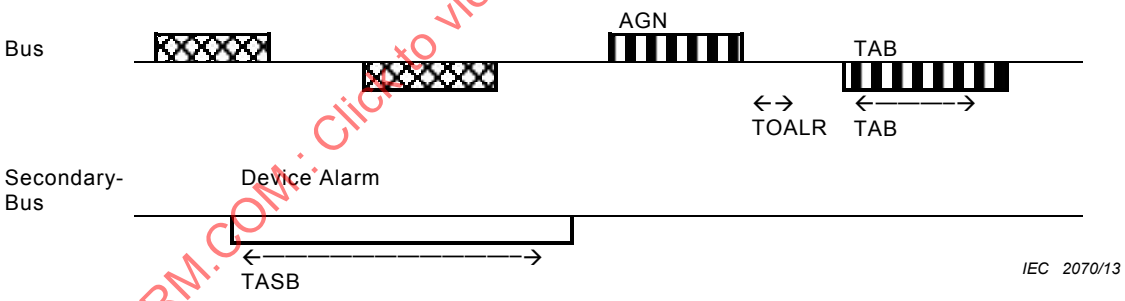


Figure 5 – Alarm event with a communication in progress

5.1.4 Physical services and service primitives

The user of the *Physical-62056-3-1* protocol can use the services and service primitives given in Table 3.

Table 3 – Physical services and service primitives

Service	Service primitive
Phy_DATA	Phy_DATA.req(Frame) Phy_DATA.ind(Frame)
Phy_UNACK	Phy_UNACK.req(Frame)
Phy_APPG	Phy_APPG.req(TypeAG) Phy_APPG.ind()
Phy_ASO	Phy_ASO.req(Frame) Phy_ASO.ind(Frame)
Phy_RSO	Phy_RSO.req(Frame, Window)
Phy_COLL	Phy_COLL.ind()
Phy_ALARM	Phy_ALARM.req() Phy_ALARM.ind()
Phy_ABORT	Phy_ABORT.req() Phy_ABORT.ind(ErrorNb)

The role assigned to each primitive is as follows:

- Phy_DATA.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a frame Frame;
- Phy_DATA.ind(Frame) enables the *Physical* layer to inform the *Data Link* layer that a frame Frame is available;
- Phy_UNACK.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a frame Frame without waiting for acknowledgement;
- Phy_APPG.req(TypeAG) enables the *Data Link* layer to request the *Physical* layer to transmit a “Wakeup Call” signal. The duration TypeAG of this signal is either AGN or AGT;
- Phy_APPG.ind() enables the *Physical* layer to inform the *Data Link* layer of the end of the transmission of a “Wakeup Call” signal;
- Phy_ASO.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a “Forgotten Stations Call” frame;
- Phy_ASO.ind(Frame) enables the *Physical* layer to inform the *Data Link* layer that a frame Frame has been received in one of the time slots of the forgotten stations;
- Phy_RSO.req(Frame, Window) enables the *Data Link* layer to request the *Physical* layer to transmit a Forgotten Stations Call frame Frame in the time slot number Window;
- Phy_COLL.ind() enables the *Physical* layer to inform the *Data Link* layer that a collision has been detected in one of the time slots of the forgotten stations;
- Phy_ALARM.req() enables the *Data Link* layer to request the *Physical* layer to transmit an Alarm;
- Phy_ALARM.ind() enables the *Physical* layer to inform the *Data Link* layer of the arrival of an alarm;
- Phy_ABORT.req() enables the *Data Link* layer to request the *Physical* layer to end its activity;
- Phy_ABORT.ind(ErrorNb) enables the *Physical* layer to inform the *Data Link* layer of the occurrence of a fatal error identified by the number ErrorNb.

5.1.5 State transitions

Table 4 – *Physical-62056-3-1* state transitions: Primary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	$true()	MaxRSO=3 MaxIndex=128 Collision=FALSE SessionAGT=FALSE wait_time(TICB)	Stopped
Stopped	Phy_APPG.req(AG) & AG=AGN	stop_timer(TOAG) FlagAbort=FALSE TypeAG=AGN send_AG(TypeAG)	W.AG
Stopped	Phy_APPG.req(AG) & AG=AGT	SessionAGT=TRUE FlagAbort=FALSE TypeAG=AGT send_AG(TypeAG)	W.AG
Stopped	time_out(TOAG)	Phy_ABORT.ind(EP-2) SessionAGT=FALSE	Stopped
Stopped	Phy_ABORT.req()	$none()	Stopped
Stopped	data-carrier_on	init_timer(TAB) init_timer (TASB)	W.ETABS
W.ETABS	data_carrier_off	stop_timer(TASB) stop_timer(TAB)	Stopped
W.ETABS	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind()	W.TASB
W.AG	AG_sent_event	Phy_APPG.ind() init_timer(TEMPO)	W.TAB
W.AG	Phy_ABORT.req()	FlagAbort=TRUE	W.AG
W.TAB	data-carrier_on	Carrier = TRUEinit_timer(TAB) init_timer (TASB)	W.TAB
W.TAB	data-carrier_off	Carrier = FALSE stop_timer(TAB) stop_timer (TASB)	W.TAB
W.TAB	time_out(TEMPO) & not(FlagAbort) & not(Carrier)	init_timer(TOL)	M.Send
W.TAB	time_out(TEMPO) & FlagAbort & not(Carrier)	wait_time (TOL)	*T.Session*
W.TAB	time_out(TEMPO) & Carrier	init_timer(TOL)	W.ETAB
W.TAB	Phy_ABORT.req()	FlagAbort=TRUE	W.TAB
W.ETAB	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind() stop_timer(TOL)	W.TASB
W.ETAB	data_carrier_off & not(FlagAbort)	stop_timer(TAB) stop_timer (TASB)	M.Send
W.ETAB	data_carrier_off & FlagAbort	stop_timer(TAB) stop_timer (TASB)	W.TOL
W.ETAB	Phy_ABORT.req()	FlagAbort=TRUE	W.ETAB
W.TASB	time_out(TASB)	$none()	Stopped
W.TOL	time_out(TOL)	$none()	*T.Session*
M.Send	Phy_DATA.req(Frame)	Service=NORMAL	*SendFirst*
M.Send	Phy_UNACK.req(Frame)	Service=UNACKNOWLEDGED	*SendFirst*
M.Send	Phy_ASO.req(Frame)	Service=ASO	*SendFirst*
M.Send	Phy_ABORT.req()	$none()	M.Send
M.Send	time_out(TOL)	$none()	*T.Session*

Initial state	Triggering condition	Set of actions	Final state
T.Session	SessionAGT=TRUE	init_timer(TOAG) Phy_ABORT.ind(EP-1) wait_time(TEMPO)	Stopped
T.Session	SessionAGT=FALSE	Phy_ABORT.ind(EP-1) wait_time(TEMP0)	Stopped
SendFirst	$true()	stop_timer(TOL) Size=size(Frame) Index=1 send_octet(Frame, Index) Size=Size-1 init_timer(TOE)	Sending
Sending	octet_sent_event & Size>0	Index=Index+1 send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0	stop_timer(TOE) wait_time(TAO) Index=1 Frame=""	*Answer*
Sending	Phy_ABORT.req()	stop_timer(TOE) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Answer	Service=NORMAL I Service=UNACKNOWLEDGED	init_timer(TA1O)	M.Rec
Answer	Service=ASO	WinRSO=1 init_timer(TARSO) init_timer(TA1O)	M.Rec
M.Rec	octet_received_event	stop_timer(TA1O) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
M.Rec	collision_detected_event	stop_timer(TA1O) Collision=TRUE init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	$none()	*Received*
M.Rec	Phy_ABORT.req()	FlagAbort=TRUE	M.Rec
Receiving	octet_received_event & Index<=MaxIndex	stop_timer(TAO) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Phy_ABORT.ind(EP-4F) wait_time(TAO) FlagAbort=TRUE	*Received*
Receiving	collision_detected_event	stop_timer(TAO) Collision=TRUE init_timer(TAO)	Receiving
Receiving	time_out(TAO)	$none()	*Received*
Receiving	time_out(TARSO)	Phy_ABORT.ind(EP-5F) wait_time(TAO) FlagAbort=TRUE	*Received*
Receiving	Phy_ABORT.req()	Flagabort = TRUE	Receiving
Received	Service=NORMAL & not(Flagabort)	Phy_DATA.ind(Frame) init_timer(TOL)	M.Send

Initial state	Triggering condition	Set of actions	Final state
Received	(Service=NORMAL & Flagabort) | Service=UNACKNOWLEDGED	wait_time(TOL)	*T.Session*
Received	Service=ASO & Collision & not(Flagabort)	Phy_COLL.ind() Collision=FALSE	*T.RSO*
Received	Service=ASO & not(Collision) & not(Flagabort)	Phy_ASO.ind(Frame)	*T.RSO*
Received	Service=ASO & Flagabort	$none()	*T.RSO*
T.RSO	(TypeAG=AGT) I (WinRSO>=MaxRSO) & (TypeAG=AGN)	stop_timer(TARSO)	*T.Session*
T.RSO	(WinRSO<MaxRSO) & (TypeAG=AGN)	Index=1 Frame=""	W.RSO
W.RSO	time_out(TARSO)	WinRSO=WinRSO+1 init_timer(TARSO) init_timer(TA1O)	M.Rec
W.RSO	Phy_ABORT.req()	Flagabort=TRUE	W.RSO

Table 5 – Power supply management state transitions (only for non-energized Secondary Station)

Initial state	Triggering condition	Set of actions	Final state
Initial	alarm_detection()	Flagalarm = TRUE FlagSendAlarm =FALSE station_power(ON)	Stopped
Initial	not(alarm_detection())	Flagalarm = FALSE	Stopped
Stopped	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB2
Stopped	occur(data_carrier_on)	init_timer(TOSEUIL) init_timer(TAGT)	W.TOSEUIL
W.TOSEUIL	time_out(TOSEUIL)& not(Flagalarm)	station_power(ON)	W.AGT
W.TOSEUIL	occur(data_carrier_off) & not(Flagalarm)	stop_timer(TOSEUIL) stop_timer(TAGT)	Stopped
W.TOSEUIL	time_out(TOSEUIL) & Flagalarm	station_signal(ON) Tend = TOAG	W.AGT
W.TOSEUIL	occur(data_carrier_off) & Flagalarm	stop_timer(TOSEUIL) stop_timer(TAGT) Tend = TOAGN init_timer(Tend)	Hide
W.TOSEUIL	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB1
W.AGT	occur(data_carrier_off)	stop_timer(TAGT) init_timer(TOAPPEL)	W.Sel
W.AGT	time_out(TAGT) & not(Flagalarm)	station_power(OFF)	Stopped
W.AGT	time_out(TAGT) & Flagalarm	init_timer(Tend)	Hide
W.Sel	occur(octet_received_event)	stop_timer(TOAPPEL) init_timer(TOBAVARD) init_timer(TAO)	Select
W.Sel	time_out(TOAPPEL) & not(Flagalarm)	station_power(OFF)	Stopped
W.Sel	time_out(TOAPPEL) & Flagalarm	station_signal(OFF)	Stopped
W.Sel	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Select	occur(octet_received_event)	stop_timer(TAO) init_timer(TAO)	Select
Select	time_out(TAO)	stop_timer(TOBAVARD) init_timer(TOPRE)	W.Answer
Select	time_out(TOBAVARD) & not(Flagalarm)	stop_timer(TAO)	Stopped

Initial state	Triggering condition	Set of actions	Final state
		station_power(OFF)	
Select	time_out(TOBAVARD) & Flagalarm	stop_timer(TAO) init_timer(Tend)	Hide
W.Answer	occur(octet_sent_event)	stop_timer(TOPRE) init_timer(Tend)	Hide
W.Answer	time_out(TOPRE) & not(Flagalarm)	station_power(OFF)	Stopped
W.Answer	time_out(TOPRE) & Flagalarm	init_timer(Tend)	Hide
W.Answer	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Hide	occur(octet_received_event) |occur(octet_sent_event) I (occur(data_carrier_on) & not(FlagSendAlarm))	stop_timer(Tend) init_timer(Tend)	Hide
Hide	occur(data_carrier_on) & FlagSendAlarm	stop_timer(Tend)	W.AGend
Hide	time_out(Tend) & not(Flagalarm)	station_power(OFF)	Stopped
Hide	time_out(Tend) & Flagalarm & not(FlagSendAlarm)	station_signal(OFF)	Stopped
Hide	time_out(Tend) & Flagalarm & FlagSendAlarm	Send_AG(AGN)	W.AB
Hide	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
W.AGend	occur(data_carrier_off)	wait_time(TOALR) Send_AG(AGN)	W.AB
W.TVASB1	occur(cpt_carrier_off)	stop_timer(TVASB) init_timer(Tend)	Hide
W.TVASB1	time_out(TVASB)	FlagSendAlarm = TRUE init_timer(Tend)	Hide
W.TVASB1	time_out(Tend)	$none()	W.TVASB2
W.TVASB2	occur(cpt_carrier_off)	stop_timer(TVASB) station_signal(OFF)	Stopped
W.TVASB2	occur(data_carrier_on)	$none()	W.TVASB1
W.TVASB2	time_out(TVASB)	Send_AG(AGN)	W.AB
W.AB	AG_sent_event	FlagSendAlarm = FALSE station_signal(OFF)	Stopped

Table 6 – *Physical-62056-3-1* state transitions: Secondary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	energized()	MaxIndex=128 FlagRSO=FALSE FirstWinRSO=FALSE	Stopped
Initial	not(energized())	MaxIndex=18 FlagRSO=FALSE FirstWinRSO=TRUE	Stopped
Stopped	AG_received_event	Stop_timer(TOAG) init_timer(TA1O)	M.Rec
Stopped	Phy_ALARM.req()	TypeAG=ASB Send_AG(TypeAG)	W.ASB
Stopped	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	Stopped
M.Rec	octet_received_event	Stop_timer(TA1O) Index=2 Frame="" Read_data(RecB) Concat(Frame, RecB) init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	Phy_ABORT(EP-1)	*WTOAG*
M.Rec	Phy_ABORT.req()	Stop_timer(TA1O)	*WTOAG*
Receiving	octet_received_event & Index<=MaxIndex	Stop_timer(TAO) Index=Index+1 Read_data(RecB) Concat(Frame, RecB) init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Stop_timer(TAO) Phy_ABORT.ind(EP-4F)	*WTOAG*
Receiving	time_out(TAO)	Phy_DATA.ind(Frame) init_timer(TOL)	M.Send
Receiving	Phy_ABORT.req()	Stop_timer(TAO)	*WTOAG*
M.Send	Phy_DATA.req(Frame)	Stop_timer(TOL) MaxIndex=128 Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 init_timer(TOE)	Sending
M.Send	Phy_RSO.req(Frame, Window)	Stop_timer(TOL) MaxIndex=128 Wait_window(FirstWinRSO, Window) FirstWinRSO=FALSE Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 FlagRSO=TRUE init_timer(TOE)	Sending
M.Send	time_out(TOL)	init_timer(TA1O)	M.Rec
M.Send	Phy_ABORT.req()	Stop_timer(TOL)	*WTOAG*
Sending	octet_sent_event & Size>0	Index=Index+1 Send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0 & not(FlagRSO)	Stop_timer(TOE) init_timer(TA1O)	M.Rec
Sending	octet_sent_event & Size=0 & FlagRSO	Stop_timer(TOE) Wait_time(TAO) FlagRSO=FALSE	WTOAG
Sending	Phy_ABORT.req()	Stop_timer(TOE)	WTOAG
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F)	WTOAG
W.ASB	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	W.ASB
W.ASB	AG_sent_event	$none()	Stopped
WTOAG	not(energized)	init_timer(TOAG)	Stopped
WTOAG	Energized	$none()	Stopped

Table 7 – Meaning of the states listed in the previous tables

State	Meaning
Initial	Initialization of the variables of the layer
Stopped	Waiting state for a “Wakeup Call” signal
W.ETABS (Wait for end of “Alarm-Bus”)	Waiting for the end of an “Alarm-bus” signal received in a Stopped state
W.AG (Wait for end of “Wakeup Call”)	Waiting for the end of a “Wakeup Call” signal transmission
W.TAB (Wait “Alarm-Bus”)	Waiting for an “Alarm-Bus” signal during delay at the end of transmission of a “Wakeup call” signal
W.ETAB (Wait for end of “Alarm-Bus”)	Waiting for the end of an “Alarm-Bus” signal received after the transmission of a “Wakeup call” signal
W.TASB	Waiting for the triggering of wakeup TASB after the beginning of the reception of an “Alarm-Bus” signal
W.TOL	Waiting for the triggering of wakeup TOL
M.Send (Must Send)	Initial state of the transmitter waiting for a frame to send
T.Session	Testing the type of the session (with an energized or not energized Secondary Station)
SendFirst	Sending the first byte of the frame to be sent
Sending	Recurrent state of the transmitter transmitting one byte at a time
Answer	Branching depending on the service requested
M.Rec (Must Receive)	Initial state of the receiver waiting for the first byte of a frame
Receiving	Recurrent state of the receiver receiving one byte at a time
Received	Processing the received frame depending on the service requested
T.RSO (Test last RSO)	Testing the end of the last time slot for RSO frame reception
W.RSO (Wait for end of an RSO time slot)	Waiting for the end of a time slot for RSO frame reception
W.ASB	Waiting for the end of a “Alarm Secondary-Bus” signal transmission
W.TOAG	Initializing the “end of session” TOAG timer if needed
W.TOSEUIL	Waiting for the triggering of wakeup TOSEUIL
W.AGT	Waiting for an AGT “Wakeup Call” signal
W.Sel (Wait for preSelection)	Waiting for a preselection frame
Select	Receiving a preselection frame
W.Answer	Waiting for an answer frame from a selected station
Hide	Waiting for the end of selection
W.AGend	Waiting for the end of AG reception
W.TVASB1	Waiting for the triggering of wakeup TVASB for an “Alarm Secondary-Bus” signal during a session
W.TVASB2	Waiting for the triggering of wakeup TVASB for an “Alarm Secondary-Bus” signal at the end of session
W.AB	Waiting for the end of a “Alarm Bus” signal transmission

Table 8 – Definition of the procedures, functions and events classified in alphabetical order

Procedure, function or event	Definition
AG_received_event	Event from the modem reporting that an AGN “Wakeup Call” signal has been correctly detected
AG_sent_event	Event from the modem reporting the end of the transmission of a “Wakeup Call” signal
alarm_detection()	Check that the station status of alarm mode is Active
collision_detected_event	Event from the modem reporting the detection of a framing error on reception of a byte
concat(Frame, RecB)	Concatenation of the byte RecB in the being built frame Frame
data_carrier_on, data_carrier_off	Occurrence of the detection on the bus of the data carrier on, the data carrier off
energized()	Check that the station is energized
init_timer(TOAPPEL), init_timer(TOSEUIL), init_timer(TAGT), init_timer(TOBAVARD), init_timer(TOPRE), init_timer(TOL), init_timer(TOE), init_timer(TAO), init_timer(TA1O), init_timer(TARSO) init_timer(TOAG), init_timer(TVASB) or init_timer(TAB)	setting of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB or TAB
occur(cpt_carrier_on), occur(cpt_carrier_off), occur(data_carrier_on), occur(data_carrier_off), occur(octet_received_event) or occur(octet_sent_event)	Occurrence (reporting without consuming) of the detection on the secondary bus of the data carrier on, the data carrier off, on the bus of the data carrier on, the data carrier off, the reception of a byte or the emission of a byte
octet_received_event	Event from the modem reporting that a byte has been received
octet_sent_event	Event from the modem reporting that a byte has been sent
read_data(RecB)	Processing of the byte_received_event by reading the received RecB byte (bits are transmitted in ascending order)
send_AG(TypeAG)	Request to the modem for transmission of a “Wakeup Call” signal of duration TypeAG (AGN or AGT)
send_octet(Frame, Index)	Transmission of the byte of rank Index in the frame Frame (bits are transmitted in ascending order)
size(Frame)	Calculation of the number of bytes of the frame Frame
station_power(ON) or station_power(OFF)	Turning ON or OFF the energy supply to the device
station_signal(ON) or station_signal(OFF)	Turning ON or OFF the signal transmission to the device on the secondary bus
stop_timer(TOAPPEL), stop_timer(TOSEUIL), stop_timer(TAGT), stop_timer(TOBAVARD), stop_timer(TOPRE), stop_timer(TOL), stop_timer(TOE), stop_timer(TAO), stop_timer(TA1O), stop_timer(TVASB) or stop_timer(TAB)	Stopping of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO,TA1O, TVASB or TAB
stop_timer(TOAG) or stop_timer(TARSO)	Stopping of wakeup TOAG or TARSO only if it has previously been set

Procedure, function or event	Definition
time_out(TOAPPEL), time_out(TOSEUIL), time_out(TAGT), time_out(TOBAVARD), time_out(TOPRE), time_out(TOL), time_out(TOE), time_out(TAO), time_out(TA1O), time_out(TARSO), time_out(TOAG), time_out(TVASB) or stop_timer(TAB)	Triggering of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB or TAB
wait_time(TAO), wait_time(TICB), wait_time(TOL) or wait_time(TOALR)	Calculated delay during time TAO, TICB, TOL or TOALR
wait_window(FirstWinRSO, Window)	Wait-time calculated as follows: when FirstWinRSO=TRUE or Window=0 ==> 0 ms when FirstWinRSO=FALSE and Window>0 ==> 40 ms + (TARSO*Window) ms (The 40 ms delay guarantees that the transmission has taken place in the time slot)

5.1.6 List and processing of errors

Errors are listed using the following codes:

EP = error in the *Physical* layer

– = separator

N = error number

F = fatal error

Table 9 – Error summary table

EP-1	Expiry of TOL wakeup (Primary Station) before the *Data Link* layer requests a frame transmission or expiry of TA1O wakeup (Secondary Station) before any character has been received from the Primary station
	This error leads to the expectation of a “Wakeup Call” signal after having informed the *Data Link* layer
EP-2	Expiry of TOAG wakeup before any “Wakeup Call” signal
	This error leads to the expectation of a “Wakeup Call” signal after having informed the *Data Link* layer
EP-3	An alarm has been received
	This error leads to the reinitialization of the *Physical* layer after having informed the *Data Link* layer
EP-3F	Abnormal length of transmission detected after expiry of TOE wakeup
	This error leads to the reinitialization of the *Physical* layer after having informed the *Data Link* layer
EP-4F	Number of bytes received higher than MaxIndex (Transmitter too talkative)
	This error leads to the reinitialization of the *Physical* layer after having informed the *Data Link* layer
EP-5F	Expiry of TARSO wakeup while receiving an RSO frame (Primary Station only)
	This error leads to the reinitialization of the *Physical* layer after having informed the *Data Link* layer

If any of these errors occurs, it is sent up locally by means of the Phy_ABORT.ind service primitive. The complete list of fatal error numbers is given in Annex C.

5.2 Data Link layer

5.2.1 Link-62056-3-1 protocol

The *Link-62056-3-1* protocol of the *Data Link* layer of the local bus data exchange profile without DLMS behaves asymmetrically. The state machine of the Primary Station is therefore different from that of the Secondary Station.

The *Data Link* layer transforms the physical channel used by the *Physical* layer to a logic channel able to transmit reliable information. Its main functions are:

- to carry out a serialization and a deserialization of the data (if the physical channel functions serially one bit at a time);
- to synchronize the transmission and reception frames;
- to filter the frames according to primary and secondary addresses;
- to ensure efficient protection against transmission errors.

5.2.2 Management of exchanges

On the Primary station, the *Link-62056-3-1* protocol takes over the transmission of an AGN or AGT “Wakeup Call” signal according to the type of Secondary Station. Detection of incompatibility in the addresses of a DSDU received from the upper layer indicates a fatal error and the stop of the *Link-62056-3-1* protocol.

On the Secondary Station, reception of an incorrect frame does not require any processing, as recovering is left to the Primary Station.

For non-energized stations, the detection of a “Forgotten Stations Call” after an AGT “Wakeup Call” signal leads to an RSO response which always takes place in the first RSO time slot. Thus, the Primary Station, when detecting a collision after such a sequence, performs a second “Forgotten Stations Call”, but this time after an AGN “Wakeup Call” signal. Nevertheless, when no collision is detected after the first call, there is one or no forgotten station and no need for a second call.

5.2.3 Data Link services and service primitives

The user of the *Link-62056-3-1* protocol can use the services and service primitives given in Table 10.

Table 10 – Data Link services and service primitives

Service	Service primitive
DL_DATA	DL_DATA.req(DSDU) DL_DATA.ind(DSDU)
DL_ALARM	DL_ALARM.req() DL_ALARM.ind()
DL_ABORT	DL_ABORT.req() DL_ABORT.ind(ErrorNb)

The role assigned to each primitive is as follows:

- DL_DATA.req(DSDU) enables the *Application* layer to request the *Data Link* layer to transfer a DSDU data packet;
- DL_DATA.ind(DSDU) enables the *Data Link* layer to inform the *Application* layer of the arrival of a DSDU data packet;

- DL_ALARM.req() enables the *Application* layer to request the *Data Link* layer to transfer an alarm;
- DL_ALARM.ind() enables the *Data Link* layer to inform the *Application* layer of the arrival of an alarm;
- DL_ABORT.req() enables the *Application* layer to request the *Data Link* layer to end its activity;
- DL_ABORT.ind (ErrorNb) enables the *Data Link* layer to inform the *Application* layer of the occurrence of a fatal error identified by the number ErrorNb.

5.2.4 Data Link parameters

For the Primary Station, the value of the number of repeat transmissions for a given frame before disconnection, MaxRetry, is set to 2.

The value of the number of sequences linked with no “Wakeup Call” signal for remote reading and remote transfer, MaxChain, is set to 5, for compatibility with Secondary Stations using a previous version of the protocol.

The value of the maximum number, MaxRSO, of RSO time slots for the processing of a “Forgotten Stations Call” is set to 3 after an AGN “Wakeup Call” signal, at 1 after an AGT “Wakeup Call” signal.

The Secondary Station shall know the list of Primary Station addresses and the list of TABi to which it has been programmed.

The station may also be solicited by the general primary address APG. In this case, it replies with the first primary address to which it has been programmed.

5.2.5 State transitions

Table 11 – *Link-62056-3-1* state transitions: Primary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	$true()	MaxRetry=2 MaxChain=5	Stopped
Stopped	DL_DATA.req(DSDU) & check_req(DSDU)	NbChain=0 MaxRSO=3 PreSel=FALSE NoRetry=FALSE RepeatASO=FALSE EP-1=FALSE context(ADS, ADP, TypeAG) Com=command(DSDU) init(Com, TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	DL_DATA.req(DSDU) & not(check_req(DSDU))	DL_ABORT.ind(EL-1F)	Stopped
Stopped	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
W.AG	Phy_APPG.ind() & not(PreSel) & not(NoRetry) & not(RepeatASO)	$none()	*T.Req*
W.AG	Phy_APPG.ind() & PreSel	Index=MaxRetry+1 Fr=PRE Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec

Initial state	Triggering condition	Set of actions	Final state
W.AG	Phy_APPG.ind() & NoRetry	Index=MaxRetry+1 Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) NbChain=1 Phy_DATA.req(Fr) NoRetry=FALSE	M.Rec
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
W.EndS	Phy_ABORT.ind(ErrorNb)	$none()	*T.Error*
W.EndS	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
T.Error	((Error_Nb = EP-1 & TypeAG = AGN) I (Error_Nb = EP-2 & TypeAG = AGT)) & Com <>IB & Com <> TRB	$none()	Stopped
T.Error	((Error_Nb = EP-1 & TypeAG = AGN) I (Error_Nb = EP-2 & TypeAG = AGT)) & (Com=IB | Com=TRB)	DL_ABORT.ind(Error_Nb)	Stopped
T.Error	Error_Nb <> EP-1 & Error_Nb <> EP-2	DL_ABORT.ind(Error_Nb)	W.EndS
T.Error	(Error_Nb = EP-1) & TypeAG = AGT	$none()	W.EndS
T.Req	Com=IB | Com=TRB	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	Com=ASO & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	Com=ASO & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	((NbChain<MaxChain) & (Com=ENQ | Com=TRF)) | (NbChain=0 & Com=REC)	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Index=1 NbChain=NbChain+1 Phy_DATA.req(Fr)	M.Rec
T.Req	Com=AUT	Fr=DSDU Fr=concat(size_frame(Fr),ADS,ADP,Fr) Fr=concat(Fr, crc(Fr)) Index=1 NbChain=MaxChain Phy_DATA.req(Fr)	M.Rec
T.Req	(NbChain>=MaxChain) | (NbChain<>0 & Com=REC)	NbChain=0 Phy_APPG.req(AGN)	W.AG

Initial state	Triggering condition	Set of actions	Final state
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & not(PreSel)	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU)	M.Send
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & command(Frame)=SEL & PreSel	PreSel=FALSE	*T.Req*
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & command(Frame)<>SEL & PreSel	Phy_ABORT.req() DL_ABORT.ind(EL-2F)	W.EndS
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame)) & Index<=MaxRetry	Index=Index+1 Phy_DATA.req(Fr)	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame)) & Index>MaxRetry	Phy_ABORT.req() DL_ABORT.ind(EL-2F)	W.EndS
M.Rec	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & size(Frame)<>0 & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & size(Frame)<>0 & not(check_frame(Frame))	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.int(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DSDU=rso(RSO, Collision, ListRSO) DL_DATA.ind(DSDU)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	DL_DATA.req(DSDU) & check_req(DSDU) & not(EP-1)	Com=command(DSDU)	*T.Req*
M.Send	DL_DATA.req(DSDU) & check_req(DSDU) & EP-1	Com=command(DSDU) NbChain=0 EP-1=FALSE Phy_APPG.req(AGN)	W.AG
M.Send	DL_DATA.req(DSDU) & not(check_req(DSDU))	Phy_ABORT.req() DL_ABORT.ind(EL-1F)	W.EndS
M.Send	Phy_ABORT.ind(EP-1)	EP-1=TRUE	M.Send
M.Send	Phy_ABORT.ind(EP-2)	$none()	Stopped
M.Send	DL_ABORT.req() & not(EP_1 & TypAG=AGN)	Phy_ABORT.req()	W.EndS
M.Send	DL_ABORT.req() & EP_1 & TypAG=AGN	$none()	Stopped
M.Send	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS

Table 12 – *Link-62056-3-1* State transitions: Secondary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	$true()	FlagDSO=TRUE Discovered=FALSE Flag_alarm = FALSE	Stopped
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & check_address(Frame)	ADP=extract_ADP(Frame) Com=command(Frame)	*T.Com*
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & not(check_address(Frame))	Phy_ABORT.req()	Stopped
Stopped	Phy_DATA.ind(Frame) & not(check_frame(Frame))	$none()	Stopped
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	DL_ALARM.req()	Phy_ALARM.req() Flag_alarm = TRUE	Stopped
T.Com	Com=IB	FlagDSO=TRUE Discovered=FALSE Phy_ABORT.req()	Stopped
T.Com	Com=TRB	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU) Phy_ABORT.req()	Stopped
T.Com	Com=PRE	Fr=SEL Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	Stopped
T.Com	Com=ASO & test_TABi(Frame, TAB)	Fr=concat(RSO, TAB, ADS) Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_RSO.req(Fr, window_RSO())	Stopped
T.Com	Com=ASO & not(test_TABi(Frame, TAB))	Phy_ABORT.req()	Stopped
T.Com	Com=ENQ | Com=REC | Com=TRF | COM=AUT	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU)	M.Send
M.Send	DL_DATA.req(DSDU)	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) update_flag_DSO(command(Fr))	Stopped
M.Send	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped

Table 13 – Meaning of the states listed in the previous tables

State	Meaning
Initial	Initialization of the variables of the layer
Stopped	Waiting state for the first request from the upper layer or the first indication from the lower layer
W.AG (Wait for end of “Wakeup Call”)	Waiting for the end of a requested “Wakeup Call” signal
W.EndS (Wait for End of Session)	Waiting for the end of the session
T.Req (Test Request)	Test the nature of a request from the upper layer
M.Rec (Must Receive)	Waiting state for an indication from the lower layer
M.RSO (Must receive RSO)	Waiting for an RSO frame following an ASO frame sent
T.RSO (Test last RSO)	Testing the end of the last time slot for RSO frame reception
M.Send (Must Send)	Waiting state for a request from the upper layer
T.Com (Test Command)	Test of the command code of a received frame

Table 14 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
build_RSO(ListRSO, Frame)	Extraction of the RSO elements (TAB and ADS fields) from the received RSO frame Frame and concatenation with the preceding list ListRSO
check_address(Frame)	Check that the ADP and ADS addresses are recognized according to the following criteria: – ADP is APG or the station has been programmed to the ADP address; – if the command code is ASO, IB or TRB, then ADS is ADG; – if the command code is not ASO, IB nor TRB, then ADS is the secondary station address
check_frame(Frame)	Check that the frame Frame received is correct: – number of bytes greater than or equal to 11 and lower than or equal to 128; – CRC correct; – number of bytes compatible with the field N; – command code recognized and number of bytes compatible with the command code
check_req(DSDU)	Check that the requested command code inside a DSDU is compatible with ADP and ADS addresses defined in the communication context
command(DSDU) or command(Frame)	Extraction of the value of the command code of a DSDU to send or a received frame Frame
concat(N, ADS, ADP, DSDU) or concat(Frame, CRC)	Concatenation of the fields N, ADS and ADP with the DSDU or concatenation of the CRC at the end of the frame Frame
context(ADS, ADP, TypeAG)	Extraction of the corresponding values from the communication context
crc(Frame)	Calculation of the CRC of the frame Frame to send

Procedure or function	Definition
extract_ADP(Frame)	If either the ADP value used in the frame is not APG or if it is the APG value but the list of the ADP values to which the Secondary Station has been programmed is empty, then extraction of this value, otherwise extraction of the first ADP value to which the Secondary Station has been programmed
extract_DSDU(Frame)	Extraction of the DSDU (COM and DATA fields) from the frame Frame received
init(COM, TypeAG)	Set PreSel to TRUE if both TypeAG equals AGT and the size of the frame is greater than 18 bytes. Set NoRetry to TRUE if both TypeAG equals AGT and the size of the frame is lower than or equal to 18 bytes.
rso(RSO, Collision, ListRSO)	Concatenation of the RSO command code, the collision indicator and the list of the RSO elements (TAB and ADS fields)
size(Frame)	Calculation of the size of the frame Frame received
size_frame(DSDU)	Calculation of the size of the frame associated with a DSDU to be sent (size(DSDU) + 10)
test_TABi(Frame, TAB)	If the first TABi contained in the received ASO frame Frame equals 00, check that Discovered=FALSE then, after provision of a whole random integer between 0 and 100, check that this integer is smaller than the response probability (second TABi). In this case, 00 is memorized in the TAB variable; If the first TABi contained in the received ASO frame Frame equals FF, check that Flag_alarm=TRUE then, in this case, Flag_alarm is set to FALSE and FF is memorized in the TAB variable; If the first TABi contained in the received ASO frame Frame does not equal 00 or FF, check that FlagDSO=TRUE and check that the Secondary Station has been programmed to one of the TABi contained in the received ASO frame Frame. In this case, the first of these values is memorized in the TAB variable
update_flag_DSO(COM)	Set FlagDSO to FALSE and Discovered to TRUE if COM equals ENQ
window_RSO()	Provision of a whole random integer between 0 and MaxRSO-1 used as the number of the RSO time slot in which the station shall reply (refer to Annex F)

5.2.6 List and processing of errors

Errors are listed using the following codes:

EL = error in the *Data Link* layer

– = separator

N = error number

F = fatal error

Table 15 – Error summary table

EL-1F	Reception of a command code not compatible with the ADP and ADS addresses memorized in the communication context (Primary Station only)
	This error leads to the reinitialization of the *Data Link* layer after having informed the *Application* layer
EL-2F	Incorrect response from the Secondary Station after either a non-energized station preselection frame or MaxRetry repeated transmissions of a request (Primary Station only)
	This error leads to the re-initialization of the *Data Link* layer after having informed the *Application* layer

If any of these errors occurs, it is sent up locally by means of the DL_ABORT.ind service primitive. The complete list of fatal error numbers is given in Annex C.

5.3 Application layer

5.3.1 Application-62056-3-1 protocol

The *Application-62056-3-1* protocol of the *Application* layer of the local bus data exchange profile without DLMS behaves asymmetrically. The state machine of the Primary Station is therefore different from that of the Secondary Station.

The *Application-62056-3-1* protocol of the *Application* layer of the local bus data exchange profile without DLMS controls and links successive messages by analysing the command code supplied by the user application.

5.3.2 Application services and service primitives

The user of the *Application-62056-3-1* protocol can use the services and service primitives given in Table 16.

Table 16 – Application services and service primitives

Service	Service primitive
A_DATA	A_DATA.req(COM, ASDU) A_DATA.ind(ASDU)
A_ALARM	A_ALARM.req() A_ALARM.ind()
A_ABORT	A_ABORT.req() A_ABORT.ind(ErrorNb)

The role assigned to each primitive is as follows:

- A_DATA.req(COM, ASDU) enables the application to request the *Application* layer to transfer a COM (ENQ, REC, TRF, TRB, IB or ASO for a Primary Station and DAT, DRJ, EOS or TRA for a Secondary Station) command code linked with an ASDU information unit;
- A_DATA.ind(ASDU) enables the *Application* layer to inform the application of the arrival of an ASDU information unit;
- A_ALARM.req()enables the application to request the *Application* layer to send an alarm;
- A_ALARM.ind() enables the *Application* layer to inform the application of the arrival of an alarm;
- A_ABORT.req() enables the application to request the *Application* layer to end its activity;
- A_ABORT.ind(ErrorNb) enables the *Application* layer to inform the application of the occurrence of a fatal error identified by the number ErrorNb.

5.3.3 Application parameters

The Primary Station shall know the DES ciphering keys of all Secondary Stations for which a remote programming is to be carried out.

In case of remote programming, the Secondary Station shall know the DES ciphering key that could be used by the Primary Station.

5.3.4 State transitions

Table 17 – *Application-62056-3-1* state transitions: Primary Station

Initial state	Triggering condition	Set of actions	Final state
Stopped	A_DATA.req(Com, ASDU) & (Com=ASO | Com=ENQ | Com=TRF)	APDU=concat(Com, _, _, ASDU) DL_DATA.req(APDU)	M.Rec
Stopped	A_DATA.req(Com, ASDU) & (Com=IB | Com=TRB)	APDU=concat(Com, _, _, ASDU) DL_DATA.req(APDU)	W.EndS
Stopped	A_DATA.req(Com, ASDU) & Com=REC	Na1=randomize() Zdt=zdt(ASDU) APDU=concat(REC, Na1, 0, Zdt) DL_DATA.req(APDU) Na1k=cipher(Na1)	M.Rec
Stopped	DL_ALARM.ind()	A_ALARM.ind()	Stopped
Stopped	A_ABORT.req()	DL_ABORT.req()	Stopped
M.Rec	DL_DATA.ind(DSDU)	Resp=command(DSDU)	*T.Resp*
M.Rec	A_ABORT.req()	DL_ABORT.req()	Stopped
M.Rec	DL_ABORT.ind(ErrorNb) & Error_Nb <> EP_1 & Error_Nb <> EP_2	A_ABORT.ind(ErrorNb)	Stopped
M.Rec	DL_ABORT.ind(ErrorNb) & (Error_Nb = EP_1 I Error_Nb = EP_2)	$none()	M.Rec
W.EndS	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped
W.EndS	A_ABORT.req()	DL_ABORT.req()	W.EndS
T.Resp	(Com=ASO & Resp=RSO) | (Com=ENQ & Resp=DAT) | (Com=ENQ & Resp=DRJ) | (Com=TRF & Resp=TRA) | (Com=TRF & Resp=DRJ) | (Com=AUT & Resp=EOS) | (Com=AUT & Resp=DRJ)	A_DATA.ind(DSDU)	Stopped
T.Resp	Com=REC & Resp=ECH & na1k(DSDU)=Na1k & Zdt=zdt(DSDU)	Na2=na2(DSDU) Na2k=cipher(Na2) Com=AUT APDU=concat(AUT, 0, Na2k, ‘’’’) DL_DATA.req(APDU)	M.Rec
T.Resp	(Com=ASO & Resp<>RSO) | (Com=ENQ & Resp<>DAT & Resp<>DRJ) | (Com=TRF & Resp<>TRA & Resp<>DRJ) | (Com=REC & Resp<>ECH) | (Com=AUT & Resp<>ARJ & Resp<>DRJ & Resp<>EOS)	A_ABORT.ind(EA-1F) DL_ABORT.req()	Stopped
T.Resp	Com=REC & (Resp<>ECH | (na1k(DSDU)<>Na1k) | (Zdt<>zdt(DSDU))	A_ABORT.ind(EA-2F) DL_ABORT.req()	Stopped
T.Resp	Com=AUT & Resp=ARJ	A_ABORT.ind(EA-3F) DL_ABORT.req()	Stopped

Table 18 – *Application-62056-3-1* state transitions: Secondary Station

Initial state	Triggering condition	Set of actions	Final state
Stopped	DL_DATA.ind(DSDU) & (command(DSDU)=ENQ | command(DSDU)=TRF)	A_DATA.ind(DSDU) Req=command(DSDU)	M.Send
Stopped	DL_DATA.ind(DSDU) & command(DSDU)=TRB	A_DATA.ind(DSDU)	Stopped
Stopped	DL_DATA.ind(DSDU) & command(DSDU)=REC	Zdt=zdt(DSDU) Na1=na1(DSDU) Na1k=cipher(Na1) Na2=randomize() APDU=concat(ECH, Na1k, Na2, Zdt) DL_DATA.req(APDU) Na2k=cipher(Na2) Req=REC	W.AUT
Stopped	A_ALARM.req() & alarm_detection()	DL_ALARM.req()	Stopped
M.Send	A_DATA.req(COM, ASDU) & (((COM=DAT | COM=DRJ) & Req=ENQ) | ((COM=TRA | COM=DRJ) & Req=TRF) | (COM=DRJ & Req=REC))	APDU=concat(COM, _, _, ASDU) DL_DATA.req(APDU)	Stopped
M.Send	A_DATA.req(COM, ASDU) & (COM=EOS & Req=REC)	APDU=concat(EOS, 0, 0, ‘’’’) DL_DATA.req(APDU)	Stopped
M.Send	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped
W.AUT	DL_DATA.ind(DSDU) & command(DSDU)=AUT & na2k(DSDU)=Na2k	ASDU=concat(REC, _, _, Zdt) A_DATA.ind(ASDU)	M.Send
W.AUT	DL_DATA.ind(DSDU) & command(DSDU)=AUT & na2k(DSDU)<>Na2k	APDU=concat(ARJ, _, _, ‘’’’) DL_DATA.req(APDU)	Stopped
W.AUT	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped

Table 19 – Meaning of the states listed in the previous tables

State	Meaning
Stopped	Waiting state for the first request from the upper layer or the first indication from the lower layer
M.Rec (Must Receive)	Standby for the response to the request transmitted
T.Resp (Test Response)	Processing of the response received
M.Send (Must Send)	Waiting for a response to a received request
W.AUT (Wait for AUT frame)	Waiting for an AUT frame following an ECH response frame sent

Table 20 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
alarm_detection()	Check that the status of Alarm mode is Active
cipher(Na1) or cipher(Na2)	Ciphering of random number Na1 or Na2 by means of DES algorithm with key contained in the communication context
command(DSDU)	Extraction of the value of the command code of a received DSDU
concat(COM, _, _, ASDU), concat(COM, ZA1, ZA2, ZDT) or concat(COM, 0, 0, _)	Concatenation of a COM command code and an ASDU or concatenation of a COM command code with an encrypted value ZA1, an encrypted value ZA2 and a data field ZDT (TAB and DATA fields) or concatenation of a COM command code and the ZA1 = 0 and ZA2 = 0 fields
na1(DSDU)	Extraction of the Na1 value from the ZA1 field of a received REC frame
na1k(DSDU)	Extraction of the Na1k value from the ZA1 field of a received ECH frame
na2(DSDU)	Extraction of the Na2 value from the ZA2 field of a received ECH frame
na2k(DSDU)	Extraction of the Na2k value from the ZA2 field of a received AUT frame
randomize()	Generation of a random number according to the procedure described in Annex G
zdt(ASDU) or zdt(DSDU)	Extraction of data (TAB and DATA fields) from a REC request, a REC frame or an ECH frame

5.3.5 List and processing of errors

Errors are listed using the following codes:

EA = error in the *Application* layer

– = separator

N = error number

F = fatal error

Table 21 – Error summary table

EA-1F	The response command code of the received frame does not correspond with the request command code (Primary Station only)
	This error leads to the re-initialization of the *Application* layer after having informed the application
EA-2F	The Secondary Station has not been correctly authenticated (Primary Station only)
	This error leads to the re-initialization of the *Application* layer after having informed the application
EA-3F	The Primary Station has not been correctly authenticated (Primary Station only)
	This error leads to the re-initialization of the *Application* layer after having informed the application

If any of these errors occurs, it is sent up locally by means of the A_ABORT.ind service primitive. The complete list of fatal error numbers is given in Annex C.

6 Local bus data exchange with DLMS

6.1 Physical layer

The *Physical-62056-3-1* protocol of the *Physical* layer of the local bus data exchange profile with DLMS is exactly the same as the one defined for the profile without DLMS.

6.2 Data Link layer

6.2.1 Link-E/D protocol

The *Link-E/D* protocol of the *Data Link* layer of the local bus data exchange profile with DLMS behaves asymmetrically. The state machine of the Primary Station is therefore different from that of the Secondary Station.

The *Data Link* layer transforms the physical channel used by the *Physical* layer into a logic channel able to transmit reliable information. Its main functions are:

- to manage directly the Bus Initialization and Forgotten Stations Call services;
- to carry out serialization and deserialization of the data (if the physical channel functions serially one bit at a time);
- to synchronize the transmission and reception of the frames;
- to filter the frames according to primary and secondary addresses;
- to ensure efficient protection against transmission errors.

6.2.2 Management of exchanges

Bus Initialization, Alarm and Forgotten Stations Call services are provided by the *Link E/D* protocol of the *Data Link* layer of the local bus data exchange profile with DLMS, but their operation takes place outside the *Application* layer. In particular, the forgotten station flag can be updated by the Application Programming Interface when a remote reading exchange occurs.

Except during opening of the communication session and Bus Initialization, Alarm report or management of Forgotten Stations Call, the protocol is perfectly symmetrical, and both stations take turns to act as Transmitter and Receiver.

After sending a frame, the *Data Link* layer of the Transmitter end always waits for a frame from the *Data Link* layer of the Receiver before transmitting again. This wait is controlled by the T1 wakeup, whose duration is 10 s.

After sending a frame and receiving an acknowledgement of the previously sent one, the current frame is retransmitted. The number of repeat transmissions is limited to MaxRetry. Above this number, the communication is stopped at the *Data Link* level and the *Application* layer is informed.

Each time a frame is received by one of the systems, an answer frame is immediately transmitted. This transmitted frame may contain data from the *Application* layer. It always contains a *Send* number and a *Confirm* number calculated according to the values of those previously sent and received. The following algorithm is used to calculate these numbers:

- if the last frame received is error-free and its *Send* number is equal to the 1's complement of the previous *Confirm* number in transmission, then the data packet is transmitted to the *Application* layer and the next frame sent will have a *Confirm* number equal to the received *Send* number. If not, the *Confirm* number is not modified and the data packet is not transmitted to the *Application* layer;
- if the last frame received is error-free and its *Confirm* number is identical to the previous *Send* number in transmission, then the *Send* number in transmission is made up to 1 for the next frame on the assumption that a new data packet has to be sent;
- if the last frame received is incorrect or if its *Confirm* number is not identical to the previous *Send* number in transmission, then the same frame is transmitted again, on condition that the number of repeat transmissions remains less than or equal to MaxRetry.

6.2.3 Data Link services and service primitives

The user of the *Link-E/D* protocol can use the services and service primitives given in Table 22.

Table 22 – Data Link services and service primitives

Service	Service primitive
DL_DATA	DL_DATA.req(Pr, DSDU) DL_DATA.ind(Pr, DSDU)
DL_IB	DL_IB.req()
DL_ASO	DL_ASO.req(DSDU) DL_ASO.ind(Collision, List)
DL_ALARM	DL_ALARM.req()
DL_ABORT	DL_ABORT.req(Strong) DL_ABORT.ind(ErrorNb)

The role assigned to each primitive is as follows:

- DL_DATA.req(Pr, DSDU) enables the *Application* layer to request the *Data Link* layer to transfer a DSDU data packet with the priority Pr [3)];
- DL_DATA.ind(Pr, DSDU) enables the *Data Link* layer to inform the *Application* layer of the arrival of a DSDU data packet with the priority Pr;
- DL_IB.req() enables the *Application* Programming Interface to request the *Data Link* layer to send a Bus Initialization frame;
- DL_ASO.req(DSDU) enables the *Application Programming Interface* to request the *Data Link* layer to send a Forgotten Stations Call frame in accordance with a DSDU data packet;
- DL_ASO.ind(Collision, List) enables the *Data Link* layer to inform the *Application Programming Interface* of the result of a Forgotten Stations Call;
- DL_ALARM.req() enables the *Application Programming Interface* to request the *Data Link* layer to transfer an alarm;
- DL_ABORT.req(Strong) enables the *Application* layer to request the *Data Link* layer to terminate its activity with the priority Strong [4)];
- DL_ABORT.ind(ErrorNb) enables the *Data Link* layer to inform the *Application* layer of the occurrence of a fatal error identified by the number ErrorNb.

6.2.4 Data Link parameters

The value of the number of repeat transmissions for a given frame before disconnection, MaxRetry, is set to 2.

For the Primary Station, the value of the maximum number, MaxRSO, of RSO time slots for the processing of a “Forgotten Stations Call” is set to 3.

The Secondary Station shall know the list of Primary Station addresses and the list of TABi to which it has been programmed.

---

3) The priority level Pr differentiates the processing of emergency services such as InformationReport (level Pr=1) from that of the other DLMS services (level Pr=0).

4) The Strong parameter differentiates the processing of fatal errors (Strong=1) from that of the other physical disconnection requests (Strong=0) initialized by the *Application* sub-layer.

The station may also be solicited by the general primary address APG. In this case, it replies with the first primary address to which it has been programmed.

6.2.5 State transitions

Table 23 – *Link-E/D* state transitions: Primary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	$true()	MaxRetry=2 MaxChain = 5 init_incrChain()	Stopped
Stopped	exist_dl_req()	NbChain = 0 RepeatASO=FALSE context(ADP, ADS, TypeAG) init(TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	Phy_ALARM.ind()	create_alarm(TPDU) DL_DATA.ind(Pr = 1, TPDU)	Stopped
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain = 0	$none()	*T.Req*
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain <> 0	NbChain = 0	*M.Send*
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
W.EndS	(Phy_ABORT.ind(EP-2) & TypeAG=AGT) I (Phy_ABORT.ind(EP-1) & TypeAG=AGN)	$none()	Stopped
W.EndS	Phy_ALARM.ind	create_alarm(TPDU) DL_DATA.ind(Pr = 1, TPDU)	Stopped
W.EndS	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS
T.Req	exist_dl_req(DL_IB.req())	Fr="" Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, IB, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	exist_dl_req(DL_ASO.req(DSDU)) & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	exist_dl_req(DL_ASO.req(DSDU)) & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO

Initial state	Triggering condition	Set of actions	Final state
T.Req	not (exist_dl_req(DL_IB.req()) | exist_dl_req(DL_ASO.req(_)))	Pr=0 Send="00"B Confirm="11"B Fr="" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & not(check_frame(Frame)) & size(Frame)<>0	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DL_ASO.ind(Collision, ListRSO)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, DSDU)) & NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, DSDU)) & NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(_, _)) & NbChain < MaxChain	Pr=0 Fr="" Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	NbChain >= MaxChain	Phy_APPG.req(AGN)	W.AG

Initial state	Triggering condition	Set of actions	Final state
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	DL_DATA.ind(extract_prty(Frame), extract_text(Frame)) Confirm=incr(Confirm) Ack_expected=FALSE Index = 0	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	Ack_expected=FALSE Index = 0	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	DL_ABORT.ind(EL-2F) Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=1)	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS

Table 24 – *Link-E/D* state transitions: Secondary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	$true()	MaxRetry=2 FlagDSO=TRUE Discovered=FALSE Flag_alarm = FALSE	Stopped
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & check_address(Frame)	ADP=extract_ADP(Frame) Com=command(Frame)	*T.Com*
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & not(check_address(Frame))	Phy_ABORT.req()	Stopped
Stopped	Phy_DATA.ind(Frame) & not(check_frame(Frame))	$none()	Stopped
Stopped	DL_ALARM.req() & alarm_detection()	Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
T.Com	Com=IB	FlagDSO=TRUE Discovered=FALSE Phy_ABORT.req()	Stopped
T.Com	Com=ASO & test_TABi(Frame, TAB)	Fr=concat(RSO, TAB, ADS) Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_RSO.req(Fr, window_RSO())	Stopped
T.Com	Com=ASO & not(test_TABi(Frame, TAB))	Phy_ABORT.req()	Stopped
T.Com	is_data+(Frame) & is_text(Frame)	Ack_expected=FALSE Send="11"B Confirm="00"B DL_DATA.ind(extract_prty, extract_text(Frame))	*M.Send*

Initial state	Triggering condition	Set of actions	Final state
T.Com	is_data+(Frame) & not(is_text(Frame))	Ack_expected=FALSE Send="11"B Confirm="00"B	*M.Send*
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, DSDU))	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, DSDU))	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(_, _))	Pr=0 Fr="" Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	stop_timer(T1) Confirm=incr(Confirm) Ack_expected=FALSE DL_DATA.ind(extract_prty, extract_text(Frame))	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	stop_timer(T1) Ack_expected=FALSE	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	stop_timer(T1) Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	stop_timer(T1) DL_ABORT.ind(EL-2F) Phy_ABORT.req()	Stopped
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	stop_timer(T1) Phy_ABORT.req()	Stopped
M.Rec	DL_ALARM.req() & alarm_detection()	stop_timer(T1) DL_ABORT.ind(EL_1F) Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
M.Rec	DL_ABORT.req(Strong=1)	stop_timer(T1) Phy_ABORT.req()	Stopped
M.Rec	Phy_ABORT.ind(EP-1)	init_timer(T1)	M.Rec
M.Rec	Phy_ABORT.ind(ErrorNb) & ErrorNb <> EP-1	stop_timer(T1) DL_ABORT.ind(ErrorNb)	Stopped
M.Rec	time_out(T1)	DL_ABORT.ind(EL_3F)	Stopped

Table 25 – Meaning of the states listed in the previous tables

State	Meaning
Initial	Initialization of the variables of the layer
Stopped	Waiting state for the first request from the upper layer or the first indication from the lower layer
W.AG (Wait for end of “Wakeup Call”)	Waiting for the end of a requested “Wakeup Call” signal
W.EndS (Wait for End of Session)	Waiting for the end of the session
T.Req (Test Request)	Test the nature of a request from the upper layer
M.RSO (Must receive RSO)	Waiting for an RSO frame following an ASO frame sent
T.RSO (Test last RSO)	Testing the end of the last time slot for RSO frame reception
M.Send (Must Send)	A frame shall be sent (possibly with the Text field empty)
M.Rec (Must Receive)	Waiting state for an indication from the lower layer
T.Com (Test Command)	Test of the command code of a received frame

Table 26 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
Alarm_detection()	Check that the status of alarm mode is Active
build_RSO(ListRSO, Frame)	Extraction of the RSO elements (TAB and ADS fields) from the received RSO frame Frame and concatenation with the preceding list ListRSO
check_address(Frame)	Check that the ADP and ADS addresses are recognized according to the following criteria: – ADP is APG or the station has been programmed to the ADP address; – if the command code is ASO or IB, then ADS is ADG; – if the command code is not ASO nor IB, then ADS is the secondary station address
check_frame(Frame)	check that the frame Frame received is correct: – number of bytes greater than or equal to 11 and lower than or equal to 128; – CRC correct; – number of bytes compatible with the field Size; – command code recognized
com(DATA+, Pr, Send, Confirm)	Concatenation of the corresponding bit fields to produce a specific command code
command(Frame)	Extraction of the value of the command code of the received frame Frame
concat(Size, ADS, ADP, COM,Text) or concat(Frame, CRC)	Concatenation of the fields Size, ADS, ADP, COM and Text or concatenation of the CRC at the end of the frame Frame
context(ADS, ADP, TypeAG)	Extraction of the corresponding values from the communication context
crc(Frame)	Calculation of the CRC of the frame Frame to send
create_alarm(TPDU)	Calculation of a TPDU where STSAP = 0, DTSAP = 0 and an UnsolicitedReqPDU with: client-type = FFFF, serveridentifier = 0, object-name = FFFF variable type = boolean, value = TRUE
exist_dl_data_req(DL_DATA.req(Pr, DSDU))	Consumption of a DL_DATA.req(Pr, DSDU) event

Procedure or function	Definition
exist_dl_req()	Check the existence of a DL_IB.req(), DL_ASO.req(DSDU) or DL_DATA.req(Pr, DSDU) event and check the compatibility with ADP and ADS addresses defined in the communication context
exist_dl_req(DL_IB.req()) or exist_dl_req(DL_ASO.req(DSDU))	Consumption of a DL_IB.req() or DL_ASO.req(DSDU) event
extract_ADP(Frame)	If either the ADP value used in the frame is not APG or if it is the APG value but the list of the ADP values to which the Secondary Station has been programmed is empty, then extraction of this value, otherwise extraction of the first ADP value to which the Secondary Station has been programmed
extract_prty(Frame)	Extraction of the Priority field from a received frame Frame
extract_text(Frame)	Extraction of the Text field from a received frame Frame
init(TypeAG)	Set Index to MaxRetry if TypeAG equals AGT, otherwise to 0
init_incrChain()	Set IncrChain to 0 if alarm supported, otherwise to 1
init_timer(T1)	Setting of wakeup T1
is_ack(Frame)	Check that the received frame Frame contains a Confirm field equal to the Send field of the last frame transmitted
is_data+(Frame)	Check that the received frame Frame contains a correct DATA+ field ("111"B)
is_text(Frame)	Check that the received frame Frame contains a non-empty Text field and that the Send field equals the complement of the Confirm field of the last frame transmitted
size(Frame)	Calculation of the size of the frame Frame received
size_frame(DSDU)	Calculation of the size of the frame to build with the DSDU data unit (size(DSDU) + 11)
stop_timer(T1)	Stopping of wakeup T1
test_TABi(Frame, TAB)	If the first TABi contained in the received ASO frame Frame equals 00, check that Discovered=FALSE then, after provision of a whole random integer between 0 and 100, check that this integer is smaller than the response probability (second TABi). In this case, 00 is memorized in the TAB variable; if the first TABi contained in the received ASO frame Frame equals FF, check that Flag_alarm=TRUE then, in this case, Flag_alarm is set to FALSE and FF is memorized in the TAB variable; If the first TABi contained in the received ASO frame Frame does not equal 00 or FF, check that FlagDSO=TRUE and check that the Secondary Station has been programmed to one of the TABi contained in the received ASO frame Frame. In this case, the first of these values is memorized in the TAB variable
time_out(T1)	Triggering of wakeup T1
window_RSO()	Provision of a whole random integer between 0 and MaxRSO-1 used as the number of the RSO time slot in which the station shall reply (refer to Annex F)

6.2.6 List and processing of errors

Errors are listed using the following codes:

EL = error in the *Data Link* layer

– = separator

N = error number

F = fatal error

Table 27 – Error summary table

EL-1F	DL_ALARM.req() received during an association
	This request leads to the re-initialization of the *Data Link* layer after having requested the *Physical* layer to transmit an Alarm
EL-2F	Incorrect response from the Secondary Station after MaxRetry repeated transmissions of a request
	This error leads to the re-initialization of the *Data Link* layer after having informed the *Application* layer and caused the *Physical* layer to abort
EL-3F	End of communication due to expiration of delay T1
	This error leads to the re-initialization of the *Data Link* layer after having informed the *Application* layer and caused the *Physical* layer to abort

If any of these errors occur, it is sent up locally by means of the DL_ABORT.ind service primitive. The complete list of fatal error numbers is given in Annex C.

6.3 Application layer

6.3.1 General

The *Application* layer specification is described in IEC 62056-51. This subclause simply clarifies the operating profile for the local bus data exchange profile with DLMS.

6.3.2 Transport sub-layer

The value of the number MaxPktSize (refer to IEC 62056-51), maximum size of *Packet* field, shall be set at 114.

6.3.3 Application sub-layer

As stated in IEC 62056-51, the client_connect and server_connect functions shall be clarified according to the communication medium used. For the local bus data exchange profile with DLMS, the server_connect function is not accepted because the unsolicited service management is not supported. The client_connect function is defined in Table 28.

Table 28 – Client_connect function definition

Procedure or function	Definition
client_connect(Adp, Ads)	If there is no active application association, the function sets the context parameters ADS, ADP and TypeAG used by the lower layers. The function is transparent if there is already an application association active with the same (ADP, ADS) addresses. The function fails if there is already an application association with different (ADP, ADS) addresses.

7 Local bus data exchange with DLMS/COSEM

7.1 Model

The profile with DLMS/COSEM is different from the two other profiles in that the application layer is the DLMS/COSEM application layer specified in IEC 62056-5-3 Ed. 1.0:—. This enables data exchange over the Euridis bus with equipment that follow the DLMS/COSEM application layer and the COSEM.

Another difference in this profile is the presence of a *Transport* layer and a *Support Manager* Layer.

7.2 Physical Layer

7.2.1 General

The protocol of the physical layer in the profile with DLMS/COSEM is identical to the protocol of the physical layer in the profiles with or without DLMS, up to the end of the correct negotiation of the transmission speed included. Once the speed negotiation phase is correctly completed, the DLMS/COSEM physical layer is applied. This is not different from the physical layer with or without DLMS, except the transmission speed, the MaxIndex value which shall be 255 and the modification of the time-out values TOL and TA10.

After a 'Wakeup Call' signal, asynchronous semi-duplex communication starts at 1 200 bauds, 8 bits without parity on the bus during the whole communication period without speed negotiation. After a successful speed negotiation, communication is asynchronous, semi-duplex, 8 bits without parity at the negotiated speed.

This speed negotiation only takes place for non energized equipment. For energized equipment, in order to respect consumption constraints, the communication speed stays at 1 200 bauds for the entire duration of the communication and the MaxIndex at 128 bytes. DLMS/COSEM operation under Euridis remains possible with these initial settings.

For non-energised equipment, operation with a DLMS/COSEM Application layer is possible with or without a change of speed. In the case where the application decides to change the speed, the relative speed negotiation settings takes place before the application associations are established.

7.2.2 Physical Parameters

Communication always starts at 1 200 bauds without parity, one stop bit. The maximum frame size is 128 bytes.

After speed negotiation, the newly negotiated speed is applied. The maximum frame size, MaxIndex, becomes 255 bytes.

The value of the maximum number of RSO time slots for the processing of a 'Forgotten Stations Call', MaxRSO, is set to 3.

The TOL timeout maximum response waiting time from an upper layer is increased to 1 s.

7.2.3 Speed negotiation

Management of the speed negotiation occurs in the *Support Manager* layer. The physical layer is informed of the new speed to apply, affecting the MaxIndex and TOL. As a consequence of the modification of the TOL value, following a speed negotiation, the maximum waiting time of the first received byte of a frame rises to a value higher than the TOL, depending on the application. By default this value is set to 1 100 ms.

7.2.4 E/COSEM Physical Services and service primitives

The *Physical E/COSEM* protocol includes services and service primitives listed in Table 29.

Table 29 – E/COSEM Physical services and service primitives

Service	Service primitive
Phy_DATA	Phy_DATA.req(Frame) Phy_DATA.ind(Frame)
Phy_UNACK	Phy_UNACK.req(Frame)
Phy_APPG	Phy_APPG.req(TypeAG) Phy_APPG.ind()
Phy_ASO	Phy_ASO.req(Frame) Phy_ASO.ind(Frame)
Phy_RSO	Phy_RSO.req(Frame, Window)
Phy_COLL	Phy_COLL.ind()
Phy_ALARM	Phy_ALARM_req() Phy_ALARM_ind()
Phy_ABORT	Phy_ABORT.req() Phy_ABORT.ind(ErrorNb)
Phy_SETUP	PHY_SETUP.req(params)

The role assigned to each primitive is a follows:

- Phy_DATA.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a frame Frame;
- Phy_DATA.ind(Frame) enables the *Physical* layer to inform the *Data Link* layer that a frame Frame is available;
- Phy_UNACK.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a frame Frame without waiting for acknowledgement;
- Phy_APPG.req (TypeAG) enables the *Data Link* layer to request the *Physical* layer to transmit a “Wakeup Call” signal. The duration TypeAG of this signal is either AGN or AGT;
- Phy_APPG.ind() enables the *Physical* layer to inform the *Data Link* layer of the end of the transmission of a “Wakeup Call” signal;
- Phy_ASO.req(Frame) enables the *Data Link* layer to request the *Physical* layer to transmit a “Forgotten Stations Call” frame;
- Phy_ASO.ind(Frame) enables the *Physical* layer to inform the *Data Link* layer that a frame Frame has been received in one of the time slots of the forgotten stations;
- Phy_RSO.req(Frame, Window) enables the *Data Link* layer to request the *Physical* layer to transmit a Forgotten Stations Call frame Frame in the time slot number Window;
- Phy_COLL.ind() enables the *Physical* layer to inform the *Data Link* layer that a collision has been detected in one of the time slots of the forgotten stations;
- Phy_ALARM.req() enables the *Data Link* layer to request the *Physical* layer to transmit an Alarm;
- Phy_ALARM.ind() enables the *Physical* layer to inform the *Data Link* layer of the arrival of an alarm;
- Phy_ABORT.req() enables the *Data Link* layer to request the *Physical* layer to end its activity;
- Phy_ABORT.ind(ErrorNb) enables the *Physical* layer to inform the *Data Link* layer of the occurrence of a fatal error identified by the number ErrorNb.

- Phy SETUP.req(params) enables the *Data Link* layer to request the *Physical* Layer to reconfigure itself according to the parameters related to the speed negotiation.

7.2.5 State transitions

Table 30 – *E/COSEM Physical* state transitions: Primary Station

Initial status	Initiation conditions	Actions	Final Status
Initial	$true()	MaxRSO=3 MaxIndex=128 Collision=FALSE SessionAGT=FALSE wait_time(TICB) Set_baudrate(1200) TOL=100ms TA10=120ms	Stopped
Stopped	Phy_APPG.req(AG) & AG=AGN	stop_timer(TOAG) FlagAbort=FALSE TypeAG=AGN send_AG(TypeAG)	W.AG
Stopped	Phy_APPG.req(AG) & AG=AGT	SessionAGT=TRUE FlagAbort=FALSE TypeAG=AGT send_AG(TypeAG)	W.AG
Stopped	time_out(TOAG)	Phy_ABORT.ind(EP-2) SessionAGT=FALSE	*Initial*
Stopped	Phy_ABORT.req()	$none()	*Initial*
Stopped	data-carrier_on	init_timer(TAB) init_timer (TASB)	W.ETABS
W.ETABS	data_carrier_off	stop_timer(TASB) stop_timer(TAB)	*Initial*
W.ETABS	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind()	W.TASB
W.AG	AG_sent_event	Phy_APPG.ind() init_timer(TEMPO)	W.TAB
W.AG	Phy_ABORT.req()	FlagAbort=TRUE	W.AG
W.TAB	data-carrier_on	Carrier = TRUE init_timer(TAB) init_timer (TASB)	W.TAB
W.TAB	data-carrier_off	Carrier = FALSE stop_timer(TAB) stop_timer (TASB)	W.TAB
W.TAB	time_out(TEMPO) & not(FlagAbort) & not(Carrier)	init_timer(TOL)	M.Send
W.TAB	time_out(TEMPO) & FlagAbort & not(Carrier)	wait_time (TOL)	*T.Session*
W.TAB	time_out(TEMPO) & Carrier	init_timer(TOL)	W.ETAB
W.TAB	Phy_ABORT.req()	FlagAbort=TRUE	W.TAB
W.ETAB	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind() stop_timer(TOL)	W.TASB
W.ETAB	data_carrier_off & not(FlagAbort)	stop_timer(TAB) stop_timer (TASB)	M.Send
W.ETAB	data_carrier_off & FlagAbort	stop_timer(TAB) stop_timer (TASB)	W.TOL
W.ETAB	Phy_ABORT.req()	FlagAbort=TRUE	W.ETAB
W.TASB	time_out(TASB)	$none()	*Initial*
W.TOL	time_out(TOL)	$none()	*T.Session*

Initial status	Initiation conditions	Actions	Final Status
M.Send	Phy_DATA.req(Frame)	Service=NORMAL	*SendFirst*
M.Send	Phy_UNACK.req(Frame)	Service=UNACKNOWLEDGED	*SendFirst*
M.Send	Phy_ASO.req(Frame)	Service=ASO	*SendFirst*
M.Send	Phy_ABORT.req()	$none()	M.Send
M.Send	time_out(TOL)	$none()	*T.Session*
M.Send	PHY_SETUP(params)	Set baudrate(new baudrate) MaxIndex=255 TOL=2000ms TA10=2100ms	M.Send
T.Session	SessionAGT=TRUE	init_timer(TOAG) Phy_ABORT.ind(EP-1) wait_time(TEMPO)	*Initial*
T.Session	SessionAGT=FALSE	Phy_ABORT.ind(EP-1) wait_time(TEMPO)	*Initial*
SendFirst	$true()	stop_timer(TOL) Size=size(Frame) Index=1 send_octet(Frame, Index) Size=Size-1 init_timer(TOE)	Sending
Sending	octet_sent_event & Size>0	Index=Index+1 send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0	stop_timer(TOE) wait_time(TAO) Index=1 Frame=""	*Answer*
Sending	Phy_ABORT.req()	stop_timer(TOE) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Answer	Service=NORMAL | Service=UNACKNOWLEDGED	init_timer(TA1O)	M.Rec
Answer	Service=ASO	WinRSO=1 init_timer(TARSO) init_timer(TA1O)	M.Rec
M.Rec	octet_received_event	stop_timer(TA1O) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
M.Rec	Collision_detected_event	stop_timer(TA1O) Collision=TRUE init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	$none()	*Received*
M.Rec	Phy_ABORT.req()	FlagAbort=TRUE	M.Rec
Receiving	octet_received_event & Index<=MaxIndex	stop_timer(TAO) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Phy_ABORT.ind(EP-4F) wait_time(TAO) FlagAbort=TRUE	*Received*

Initial status	Initiation conditions	Actions	Final Status
Receiving	Collision_detected_event	stop_timer(TAO) Collision=TRUE init_timer(TAO)	Receiving
Receiving	time_out(TAO)	$none()	*Received*
Receiving	time_out(TARSO)	Phy_ABORT.ind(EP-5F) wait_time(TAO) FlagAbort=TRUE	*Received*
Receiving	Phy_ABORT.req()	Flagabort = TRUE	Receiving
Received	Service=NORMAL & not(Flagabort)	Phy_DATA.ind(Frame) init_timer(TOL)	M.Send
Received	(Service=NORMAL & Flagabort) | Service=UNACKNOWLEDGED	wait_time(TOL)	*T.Session*
Received	Service=ASO & Collision & not(Flagabort)	Phy_COLL.ind() Collision=FALSE	*T.RSO*
Received	Service=ASO & not(Collision) & not(Flagabort)	Phy_ASO.ind(Frame)	*T.RSO*
Received	Service=ASO & Flagabort	$none()	*T.RSO*
T.RSO	(TypeAG=AGT) I (WinRSO>=MaxRSO) & (TypeAG=AGN)	stop_timer(TARSO)	*T.Session*
T.RSO	(WinRSO<MaxRSO) & (TypeAG=AGN)	Index=1 Frame=""	W.RSO
W.RSO	time_out(TARSO)	WinRSO=WinRSO+1 init_timer(TARSO) init_timer(TA1O)	M.Rec
W.RSO	Phy_ABORT.req()	Flagabort=TRUE	W.RSO

Table 31 – Power supply management state transitions (only for non-energized Secondary Station)

Initial State	Triggering condition	Set of actions	Final state
Initial	alarm_detection()	Flagalarm=TRUE FlagSendAlarm =FALSE station_power(ON) Set_baudrate(1200) MaxIndex=128 TOL=100ms TA10=120ms	Stopped
Initial	not(alarm_detection())	Flagalarm=FALSE Set_baudrate(1200) MaxIndex=128 TOL=100ms TA10=120ms	Stopped
Stopped	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB2
Stopped	occur(data_carrier_on)	init_timer(TOSEUIL) init_timer(TAGT)	W.TOSEUIL
W.TOSEUIL	time_out(TOSEUIL)& not(Flagalarm)	station_power(ON)	W.AGT
W.TOSEUIL	occur(data_carrier_off) & not(Flagalarm)	stop_timer(TOSEUIL) stop_timer(TAGT)	*Initial*
W.TOSEUIL	time_out(TOSEUIL) & Flagalarm	station_signal(ON) Tend = TOAG	W.AGT
W.TOSEUIL	occur(data_carrier_off) & Flagalarm	stop_timer(TOSEUIL) stop_timer(TAGT) Tend = TOAGN init_timer(Tend)	Hide
W.TOSEUIL	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB1
W.AGT	occur(data_carrier_off)	stop_timer(TAGT) init_timer(TOAPPEL)	W.Sel
W.AGT	time_out(TAGT) & not(Flagalarm)	station_power(OFF)	*Initial*
W.AGT	time_out(TAGT) & Flagalarm	init_timer(Tend)	Hide
W.Sel	occur(octet_received_event)	stop_timer(TOAPPEL) init_timer(TOBAVARD) init_timer(TAO)	Select
W.Sel	time_out(TOAPPEL) & not(Flagalarm)	station_power(OFF)	*Initial*
W.Sel	time_out(TOAPPEL) & Flagalarm	station_signal(OFF)	*Initial*
W.Sel	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Select	occur(octet_received_event)	stop_timer(TAO) init_timer(TAO)	Select
Select	time_out(TAO)	stop_timer(TOBAVARD) init_timer(TOPRE)	W.Answer
Select	time_out(TOBAVARD) & not(Flagalarm)	stop_timer(TAO) station_power(OFF)	*Initial*
Select	time_out(TOBAVARD) & Flagalarm	stop_timer(TAO) init_timer(Tend)	Hide
W.Answer	occur(octet_sent_event)	stop_timer(TOPRE) init_timer(Tend)	Hide
W.Answer	time_out(TOPRE) & not(Flagalarm)	station_power(OFF)	*Initial*
W.Answer	time_out(TOPRE) & Flagalarm	init_timer(Tend)	Hide
W.Answer	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Hide	occur(octet_received_event) | occur(octet_sent_event) I (occur(data_carrier_on) & not(FlagSendAlarm))	stop_timer(Tend) init_timer(Tend)	Hide

Initial State	Triggering condition	Set of actions	Final state
Hide	occur(data_carrier_on) & FlagSendAlarm	stop_timer(Tend)	W.AGend
Hide	time_out(Tend) & not(Flagalarm)	station_power(OFF)	*Initial*
Hide	time_out(Tend) & Flagalarm & not(FlagSendAlarm)	station_signal(OFF)	*Initial*
Hide	time_out(Tend) & Flagalarm & FlagSendAlarm	Send_AG(AGN)	W.AB
Hide	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
W.AGend	occur(data_carrier_off)	wait_time(TOALR) Send_AG(AGN)	W.AB
W.TVASB1	occur(cpt_carrier_off)	stop_timer(TVASB) init_timer(Tend)	Hide
W.TVASB1	time_out(TVASB)	FlagSendAlarm = TRUE init_timer(Tend)	Hide
W.TVASB1	time_out(Tend)	$none()	W.TVASB2
W.TVASB2	occur(cpt_carrier_off)	stop_timer(TVASB) station_signal(OFF)	*Initial*
W.TVASB2	occur(data_carrier_on)	$none()	W.TVASB1
W.TVASB2	time_out(TVASB)	Send_AG(AGN)	W.AB
W.AB	AG_sent_event	FlagSendAlarm = FALSE station_signal(OFF)	*Initial*

Table 32 – *E/COSEM Physical* State transitions: Secondary Station

Initial state	Triggering condition	Set of actions	Final state
Initial	energized()	MaxIndex=128 FlagRSO=FALSE FirstWinRSO=FALSE Set_baudrate(1200) MaxIndex=128 TOL=100ms TA10=160ms	Stopped
Initial	not(energized())	MaxIndex=18 FlagRSO=FALSE FirstWinRSO=TRUE Set_baudrate(1200) TOL=100ms TA10=120ms	Stopped
Stopped	AG_received_event	Stop_timer(TOAG) init_timer(TA1O)	M.Rec
Stopped	Phy_ALARM.req()	TypeAG=ASB Send_AG(TypeAG)	W.ASB
Stopped	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	*Initial*
M.Rec	octet_received_event	Stop_timer(TA1O) Index=2 Frame="" Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	Phy_ABORT(EP-1)	*WTOAG*
M.Rec	Phy_ABORT.req()	Stop_timer(TA1O)	*WTOAG*

Initial state	Triggering condition	Set of actions	Final state
M.Rec	PHY_SETUP(params)	set_baudrate(new baudrate) MaxIndex=255 TOL=2000ms TA10=2100ms	M.Rec
Receiving	octet_received_event & Index<=MaxIndex	Stop_timer(TAO) Index=Index+1 Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Stop_timer(TAO) Phy_ABORT.ind(EP-4F)	*WTOAG*
Receiving	time_out(TAO)	Phy_DATA.ind(Frame) Init_timer(TOL)	M.Send
Receiving	Phy_ABORT.req()	Stop_timer(TAO)	*WTOAG*
M.Send	Phy_DATA.req(Frame)	Stop_timer(TOL) Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 Init_timer(TOE)	Sending
M.Send	Phy_RSO.req(Frame, Window)	Stop_timer(TOL) Wait_window(FirstWinRSO, Window) FirstWinRSO=FALSE Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 FlagRSO=TRUE Init_timer(TOE)	Sending
M.Send	time_out(TOL)	Init_timer(TA1O)	M.Rec
M.Send	Phy_ABORT.req()	Stop_timer(TOL)	*WTOAG*
Sending	octet_sent_event & Size>0	Index=Index+1 Send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0 & not(FlagRSO)	Stop_timer(TOE) init_timer(TA1O)	M.Rec
Sending	octet_sent_event & Size=0 & FlagRSO	Stop_timer(TOE) Wait_time(TAO) FlagRSO=FALSE	*WTOAG*
Sending	Phy_ABORT.req()	Stop_timer(TOE)	*WTOAG*
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F)	*WTOAG*
W.ASB	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	W.ASB
W.ASB	AG_sent_event	$none()	*Initial*
WTOAG	Not(energized)	init_timer(TOAG)	*Initial*
WTOAG	energized	$none()	*Initial*

Table 33 – Meaning of the states listed in the previous tables

State	Definition
Initial	Initialisation of the variables of the layer
Stopped	Waiting for a 'Wakeup Call' signal
W.ETABS (Wait for end of " Alarm-Bus ")	Waiting for the end of an 'Alarm Bus' signal, received in a Stopped state
W.AG (Wait for end of " Wakeup Call ")	Waiting for the end of a 'Wakeup Call' signal transmission.
W.TAB (Wait " Alarm-Bus ")	Waiting for an alarm bus signal during safety delay at the end of transmission of a 'wakeup Call' signal
W.ETAB (Wait for end of " Alarm-Bus ")	Waiting for an end of an 'Alarm Bus' signal received after the transmission of a 'Wakeup Call' signal.
W.TASB	Waiting for the triggering of wakeup TASB after the begin of the reception of an "Alarm-Bus" signal
W.TOL	Waiting for the triggering of wakeup TOL
M.Send (Must Send)	Initial state of the transmitter waiting for a frame to send
T.Session	Testing the type of the session (with an energized or not energized Secondary Station)
SendFirst	Sending the first byte of the frame to be sent
Sending	Recurrent state of the transmitter transmitting one byte at a time
Answer	Branching depending on the service requested
M.Rec (Must Receive)	Initial state of the receiver waiting for the first byte of a frame
Receiving	Recurrent state of the receiver receiving one byte at a time
Received	Processing the received frame
T.RSO (Test last RSO)	Testing the end of the last time slot for RSO frame reception
W.RSO (Wait for end of an RSO time slot)	Waiting for the end of a time slot for RSO frame reception
W.ASB	Waiting for the end of a "Alarm Secondary-Bus" signal transmission
W.TOAG	Initializing the "end of session" TOAG timer if needed
W.TOSEUIL	Waiting for the triggering of wakeup TOSEUIL
W.AGT	Waiting for an AGT "Wakeup Call" signal
W.Sel (Wait for preSelection)	Waiting for a preselection frame
Select	Receiving a preselection frame
W.Answer	Waiting for an answer frame from a selected station
Hide	Waiting for the end of selection
W.Agend	Waiting for the end of AG reception
W.TVASB1	Waiting for the triggering of wakeup TVASB for an "Alarm Secondary-Bus" signal during a session
W.TVASB2	Waiting for the triggering of wakeup TVASB for an "Alarm Secondary-Bus" signal at the end of session
W.AB	Waiting for the end of a "Alarm Bus" signal transmission

Table 34 – Definition of the procedures, functions and events classified in alphabetical order

Procedure, function or event	Definition
AG_received_event	Event from the modem reporting that an AGN “Wakeup Call” signal has been correctly detected
AG_sent_event	Event from the modem reporting the end of the transmission of a “Wakeup Call” signal
alarm_detection()	Check that the station status of alarm mode is Active
collision_detected_event	Event from the modem reporting the detection of a framing error on reception of a byte
concat(Frame, RecB)	Concatenation of the byte RecB in the being built frame Frame
data_carrier_on, data_carrier_off	Occurrence of the detection on the bus of the data carrier on, the data carrier off
energized()	Check that the station is energized
init_timer(TOAPPEL), init_timer(TOSEUIL), init_timer(TAGT), init_timer(TOBAVARD), init_timer(TOPRE), init_timer(TOL), init_timer(TOE), init_timer(TAO), init_timer(TA1O), init_timer(TARSO) init_timer(TOAG), init_timer(TVASB) or init_timer(TAB)	setting of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB or TAB
occur(cpt_carrier_on), occur(cpt_carrier_off), occur(data_carrier_on), occur(data_carrier_off), occur(octet_received_event) or occur(octet_sent_event)	Occurrence (reporting without consuming) of the detection on the secondary bus of the data carrier on, the data carrier off, on the bus of the data carrier on, the data carrier off, the reception of a byte or the emission of a byte
octet_received_event	Event from the modem reporting that a byte has been received
octet_sent_event	Event from the modem reporting that a byte has been sent
read_data(RecB)	Processing of the byte_received_event event by reading the received RecB byte (bits are transmitted in ascending order)
send_AG(TypeAG)	Request to the modem for transmission of a “Wakeup Call” signal of duration TypeAG (AGN or AGT)
send_octet(Frame, Index)	Transmission of the byte of rank Index in the frame Frame (bits are transmitted in ascending order)
Setup_params(baurate, TOL, TA10)	Setup of the parameters baudrate, TOL and TA10.
size(Frame)	Calculation of the number of bytes of the frame Frame
station_power(ON) or station_power(OFF)	Turning ON or OFF the energy supply to the device
station_signal(ON) or station_signal(OFF)	Turning ON or OFF the signal transmission to the device on the secondary bus
stop_timer(TOAPPEL), stop_timer(TOSEUIL), stop_timer(TAGT), stop_timer(TOBAVARD), stop_timer(TOPRE), stop_timer(TOL), stop_timer(TOE), stop_timer(TAO), stop_timer(TA1O), stop_timer(TVASB) or stop_timer(TAB)	Stopping of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO,TA1O, TVASB or TAB
stop_timer(TOAG) or stop_timer(TARSO)	Stopping of wakeup TOAG or TARSO only if it has previously been set

Procedure, function or event	Definition
time_out(TOAPPEL), time_out(TOSEUIL), time_out(TAGT), time_out(TOBAVARD), time_out(TOPRE), time_out(TOL), time_out(TOE), time_out(TAO), time_out(TA1O), time_out(TARSO), time_out(TOAG), time_out(TVASB) or stop_timer(TAB)	Triggering of wakeup TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB or TAB
wait_time(TAO), wait_time(TICB), wait_time(TOL) or wait_time(TOALR)	Calculated delay during time TAO, TICB, TOL or TOALR
wait_window(FirstWinRSO, Window)	Wait-time calculated as follows: when FirstWinRSO=TRUE or Window=0 ==> 0 ms when FirstWinRSO=FALSE and Window>0 ==> 40 ms + (TARSO*Window) ms (The 40 ms delay guarantees that the transmission has taken place in the time slot)

Table 35 – Error summary table

EP-1	Expiry of TOL wakeup (Primary Station) before the Data Link layer requests a frame transmission or expiry of TA1O wakeup (Secondary Station) before any character has been received from the Primary station
	This error leads to the expectation of a “Wakeup Call” signal after having informed the Data Link layer
EP-2	Expiry of TOAG wakeup before any “Wakeup Call” signal
	This error leads to the expectation of a “Wakeup Call” signal after having informed the Data Link layer
EP-3	An alarm has been received
	This error leads to the reinitialization of the Physical layer after having informed the Data Link layer
EP-3F	Abnormal length of transmission detected after expiry of TOE wakeup
	This error leads to the reinitialization of the Physical layer after having informed the Data Link layer
EP-4F	Number of bytes received higher than MaxIndex (Transmitter too talkative)
	This error leads to the reinitialization of the Physical layer after having informed the Data Link layer
EP-5F	Expiry of TARSO wakeup while receiving an RSO frame (Primary Station only)
	This error leads to the reinitialization of the Physical layer after having informed the Data Link layer

If any of these errors occurs, it is sent up locally by means of the Phy_ABORT.ind service primitive. The complete list of fatal error numbers is given in Annex C.

7.3 Data Link layer

7.3.1 General

The *Data Link* Layer used for local bus data exchange with DLMS/COSEM is based on the same principle as the *Data Link E/D* layer used for data exchange with DLMS. The only difference is the interaction with the upper layers and processing of the speed negotiation.

At the upper level, the *Data link* layer interfaces the transport layer and the communication *Support Manager* layer.

7.3.2 Identification of data units

The Data Link layer delivers all PDUs identified as DATA+ to the DLMS/COSEM Transport layer. All other PDUs are directed towards the Support Manager layer, which recognises and processes known PDUs; unknown PDUs are simply ignored. See Table 41 for the Support Manager commands.

7.3.3 Role of the Data Link layer

The purpose of the *Data Link* layer is to:

- carry out serialization and deserialization of the data;
- synchronise the transmission and reception frames;
- filter the frames according to primary and secondary addresses;
- ensure efficient protection against transmission errors.

7.3.4 Management of exchanges

At the Transmitter end, all data frames that are sent shall receive a positive acknowledgement from the Receiver station before sending the next data frame. After sending a frame and receiving the acknowledgement of the previously sent frame, the current frame is transmitted. The number of repetitions is limited to MaxRetry. Above this number, the communication is stopped at the Data Link level and a notification is sent to the Transport layer and Manager Support layer.

Each time a frame is received, an answer frame is transmitted within a delay compatible with the TOL managed by the Physical layer. If the transport layer does not have any data available to transmit, *a* DSDU with a Text empty field is sent, notifying the acknowledgement or non-acknowledgement of the received DPDU.

The management principle of acknowledgement / non-acknowledgement is identical to that specified in 6.2.2.

7.3.5 Data Link services and service primitives

Table 36 – Data Link services and service primitives

Service	Service primitive
DL_DATA	DL_DATA.req(Pr ,Service class, DSDU) DL_DATA.ind(Pr,Service class, DSDU)
DL_ALARM	DL_ALARM.req() DL_ALARM.ind()
DL_ABORT	DL_ABORT.req(Strong) DL_ABORT.ind(ErrorNb)
DL_IB	DL_IB.req() DL_IB.ind()
DL_Discover	DL_Discover.req() DL_Discover.ind()
DL_ChangeBaudrate	DL_ChangeBaudrate.req() DL_ChangeBaudrate.ind()
DL_PhysicalSetupParameters	DL_PhysicalSetup.req(params)

The role assigned to each primitive is as follows:

DL_DATA.req(Pr, Sevice class, DSDU) enables the upper layer (*Transport* or *Support Manager*) to request the *Data Link* layer to transfer a DSDU data packet with the priority Pr[5], with service class CONFIRMED or UNCONFIRMED. In the request, the service class parameter only concerns the primary station. As long as the service class is UNCONFIRMED, the *Data Link* layer of the primary station shall use a broadcast destination address;

DL_DATA.ind(Pr, Service class, DSDU) enables the *Data Link* layer to inform the upper layer of the arrival of a DSDU data packet with the priority Pr. In the indication, the service class parameter only concerns the secondary station. As long as the destination address is a broadcast address, the secondary station data link layer shall set the Service_class parameter to UNCONFIRMED for the upper layer;

DL_ALARM.req()enables the *Support Manager* layer of the secondary station to request the *Data Link* layer to send an alarm;

DL_ALARM.ind()enables the *Data Link* layer of the primary station to warn the *Support Manager* layer of the presence of an Alarm;

DL_ABORT.req(Strong[6])enables the upper layers to request the *Data Link* layer to end its activity with the priority Strong;

DL_ABORT.ind(ErrorNb) enables the *Data Link* layer to inform the *Support Manager* layer of the occurrence of a fatal error identified by the number ErrorNb;

DL_IB.req() enables the *Support Manager* layer of the primary station to request the *Data Link layer to* initialise the bus;

DL_IB.ind() enables the secondary station *Data Link* layer to inform the *Support Manager* layer of the presence of an initialisation of the bus;

DL_Discover.req() enables the *Support Manager* layer to request the *Data Link* layer to send a call to forgotten stations frame;

DL_Discover.ind() enables the *Data Link* layer to inform the *Support Manager* layer of the presence of a call to forgotten stations frame;

DL_ChangeBaudrate.req() enables the *Support Manager* layer to request the *Data Link* layer to send a speed negotiation frame;

DL_ChangeBaudrate.ind() enables the *Data Link* layer to inform the *Support* Manager layer of the presence of a speed negotiation frame;

DL_PhysicalSetupParameters.req(Params) enables the *Support Manager* layer to request the *Data Link* layer to carry out the modification of the settings related to a change of speed. These settings consist of the negotiated baudrate, the Timeout TOL that has passed 1 second, the timeout TA10 of 1 100 ms and MaxIndex which increases from 128 to 255 bytes.

---

5 The priority level Pr differentiates the processing of emergency services such as InformationReport (level Pr=1) from that of the other DLMS services (level Pr=0).

6 The Strong parameter differentiates the processing of fatal errors (Strong=1) from that of the other physical disconnection requests (Strong=0) initialized by the *Application* sub-layer.

7.3.6 Data Link parameters

The Data Link parameters are the same as for the Data Link layer in the profile with DLMS, see 6.2.4.

At the end of the baud rate change process, at the primary device side the Data Link layer manages a timeout TES (time out end of Setup) of 50 ms before processing all new requests destined for the secondary device. This is to allow the equipment to correctly set up the UART.

7.3.7 State transitions

Table 37 – *DLMS/COSEM Data Link E/D* state transitions: Primary Station

Initial State	Triggering Conditions	Set of actions	Final State
Initial	$true()	MaxRetry=2 MaxChain = 5 init_incrChain()	Stopped
Stopped	exist_dl_req()	NbChain = 0 MaxIndex=0 RepeatASO=FALSE context(ADP, ADS, TypeAG) init(TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain = 0	$none()	*T.Req*
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain <> 0	NbChain = 0	*M.Send*
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
W.EndS	(Phy_ABORT.ind(EP-2) & TypeAG=AGT) I (Phy_ABORT.ind(EP-1) & TypeAG=AGN)	$none()	Stopped
W.EndS	Phy_ALARM.ind	DL_ALARM.ind()	Stopped
W.EndS	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS
T.Req	exist_dl_req(DL_IB.req())	Fr="" Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, IB, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	exist_dl_req(DL_Discover.req(DSDU)) & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO

Initial State	Triggering Conditions	Set of actions	Final State
T.Req	exist_dl_req(DL_Discover.req(DSDU)) & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	not (exist_dl_req(DL_IB.req()) | exist_dl_req(DL_Discover.req(_)))	Pr=0 Send="00"B Confirm="11"B Fr="" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
T.Req	exist_dl_req(DL_XBR.req(proposed_baudrate)	Fr="proposed_baudrate" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, XBR, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & not(check_frame(Frame)) & size(Frame)<>0	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DL_Discover.ind (Collision, ListRSO)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, Service class DSDU)) & ((& not(TreqTimout))) NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec

Initial State	Triggering Conditions	Set of actions	Final State
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, Service class DSDU)) & ((& not(TreqTimout))) NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec
M.Send	not(DL_DATA.req(_, _)) & TreqTimeout()& NbChain < MaxChain	Pr=0 Fr="" Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) initTimer(Treq)	M.Rec
M.Send	NbChain >= MaxChain	Phy_APPG.req(AGN)	W.AG
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	DL_DATA.ind(extract_prty(Frame), extract_text(Frame)) Confirm=incr(Confirm) Ack_expected=FALSE Index = 0 initTimer(Treq)	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	Ack_expected=FALSE Index = 0 initTimer(Treq)	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	DL_ABORT.ind(EL-2F) Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=1)	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & (check_address(Frame) & (Com =XBA)	DL_ChangeBaudrate.ind(accepted_baurate).	M.Rec
M.Rec	exist_dl_physical_setup_req(params)	Phy_SETUP(params) wait(TES)	T.Req

Table 38 – *DLMS/COSEM Link E/D* state transitions: Secondary Station

Initial State	*Triggering conditions*	Set of actions	Final State
Initial	*$true()*	MaxRetry=2 FlagDSO=TRUE Discovered=FALSE Flag_alarm=FALSE	Stopped
Stopped	*Phy_DATA.ind(Frame) & check_frame(Frame) & check_address(Frame)*	ADP=extract_ADP(Frame) Com=command(Frame)	*T.Com*
Stopped	*Phy_DATA.ind(Frame) & check_frame(Frame) & not(check_address(Frame))*	Phy_ABORT.req()	Stopped
Stopped	*Phy_DATA.ind(Frame) & not(check_frame(Frame))*	$none()	Stopped
Stopped	*DL_Alarm.req()*	Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
T.Com	*Com=IB*	Discovered=FALSE DL_IB.ind() Phy_ABORT.req()	Stopped
T.Com	*Com=ASO & test_TABi(Frame, TAB)*	DL_Discover.ind(TAB)	W.MM
T.Com	*Com=XBR*	DL_ChangeBaudrate.ind(proposed_baudrate)	W.MM
T.Com	*is_data+(Frame) & is_text(Frame)*	Ack_expected=FALSE Send="11"B Confirm="00"B DL_DATA.ind(extract_prty, extract_text(Frame)) initTimer(Treq)	*M.Send*
T.Com	*is_data+(Frame) & not(is_text(Frame))*	Ack_expected=FALSE Send="11"B Confirm="00"B initTimer(Treq)	*M.Send*
W.MM	*exist_dl_discover_req(TAB)*	Fr=concat(RSO, TAB, ADS) Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_RSO.req(Fr, window_RSO())	*stopped*
W.MM	*exist_dl_discover_req(not(TAB))*	Phy_ABORT.req()	*stopped*
W.MM	*exist_dl_change_baud_rate.req(acceptedBaudrate)*	Discovered = TRUE Fr= acceptedBaudrate Index=1 Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, XBA, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	*M.Rec*
M.Send	*exist_dl_data_req(DL_DATA.req(Pr=1, DSDU)) & not(Treqtimeout())*	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec

Initial State	*Triggering conditions*	Set of actions	Final State
M.Send	*not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, DSDU)) & not(Treqtimeout())*	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec
M.Send	*not(DL_DATA.req(_, _))& TreqTimeout()*	Pr=0 Fr="" Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) initTimer(Treq)	M.Rec
M.Rec	*Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)*	Confirm=incr(Confirm) Ack_expected=FALSE DL_DATA.ind(extract_prty, extract_text(Frame)) initTimer(Treq)	*M.Send*
M.Rec	*Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))*	Ack_expected=FALSE initTimer(Treq)	*M.Send*
M.Rec	*Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry*	Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	*Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry*	DL_ABORT.ind(EL-2F) Phy_ABORT.req()	Stopped
M.Rec	*DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE*	Phy_ABORT.req()	Stopped
M.Rec	*DL_ALARM.req() & alarm_detection()*	DL_ABORT.ind(EL_1F) Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
M.Rec	*DL_ABORT.req(Strong=1)*	stop_timer(T1) Phy_ABORT.req()	Stopped
M.Rec	*Phy_ABORT.ind(EP-1)*	DL_ABORT.ind(EL_3F)	M.Rec
M.Rec	*Phy_ABORT.ind(ErrorNb) & ErrorNb <> EP-1*	DL_ABORT.ind(ErrorNb)	Stopped
M.Rec	*exist_dl_physical_setup_req()*	Phy_SETUP(params)	M.Rec

Table 39 – Meaning of the states listed in the previous tables

State	Meaning
Initial	Initialization of the variables of the layer
Stopped	Waiting for the first request from the upper layer or for the first indication from the lower layer.
W.AG (Wait for end of " Wake-up Calll")	Waiting for the end of a “Wakeup Call” signal transmission
W.EndS (Wait for end of Session)	Waiting for the end of a session
T.Req (Test Request)	Testing the nature of a request coming from an upper layer
M.RSO (Must Receive RSO)	Waiting for responses from a call o forgotten stations
T.RSO (Test last RSO)	Testing the end of the last time slot for RSO frame reception
M.Send (Must Send)	Test state of the frame to transmit. (The field Text may be empty)
M.Rec (Must Receive)	Initial state of the receiver waiting for the first byte of a frame
T.Com (Test Command)	Testing the com field of a received frame
W.MM (Wait for Support Manager event)	Waiting to receive an event from the Support Manager layer

Table 40 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
alarm_detection()	Check that Alarm mode is active
build_RSO(ListRSO, Frame)	Extraction of the RSO elements (TAB and ADS fields) from the received RSO frame Frame and concatenation with the preceding list ListRSO
check_address(Frame)	Check that the ADP and ADS addresses are recognized according to the following criteria: – DP is APG or the station has been programmed to the ADP address; – if the command code is ASO, IB or TRB, then ADS is ADG; – if the command code is not ASO, IB nor TRB, then ADS is the secondary station address
check_frame(Frame)	Check that the frame Frame received is correct: – number of bytes greater than or equal to 11 and lower than or equal to MaxIndex; – CRC correct; – number of bytes compatible with the field Size; – command code recognized and number of bytes compatible with the command code
com(DATA+, Pr, Send, Confirm)	Concatenation of corresponding binary fields to obtain a specific command
command(Frame)	Extraction of the value of the command code of a received frame Frame
concat(Size, ADS, ADP, COM, Text), or concat(Frame, CRC)	Concatenation of the fields Size, ADS, ADP, COM and Text or concatenation of the CRC at the end of the frame Frame
context(ADS, ADP, TypeAG)	Extraction of the corresponding values from the communication context
crc(Frame)	Calculation of the CRC of the frame Frame to send
create_alarm(TPDU)	Calculation of a TPDU with STSAP = 0, DTSAP = 0 and an UnsolicitedReqPDU with: client-type = FFFF, serveridentifier = 0 , object-name = FFFF variable type = boolean, value = TRUE
exist_dl_data-req(DL_DATA.req(Pr, DSDU))	Consumption of a DL_DATA.req(Pr, DSDU) event
exist_dl_req()	Check for the existence of a DL_IB.req(), DL_ASO.req(DSDU) or DL_DATA.req(Pr, DSDU) event and compatibility check with ADS and ADP adresses defined in the communication context
exist_dl_req(DL_IB.req()) or exist_dl_req(DL_ASO.req(DSDU))	Consumption of a DL_IB.req or DL_ASO.req(DSDU) event
exist_dl_physical_setup_req	Consumption of a baudrate reconfiguration event.
extract_ADP(Frame)	If either the ADP value used in the frame is not APG or if it is the APG value but the list of the ADP values to which the Secondary Station has been programmed is empty, then extraction of this value, otherwise extraction of the first ADP value to which the Secondary Station has been programmed
extract_prty(Frame)	Extraction of the Priority field from a received frame Frame
extract_text(Frame)	Extraction of the Text field from a received frame Frame
init(TypeAG)	Set Index to MaxRetry if TypeAG equals AGT, otherwise to 0
init_incrChain()	Set IncrChain at 0 if the alarms are not managed, otherwise to 1
init_timer(T1)	Setting of wakeup T1
initTimer(Req)	Initialisation of the waiting for a request from upper layers timer.
is_ack (Frame)	Check that the received frame Frame contains a Confirm field equal to the Send field of the last frame transmitted
is_data+ (Frame)	Check that the received frame Frame contains a correct DATA+ field("111"B)
is_text(Frame)	Check that the received frame Frame contains a non-empty text field and that the Send field equals the 1' complement of theConfirm field of the last frame transmitted

Procedure or function	Definition
size(Frame)	Calculation of the size of the frame Frame received
size_frame(DSDU)	Calculation of the size of the frame to build with the DSDU data unit (size(DSDU) + 11)
stop_timer(T1)	Stopping of wakeup T1
test_TABi(Frame, TAB)	If the first TABi contained in the received ASO frame Frame equals 00, check that Discovered=FALSE then, after provision of a whole random integer between 0 and 100, check that this integer is smaller than the response probability (second TABi). In this case, 00 is memorized in the TAB variable If the first TABi contained in the received ASO frame Frame equals FF, check that Flag_alarm=TRUE then, in this case, Flag_alarm is set to FALSE and FF is memorized in the TAB variable; If the first TABi contained in the received ASO frame Frame does not equal 00 or FF, check that FlagDSO=TRUE and check that the Secondary Station has been programmed to one of the TABi contained in the received ASO frame Frame. In this case, the first of these values is memorized in the TAB variable
time_out(T1)	Triggering of wakeup T1
TreqTimeout()	Verification that the waiting time of a request from upper layers does not exceed a time t so that the time out duration TOL managed by the physical layer might happen eventually. This waiting time is necessarily less than TOL and is armed once the indication is sent to the upper layer
window_RSO()	Provision of a whole random integer between 0 and MaxRSO-1 used as the number of the RSO time slot in which the station shall reply (refer to Annex F)

7.4 Support Manager layer

7.4.1 Overview

The Support Manager layer processes all services related to communication support. These services are:

- initialisation of the bus;
- discovery management;
- alarm management;
- speed negotiation;

These services are managed in such a way that they are consistent with their management under profiles with or without DLMS. For these services, the identifiers have the following values:

Table 41 – Commands managed by the Support Manager layer

Identifier	Hexadecimal Value	Role	P/S
ASO	07	Discovery request	Primary Station
RSO	08	Discovery request response	Secondary Station
IB	09	Initialisation of the bus	Primary Station
XBR	0x12	Change Baudrate Request	Primary Station
XBA	0x13	Change Baudrate Answer	Secondary Station

7.4.2 Initialisation of the bus

Management complies with 4.4.6. Although the initialisation is managed by the Support manager layer, processing is entirely carried out by the Data link layer.

7.4.3 Discover service

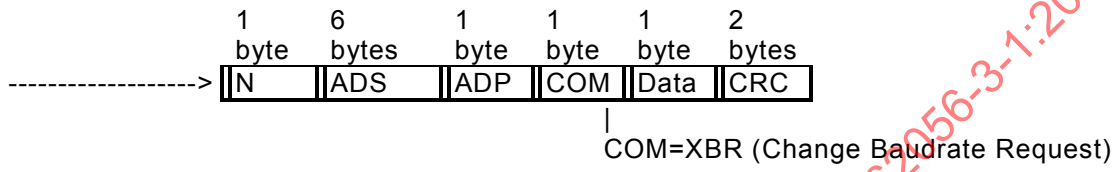
Management of the Discover service complies with 4.4.7 and Annex H. The TAB field contains 2 parameters of one byte each: the first has a value of 0, the value at which all equipment is programmed. The second is the probability of a response.

7.4.4 Speed negotiation

Once the connection at data link level is established, the Support Manager layers can negotiate the communication speed. This negotiation is always initiated by the primary device.

Authorised speeds are 1 200, 2 400, 4 800 and 9 600 bauds.

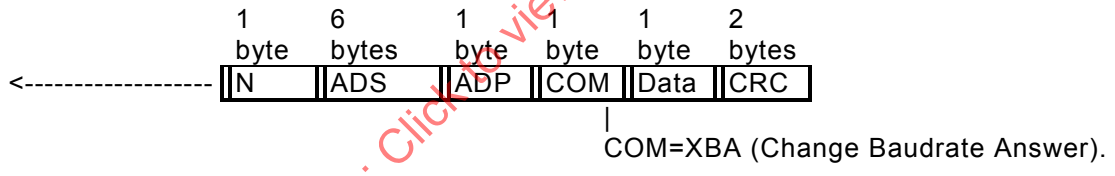
- Communication Speed Negotiation Frame



The Data field contains the speed at which the primary equipment wants to communicate. The values are as follows:

- 0x00 1 200 bauds,
- 0x01 2 400 bauds,
- 0x02 4 800 bauds,
- 0x03 9 600 bauds.

- Secondary station response frame:



The Data field contains the speed selected by the secondary station.

The primary station observes a TES delay of 50 ms at the end of the process in order to allow the two stations present to adjust the connected devices to the selected speed.

After a successful change of speed, the following parameters are modified:

Table 42 – List of parameters

Parameter	Old value	New value
Baudrate	1 200 baud	Value Negotiated
MaxIndex	128	255
TOL	100 ms	1 000 ms
TA10	160 ms	1 100 ms

7.4.5 Support Manager parameters

For a primary station, the value of the maximum number of RSO timeslots for the processing of a ‘discover’ frame is set to 3.

7.4.6 State transitions

Table 43 – Support Manager layer state transitions: Primary Station

Initial State	Triggering Conditions	Set of actions	Final State
Stopped	Exist_req(discover)	TABi=0 DL_Discover.req()	M.Rec
Stopped	Exist_req(change_baud_rate)	Baudrate = proposed_baudrate DL_ChangeBaudrate.req()	M.Rec
Stopped	DL_abort.ind(ErrorNb)	T Abort.ind() Update(FatalError)	Stopped
Stopped	Exist_req(init_bus)	DL_InitBus_req()	Stopped
Stopped	Exist_req(abort)	DL_Abort.req()	Stopped
Stopped	DL_alarm.ind()	MM alarm.ind()	Stopped
M.Rec	Exist_cnf(discover)	MM discover.cnf(disc list)	Stopped
M.Rec	Exist_cnd(change_baud_rate_OK)	Baud_rate = accepted_baudrate DL_PhysicalSetup.req(accepted_baudrate)	Stopped
M.Rec	Exist_cnf(change_baud_rate_NOK)	$none	Stopped

Table 44 – Support Manager layer state transitions: Secondary Station

Initial State	Triggering Conditions	Set of actions	Final State
Stopped	exist_ind(DL_IB_ind)	MM_IB.ind()	Stopped
Stopped	exist_ind(discover) & test_TABi(TAB)	MM_Discover.cnf()	Stopped
Stopped	exist_ind(discover) & not(test_TABi(TAB)	MM_Discover.cnf(not_TABi)	Stopped
Stopped	exist_ind(dl_change_baud_rate) & check(proposed_baudrate) & not(energized_station)	accepted_baudrate =min(received_baudrate, programmed_baudrate) DL_SetupBaudrate.req(accepted_baudrate) Init_timer(TWS)	Sending
Stopped	exist_ind(dl_change_baud_rate) & not(check(received_baudrate))	DL_ABORT.req(Strong) T_ABORT.ind()	Stopped
Stopped	DL_abort.ind(ErrorNb)	T_ABORT.ind() Update(FatalError)	Stopped
Stopped	Alarm_detection() & DL_ALARM.req()	DL_ALARM.req()	Stopped
Sending	time_out(TWS)	DL_PhysicalSetup.req(accepted_baudrate)	Stopped

Table 45 – Meaning of the states listed in the previous table

State	Meaning
Stopped	Waiting state for a request or the first indication from the lower layer.
M.Rec	Waiting state for a response following the sending of a request
Sending	Waiting for the end of transmission from lower layers; This wait is limited by a timeout TWS with a value of 150 ms.

Table 46 – Definition of procedures, functions and events

State	Meaning
Alarm_detection()	Check that alarm mode is active
exist_ind(indication)	Check for the presence of a change of baudrate, IB or Discover indicator.
exist_ind(request)	Check for the presence of a baudrate change, IB or Discover request
Init_timer(TWS)	Initialisation of the TWS timer from the moment that the speed change request from the secondary station is sent by the lower layers.
check(received_baudrate)	Check that the received baudrate falls within the acceptable range.
min(received_baudrate, programmed_baudrate)	Determination of the accepted speed. This is the smallest value between the proposed value and the maximum at which the secondary equipment can function.
Test TABI(TAB)	If the first of the ASO TABi contained in the frame Frame has a value of 00, check that the Discovered variable is FALSE and verify, after generating a random number between 1 and 100, that this number is less than the probability of the response expected in the second TABi. In this case the value 00 is memorised in the TAB variable. If the first of the ASO TABi contained in the frame Frame has a value of FF, check that the Flag_alarm variable is TRUE, and in this case, allocation of the value False to the variable Flag_alarm and memorisation of the value FF in the TAB variable. All other values of TAB are not valid.

7.5 Transport Layer

7.5.1 General

The Transport layer is totally identical at the primary and secondary station. Its role is to ensure the fragmentation and reassembly of the application protocol data units.

In the transmission, the transport layer receives the application data units, fragments them into many transport data units as necessary in order to allow the Data Link layer to send them to the transport layer of the peer station.

In reception, Transport Data Units received by the Transport layer, when their size exceeds the link layer payload, are fragmented. The receiving Transport entity makes sure that the Transport units being received are complete before closing the reception.

The end of the reception is indicated by the End field to TRUE.

7.5.2 Transport Data Units

The structure of the first transport data unit is

3 bits	1 bit	1 bit	3 bits	8 bits	8 bits	N bytes
DSap	End	First	Reserved	STSAP	DTSAP	Payload

The structure of the Data units that follow is:

3 bits	1 bit	1 bit	3 bits	N bytes
DSap	End	First	Reserved	Payload

- The Dsap field comprises 3 bits. It identifies the data units under DLMS/COSEM. It has a value of 6;
- The End field indicates if the transport unit is the last. As long as the Application data unit is fragmented and the current transport data unit is not the last, its value stays as FALSE;
- The First field indicates if the Transport unit is the first or not. Whilst the first field is TRUE, the TPDU shall contain the source and destination adresses. All the transport units that follow are relative to the connection defined by STSAP and DTSAP carried on the first data unit, until the arrival of the transport unit with the End field. As long as the TPDU is the first and last, the two fields, End and first are both TRUE. A TPDU that does not contain a STSAP and DTSAP cannot have the field First as TRUE;
- The Reserved field comprises 3 bits which shall always be at 0;
- The STSAP field of 8 bits contains the address of the application layer data source;
- The DTSAP field of 8 bits contains the address of the application layer data destination;
- The payload field contains the application data. The first TSDU has a maximum size of 241 bytes and 243 for those that follow. Until the speed negotiation has taken place, the sizes are limited to 114 and 116 bytes respectively.

Table 47 – Transport services and services primitive

Service	Service primitive
T_DATA	T_DATA.req(STSAP, DTSAP, Pr, Service class, TSDU) T_DATA.ind(STSAP, DTSAP, Service class, TSDU)
T_ABORT	T_ABORT.req() T_ABORT.ind()

- The primitive T_DATA.req(STSAP, DTSAP, Pr, Service class, TSDU) enables the *Application* layer to request the *Transport* layer to send a transport data unit from a source address STSAP to a destination address DTSAP with a priority Pr., using a confimed or unconfirmed service;
- The primitive T_DATA.ind(STSAP, DTSAP, Pr, Service class, TSDU) enables the *Transport* layer to inform the *Application* layer of the arrival of a transport data unit from a source address STSAP at a destination address DTSAP, transported with a confirmed or unconfirmed service;
- The primitive T_ABORT.req enables the *Application* layer to request the *Transport* layer to end its activity;
- The primitive T_ABORT.ind enables the *Transport* layer to inform the *Application* layer of an error necessitating an end to communication.

7.5.3 State transitions

Table 48 – Transport state transitions

Initial State	Triggering conditions	Set of actions	Final State
stopped	$true	Init()	Idle
idle	T_DATA.req(STsap, Dtsap, TSDU, Service_class,)	SMsg = TSDU	M.FirstFgt
Idle	DL_DATA.ind(Pr, TPDU) & check_fgt(TPDU) & first_fgt(TPDU) & not(last_fgt(TPDU))	Tsap(TDPU, STsap, DTsap) initBuffer() updateBufferRec(TPDU)	Rec
Idle	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & fisrt_fgt(TPDU) & last_sgt(TPDU) & isSizeOK ()	Tsap(TDPU, STsap, DTsap) initBuffer() updateBufferRec(TPDU) T_DATA.ind(TPDU , STsap, DTsap)	Idle
Idle	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & fisrt_fgt(TPDU) & last_sgt(TPDU) & not(isSizeOK())	Tsap(TDPU, STsap, DTsap) initBuffer() T_DATA.ind(TPDU , STsap, DTsap Service_class,)	Idle
Idle	DL_DATA.ind(Pr, Service_class, TPDU) & not(check_fgt(TPDU)	DL_ABORT.req() T_ABORT.ind()	*stopped*
Idle	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & not(fisrt_fgt(TPDU)) & last_sgt(TPDU))	initBuffer() T_DATA.ind(TPDU , STsap, DTsap Service_class,)	Idle
Idle	T_ABORT.req()	DL_ABORT.req()	*stopped*
Idle	DL_ABORT.ind(ErrorNb)	T_ABORT.ind()	*stopped*
M. FirstFgt	Size(SMsg)>FirstPayload()	End=0 First=1 DL_DATA.req(DSap,End, STsap, DTsap, Service_class, FirstPayload()) update(SMsg)	M.NextFgt
M. FirstFgt	Size(SMsg)<=FirstPayload()	End=1 First=1 DL_DATA.req(DSap,End, STsap, DTsap, Service_class, FirstPayload())	Idle
M.FirstFgt	DL_ABORT.ind(ErrorNb)	T_ABORT.ind()	*stopped*
M.NextFgt	Size(SMsg)>NextPayload()	End=0 First=0 DL_DATA.req(DSap,End, STsap, Service_class, min(SMsg, NextPayload())) update(SMsg)	M.NextFgt
M.NextFgt	Size(SMsg)<=NextPayload()	End=1 First=0 DL_DATA.req(DSap,End, STsap, Service_class, NextPayload())	Idle
M.NextFgt	DL_ABORT.ind(ErrorNb)	T_ABORT.ind()	*stopped*
Rec	DL_DATA.ind(Pr, Service_class, TPDU) &	initBuffer()	Idle

Initial State	Triggering conditions	Set of actions	Final State
	not(check_fgt(TPDU))	T_DATA.ind(TPDU , STsap, DTsap, Service_class,)	
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & not(fisrt_fgt(TPDU)) & not(last_fgt(TPDU))) & isSizeOK ()	Tsdu(TDPU) updateBufferRec(TPDU) DL_DATA.req(DSap,End, STsap, DTsap, Service_class, text=NULL)	Rec
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & not(fisrt_fgt(TPDU)) & not(last_fgt(TPDU)) & not(isSizeOK ())	Tsap(TDPU, STsap, DTsap) InitBuffer () T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & not(fisrt_fgt(TPDU)) & last_fg(TPDU)) & isSizeOK	Tsdu(TDPU) updateBufferRec (TPDU) T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & not(fisrt_fgt(TPDU)) & last_fg(TPDU) & not(isSizeOK ())	Tsap(TDPU, STsap, DTsap) InitBuffer () T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & first_fgt(TPDU) & not(last_fgt(TPDU)))) & isSizeOK ()	Tsap(TDPU, STsap, DTsap) InitBuffer () updateBufferRec(TPDU) DL_DATA.req(DSap,End, STsap, DTsap, Service_class, text=NULL)	Rec
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & first_fgt(TPDU) & not(last_fgt(TPDU)))) & not(isSizeOK ())	Tsap(TDPU, STsap, DTsap) InitBuffer () T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & first_fgt(TPDU) & last_fgt(TPDU) & isSizeOK ()	Tsap(TDPU, STsap, DTsap) InitBuffer () updateBufferRec(TPDU) T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_DATA.ind(Pr, Service_class, TPDU) & check_fgt(TPDU) & first_fgt(TPDU) & last_fgt(TPDU) & not(isSizeOK())	Tsap(TDPU, STsap, DTsap) InitBuffer () T_DATA.ind(STsap, DTsap, Service_class, TPDU)	Idle
Rec	DL_ABORT.ind(ErrorNb)	T_ABORT.ind()	*stopped*

Table 49 – Meaning of the states listed in the previous table

State	Meaning
Stopped	Stopped
Idle	Waiting state for a request from the application layer or an indication from the Data Link Layer
M.FirstFgt (Must first Fragment)	Initial transmission state with fragmentation of the application data unit
M.NextFgt (Must next Fragment)	Transmission state with fragmentation of the application data unit for the following fragments
Rec	Reception state with fragmentation of the received data units

Table 50 – Definition of the procedures and functions classified in alphabetical order

Procedure or Function	Definition
check_fgt(TPDU)	Check that the TPDU or TSDU carry the indentifier Dsap=6
first_fgt(TPDU)	Check if the fragment is the first or not. The first fragment shall have the fields DTSAP and STSAP correctly set and the field First = TRUE
FirstPayload()	Determination of the FirstPayload size. This determination invokes the Support Manager layer which returns back the value 114 or 241 depending whether the speed negotiation took place or not.
init()	Initialisation of state chart
initBuffer(TPDU)	Processing related to a first fragment The initial value of the APDU descriptor points to the top of the buffer, and the size of receivable data is equal to the size of the APDU. These parameters are provided by the Application layer.
IsSizeOK()	Check that the dat unit size is not greater than the available memory at the destination.
last_fgt(TPDU)	Check if the fragment is the last or not. The last fragment always has the field END= TRUE
min(SMsg, Payload)	Takes into account the size of the smallest of two parameters.
NextPayload()	Determination of the NextPayload size. This determination invokes the Support Manager layer which returns back the value 116 or 243 depending whether the speed negociation takes place or not.
Size(SMsg)	Calculate the size of the message SMsg in bytes
tsap(TDPU, STsap, DTsap)	Extraction of the fields STSAP and DTSAP of the TPDU
tsap(TDPU)	Extraction of the data fields of the TPDU
UpdateBufferRec(TPDU)	Update of the received Data buffer: Transfer of the TPDU data received into the Application buffer without exceeding the maximum size of the APDU; then the pointer describing the buffer, and the total length of the data received are increased accordingly. If the size of the data received exceeds the available buffer size, the buffer is reinitialised and the upper layer is informed of reception with an empty buffer.
Update(SMsg)	Update of the APDU after the transmission of a TPDU. The descriptor of the APDU is increased by the number of bytes sent and the length is reduced by the same amount.

7.6 Application Layer

7.6.1 General

The specification of the Application layer can be found in IEC 62056-5-3 Ed. 1.0:—. However, given the nature of the medium used and the characteristics of the lower layers, the following restrictions and behaviour apply.

7.6.2 Broadcast Management

All exchanges use confirmed services by default. Only broadcasting uses unconfirmed services. Broadcasting can only be used by a primary station. The type of service used will be propagated from the Application layer to the Data Link layer, which, depending on the service type specified, will use a specific destination address if the service type is confirmed or a broadcast address if the service type is unconfirmed

The secondary station shall not respond to requests carried by an unconfirmed service. This lack of response is managed by the application layer depending on the type of service.

Broadcasting will always be done without a speed negotiation, that is, at 1 200 bauds, with a data size of 114 bytes at the application layer level.

7.6.3 Management of EventNotifications or InformationReports

Secondary stations cannot transmit on the bus without having been requested. Management of event notifications or information reports is done in Euridis with the use of an alarm. The primary station is responsible for engaging the necessary actions to interrogate the equipment generating an alarm.

7.6.4 Priority Management

In the Euridis profile, priority management occurs at the Data Link layer. The DLMS/COSEM Application layer also has priority management tools. The DLMS/COSEM layer tools take precedence over those of Euridis for greater homogenisation.

However, in order to maintain compatibility with the DLMS profile, the priority management field (Pr) is maintained up to the Transport level. The transport layer simply ignores it in its relation with the application layer.

7.6.5 Management of releasing Application Associations

DLMS/COSEM Application Associations can be active only as long as the supporting layer of the DLMS/COSEM Application Layer is active. The Association is terminated when the activities of the lower layers are terminated.

Application Associations can be released either by closing the supporting layer or by using the RLRQ / RLRE services.

Note that if an Application Association is released by using the RLRQ / RLRE services, this does not affect the supporting layer.

8 Local bus data exchange – Hardware

8.1 General

The protocol describes the data exchanges between a Primary Station and Secondary Stations connected in parallel on a hardware bus. The Primary Station is connected to the bus by a passive magnetic plug.

This clause describes the following items:

a) the signal characteristics;
b) the bus characteristics;
c) the magnetic plug;
d) the Primary Station;
e) the Secondary Station;
f) the energy supply characteristics.

8.2 General characteristics

8.2.1 Signal transmission at 50 kHz

Transmission includes:

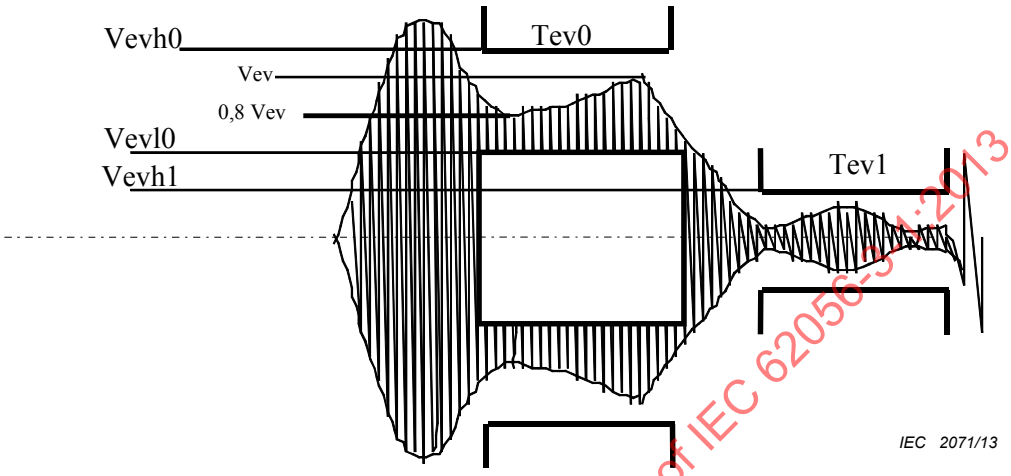
a) binary data transmission;
b) bi-directional, half-duplex;
c) baud rate: 1 200 Bd ± 1 %, 2 400Bd ± 1 %, 4 800Bd ± 1 %, 9 600Bd ± 1 %;
d) equal duration of bits 0 and 1;

e) amplitude signal modulation (ASM) of a 50 kHz ± 3 % carrier;

f) polarity:

- 0 = carrier detected,
- 1 = carrier not detected

The signal characteristics are defined by the carrier envelope described in Figure 6.



Key

Vevh1 is the maximum level for transmission of a "1"

Vevl0 is the minimum level for transmission of a "0"

Vevh0 is the maximum level for transmission of a "0"

Tev1 is the minimum guaranteed time for an envelope to remain lower than Vevh1

Tev0 is the minimum guaranteed time for an envelope to remain between Vevl0 and Vevh0

Figure 6 – Signal envelope on the bus

g) Vevl0 and Vevh0 are not the extremes of the envelope, but rather the low and high limits for correct operation;

h) during Tev0 the level of the envelope shall not vary by more than 20 %;

i) during the gaps between Tev0 and Tev1, the envelope rise or fall is exponential or damped sinusoid, with addition of frequency transients;

j) total harmonic distortion of the signal during continuous wave transmission is less than 15 %, with a resistor of 100 Ω or a capacitor of 31,8 nF in place of the bus;

k) all the voltages are specified in peak values;

l) for Time Bit definition (time of "1" or "0"), several parameters have to be taken into account:

- maximum time guaranteed for "0" transmission signal ≥ Vevl0;
- maximum time non-guaranteed for "0" transmission signal ≥ Vevh1;
- maximum time guaranteed for "1" transmission signal ≤ Vevh1;
- maximum time non-guaranteed for "1" transmission signal ≤ Vevl0;

8.2.2 Energy supply signal transmission

8.2.2.1 Characterization of the remote supply of energy

Figure 7 gives the representation of the bus for the remote supply of energy.

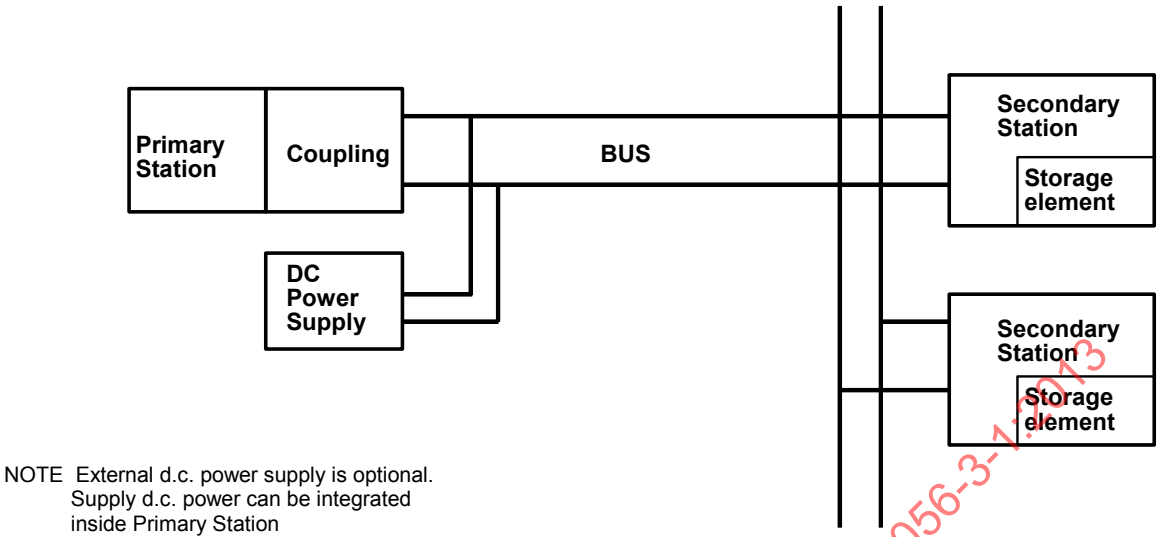


Figure 7 – Bus representation

The three main elements for the remote supply of energy are:

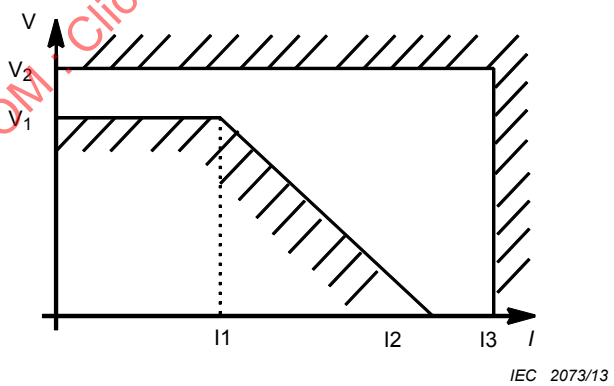
a) a.d.c. energy supply source,

b) a storage element in the Secondary Station,

c) the consumption of a Secondary Station.

8.2.2.2 Energy supply source

The energy supply source provides the bus with the values of voltage and current according to the template of Figure 8.



Key

V1 = 22 V, V2 = 35 V

I1 = 80 mA, I2 = 250 mA, 350 mA ≤ I3 ≤ 1 000 mA

Figure 8 – Power supply characteristics

The ripple noise will be less than 10 mV peak from 1 kHz to 1 MHz and 100 mV peak for f < 1 kHz.

The nature of the energy supply is not taken in account in this specification.

8.2.2.3 Storage Element in Secondary Station

A storage element associated with each Secondary Station allows the local accumulation of energy in order to absorb peaks of consumption during exchanges and to optimize the remote energy supply source. (Maximum value of 470 μF with a relative tolerance of + 20 %.)

8.2.2.4 Consumption of a Secondary Station

States associated to an exchange session are shown in Figure 9 and Figure 10.

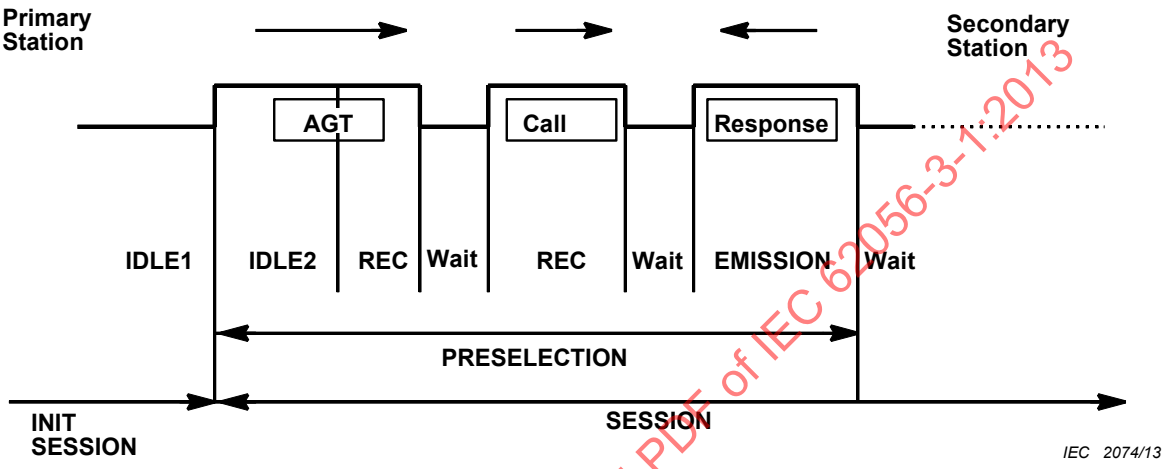


Figure 9 – States associated to a session: for selected Secondary Station

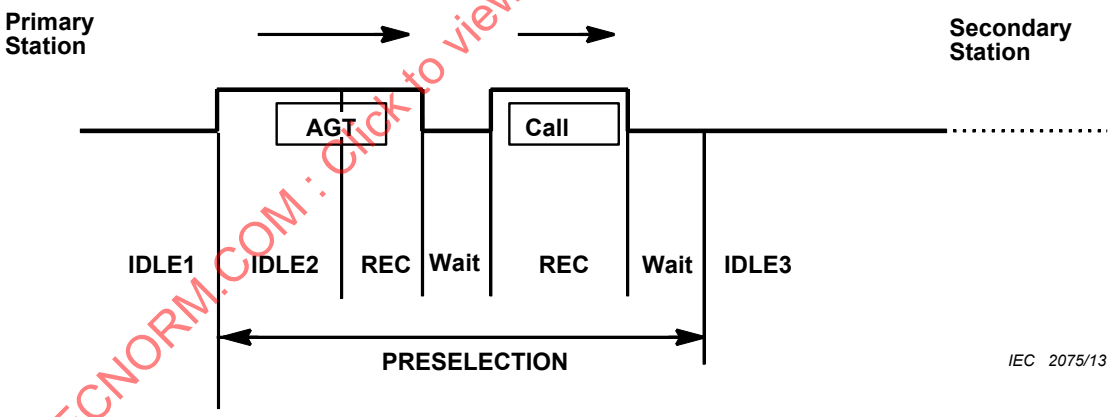


Figure 10 – States associated to a session: for non-selected Secondary Station

The consumption, depending on the states, is specified below.

These values represent maximum average consumption: peak can go over but average value remains under:

- Idle mode 1 15 mW max. (established mode);
- Idle mode 2 15 mW max.;
- Idle mode 3 15 mW max.;

- Reception mode 40 [7)] mW max. + nx10[7)] mW max. (n = number of devices);
- Wait mode 40 mW max. + nx10 mW max. (n = number of devices);
- Emission state 140 [7)] mW max. + nx10 mW max. (n = number of devices)

Worst case is the transmission of 9/10 "0" bits:

- Reception mode 50 [7)] mW max. + nx14[7)] mW max. (n = number of devices);
- Emission state 180 [7)] mW max. + nx14[7)] mW max. (n = number of devices)

The detailed description of these different modes (with the associated timing) is given in the state transitions.

8.2.3 Simple Secondary Station and multiple Secondary Station

A simple Secondary Station is equivalent to a logical address ADS.

A multiple Secondary Station is equivalent to several logical addresses ADS.

The notion of multiple Secondary Station, see Figure 11, allows the addressing of different devices located inside a Secondary Station on a secondary bus.

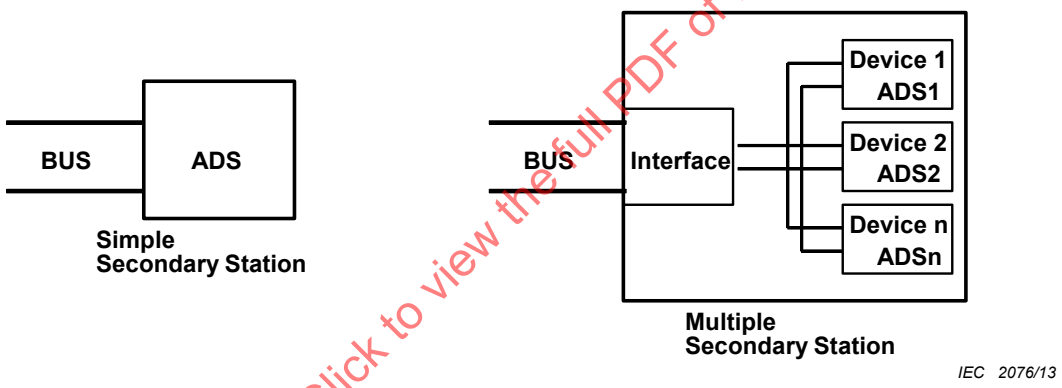


Figure 11 – Simple and multiple Secondary stations

The secondary bus structure fundamental points are as follows:

a) the length of secondary bus is not included in the length of main bus;

b) the secondary bus is four wires with two wires dedicated to signal and two wires dedicated to energy supply;

c) a Secondary non-energized Station, simple or multiple, is always equivalent to a maximum of two Secondary Stations towards physical communication parameters;

d) the primary and secondary bus protocol are the same, except for modulation aspects: signals on secondary bus are baseband (1 200 Bd to 9 600 Bd): signal wires. They are in accordance with EIA 485 and with ISO/IEC 8482;

e) the maximum length of the secondary bus is 50 m;

f) the maximum number of devices on the secondary bus is 6;

g) the cable is the same as the one used on the primary bus;

---

7) Reception and Emission consumption depends on the message content number of "0". Those values are average consumptions with an equal number of "0" and "1" bits.

h) it is possible to transmit alarms from a device to a primary station though the interface, when power supply is permanent on this bus. This report is made by carrier transmission TAB. The primary station, the interface and the devices shall be in Alarm mode active (see 4.4.12).

Main bus is able to support the following stations:

1) Simple energized Secondary Station;
2) Simple non-energized Secondary Station;
3) Multiple non-energized Secondary Station.

8.3 Bus specification

8.3.1 General characteristics

a) Specific support for remote reading and programming. The bus always has a magnetic socket and one Secondary Station (minimum);

b) The bus topology is unimportant, and can be linear, or star or tree without loop, provided that the total wired length of cable does not exceed 500 m (cable of power supply included). The secondary bus, inside a Secondary Station, is not included in this value;

c) The bus allows energy supply for Secondary Stations. In this way, the bus can support energized and non-energized stations;

d) Galvanic isolation is maintained between the bus and all the electronics of the transmitters and receivers, with voltage ratings which are required in the standards applied to Secondary Stations;

e) Supply can be permanent on the bus, in which case Alarm mode can be active;

f) From 1 to 100 Secondary Stations can be connected in parallel on the bus:

- The maximum number of non-energized Secondary Stations (simple or multiple) on the bus is 50;
- The maximum number of energized Secondary Stations on the bus is 100;
- The maximum number of devices associated to Secondary Stations (simple or multiple) on the bus is 50;
- In case of a mix of Secondary Station types, Energized and non-energized, connected on a same bus, a rule gives the maximum number of each type:
  - if $N1$ is the number of non-energized multiple Secondary Stations on the bus;
  - if $N2$ is the number of non-energized simple Secondary Stations on the bus;
  - if $N3$ is the number of energized Secondary Stations on the bus, then these inequations have to be respected:
    1) $2*(N1 + N2) + N3 \leq 100$ when power supply is integrated to the Primary Station;
    2) $2*(N1 + N2) + N3 \leq 98$ when power supply is external, connected directly on the bus

g) one of these Secondary Stations can accidentally stay in low impedance (transmission mode but without emission of “0” transmission), without interrupting the communication;

h) communication with the Primary Station is via a magnetic plug (one only);

i) the bus shall withstand the accidental connection of the 230 V mains. Control procedure is to apply five times successively 250 V a.c. Each application lasts 5 min and 5 s between each application.

8.3.2 Cable characteristics

Indoor telephone cable of type:

- single twisted pair and screen (aluminum) with drain wire;

- conductor: solid tinned copper of 0,5 mm to 0,6 mm nominal diameter;
- insulation PVC.

Electrical characteristics:

- DC looped resistance at 20 °C: 117 Ω/km to 192 Ω/km
- AC 50 kHz, between –15 °C to +45 °C:
  a) linear looped resistance: 154 Ω/km to 220 Ω/km
  b) linear looped inductivity: 500 μH/km to 800 μH/km
  c) linear mutual capacity: 80 nF/km to 130 nF/km
  d) loss factor of capacity: 5 % maximum
  e) capacity unbalance, wires to screen: 5 % maximum
  f) complex characteristic impedance: 74 Ω to 115 Ω
  g) linear phase shift (50 kHz): 150°/km maximum

The above characteristics are given for a symmetric source isolated from screen with the impedances *Z* and *Z´* greater than 1 000 Ω at 50 kHz (Figure 12).

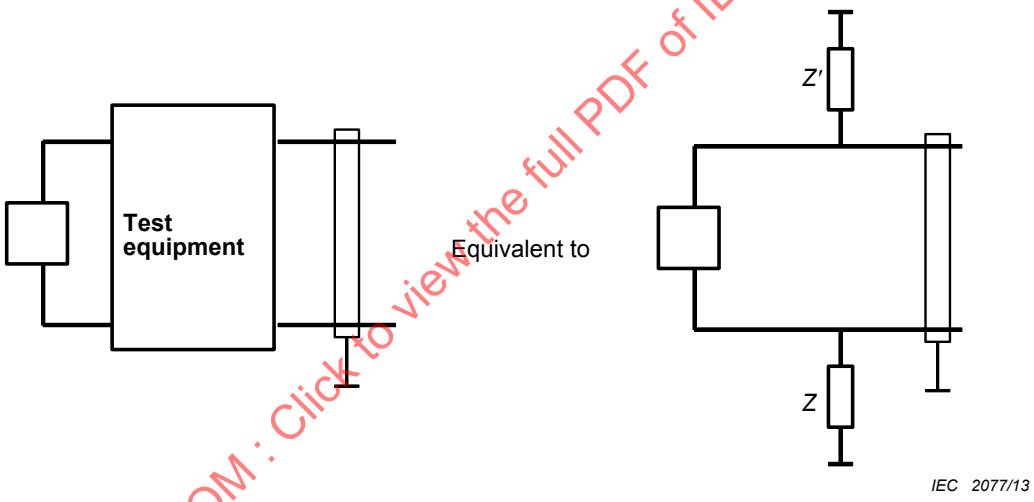


Figure 12 – Equivalent diagram of the test equipment

8.3.3 Wiring

a) connection of Secondary Stations shall ensure the continuity of the drain wire (e.g. three terminal distribution boxes);

b) one point of the drain wire shall be connected to earth, if any, or to an equivalent reference potential;

c) no impedance (except the cable), of less than 1 000 Ω at 50 kHz, shall be connected between wires of the bus and screen or earth.

For use of cables slightly outside the above specifications, the following should be noted:

– A cable with a higher linear capacity or resistance needs a lower length of wired cable. Ratio of length is approximately as the inverse ratio of linear capacity or resistance.

– A cable with a lower linear capacity or resistance could give overvoltage on receiver inputs for a long empty bus. This can be overcome by connecting between the wires of the bus, near the end opposite the magnetic plug, a damping resistor (330 Ω to 1 000Ω, 0,25 W, depending on the overvoltage ratio). To ensure that the bus withstands an accidental connection to 230 V mains, a 47 nF capacitor of proper voltage capability should be in series with this resistor.

8.4 Magnetic plug

8.4.1 Function

8.4.1.1 Simple magnetic plug

The magnetic plug consists of a mobile plug (primary) and a fixed socket (secondary).

When the two halves are joined, the magnetic plug transfers signals between the HHU connected to the plug and the bus connected to the socket, in both directions.

Each part consists of half a ferrite transformer with an air-gap in the magnetic circuit.

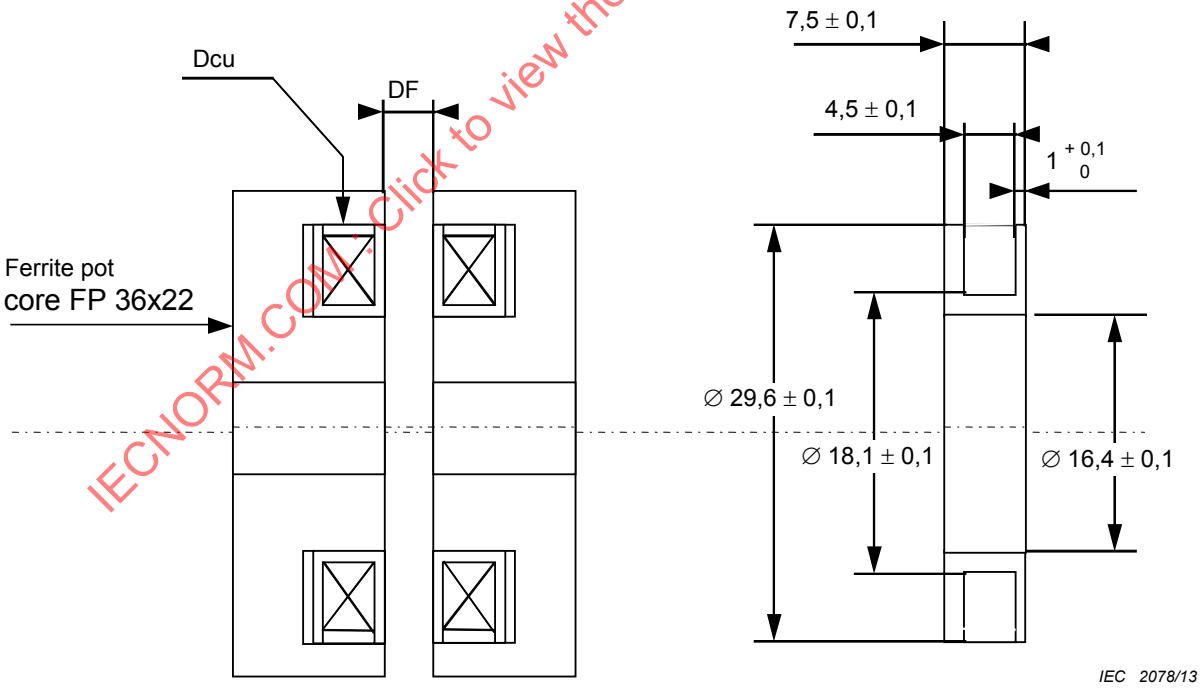
To compensate the high series inductance and the low parallel inductance of such a transformer, a resonating capacitor and a damping resistor on both sides convert this transformer into a fourth order band-pass filter centered near 50 kHz, with a Q factor <3.

This allows the use of a simple square wave source for the transmission, and eliminates the frequency transients.

8.4.1.2 Magnetic plug with energy supply

In addition to the previous characteristics, the magnetic plug with energy supply allows the transmission between 400 kHz and 600 kHz signal. The default value is fixed at 500 kHz. This signal is rectified and filtered before injection of d.c. signal on the bus.

8.4.2 Common mechanical characteristics



Key *Dimensions in millimetres*

FP ferromagnetic pot

DF distance between ferrites

Dcu diametre of core unit

Figure 13 – Ferrite pot and bobbin

Each half of the magnetic plug comprises a bobbin in a half pot ferrite core, enclosed in a robust plastic housing, see Figure 13. When joined, the bobbins and ferrite cores are nearly coaxial and symmetric about the mid plane, with:

- magnetic air gap DF = 4,25 mm ± 0,25 mm;
- maximum error of coaxiality, near mid point: 0,25 mm.

When assembled and joined, both ferrite cores and bobbins shall be fixed or pushed closer to the mid plane. For example, the axial gap between bobbin and core, due to tolerances or sizes shall be at the rear side of the bobbin.

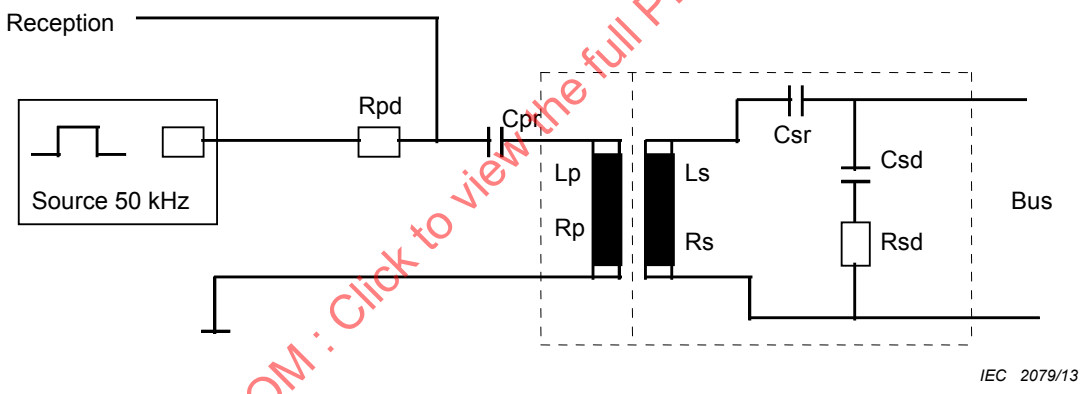
Ferrite is a standard type for a maximum frequency of less than 100 kHz:

- Initial permeability more than 1 800;
- Tan delta maximum at 100 kHz about 2 %;
- No saturation due to leakage fields (practically saturation comes around 0,4 T).

8.4.3 Electrical block diagram with simple plug

8.4.3.1 General

The general electrical bloc diagram with simple plug is shown in Figure 14, with its associatied components.



Key

Rpd	primary damping resistor	Lp	primary inductance
Cpr	primary resonance capacitor	Ls	secondary inductance
Csr	secondary resonance capacitor	Rp	primary resistance
Csd	secondary damping capacitor	Rs	secondary resistance
Rsd	secondary damping resistor		

Figure 14 – Associated components of the magnetic plug

8.4.3.2 Associated components in socket, bus side

Serial capacitor Csr resonating with Ls.

Parallel damping resistor Rsd.

Parallel capacitor Csd in series with Rsd to ensure immunity to accidental connection to 230 V mains.

8.4.3.3 Associated components, plug side

Serial capacitor Cpr resonating with Lp.

Serial damping resistor Rpd: (depending on the 50 kHz source, and including all series resistances (at 50 kHz), such as flexible cord, connector, etc.).

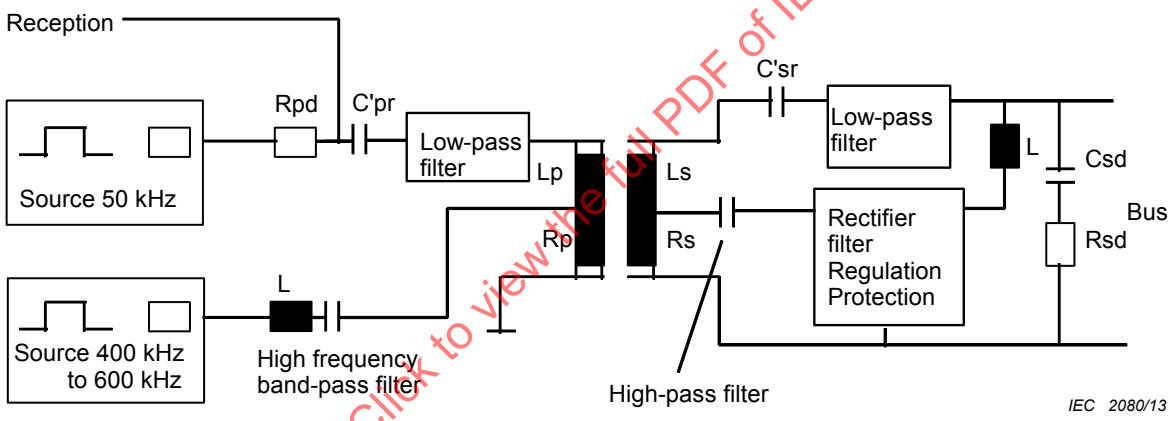
A 50 kHz source which can be a square-wave source.

Receiver circuit, demodulation and squaring, connected to the Cpr Rpd node (e.g. full wave rectifier and threshold circuit).

In receiving mode, the output impedance of the 50 kHz source shall be maintained at less than a few ohms, even for low voltage, to ensure damping of the primary circuit.

8.4.4 Electrical Block Diagram with energy supply plug

The electrical bloc diagram with energy supply plug, 8.4.1.2 is shown in Figure 15, with its associatied components.



Key

Rpd	primary damping resistor	Lp	primary inductance
Cpr	primary resonance capacitor	Ls	secondary inductance
Csr	secondary resonance capacitor	Rp	primary resistance
Csd	secondary damping capacitor	Rs	secondary resistance
Rsd	secondary damping resistor		

Figure 15 – Associated components of the energy supply plug

The 50 kHz part works like the simple plug circuit, except for:

- extra attenuation, both ways, due to the low-pass filters. The 50 kHz source receiver device shall take this attentuation into account.
- C'pr and C'sr differ slightly from Cpr and Csr due to the low-pass filter impedance at 50 kHz.

The 400 kHz-600 kHz part provides means for the remote supply.

8.5 Functional specifications of Primary Station transmitter (for 50 kHz signal)

The Primary Station transmitter is the assembly of a Primary Station transmitting source and the magnetic plug.

The signal transmitted at the bus outputs shall fit into the limits (see Figure 6), of the whole temperature range, with:

a) Tev0 and Tev1 shall comply with the values Table 51.

Table 51 – Primary station transmitter: Tev0 and Tev1 values

	1 200 bauds	2 400 bauds	4 800 bauds	9 600 bauds
Tev1(µs)	750	330	140	60
Tev0(µs)	750	330	140	60

b) Vevh1 = 0,25 V

This concerns only signals with frequency 1 kHz ≤ *f* ≤ 1 MHz.

Open circuit at bus output:

c) Vevl0 = 5,8 V

d) Vevh0 = 7,5 V

With a resistor of 100 Ω in place of the bus:

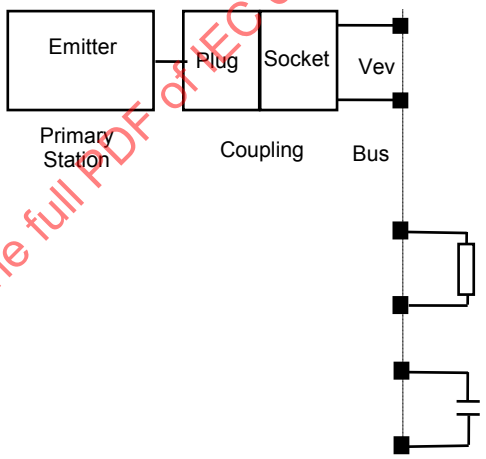
e) Vevl0 = 4 V

f) Vevh0 = 4,8 V

With a capacity of 31,8 nF in place of the bus:

g) Vevl0 = 5,2 V

h) Vevh0 = 6,5 V



In addition, the output terminals should be open-circuit or matched to a 100 Ω resistance or a capacitance of 31,8 nF.

i) the noise transmitted at the bus output, in all conditions and at all frequencies up to 1 MHz, after extinction of transients, shall not exceed 10 mV peak between 1 kHz and 1 MHz;

j) the overvoltage spike due to the switch from transmission mode to reception mode, or the reverse, shall not exceed 0,25 V peak, in all conditions.

These values have to be respected with a magnetic air gap
DF = (4,25 mm ± 0,25 mm) $\pm^{0,15}_{0}$ mm.

8.6 Functional specifications of Primary Station receiver (for 50 kHz signal)

The Primary Station receiver is the assembly of a Primary Station receiving circuit (demodulation and squaring) and the magnetic plug.

The receiver shall function correctly (defined as frame repetition rate < 10^{-5}) for a sinusoidal input signal whose characteristics are defined above and for the complete temperature range.

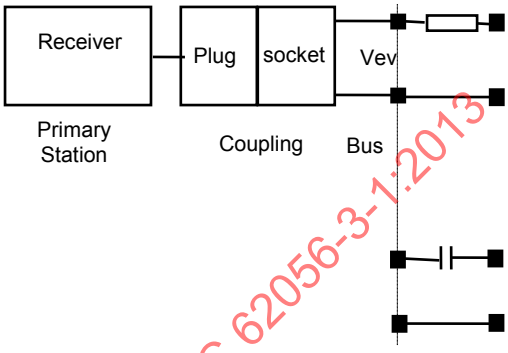
The signal shall be applied to the bus terminals of the magnetic plug through the serial impedances given below:

a) Tev0 and Tev1 shall comply with the values Table 52.

Table 52 – Primary station receiver: Tev0 and Tev1 values

	1 200 bauds	2 400 bauds	4 800 bauds	9 600 bauds
Tev1(µs)	700	290	125	50
Tev0(µs)	700	290	125	50

Through a 100 Ω resistance:



b) Vevh1 = 0,25 V

c) Vevl0 = 0,7 V

d) Vevh0 = 3,2 V

Through 31,8 nF:

e) Vevh1 = 0,20 V

f) Vevl0 = 0,55 V

g) Vevh0 = 2,5 V

In addition, the receiver in condition b) (100 Ω in series) shall function correctly (defined as frame repetition rate < 10^{-5}) with:

h) a continuous wave signal of 0,1 V peak from 1 kHz to 1 MHz

i) a square pulse of 20 V of duration 5 µs

j) a square pulse of 3,5 V of duration 200 µs

8.7 Functional specification of Secondary Station transmitter (for 50 kHz signal)

The signal transmitted at the bus outputs shall fit into the limits specified, in the whole temperature range with:

a) Tev0 and Tev1 shall comply with the values Table 53.

Table 53 – Secondary station transmitter: Tev0 and Tev1 values

	1 200 bauds	2 400 bauds	4800 bauds	9 600 bauds
Tev1(µs)	750	330	140	60
Tev0(µs)	750	330	140	60

b) Vevh1 = 0,1 V

This only concerns signals with a frequency of 1 kHz ≤ *f* ≤ 1 MHz.

With 100 Ω in place of the bus:

c) Vevl0 = 1,2 V

d) Vevh0 = 1,8 V

With a capacitor of 31,8 nF in place of the bus, the output signal measured across a resistor of 1 Ω in series with the capacitor, and the result multiplied by 100:

e) Vevl0 = 1,5 V

f) Vevh0 = 2,5 V

In addition, with the two bus terminals connected to a resistance of 100 Ω or a capacitance of 31,8 nF:

g) the overvoltage spike due to the switching from transmission mode to reception mode, or the reverse, shall not exceed 0,75 V peak, in all conditions;

h) the noise transmitted at the bus output, in all conditions and at all frequencies up to 1 MHz, after settling of transients, shall not exceed 10 mV peak between 1 kHz and 1 MHz.

In addition:

i) maximum short circuit current: 26 mA peak at 50 kHz;

j) the transmitter shall tolerate permanent short circuit and connection to 230 V mains at the bus terminals;

k) the maximum common mode capacity between bus inputs and other inputs of the device including the secondary station is fixed to 15 pF.

8.8 Functional specifications of Secondary Station receiver (for 50 kHz signal)

The receiver shall function correctly (defined as bit error rate < 10^{-5}) for a sinusoidal input signal whose characteristics are defined above and for the whole temperature range with:

a) Tev0 and Tev1 shall comply with the values Table 54.

Table 54 – Secondary station receiver: Tev0 and Tev1 values

	1 200 bauds	2 400 bauds	4 800 bauds	9 600 bauds
Tev1(μs)	700	290	125	50
Tev0(μs)	700	290	125	50

With a voltage generator with negligible internal impedance, in comparison with input impedance of the receiver:

b) Vevh1 = 0,3 V

c) Vevl0 = 2 V

d) Vevh0 = 8 V

In addition, the receiver shall be insensitive to:

e) a permanent signal of 0,25 V peak from 1 kHz to 1 MHz;

f) a pulse of 20 V of duration 5 μs;

Input impedance at 50 kHz:

g) the input impedance, whether the receiver is powered up or not, at up to 5 V peak, shall consist of a resistance in parallel with a reactance. Energized and non-energized cases have to be considered:

- For an energized station:

  resistance > 20 kΩ

  reactance > 20 kΩ (60 mH) if inductive

  >100 kΩ (30 pF) if capacitive

- For a non-energized station:

  resistance > 10 kΩ

reactance > 10 kΩ (30 mH) if inductive

> 50 kΩ (60 pF) if capacitive

h) internal clamping can occur at more than 5 V peak, provided that the dynamic input impedance above the clamping voltage is > 200 Ω at 50 kHz;

i) minimum input impedance at 50 kHz with an output locked in position transmission of a logical level 1 (no signal on the bus): 200 Ω;

j) the receiver shall tolerate a permanent connection to 230 V mains, at the bus terminals;

k) maximum common mode capacity between bus inputs and other inputs: 15 pF for an energized station and 100 pF for a non-energized station.

All impedances used for measurements as specified in Clause 8 should be of 1 % accuracy.

Annex A
(normative)

Specification language

A.1 Vocabulary and operating rules

To describe the role of each layer of the local bus data exchange profiles unambiguously, the specification uses a table formalism modelling the real behaviour by a controller with a finite number of states.

To each controller corresponds a unique logic table; this logic table may be broken down into several physical tables if it is particularly large.

To each controller occurrence corresponds an instance (distinct active copy) of the logic table of the reference controller.

Each physical table consists of lines known as state lines. Each state line describes the triggering condition (column 2) for the machine to pass from an initial state (column 1) to a final state (column 4) by executing a set of actions (column 3).

The first initial state is the start-up state of the controller. This state is unique; it is particularized by means of italic characters.

A stop state of the controller is a final state for which no state line is defined with this state as initial state. A controller is infinite when it does not have a stop state. A finite controller may have one or more stop states. These states are also represented by using italic characters. This convention means that the order in which the states are presented in a physical table is not important.

The same rule applies when several state lines refer to the same initial state, because the triggering conditions are always mutually exclusive. The order of the lines in a physical table is therefore governed by presentation considerations only. Nevertheless, it is logical to begin by describing the transitions of the start-up state.

A set of actions in a state line shall be considered as a critical section (i.e. an uninterruptible sequence). The actions described there shall be executed in the order in which they are written. An action is defined by an invocation of a named procedure instantiated with a zero list, one or more parameters between parentheses. All referenced named procedures shall be the subject of separate descriptions. However, there are two predefined actions: assignment = and empty action $none() (no action).

The triggering condition associated with a state line may be composed of several sub-conditions. The assessment of a composite triggering condition always involves the assessment of all the sub-conditions that it contains. Therefore, the order in which the sub-conditions are written is unimportant.

The operators supported for expressing composite conditions are the logic operators & (logic and), | (logic or), not() (logic no) and the comparison operators (<, >, <=, >=, = and <>).

There are two types of triggering condition.

A simple condition is assessed instantaneously, by definition. It may be composite but, in such cases, all the sub-conditions shall be of the simple type. A boolean named function is an

example of a simple condition. All referenced boolean named functions shall be the subject of separate descriptions.

An event condition expresses the wait for an event. It may be composed of several events or simple sub-conditions.

When the assessment of a triggering condition gives a true result, the condition is satisfied. The satisfaction of a triggering condition always leads to a state transition.

An event can be defined as an element contributing to the satisfaction of an event-type triggering condition.

When an event is included in an event-type triggering condition which is satisfied, it is automatically consumed. An event can be consumed only once.

Any event that occurs when the controller occurrence that is likely to consume it is in a state where this consumption is impossible, is stored chronologically in an area known as the inter-controller queue.

Each controller thus has a single queue that it shares between its own controller occurrences. The size of this queue is assumed to be quasi-infinite; its organization and its management are not described here. However, it should be noted that a partial purge of the queue (i.e. related only to the events concerning the current controller occurrence) is automatically carried out for any state transition starting from the start-up state.

There shall also be a self-purge mechanism for automatic deletion of incoming events that are manifestly not consumable. Moreover, there is a predefined named procedure $purge(), which corresponds to the action of total purge of the current inter-controller queue. All the occurrences of the corresponding controller then return to the start-up state.

Events are produced by some of the described actions in a set of actions associated with a state line. An internal event can be consumed only by the controller that has produced it. An external event is always consumed by a controller other than the one that has produced it.

It should be noted that the absence of an event (expressed by an event sub-condition encapsulated in the logic operator not()) is always a simple sub-condition.

When, for an initial state, there is a state line where the triggering condition is of a certain type (simple or event), then all the state lines having the same initial state shall have triggering conditions of the same type.

When this type is simple, the initial state is called sub-state. A sub-state is particularized by means of the italic attribute. It is transient and can always be removed; its presence in a physical table is justified only by improved clarity of presentation. In the special case of a start-up sub-state, a special condition has been predefined: $true(), which is always true.

The variables referred to in the triggering conditions and the actions described in a physical table remain local with respect to each controller occurrence. There is also a predefined variable (the unlinked variable _) intended to replace any unused parameter in any function or named procedure.

A.2 Entity and Entity Invocation

It is interesting to draw a parallel between the elements of the specification language described herein and certain concepts developed by the OSI (Open Systems Interconnection).

For each layer, for example, the notion of Entity corresponds to a controller, while the term Entity Invocation is similar to controller occurrence.

Annex B
(normative)

Timing types and characteristics

B.1 Timing type definition

Timings are classified into several types:

Logical timing: TL type

defined from Stop-bit to Stop-bit (Figure B.1)

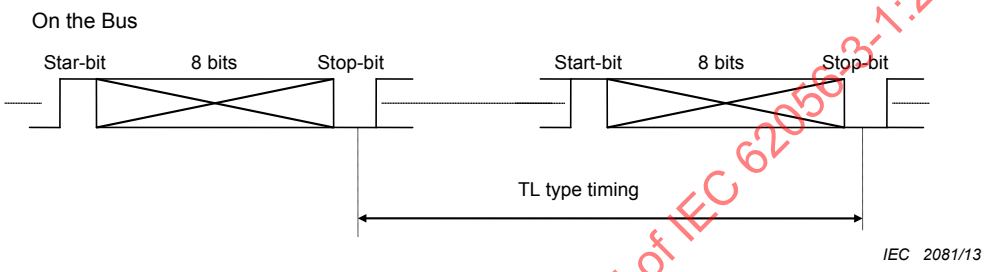


Figure B.1 – Logical timing type

Physical timing: TPFD or TPDF

Defined from End of carrier to Start of carrier for TPFD timing and from Start of carrier to End of carrier for TPDF timing (Figure B.2).

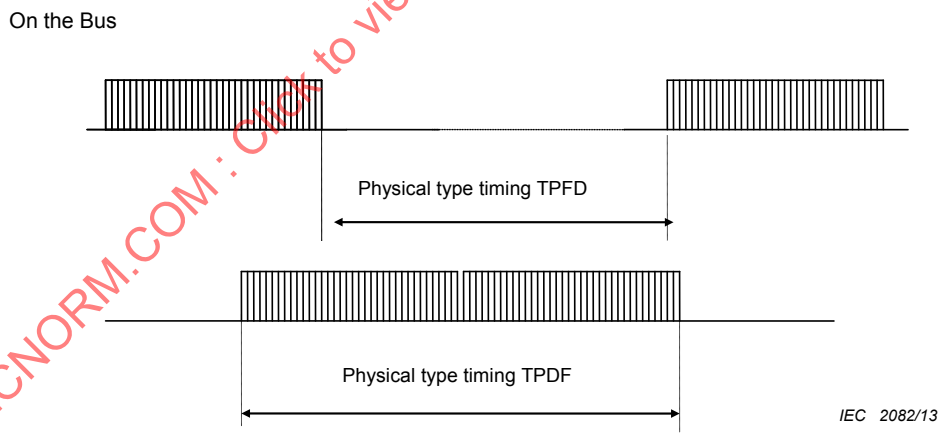


Figure B.2 – Physical timing type

Semi-Logical timing: TSL1 type

Measured from End of carrier or End of event to Stop-bit.

Semi-Logical timing: TSL2 type

Measured from End of event to Start-bit.

Chrono Timing: Tc type

Triggered by event and stopped automatically after programmed delay.

Specific Timing: Ta type

Depending of bus energy supply: TICB is Ta type.

B.2 Timing measurements and characteristics

Timing precision is ±1 %. The minimum value cannot be under ±10 ms.

Each timing limit value in the arrays takes this characteristic into account. Results shall be treated in the following way (see Figures B.3 and B.4):

a) High Limit – Tolerance > M > Low Limit + Tolerance, result is 100 % OK;

b) M < Low Limit – Tolerance, result is 100 % NOK;

c) M > High Limit + Tolerance, result is 100 % NOK;

d) (Low Limit – Tolerance) < M < (Low Limit + Tolerance), result is not determined, OK or NOK;

e) (High Limit – Tolerance) < M < (High Limit + Tolerance), result is not determined, OK or NOK.

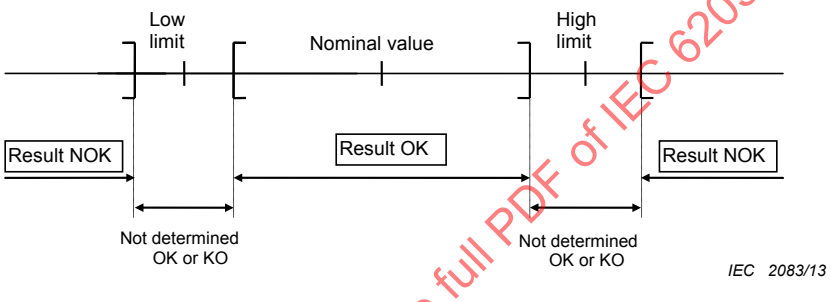


Figure B.3 – Results processing for timing defined with low and high limits

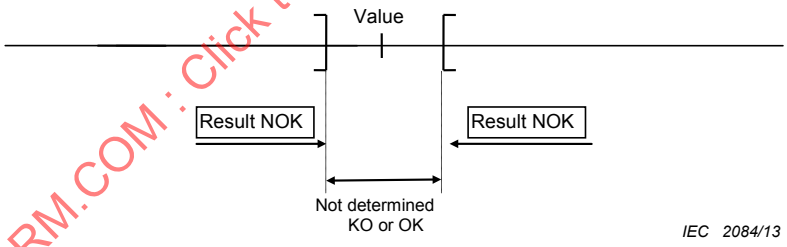


Figure B.4 – Results processing for timing defined by a nominal value

Annex C
(normative)

List of fatal errors

Any occurrence of one of the listed fatal errors is sent up locally to the *Application* layer.

Table C.1 – FatalError error numbers

Number	1	2	3	4	5	6	7	8
Error	**EP-3F**	**EP-4F**	**EP-5F**	**EL-1F**	**EL-2F**	**EA-1F**	**EA-2F**	**EA-3F**

Whatever layer is involved, the occurrence of a fatal error results in:

- if appropriate, stopping of the next lower layer by means of the service primitive abort.req;
- if appropriate, informing of the next higher level by means of the primitive abort.ind with the number of the fatal error as parameter;
- complete reinitialization of the corresponding controller occurrence.

Annex D
(normative)

Coding the command code field of frames

D.1 Command codes for local bus data exchange (Table D.1)

Table D.1 – Command codes for local bus data exchange

Abbreviation	Hexa value	Binary value	Role
The following codes are used without DLMS			
ENQ	01	0000 0001	Request for remote reading
DAT	02	0000 0010	Response of remote reading
REC	03	0000 0011	Request for remote programming
ECH	04	0000 0100	Data echo during remote programming
AUT	05	0000 0101	Authentication command
EOS	06	0000 0110	End of session
ASO	07	0000 0111	Forgotten stations call
RSO	08	0000 1000	Forgotten stations response
IB	09	0000 1001	Bus initialization
DRJ	0A	0000 1010	Rejection of remote programming data
ARJ	0B	0000 1011	Rejection of remote programming authentication
TRF	0C	0000 1100	Point to point remote transfer
TRB	0D	0000 1101	Broadcast remote transfer (not acknowledged)
TRA	0E	0000 1110	Point to point remote transfer acknowledgement
PRE	10	0001 0000	Non-energized station selection
SEL	11	0001 0001	Non-energized station selection acknowledgement
The following codes are used only with DLMS/COSEM			
XBR	12	0001 0010	Change baurate request
XBA	13	0001 0011	Change baurate response

The commands listed in Table D.1 above are also used by the Support Manager Layer in DLMS/COSEM environment and in DLMS for the management of the bus initialisation, the discover and the transmission speed negotiation processes.

D.2 Codes of commands for data exchange on the local bus with DLMS or DLMS/COSEM

In order not to confuse DATA+ frame with frames from the profile without DLMS, the DATA+, Priority, Send and Confirm fields make up a special command code COM whose values have been chosen different from the already reserved COM values (Table D.2).

Table D.2 – Command codes with DLMS and DLMS/COSEM

Abbrev.	Hexa value	Binary value	Role
ND1	E0	1110 0000	Normal data (Priority="0"B) with sequence number="0000"B
ND2	E3	1110 0011	Normal data (Priority="0"B) with sequence number="0011"B
ND3	EC	1110 1100	Normal data (Priority="0"B) with sequence number="1100"B
ND4	EF	1110 1111	Normal data (Priority="0"B) with sequence number="1111"B
UD1	F0	1111 0000	Urgent data (Priority="1"B) with sequence number="0000"B
UD2	F3	1111 0011	Urgent data (Priority="1"B) with sequence number="0011"B
UD3	FC	1111 1100	Urgent data (Priority="1"B) with sequence number="1100"B
UD4	FF	1111 1111	Urgent data (Priority="1"B) with sequence number="1111"B

Annex E
(normative)

Principle of the CRC

E.1 General

The checking of the correct transmission of the bits is made globally on an information block expressed in bytes (refer to the Data Link layer). The check key is a set of bits known as the check field. Its calculation is based on the theory of cyclic codes which uses the division of polynomials and the algebraic properties of the remainders of the division of polynomials.

E.2 Operations on the polynomials

Let A = (a_1, a_2,..., a_n) be the bit string for which the check key shall be calculated. This string can be viewed as a polynomial of degree n-1:

$$A(X) = a_1X^{n-1} + ... + a_{n-1}X + a_n$$

Take a divisor polynomial D(X) of degree m. The polynomial division of $A(X)*X^m$ by D(X) gives the following formula:

$A(X)*X^m = D(X)*Q(X) + R(X)$
where R(X) is a polynomial of degree less than or equal to m-1 representing the bit string R = (r_1, r_2,..., r_m)

Given the properties of boolean arithmetic, this formula can also be written:

$A(X)*X^m + R(X) = A(X)*X^m - R(X) = D(X)*Q(X)$
which represents the bit string (a_1, a_2,..., a_n, r_1, r_2,..., r_m)

E.3 Check procedure

The CRC (Cyclical Redundancy Check) check field is represented by the bit string R = (r_1, r_2,..., r_m). Its routine practical calculation is based on shift registers and accumulator registers enabling the key to be calculated as the data bits arrive. The corresponding algorithm will not be described here.

As the theory indicates, the Transmitter shall assess the bit string R = (r_1, r_2,..., r_m), concatenate this string with the bit sequence to be protected A = (a_1, a_2,..., a_n) and transmit the resulting bit string (a_1, a_2,..., a_n, r_1, r_2,..., r_m) to the Receiver.

The Receiver considers that there is no transmission error when the bit string received corresponds to a polynomial of degree m+n-1 exactly divisible by D(X).

E.4 Operating parameters

For the concrete implementation of the algorithm, the chosen parameters are:

m	16
D(X)	$X^{16} + X^{15} + X^2 + 1$

Annex F
(normative)

Random integer generation for response from forgotten stations

F.1 General

The forgotten station call exchanges require the generation and the processing of random integers in the range [0, MaxRSO[to manage several time slots.

F.2 Criterion for a random integer

Rather than specifying a dedicated solution or algorithm, the following criterion has been chosen for focusing on a random integer.

“Whatever an integer I in the range [0, n[, this number I is known as random when its apparition probability is always included in the range [100/n – D, 100/n + D], provided that a significant number of N draws has been respected in a relatively short time T.”

F.3 Operating parameters

In the case of RSO time slots for the processing of a “Forgotten Stations Call”, the value of the maximum number, MaxRSO, is set to 3. Under this condition, the choice of N, T and D parameters have been chosen as follows:

N	T	D
≤ 100	≤ 10 min	≤ 7

Annex G
(normative)

Random number generation for authentication (profile without DLMS)

According to the DES (Data Encryption Standard) standard algorithm, the authentication exchanges require the generation and the processing of 64 bit random numbers.

Two criteria allowing implementation control are specified rather than a specific solution:

First criterion – Hamming Distance

Random number NAi(k), generated by a primary station (i=1) or a secondary station (i=2), shall have a Hamming Distance Dh greater than 4 with each anterior random number in a window k of 400 values observed on the bus.

It is given by the formula:

For observation k, Dh [NAi(n+k), NAi(n+k-l)] > 4, with 0 < l < k

Successive observation number k is 400. Observation starts with k = 0 and first random number has the value NAi(n).

Observations occur with a minimum 1 s delay.

Second criterion – Probability of Bit Distribution

From the complete bit range of row i taken from the 400 random numbers previously mentioned, the probability of appearance of an 0 or 1 value have to be between 0,35 and 0,65.

It is given by the formula:

$0{,}35 < [Pr(BitVal\ 2^i) = 0] < 0{,}65$ for $0 \leq i \leq 63$

$0{,}35 < [Pr(BitVal\ 2^i) = 1] < 0{,}65$ for $0 \leq i \leq 63$

Annex H
(normative)

Systems management implementation

To ensure a maximum compatibility (for stations including DLMS and DLMS/COSEM or not), it is proposed to implement the systems management by using the forgotten station call mechanism Table H.1.

Table H.1 – Discovery service

Discover request	Discover response
ASO frame with 2 bytes in the TABi field (the first corresponds to the reserved TAB 0 and the second is the response probability value)	Standard RSO frame including the System Title (ADS)

Below are the service specifications of the discovery service Table H.2.

Table H.2 – Service specification

Procedure or function (Primary Station)	Definition
discover.request(energized, adp, response-probability)	energized: TRUE for discovering new energized stations and FALSE for discovering new non-energized stations ADP: Physical address of the Primary Station response-probability: required probability of answer
discover.confirm (collision, discover-list)	collision: TRUE if collision has occurred, FALSE if not discover-list: list of the ADS of the new stations discovered

On the Secondary Station, the Discover service requires the generation of random integers in the range [0, 100] to issue a Discover response. The random function shall have a random behaviour of ±10 %. Thus, for a significant number *N* of systems (*N* greater than 10), the reporting systems number shall be (Response Probability × *N*) / 100 at ±10 %.

Annex I
(informative)

Information about exchanges

I.1 Non-energized station session (Figure I.1)

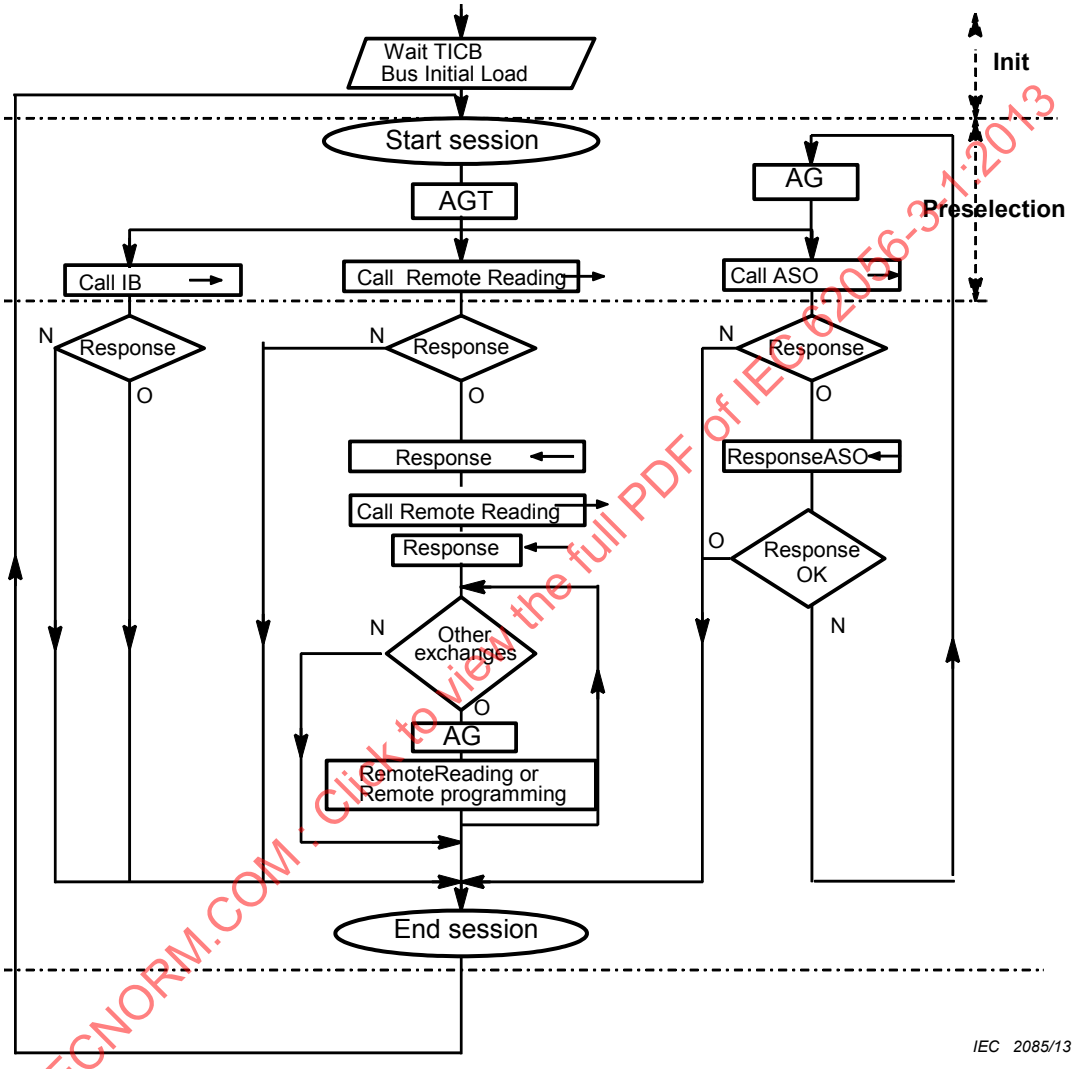


Figure I.1 – Non-energized station session

I.2 Remote reading and programming exchanges (Figure I.2)

NOTE This subclause is relevant only for the profile without DLMS.

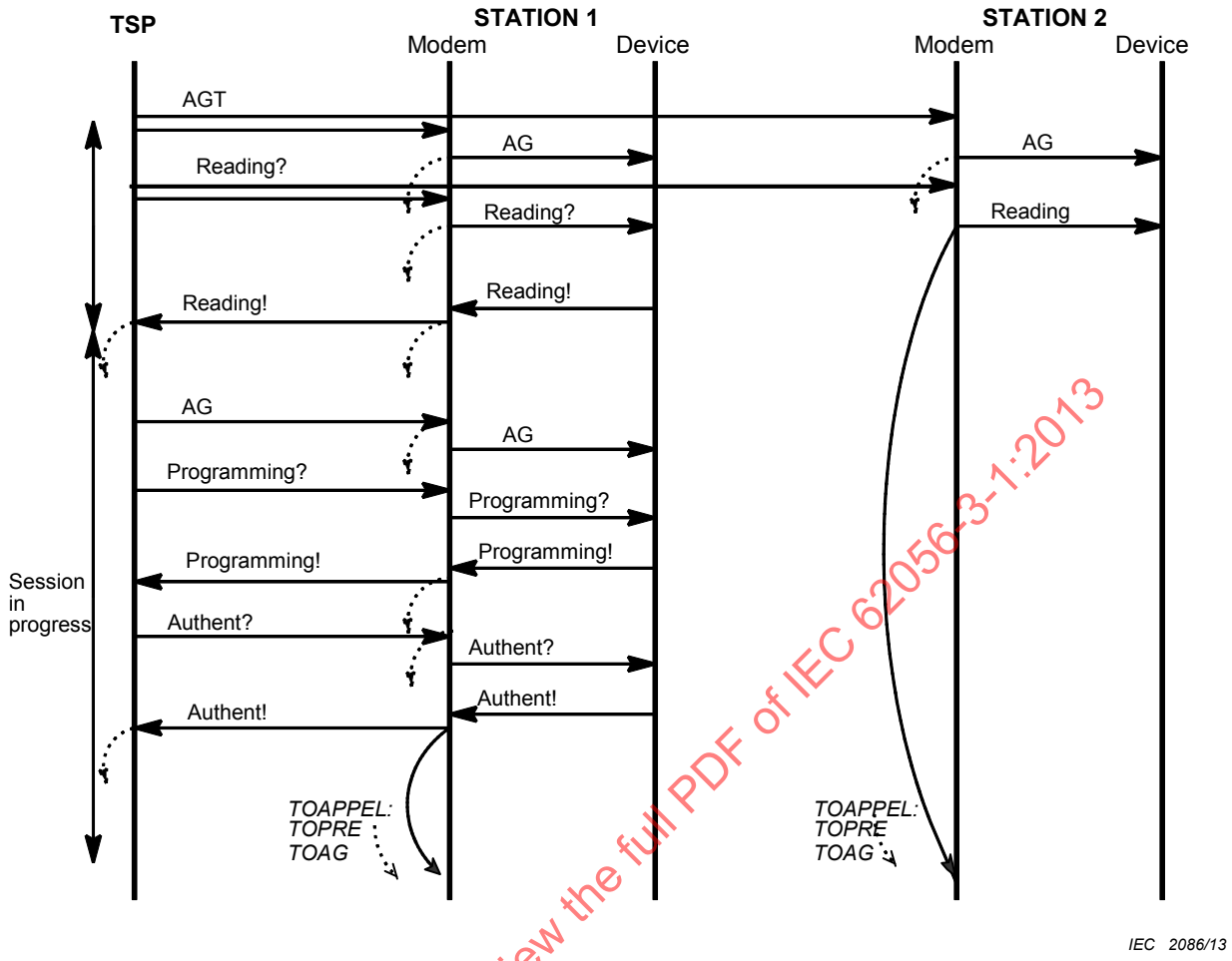


Figure I.2 – Remote reading and programming exchanges

The Primary Station sends an AGT “wakeup Call” signal towards the non-energized Secondary Station. After filtering by its modem, the Secondary Station receives an AGN signal, and it initializes the timer TOAPPEL (maximum waiting time for selection frame).

The Primary Station sends a remote reading frame to the Secondary Station 1. The Secondary Station 2 does not answer, and goes back to a low consumption state when there is a time out of TOPRE. The station 1 answers: it is selected. The example is a sequence of remote reading exchange and a remote programming exchange. The session is checked by the wakeup TOAG.

I.3 Bus initialization frame (Figure I.3)

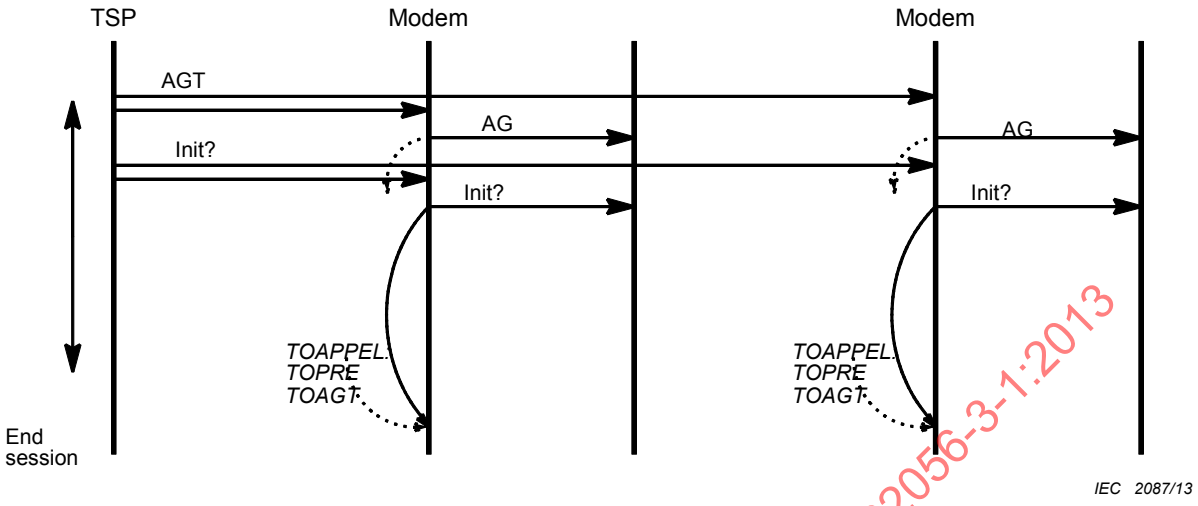


Figure I.3 – Bus initialization

The Primary Station sends an AGT “Wakeup Call” signal and a bus initialization frame. All the non-energized Secondary Stations wake up. The IB frame does not involve any frame answer, thus all the stations go back to a low consumption state.

I.4 Forgotten station call exchange (Figure I.4)

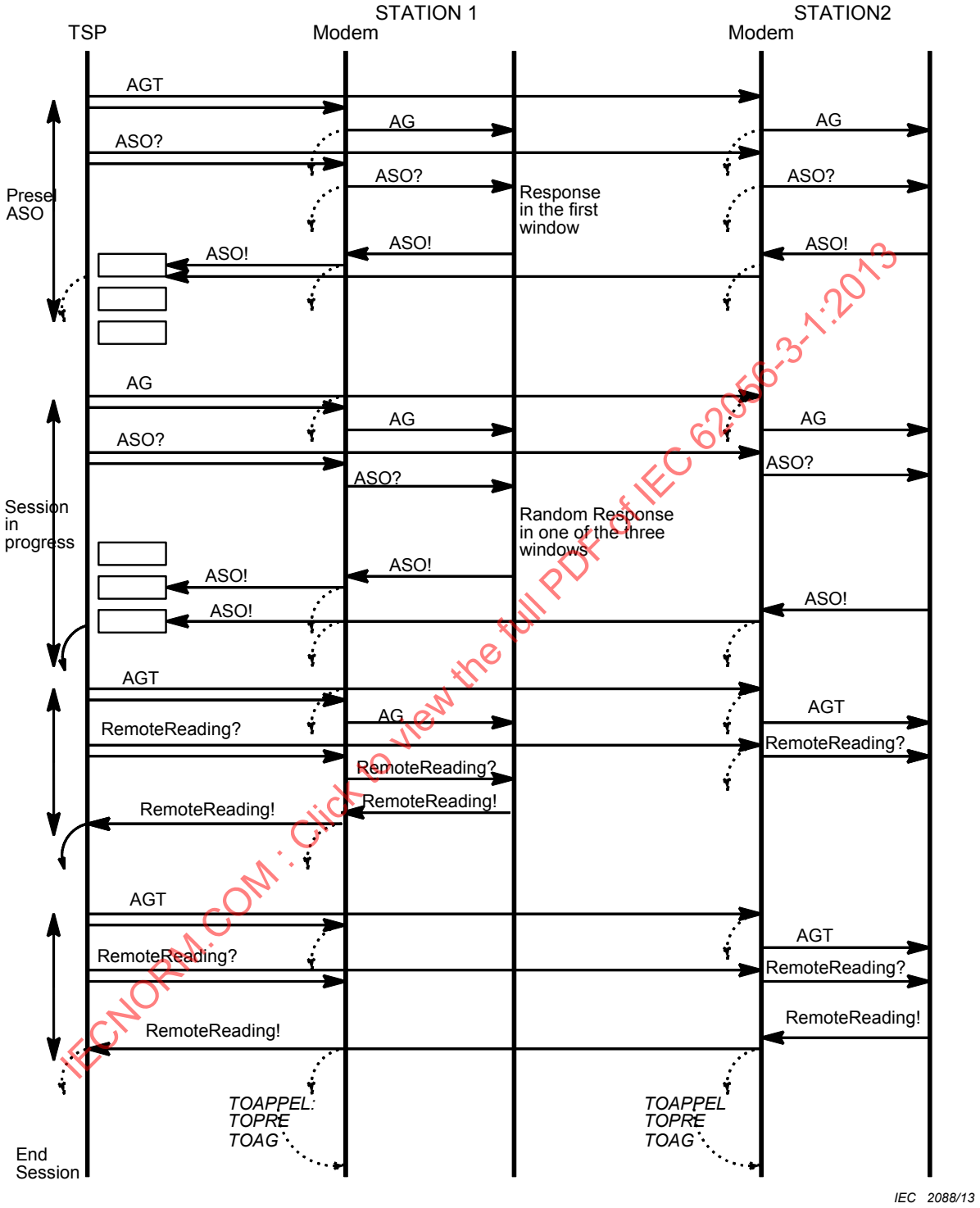


Figure I.4 – Forgotten station call exchange

In this case, two Secondary Stations are “forgotten stations”. At the first forgotten station call request (preselection request), both stations answer in the first time slot. At the next forgotten station call exchange, station 1 answers in the second time slot, while station 2 chooses the third time slot.

Bibliography

IEC 62056-6-1 Ed. 1.0: —, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object Identification System (OBIS)*

IEC 62056-6-2 Ed. 1.0:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ – LA SUITE DLMS/COSEM –

Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale CEI 62056-3-1 a été établie par le comité d'études 13 de la CEI: Mesure de l'énergie électrique, contrôle des tarifs et de la charge.

Cette première édition annule et remplace la première édition de la CEI 62056-31, parue en 1999, dont elle constitue une révision technique.

Les modifications techniques majeures par rapport à l'édition précédente sont les suivantes:

- Ajout d'un profil qui permet l'utilisation de la couche Application et la modélisation objet DLMS/COSEM de la CEI 62056,
- Révision de la couche liaison de données qui est maintenant scindée en deux parties:
  - la première est intégralement une couche de liaison de données;

  – la dernière, nommée «Gestion du Support», gère le média de communication;
- Capacité de négocier la vitesse de communication, portant la vitesse maximale jusqu'à 9 600 bauds.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
13/1546/FDIS	13/1552/RVD

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

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

Une liste de toutes les parties de la série CEI 62056, publiées sous le titre général *Échange des données de comptage de l'électricité – La suite DLMS/COSEM*, peut être consultée sur le site web de la CEI.

Les futures normes de cette série porteront dorénavant le nouveau titre général cité ci-dessus. Le titre des normes existant déjà dans cette série sera mis à jour lors de la prochaine édition.

La numérotation est passée de CEI 62056-XY à CEI 62056-X-Y. Par exemple, la CEI 62056-31 devient la CEI 62056-3-1.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ – LA SUITE DLMS/COSEM –

Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse

1 Domaine d'application

Cette partie de la CEI 62056 décrit trois profils pour échange de données par bus en local avec des stations alimentées ou non en énergie. Pour les stations télé-alimentées, le bus fournit l'énergie pour l'échange des données.

Trois différents profils sont supportés:

- Profil de base : ce profil en trois couches fournit des services de télé-relevé ;

  NOTE : Ce profil a été publié dans la CEI 61142-31:1993 et était alors connu sous le nom de Bus Euridis.

- Profil avec DLMS : ce profil permet l'utilisation des services DLMS tels qu'ils sont spécifiés dans la CEI 61334-4-41.

  NOTE Ce second profil a été publié dans la CEI 62056-31 Ed1.0 : 1999.

- Profil avec DLMS/COSEM : ce profil permet l'utilisation de la couche Application de DLMS/COSEM et le modèle objet COSEM tels qu'ils sont spécifiés respectivement dans la CEI 62056-5-3 Ed 1.0 :– et dans la CEI 62056-6-2 Ed1.0 :–.

Les trois profils utilisent la même couche physique et ils sont entièrement compatibles, c'est-à-dire que des équipements ayant implémenté l'un de ces profils peuvent opérer sur le même bus.

Le moyen de transmission est la paire torsadée par signal de porteuse et connue sous le nom de Bus Euridis.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 61334-4-41:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4: Protocoles de communication de données – Section 41: Protocoles d'application – Spécification des messages de ligne de distribution*

CEI 62056-51:1998, *Comptage de l'électricité – Echange de données pour la lecture des compteurs, le contrôle des tarifs et de la charge – Partie 51: Protocoles de couche application*

CEI 62056-5-3: Ed 1.0 —, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 5-3: Couche application DLMS/COSEM*

ISO/IEC 8482:1993, *Technologies de l'information – Télécommunications et échange d'informations entre systèmes – Interconnexions multipoints par paire torsadée* (disponible en anglais seulement)

EIA 485:—, *Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*

3 Abréviations

Abréviation	Signification
ADP	Adresse de la station primaire
ADG	Adresse secondaire générale. Adresse de diffusion.
ADS	Adresse de la station secondaire
AGN	Appel général normal
AGT	Appel général pour station en télé-alimentation
APDU	Application Protocol Data Unit (Unité de Données du Protocole d'Application)
APG	Adresse primaire générale
ARJ	Valeur du champ COM. Rejet d'authentification en téléprogrammation.
ASDU	Application Service Data Unit (Unité de données de service Application)
ASO	Valeur du champ COM. Appel de stations oubliées
AUT	Valeur du champ COM. Commande d'authentification
COM	Champ contrôle de la couche liaison
COSEM	Companion Specification for Energy Metering (Spécification d'accompagnement pour le comptage de l'énergie)
DAT	Valeur du champ COM. Réponse de télérelève
DES	Data Encryption Standard (Norme de crytage de données)
DLMS	Distribution Line Message Spécification (Specification de Message de la Ligne de Distribution)
DSDU	Data link Service Data Unit (Unité de données de Service Liaison)
DRJ	Valeur du champ COM. Data Reject. Valeur du champ COM notifiant le rejet des données de téléprogrammation.
Dsap	Etiquette des unités de données Transport. Codé sur 3 bits. Sa valeur est 6.
DTSAP	Destination Transport Service Access Point
ECH	Valeur du champ COM. Echo des données de téléprogrammation
ENQ	Demande de télérelève
EOS	Valeur du champ COM. Fin de téléprogrammation.
IB	Initialisation de bus
MaxRetry	Nombre maximum de retransmissions. Limité à 2.
MaxRSO	Nombre maximum de fenêtres d'écoute RSO. Fixé à 3
PDU	Protocol Data Unit (Unité de données du Protocole)
PRE	Valeur du champ COM. Présélection des stations télé-alimentées
REC	Valeur du champ COM. Demande de téléprogrammation
RSO	Valeur du champ COM. Réponse à, l'appel de stations oubliées
SEL	Valeur du champ COM. Acquittement de la présélection des stations télé-alimentées
STSAP	Source Transport Service Access Point
TAB	Code des données pour Euridis dans le profil sans DLMS ni DLMS/COSEM. Dans les profils DLMS ou DLMS/COSEM c'est la valeur à laquelle l'appareil est sensibilisé au Discover.
TABi	Liste de champ TAB
TASB	Durée d'un signal, d'alarme sur le bus
TOAG	Temps d'attente maximal pour une station télé-alimentée, une fois sélectionnée pour reconnaître un appel général AGN.
TOALR	Attente avant envoi d'un AGN après réception d'un AGN ou d'un AGT

Abréviation	Signification
TOL	Temps d'attente maximale d'une requête issue de la couche supérieure
TOPRE	Temps d'attente maximale de la réponse à une présélection
TPDU	Transport Protocol Data Unit (Unité de données de Protocole Transport)
TSDU	Transport Protocol Service Unit (Unité de données de Service Transport)
TRA	Valeur du champ COM. Acquittement de transfert en point à point
TRB	Valeur du champ COM. Télé transfert en diffusion non acquitté
TRF	Valeur du champ COM. Télé transfert point à point
T1	Délai de garde d'attente d'une réponse à une requête
XBA	Valeur du champ COM. Réponse à une requête de changement de vitesse
XBR	Valeur du champ COM. Requête de changement de vitesse
ZA1	Champ réservé pour l'authentification bidirectionnelle en programmation
ZA2	Champ réservé pour l'authentification bidirectionnelle en programmation

4 Présentation générale

4.1 Vocabulaire de base

Toute communication fait intervenir deux équipements représentés par les expressions Station Primaire et Station Secondaire. La Station Primaire est le système qui décide d'initialiser une communication avec un équipement distant dit Station Secondaire; ces dénominations restent valables pendant toute la durée de vie de la communication.

Une communication est décomposée en un certain nombre de transactions. Chaque transaction se traduit par une émission de l'Emetteur vers le Récepteur. Au gré de l'enchaînement des transactions, les systèmes Station Primaire et Station Secondaire jouent tour à tour le rôle d'Emetteur et de Récepteur.

Dans le cas du profil d'échange de données par bus local avec DLMS ou DLMS/COSEM, les termes Client et Serveur ont le même sens que dans le modèle DLMS (voir la CEI 61334-4-41 ou la CEI 62056-5-3 :–). Le Serveur (qui est obligatoirement une Station Secondaire) est le système qui reçoit et traite toute soumission de requête de service particulière. Le Client (qui est obligatoirement une Station Primaire) est le système qui utilise le Serveur dans un but spécifique à l'aide d'une ou de plusieurs soumissions de requête de service.

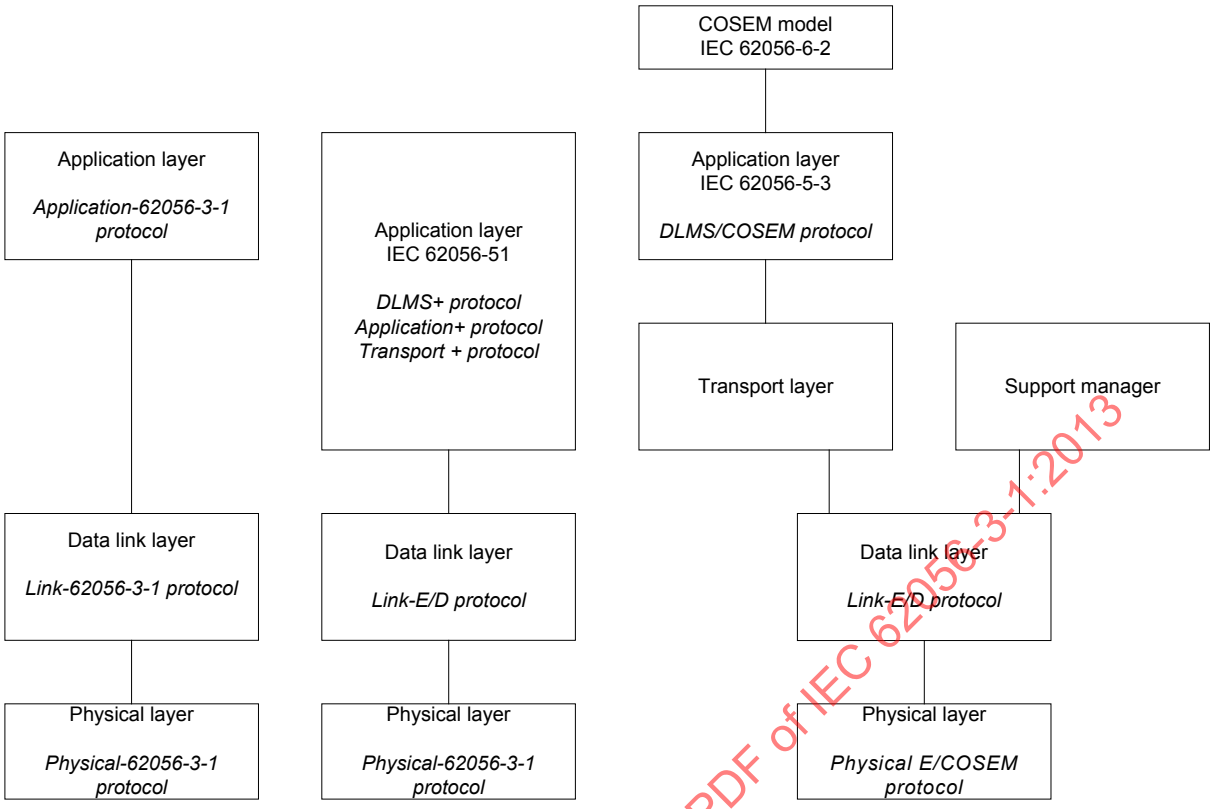
4.2 Profils, couches et protocoles

4.2.1 Généralités

Cette norme spécifie trois profils de communication, voir Figure 1.

- Profil sans DLMS, voir 4.2.2,
- Profil avec DLMS, voir 4.2.3,
- Profile avec DLMS/COSEM voir 4.2.4.

La couche physique sur les trois profils est la même à part le fait que le profil avec DLMS/COSEM permet la négociation de vitesse. Cette couche physique commune permet la coexistence de différents profils sur le même bus.



Légende

Anglais	Français
Application layer	Couche application
Application-62056-3-1 protocol	Protocole Application-62056-3-1
Data link layer	Couche liaison
Link-62056-3-1 protocol	Protocole Link-62056-3-1
Physical layer	Couche physique
Physical-62056-3-1 protocol	Protocole Physical-62056-3-1
Base architecture	Architecture de base
Application layer IEC 62056-51	Couche application CEI 62056-51
DLMS+ protocol Application+ protocol Transport+ protocol	Protocole DLMS+ Protocole Application+ Protocole Transport+
Data link layer Link-E/D protocol	Couche liaison Protocole Link-E/D
Physical layer	Couche physique
Physical-62056-3-1 protocol	Protocole Physical-62056-3-1
Architecture with DLMS	Architecture avec DLMS
COSEM model IEC 62056-6-2	Modèle COSEM CEI 62056-6-2
Application layer IEC 62056-5-3 DLMS/COSEM protocol	Couche application CEI 62056-5-3 Protocole DLMS/COSEM

Anglais	Français
Transport layer	Couche transport
Support Manager	Gestion du Support
Data link layer Link-E/D protocol	Couche liaison Protocole Link-E/D
Physical layer Physical E/COSEM protocol	Couche physique Protocole Physique E/COSEM
Architecture with DLMS/COSEM	Architecture avec DLMS/COSEM

Figure 1 – Profils de communication CEI 62056-3-1

4.2.2 Profil sans DLMS

Le profil de base sans DLMS utilise le protocole triple couches:

- La couche physique avec Physique-62056-3-1 dont le protocole est spécifié en 5.1;
- La couche liaison avec Link-62056-3-1 dont le protocole est spécifié en 5.2;
- La couche application avec Application-62056-3-1 dont le protocole est spécifiée en 5.3

Ce profil permet le télérelevé et la téléprogrammation, le transfert de données point à point – qui est un simplifié de service de téléprogrammation, le transfert par télédiffusion, la téléalimentation des stations secondaires, la détection de stations oubliées et les fonctions d'alarmes. Le détail des services concernés sont spécifiés en 4.4.

4.2.3 Profil avec DLMS

Le profil sans DLMS utilise le protocole triple couches:

- La même couche physique que le profil de base spécifié en 5.1.
- La couche liaison utilisant le protocole Link-E/D spécifié en 6.2 et
- La couche application avec Application-62056-51 utilisant les protocoles *Transport+*, *Application+* et *DLMS+*, voir 6.3.

Ce profil permet aussi l'utilisation de DLMS tel qu'il est spécifié dans la CEI 61334-4-41. Les services de communication concernés sont spécifié en 4.5.

4.2.4 Profil avec DLMS/COSEM

Le profil avec DLMS/COSEM utilise un protocole à quatre couches:

- La couche physique est similaire à celle utilisée dans le profil de base et dans le profil DLMS comme spécifié en 5.1, mais avec une négociation de vitesse, voir 7.2;
- La couche liaison de données utilise le protocole Link-/E/D. C'est la même que la couche liaison de données dans le profile avec DLMS à part qu'elle interface la couche gestion du support et la couche transport. Voir 7.3;
- La couche gestion du support prend en charge les processus spécifiques à la gestion du bus (voir 0);
- La couche transport met en place la segmentation et le réassemblage des APDU, voir 7.5;
- La couche application, telle qu'elle est spécifiée dans CEI 62056-5-3: Ed 1.0 :– , et prenant en compte quelques spécificités du bus Euridis, voir 7.6.

Le profil DLMS/COSEM permet l'utilisation de la modélisation objet COSEM et des services DLMS d'accessibilité aux objets, sur le bus Euridis.

4.3 Langage de spécification

Dans cette norme, le protocole de chaque couche est décrit par des transitions d'état représentées sous forme de tableaux. La syntaxe utilisée pour la constitution de ces tableaux est définie par un langage de spécification présenté à l'Annexe A.

En cas de divergence d'interprétation entre une partie du texte et un tableau de transitions d'état, c'est toujours le tableau qui fait référence.

4.4 Services de communication pour l'échange de données en bus local sans DLMS et COSEM

4.4.1 Généralités

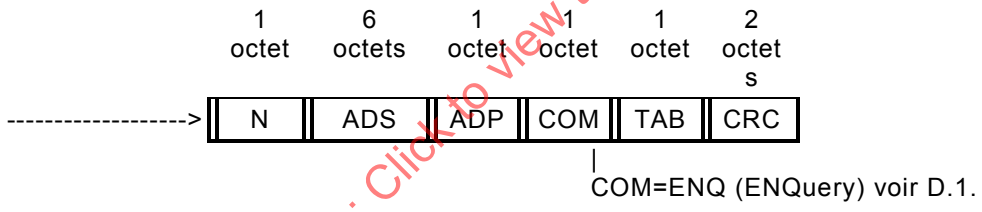
La liste des services disponibles au niveau de la couche Application est:

a) télérelève de données, 4.4.2;
b) téléprogrammation de données, 4.4.3;
c) télétransfert point à point, qui est un service de téléprogrammation simplifié, 4.4.4;
d) télétransfert en diffusion, 4.4.5;
e) initialisation du bus, 4.4.6;
f) appel des stations oubliées, 4.4.7.

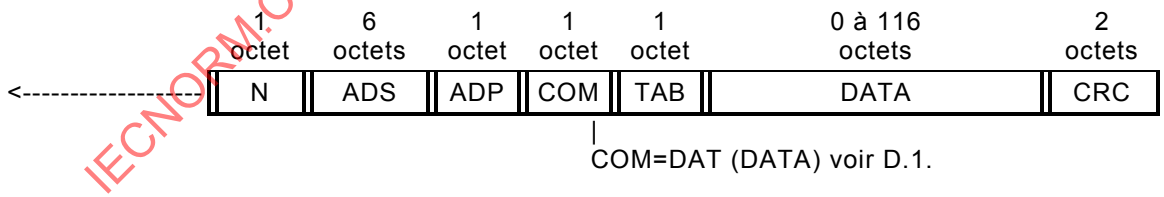
4.4.2 Télérelève

L'échange de télérelève est constitué de deux trames en une seule séquence:

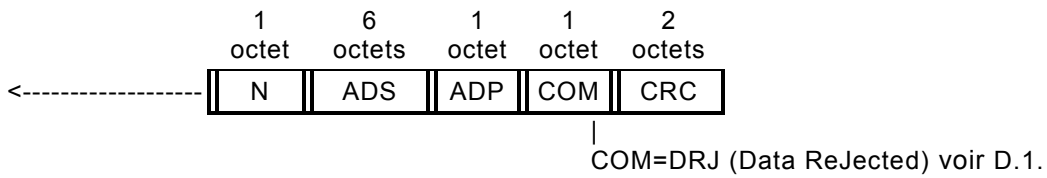
- trame de télérelève contenant dans le champ TAB la référence des données à relever:



- trame d'acquittement positif contenant les données relevées dans le champ DATA:



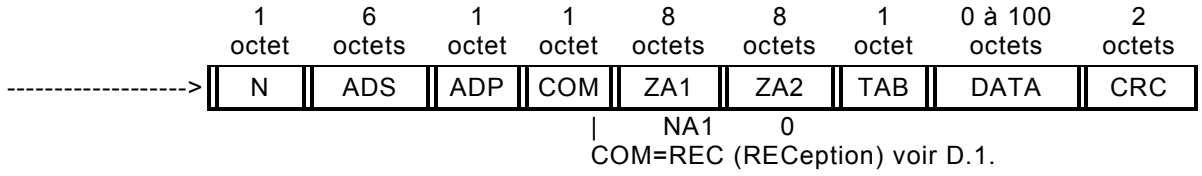
- trame d'acquittement négatif (référence TAB inconnue)



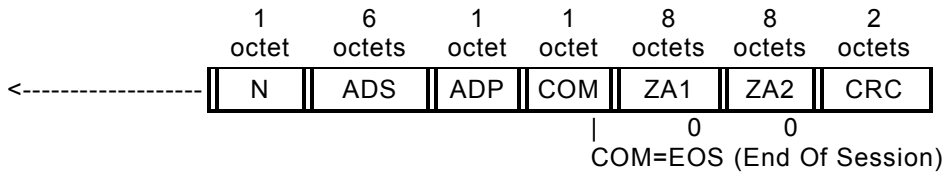
4.4.3 Téléprogrammation

L'échange REC est constitué de quatre trames en deux séquences. Comme il comporte une séquence interne pour l'authentification, il apparaît du point de vue de l'application, comme un échange de deux trames en une seule séquence:

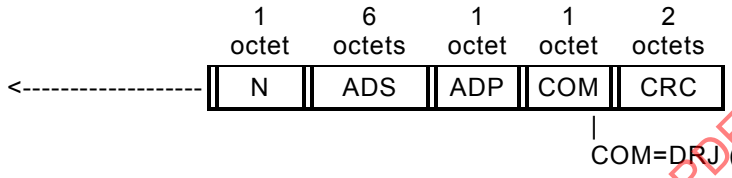
- trame de téléprogrammation contenant les données à programmer dans le champ DATA et leur référence dans le champ TAB



- trame d'acquittement positif (pas de problème d'authentification)



- trame d'acquittement négatif (pas de problème d'authentification mais les données à programmer ne sont pas valables)



L'authentification est réalisée grâce à un échange de nombres aléatoires cryptés au moyen d'une clef secrète propre à chaque Station Secondaire. Les nombres aléatoires sont générés avec une taille de 8 octets puis cryptés grâce à l'algorithme DES, en utilisant une clef Ki sur 8 octets connue des deux stations Primaire et Secondaire.

Un nombre aléatoire NA1 est d'abord généré par la Station Primaire puis transmis dans le champ ZA1 de la trame de téléprogrammation, tandis que le champ ZA2 est mis à zéro.

A l'arrivée sur la Station Secondaire, le champ ZA1 est crypté grâce à l'algorithme DES, en utilisant la clef Ki, opération qui génère le nombre aléatoire crypté NA1K. Puis les deux stations échangent les deux trames formant la séquence interne d'authentification.

La première trame (dans le sens Station Secondaire vers Station Primaire) contient le nombre aléatoire crypté NA1K dans le champ ZA1 et un nombre aléatoire NA2 généré par la station Secondaire dans le champ ZA2.

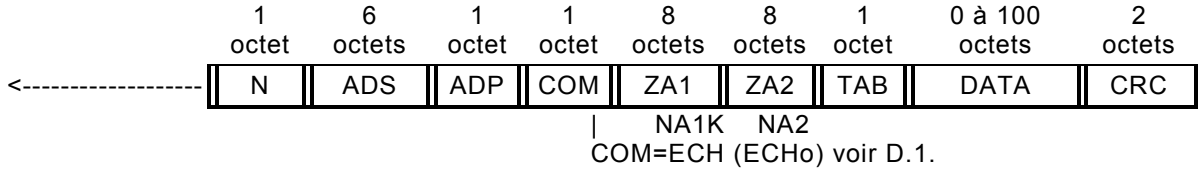
A la réception de cette trame, la Station Primaire compare le contenu du champ ZA1 au nombre NA1' obtenu en cryptant le nombre transmis NA1 au moyen de l'algorithme DES et de sa clef Ki. Si NA1' = ZA1, la Station Primaire considère que la Station Secondaire est bien authentifiée. Sinon, elle suppose que la Station secondaire n'est pas authentifiée et coupe la session de communication.

Dans le cas d'une authentification correcte de la Station Secondaire, la Station Primaire crypte le nombre aléatoire NA2 au moyen de l'algorithme DES et de sa clef Ki, obtenant le nombre aléatoire crypté NA2K qu'il transmet dans le champ ZA2, le champ ZA1 étant mis à zéro.

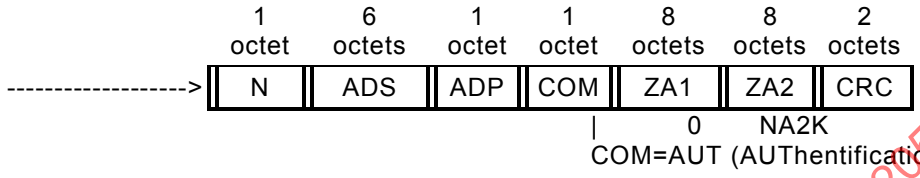
Sur réception de cette trame de réponse, la Station Secondaire compare le contenu du champ ZA2 au nombre NA2' obtenu en cryptant le nombre transmis NA2 au moyen de l'algorithme DES et de sa clef Ki. Si NA2' = ZA2, la Station Secondaire considère que la Station Primaire est bien authentifiée. Sinon, elle suppose que la Station Primaire n'est pas authentifiée et envoie une trame d'acquittement négatif.

L'échange interne d'authentification est le suivant:

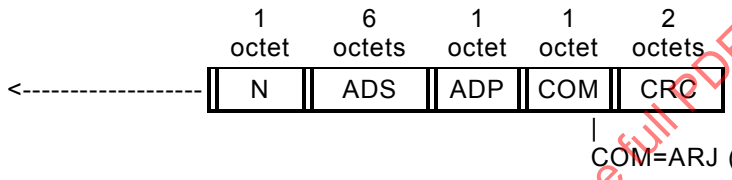
- trame interne d'authentification contenant le nombre aléatoire crypté NA1K dans le champ ZA1 et le nombre aléatoire NA2 dans le champ ZA2



- trame de réponse positive contenant le nombre aléatoire crypté NA2K dans le champ ZA2 (si la Station Secondaire est bien authentifiée)



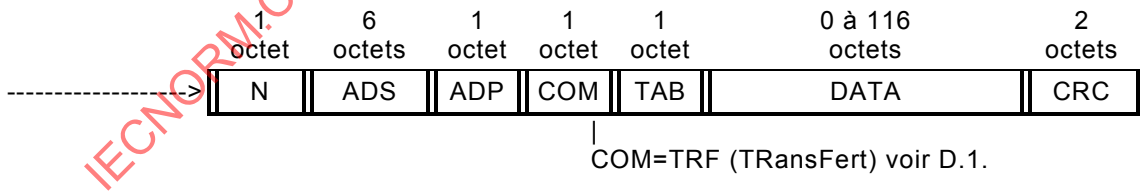
- trame de rejet d'authentification, remplaçant les trames normales EOS ou DRJ quand la Station Primaire n'a pas été convenablement authentifiée



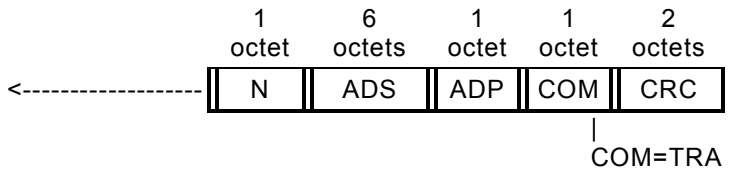
4.4.4 Télétransfert point à point

L'échange de télétransfert point à point est constitué de deux trames en une seule séquence. Du point de vue de l'application, il apparaît comme un échange de téléprogrammation en une seule séquence et sans authentification:

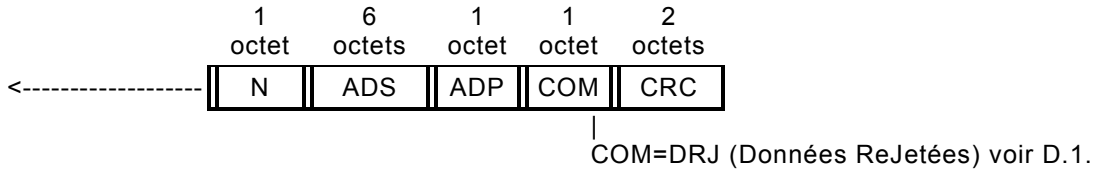
- trame de télétransfert point à point contenant les données à programmer dans le champ DATA et leur référence dans le champ TAB



- trame d'acquittement positif



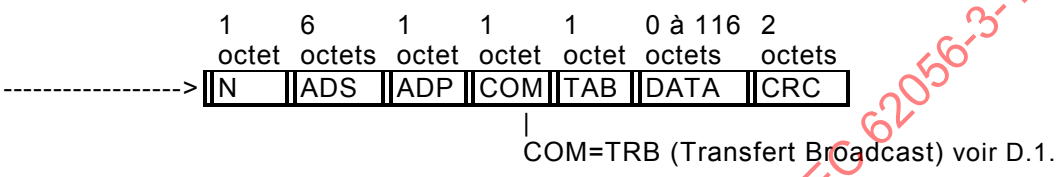
- trame d'acquittement négatif (les données transférées ne sont pas valables)



4.4.5 Télétransfert en diffusion

L'échange de télétransfert en diffusion ne comporte pas de trame de réponse. Du point de vue de l'application, il apparaît comme une séquence similaire à celle du télétransfert point à point, mais sans acquittement puisqu'il s'agit d'une diffusion.

- trame de télétransfert en diffusion contenant les données à programmer dans le champ DATA et leur référence dans le champ TAB

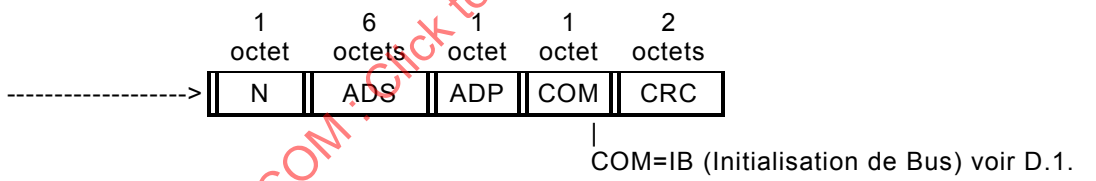


L'adresse secondaire (qui définit les Stations Secondaires destinataires) doit être une adresse de diffusion.

4.4.6 Initialisation du bus

La trame IB est sans acquittement puisqu'il s'agit d'une trame envoyée en diffusion. Elle ne véhicule pas de données, mais positionne à VRAI le drapeau de «station oubliée» pour toutes les Stations Secondaires sensibilisées à l'adresse primaire ADP véhiculée:

- trame d'initialisation de bus



L'adresse secondaire (qui définit les Stations Secondaires destinataires) doit être une adresse de diffusion.

Après le passage d'une trame d'initialisation de bus, toute Station Secondaire recevant une trame ENQ correcte avec une référence TAB connue ne sera plus considérée comme une «station oubliée».

4.4.7 Appel des stations oubliées

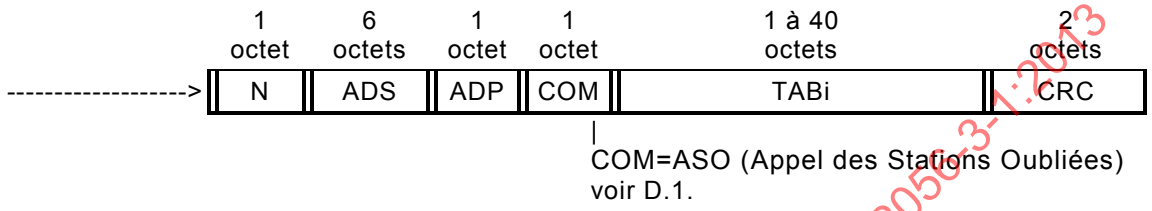
L'échange d'appel de stations oubliées est constitué de deux trames en une seule séquence. A la fin d'une séquence de télérelève, la Station Primaire peut chercher les stations dont le drapeau de «station oubliée» est à VRAI (maximum 5 sur un total de 100).

Comme un échange correct de télérelevé positionne à FAUX le drapeau de «station oubliée» de la station concernée, ce service n'est demandé normalement qu'à la fin d'une séquence de télérelève, formée d'un ou de plusieurs échanges de télérelève précédés d'une trame d'initialisation de bus.

La Station Primaire gère plusieurs intervalles de temps. Quand elle détecte une collision en retour, elle doit retenter un appel des stations oubliées. Chaque fois qu'elle reçoit une réponse correcte d'une Station Secondaire, la Station Primaire doit l'éliminer de la liste des stations oubliées en effectuant une télérelève correcte de cette station.

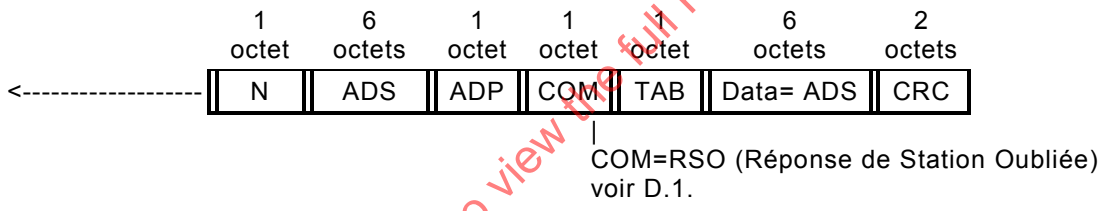
Afin de respecter les contraintes de présélection (décrites en 4.4.10), les stations télé-alimentées doivent répondre dans le premier intervalle de temps du premier échange ASO. Ensuite, seules sont sélectionnées les stations oubliées et le principe habituel peut être utilisé pour les échanges ASO suivants.

- appel des stations oubliées contenant des critères de sélection dans le champ TABi (1 à 40 références TAB)



Il convient que l'adresse secondaire (qui définit les Stations Secondaires destinataires) soit une adresse de diffusion.

- trame d'acquittement contenant la première référence TAB reconnue par l'unité, et l'ADS de la station



- Le champ données contient l'ADS de la station secondaire répondant à l'appel de stations oubliées.

4.4.8 Champs de la trame

N nombre total d'octets de la trame, y compris N.

ADS adresse physique absolue de la Station Secondaire codée comme un bit string de taille 48. Il y a une seule adresse physique de diffusion, l'adresse générale ADG codée "000000000000" en hexadécimal [1].
L'ADS correspond également au Titre-Système de la Station Secondaire.

ADP adresse physique de la Station Primaire codée comme un bit string de taille 8. La valeur "00"H est réservée au codage de l'adresse primaire générale APG [2]. N'importe quelle Station Secondaire sollicitée par une Station Primaire dont l'adresse primaire est APG, répond avec la première adresse physique primaire avec laquelle elle a été programmée.

COM code de la commande, dépendant de l'échange et de la direction de la trame (voir Annexe D).

---

[1] D'autres adresses de diffusion pourront être définies, en fonction des règles d'adressage adoptées dans les normes d'accompagnement pour la sémantique des Titres-Système, qui sont souvent basés sur un code de fabricant, une année de fabrication et un type d'équipement.

[2] D'autres adresses générales pourront être définies, en fonction des règles d'adressage adoptées dans les normes d'accompagnement pour la sémantique des identifications d'opérateurs, qui sont souvent basés sur un code utilitaire.

ZA1, ZA2 champs réservés pour l'authentification réalisée au cours d'un échange de téléprogrammation.

TAB référence des données sélectionnées, associée à certaines commandes (ENQ, DAT, REC, TRF, TRB ou RSO). La valeur "00"H est réservée aux systèmes d'administration, la valeur "FF"H à la gestion de la fonction alarme.

DATA message de données provenant de l'application; ce champ peut être vide dans le cas de certaines commandes.

CRC Champ de contrôle cyclique contenant les 16 bits redondants du CRC (Cyclic Redundancy Check) dont le principe est décrit à l'Annexe E.

Les champs de la trame sont transmis dans l'ordre croissant (de N au CRC). Quand un champ contient des données sur plusieurs octets, l'émission commence par l'octet de poids faible, et se termine par l'octet de poids fort. Seul le champ DATA est considéré comme un octet string et est émis dans un ordre croissant.

4.4.9 Principe de la télé-alimentation en énergie

Le principe général qui régit les échanges de données est préservé pour les stations télé-alimentées. Il est seulement ajouté une notion d'alimentation d'énergie à distance pour les communications entre une Station Primaire et une ou plusieurs Stations Secondaires.

Pour démarrer une session, la Station Primaire doit envoyer un «Appel Général», qui alerte le système de communication de chaque Station Secondaire connectée au bus. Cet appel correspond à la présence de la porteuse durant une durée nominale qui dépend du mécanisme d'alimentation:

- la durée du signal d'«Appel Général» est AGT pour réveiller les stations télé-alimentées,
- la durée du signal d'«Appel Général» est AGN pour réveiller les stations alimentées.

Remarque: Une station secondaire peut être configurée en mode alarme. Elle est alors télé-alimentée en continu, et peut ainsi transmettre une alarme à la station primaire, voir 4.4.11.

De plus, quel que soit le type de la station distante sélectionnée (alimentée ou non), un «Appel Général» AGN intermédiaire doit être envoyé par la Station Primaire dans les circonstances suivantes:

- avant le premier échange de type ENQ ou TRF;
- avant le sixième échange consécutif et réussi de type ENQ ou TRF avec la même Station Secondaire;
- avant le premier échange de type ENQ ou TRF avec une autre Station Secondaire que celle sélectionnée auparavant, lors du précédent échange de type ENQ ou TRF;
- avant n'importe quel échange de type REC;
- avant n'importe quelle trame TRB;
- avant n'importe quelle trame IB;
- avant n'importe quel échange de type ASO.

Cela permet à la Station Primaire de ne pas réveiller toutes les stations distantes télé-alimentées si ce n'est pas nécessaire, et ainsi de sauvegarder l'énergie.

Une Station Primaire peut être munie d'un modem spécifique assurant à la fois la télé-alimentation, et les fonctions de modulation-démodulation. La durée de communication et le nombre de stations télé-alimentées doivent être optimisés pour sauvegarder les batteries de la Station Primaire.

Dans le cas où la Station Primaire ne réaliserait que les fonctions de modulation-démodulation, une station auxiliaire fournit en continu l'énergie sur le bus.

Une Station Secondaire, qu'elle soit alimentée ou non, ne contient généralement qu'une seule application logique, référencée par son ADS.

Il convient qu'une Station Secondaire multiple (contenant plusieurs applications logiques, correspondant à plusieurs ADSs) soit une station télé-alimentée. Ce type de station est décrit plus complètement à l'Article 8.

4.4.10 Présélection d'une station télé-alimentée

Pour optimiser la consommation d'énergie, un échange de présélection permet à la Station Primaire de sélectionner une Station Secondaire télé-alimentée.

L'échange de présélection est réalisé après un «Appel Général» AGT adressé à toutes les stations télé-alimentées présentes sur le bus. Pour limiter la consommation sur le bus, il convient que la première trame envoyée par la Station Primaire soit suffisamment courte, et il convient que la Station Secondaire concernée réponde avant l'expiration du délai TOPRE. Le modem de la Station Secondaire retourne dans un état de basse consommation s'il ne détecte pas de réponse dans le temps imparti.

Durant l'échange de présélection, toutes les stations télé-alimentées consomment de l'énergie. La tension sur le bus et l'énergie stockée décroissent jusqu'à ce que les stations non sélectionnées retournent à un état de veille. Puis l'envoi continu d'énergie charge les éléments de stockage d'énergie et augmente la tension sur le bus.

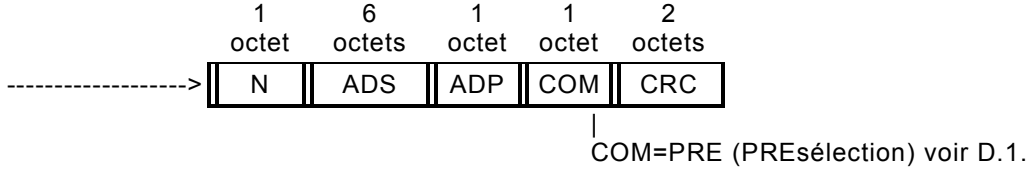
Il convient que le modem de la Station Primaire ait accumulé assez d'énergie avant la première présélection. Cette étape est garantie grâce à un temps d'attente contrôlé par le réveil TICB. A la fin de la présélection, les éléments de stockage d'énergie sont vides et la Station Primaire doit attendre la remontée de la tension sur le bus avant de réaliser une seconde présélection.

La trame de présélection doit avoir une taille maximale de 18 octets, elle peut correspondre à:

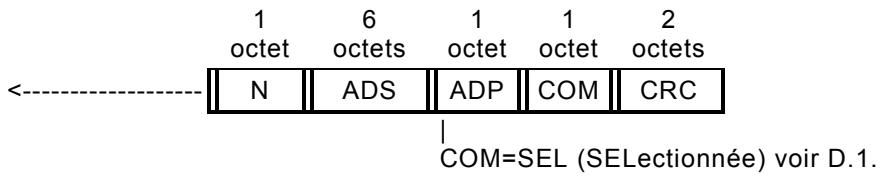
- une trame ENQ;
- une trame TRB ou TRF, si et seulement si le champ DATA a une taille inférieure ou égale à 6 octets;
- une trame IB;
- une trame ASO, si et seulement si le nombre de champs TABi est inférieur ou égal à 7.

Comme la première trame d'un échange de type REC ou TRF peut être trop grande, un service supplémentaire est fourni pour la présélection. Cet échange complètement transparent est formé d'une séquence de deux trames.

- trame de présélection d'une station télé-alimentée



- trame d'acquittement



Pour préserver l'énergie de la Station Primaire, il n'y a pas de reprise lors d'un échange de présélection. Si une station télé-alimentée ne répond pas correctement, elle n'est pas sélectionnée, et la Station Primaire doit renvoyer un nouvel «Appel Général» AGT.

4.4.11 Communication après la présélection

Après la présélection, le modem de la station télé-alimentée peut rester éveillé tout le long de la communication; les délais ne sont plus critiques puisque le nombre d'équipements connectés est limité. La Station Primaire alimente la station sélectionnée et charge les accumulateurs des stations non sélectionnées.

La fin normale d'une session a lieu différemment suivant que la Station Secondaire est alimentée ou non:

- après une courte période d'inactivité au cours de la session, alors qu'un «Appel Général» AGN intermédiaire n'est pas requis, pour les stations alimentées. Cette période est contrôlée par le réveil TOL;
- après une période d'inactivité plus longue durant une session, pour une station télé-alimentée. Cette période est contrôlée par le réveil TOAG.

Noter que, pour une station télé-alimentée, un «Appel Général» intermédiaire AGN est suffisant pour poursuivre la session courante tant qu'il n'y a pas eu déclenchement du réveil TOAG.

4.4.12 Fonction Alarme

Un équipement intégré dans une station secondaire simple ou multiple (voir 8.2.3) peut transmettre des alarmes à une station primaire, à condition d'intégrer les fonctions d'interface définies ci-dessous.

Une alarme doit être récupérée sur la station secondaire dans les 10 s au maximum.

Une configuration programmable, sur l'Interface et sur chaque équipement, permet de choisir le mode Alarme: Actif ou Inactif.

Quand le mode Alarme est Actif, un équipement à l'intérieur de la station secondaire peut générer une alarme. La fonction alarme n'est effective que si l'alimentation en énergie est présente et permanente sur le bus.

L'équipement envoie le signal d'alarme pendant un délai TASB. TASB est assez important pour forcer à un état «0» le bus secondaire et être détecté par l'Interface, même s'il y a une communication en cours.

Le mécanisme d'alarme est décrit à la Figure 2.

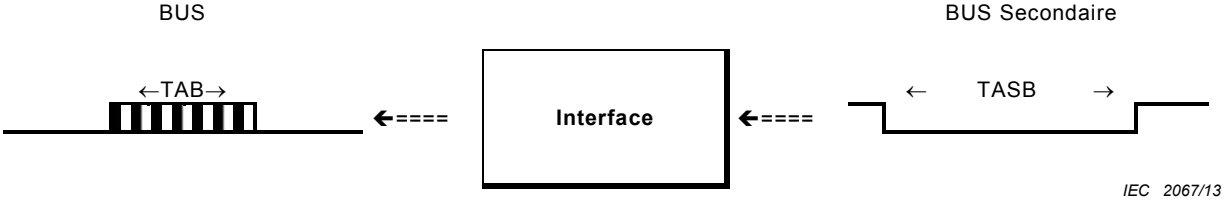


Figure 2 – Mécanisme d'alarme

L'alarme n'est pas directement transmise vers la station primaire. Elle est reçue par l'Interface qui la retransmet en envoyant des «0» (porteuse à 50 kHz) durant TAB sur le bus, dès que c'est possible:

a) Aucune communication sur le bus.

Quand l'Interface reçoit l'alarme en provenance du bus secondaire, elle la retransmet sur le bus.

b) En se synchronisant sur un AGN ou un AGT si une communication est en cours. Quand une communication est en cours sur le bus, l'Interface mémorise l'alarme reçue. Elle la retransmet sur le bus après l'un des événements suivants:

- TOALR après la fin de la réception d'un AGN ou d'un AGT;
- à la fin de la communication en cours.

Ainsi, l'Interface peut filtrer les alarmes pour éviter les conflits sur le bus.

Après la génération d'une alarme, une Station Secondaire sera considérée comme une «station oubliée» avec un critère de sélection égal à FF.

Une Station Primaire configurée en mode Alarme écoute le bus en dehors des communications et après l'émission d'un AGN ou d'un AGT pour détecter la présence d'une alarme. Après la réception d'une alarme, une Station Primaire effectue une procédure d'appel des stations oubliées, avec un critère de sélection du champ TABi égal à FF (voir 4.4.7).

La gestion des alarmes est illustrée en 0 par les diagrammes de temps.

4.5 Services de communication pour l'échange de données par bus local avec DLMS

DLMS n'offre pas de services permettant l'initialisation du bus et l'appel des stations oubliées. Néanmoins, la trame IB et l'échange ASO sont supportés et gérés comme pour l'échange de données par bus local sans DLMS. La seule différence est liée au fait que le drapeau de station oubliée est considéré comme une variable globale partagée avec l'interface de programmation de l'application.

La télérelève et le télétransfert en point à point sont réalisés directement par DLMS. Mais la téléprogrammation de données (redondante) n'est pas supportée puisque l'authentification est réservée à la couche *Application*.

Comme la sémantique des données est gérée par DLMS, le format des trames est très simple, et ne nécessite que des trames banalisées. Pour assurer la compatibilité avec le profil sans DLMS, le format des trames est défini à l'aide des neuf champs suivants:

1 octet	6 octets	1 octet	3 bits	1 bit	2 bits	2 bits	0 à 117 octets	2 octets
Size	ADS	ADP	DATA+	Priority	Send	Confirm	Text	CRC

Size nombre total d'octets de la trame, incluant Size. Si ce nombre diffère de 11, le récepteur sait que la trame contient des données dans le champ Text.

ADS même signification que pour le profil en bus local sans DLMS.

ADP même signification que pour le profil en bus local sans DLMS.

DATA+ toujours codé "111"B.

Priority niveau de priorité d'émission de la trame courante. La couche *Application* met la priorité en fonction du service demandé.

Send numéro de la dernière trame envoyée.

Confirm numéro de la dernière trame reçue sans erreur.

Text DSDU (Data link Service Data Unit) du niveau supérieur. Une trame ne contient pas nécessairement du texte. Si des données en provenance de la couche *Application* sont disponibles au moment de l'envoi de la trame, elles seront mises dans le champ *Text*, autrement celui-ci sera vide. Cela permet l'échange symétrique et bidirectionnel des données. Afin de ne pas confondre les trames de type DATA+

avec les trames du profil sans DLMS, les champs DATA+, Priority, Send et Confirm correspondent à un code de commande spécial, COM, dont les valeurs sont différentes de celles déjà réservées pour le champ COM (voir l'Annexe D).

CRC même signification que pour le profil en bus local sans DLMS.

Les champs de la trame sont transmis dans l'ordre croissant (de Size au CRC). Quand un champ est codé sur plusieurs octets, l'émission commence par l'octet de poids faible, et se termine par l'octet de poids fort. Seul, le champ Text est considéré comme un octet string et est émis dans un ordre croissant.

4.6 Système d'administration

L'objet du système d'administration est de permettre un enrôlement qui comporte l'identification des stations secondaires sur un bus. Il offre pour ce faire, le service Discover.

L'enrôlement consiste en une séquence de demande Discover effectuée par l'initiateur actif situé dans la Station Primaire. Chaque service Discover sert à informer les nouvelles stations qu'elles auront une possibilité de répondre durant les prochains intervalles de temps.

La demande Discover contient un paramètre correspondant à une probabilité de réponse spécifique. Ce paramètre est un entier compris dans l'intervalle [0, 100]. Il donne la probabilité de réponse d'une nouvelle station en pourcentage. Quand il vaut 100, toutes les nouvelles stations sur le bus doivent répondre.

Sur réception d'un Discover Indication, chaque Station Secondaire teste la valeur de son drapeau Discovered. S'il est égal à TRUE, l'Indication est détruite; sinon la station tire un nombre aléatoire entre 1 et 100. Si ce nombre est plus petit ou égal au paramètre de probabilité de réponse, la nouvelle station répond au Discover et met le drapeau associé à TRUE.

La réception d'une trame IB remet la valeur du drapeau à FAUX.

Pour assurer une compatibilité maximale (aussi bien pour les stations DLMS ou DLMS/COSEM que pour les autres), il est proposé d'implémenter le service d'administration conformément à ce qui est indiqué à l'Annexe H.

5 Echange de données par bus local sans DLMS

5.1 Couche Physique

5.1.1 Protocole Physique-62056-3-1

Le protocole *Physique-62056-3-1* de la couche *Physique* du profil d'échange de données par bus local sans DLMS adopte un comportement asymétrique. La machine d'état de la Station Primaire est donc différente de celle de la Station Secondaire.

Le protocole *Physique-62056-3-1* supporte les Stations Secondaires télé-alimentées ou non. Ainsi qu'il est indiqué dans la description générale, les stations distantes sont réveillées par un signal «Appel Général», de type AGN ou AGT, et une session se termine à l'expiration du réveil TOL ou TOAG.

Après un signal «Appel Général», les communications se déroulent en asynchrone et en semi-duplex à 1 200 bits/s sur le bus.

5.1.2 Paramètres de physique

La valeur de la taille maximale MaxIndex d'une trame en réception est fixée à 128.

La valeur du nombre maximal MaxRSO d'intervalles de temps RSO pour le traitement d'une trame «Appel des Stations Oubliées» est fixée à 3.

La durée du signal «Appel Général» AGN doit être dans l'intervalle [50, 150[ms, tandis que celle du signal «Appel Général» AGT doit être dans l'intervalle [200, 300[ms.

Les caractéristiques et les types des réveils et des temporisations sont donnés à l'Annexe B.

Les valeurs données dans le Tableau 1 sont définies pour une Station Primaire.

Tableau 1 – Temps d'une Station Primaire

	Min. ms	Nominal ms	Max. ms	Type, voir Article B.1	Définition
TA1O	–	-	120	TSL1	Temps d'attente maximal du premier octet d'une trame en réception
TAB		100		TC	Durée d'un signal d'alarme sur le bus
TAGN	–	100	–	TPDF	Durée d'un signal " Appel Général " AGN
TAO	–	–	40	TC	Temps d'attente maximal d'un octet en réception dont l'expiration indique implicitement la fin d'une trame
TARSO	–	500	–	TC	Durée d'un intervalle de temps pour le RSO
TASB		1 200		TC	Période d'attente après le début d'un signal d'alarme
TEMPO	–	40	–	TC	Délai de temporisation en fin d'émission d'un «Appel Général» ou d'une trame
TOE	–	–	2 500	TL	Délai de garde en émission pour se protéger contre un défaut matériel
TOL	–	–	100	TSL2	Temps d'attente maximal d'une requête issue de la couche supérieure
T1	-	10 000	-	TL	Temps d'attente maximal d'une réponse de la station secondaire
Spécifique d'une station télé-alimentée (Alimentation)					
TAGT	–	250	–	TPDF	Durée d'un signal «Appel Général» AGT
TICB	8 000	–	–	Ta	Délai initial de chargement du bus
TOAG	–	–	3 000	TPFD	Temps d'attente maximal pour une station télé-alimentée, une fois sélectionnée, pour reconnaître un signal «Appel Général» AGN

Les valeurs données dans le Tableau 2 sont définies pour une Station Secondaire.

Tableau 2 – Temps d'une Station Secondaire

	Min. ms	Nominal ms	Max. ms	Type[a]	Définition
TA1O	30[b]	–	160	TSL1	Temps d'attente maximal du premier octet d'une trame en réception
TAB	–	100	–	TC	Durée d'un signal d'alarme sur le bus
TAGN	50	100	150	TPDF	Durée d'un signal «Appel Général» AGN
TAO	–	–	40	TC	Temps d'attente maximal d'un octet d'une trame en réception dont l'expiration indique implicitement la fin d'une trame
TARSO	–	–	500	TC	Durée d'un intervalle de temps pour le RSO
TOALR	20	–	–	TL	Attente avant l'envoi d'un AGN après la réception d'un AGN ou d'un AGT
TOE	–	–	2 500	TL	Délai de garde en émission pour se protéger contre un défaut matériel
TOL	–	–	100	TSL2	Temps d'attente maximal d'une requête issue de la couche supérieure
Spécifique d'une station télé-alimentée (Alimentation)					
TAGT	200	250	300	TPDF	Durée d'un signal " Appel Général " AGT
TASB	–	1 200	–	TL	Durée d'un signal d'alarme sur le bus secondaire
TICB	8 000	–	–	Ta	Délai initial de chargement du bus
TOAG	–	–	3 000	TPFD	Temps d'attente maximal pour une station télé-alimentée, une fois sélectionnée, pour reconnaître un signal " Appel Général " AGN
TOAGN	–	–	300	Tc	Temps d'inactivité maximal pour reconnaître la fin d'une session de communication avec une station télé-alimentée
TOAPPEL	–	–	180	TPFD	Temps d'attente maximal de réception du premier octet d'une trame de présélection
TOBAVARD	–	–	260	TPDF	Délai de garde pour se protéger contre les trames de présélection défectueuses
TOPRE	–	–	130	TPFD	Temps d'attente maximal de la réponse à une présélection
TOSEUIL	–	150	–	TC	Durée d'un signal «Appel Général» qui réveille une station télé-alimentée
TVASB	40	–	–	TL	Durée minimale d'un signal d'alarme sur le bus secondaire

[a] Pour la définition des différents types de temps, voir Article B.1.

[b] Après un «Appel Général», une durée minimale de 30 ms est nécessaire.

5.1.3 Diagrammes de temps

Les Figure 3, Figure 4, et Figure 5 illustrent différents types d'échanges pour les stations secondaires télé-alimentées.

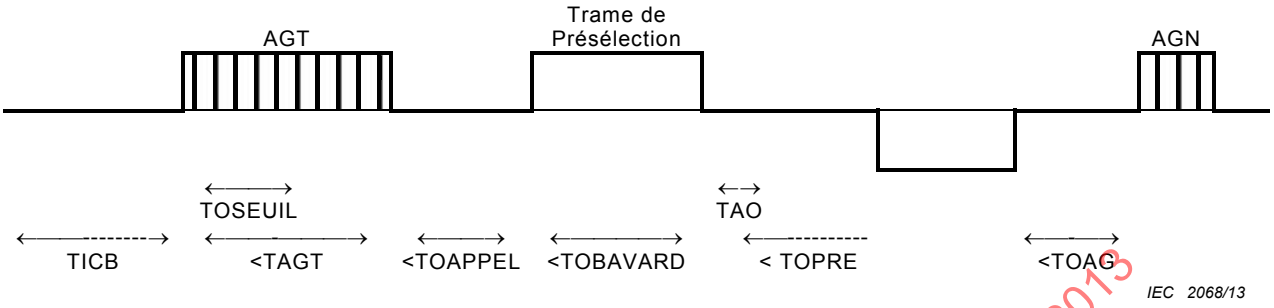


Figure 3 – Echanges sans interruption

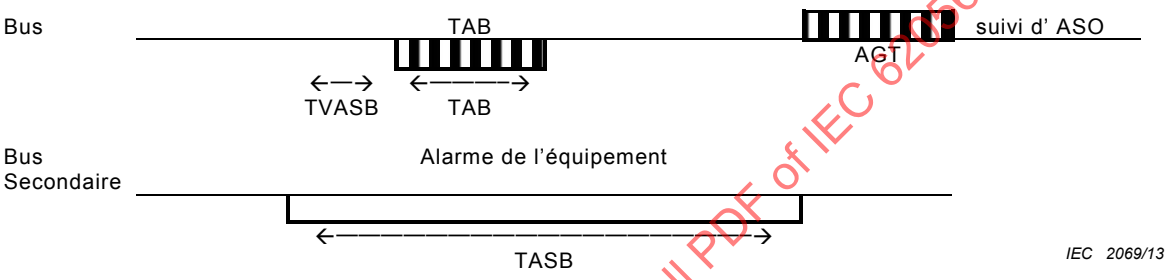


Figure 4 – Alarme alors qu'il n'y a pas de communication sur le bus

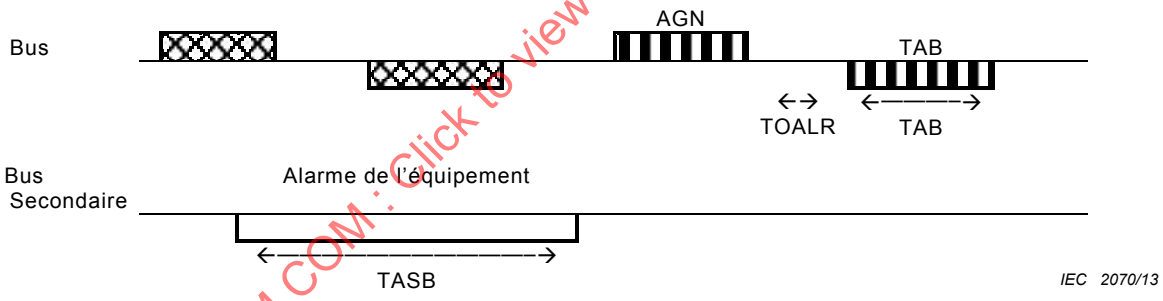


Figure 5 – Alarme alors qu'il y a une communication en cours sur le bus

5.1.4 Services et primitives de service de physique

L'utilisateur du protocole *Physique-62056-3-1* dispose des services et primitives de service donnés dans le Tableau 3.

Tableau 3 – Services et primitives de services de physique

Service	Primitive de service
Phy_DATA	Phy_DATA.req(Frame) Phy_DATA.ind(Frame)
Phy_UNACK	Phy_UNACK.req(Frame)
Phy_APPG	Phy_APPG.req(TypeAG) Phy_APPG.ind()
Phy_ASO	Phy_ASO.req(Frame) Phy_ASO.ind(Frame)
Phy_RSO	Phy_RSO.req(Frame, Window)
Phy_COLL	Phy_COLL.ind()
Phy_ALARM	Phy_ALARM.req() Phy_ALARM.ind()
Phy_ABORT	Phy_ABORT.req() Phy_ABORT.ind(ErrorNb)

Le rôle attribué à chaque primitive est le suivant:

- Phy_DATA.req(Frame) permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame Frame;
- Phy_DATA.ind(Frame) permet à la couche *Physique* d'informer la couche *Liaison* qu'une trame Frame est disponible;
- Phy_UNACK.req(Frame) permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame Frame sans attendre d'acquittement;
- Phy_APPG.req(TypeAG) permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'un signal «Appel Général». La durée TypeAG du signal est soit AGN soit AGT;
- Phy_APPG.ind() permet à la couche *Physique* d'informer la couche *Liaison* de la fin d'émission d'un signal «Appel Général»;
- Phy_ASO.req(Frame) permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame «Appel des Stations Oubliées»;
- Phy_ASO.ind(Frame) permet à la couche *Physique* d'informer la couche *Liaison* qu'une trame Frame a été reçue dans l'un des intervalles de temps réservés pour la réponse des stations oubliées;
- Phy_RSO.req(Frame, Window) permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame Frame de Réponse de Stations Oubliées dans l'intervalle de temps de numéro Window;
- Phy_COLL.ind() permet à la couche *Physique* d'informer la couche *Liaison* qu'une collision a été détectée dans l'un des intervalles de temps de réponse des stations oubliées;
- Phy_ALARM.req() permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'un signal «Alarme»;
- Phy_ALARM.ind() permet à la couche *Physique* d'informer la couche *Liaison* qu'une Alarme a été détectée;
- Phy_ABORT.req() permet à la couche *Liaison* de demander à la couche *Physique* de mettre fin à son activité;

- Phy_ABORT.ind(ErrorNb) permet à la couche *Physique* d'informer la couche *Liaison* de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

5.1.5 Transitions d'état

Tableau 4 – Transitions d'état de Physique-62056-3-1: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRSO=3 MaxIndex=128 Collision=FALSE SessionAGT=FALSE wait_time(TICB)	Stopped
Stopped	Phy_APPG.req(AG) & AG=AGN	stop_timer(TOAG) FlagAbort=FALSE TypeAG=AGN send_AG(TypeAG)	W.AG
Stopped	Phy_APPG.req(AG) & AG=AGT	SessionAGT=TRUE FlagAbort=FALSE TypeAG=AGT send_AG(TypeAG)	W.AG
Stopped	time_out(TOAG)	Phy_ABORT.ind(EP-2) SessionAGT=FALSE	Stopped
Stopped	Phy_ABORT.req()	$none()	Stopped
Stopped	data-carrier_on	init_timer(TAB) init_timer (TASB)	W.ETABS
W.ETABS	data_carrier_off	stop_timer(TASB) stop_timer(TAB)	Stopped
W.ETABS	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind()	W.TASB
W.AG	AG_sent_event	Phy_APPG.ind() init_timer(TEMPO)	W.TAB
W.AG	Phy_ABORT.req()	FlagAbort=TRUE	W.AG
W.TAB	data-carrier_on	Carrier = TRUE init_timer(TAB) init_timer (TASB)	W.TAB
W.TAB	data-carrier_off	Carrier = FALSE stop_timer(TAB) stop_timer (TASB)	W.TAB
W.TAB	time_out(TEMPO) & not(FlagAbort) & not(Carrier)	init_timer(TOL)	M.Send
W.TAB	time_out(TEMPO) & FlagAbort & not(Carrier)	wait_time (TOL)	*T.Session*
W.TAB	time_out(TEMPO) & Carrier	init_timer(TOL)	W.ETAB
W.TAB	Phy_ABORT.req()	FlagAbort=TRUE	W.TAB
W.ETAB	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind() stop_timer(TOL)	W.TASB
W.ETAB	data_carrier_off & not(FlagAbort)	stop_timer(TAB) stop_timer (TASB)	M.Send
W.ETAB	data_carrier_off & FlagAbort	stop_timer(TAB) stop_timer (TASB)	W.TOL
W.ETAB	Phy_ABORT.req()	FlagAbort=TRUE	W.ETAB
W.TASB	time_out(TASB)	$none()	Stopped
W.TOL	time_out(TOL)	$none()	*T.Session*
M.Send	Phy_DATA.req(Frame)	Service=NORMAL	*SendFirst*
M.Send	Phy_UNACK.req(Frame)	Service=UNACKNOWLEDGED	*SendFirst*

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Send	Phy_ASO.req(Frame)	Service=ASO	*SendFirst*
M.Send	Phy_ABORT.req()	$none()	M.Send
M.Send	time_out(TOL)	$none()	*T.Session*
T.Session	SessionAGT=TRUE	init_timer(TOAG) Phy_ABORT.ind(EP-1) wait_time(TEMPO)	Stopped
T.Session	SessionAGT=FALSE	Phy_ABORT.ind(EP-1) wait_time(TEMPO)	Stopped
SendFirst	$true()	stop_timer(TOL) Size=size(Frame) Index=1 send_octet(Frame, Index) Size=Size-1 init_timer(TOE)	Sending
Sending	octet_sent_event & Size>0	Index=Index+1 send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0	stop_timer(TOE) wait_time(TAO) Index=1 Frame=""	*Answer*
Sending	Phy_ABORT.req()	stop_timer(TOE) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Answer	Service=NORMAL I Service=UNACKNOWLEDGED	init_timer(TA1O)	M.Rec
Answer	Service=ASO	WinRSO=1 init_timer(TARSO) init_timer(TA1O)	M.Rec
M.Rec	octet_received_event	stop_timer(TA1O) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
M.Rec	collision_detected_event	stop_timer(TA1O) Collision=TRUE init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	$none()	*Received*
M.Rec	Phy_ABORT.req()	FlagAbort=TRUE	M.Rec
Receiving	octet_received_event & Index<=MaxIndex	stop_timer(TAO) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Phy_ABORT.ind(EP-4F) wait_time(TAO) FlagAbort=TRUE	*Received*
Receiving	collision_detected_event	stop_timer(TAO) Collision=TRUE init_timer(TAO)	Receiving
Receiving	time_out(TAO)	$none()	*Received*
Receiving	time_out(TARSO)	Phy_ABORT.ind(EP-5F) wait_time(TAO) FlagAbort=TRUE	*Received*

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Receiving	Phy_ABORT.req()	Flagabort = TRUE	Receiving
Received	Service=NORMAL & not(Flagabort)	Phy_DATA.ind(Frame) init_timer(TOL)	M.Send
Received	(Service=NORMAL & Flagabort) | Service=UNACKNOWLEDGED	wait_time(TOL)	*T.Session*
Received	Service=ASO & Collision & not(Flagabort)	Phy_COLL.ind() Collision=FALSE	*T.RSO*
Received	Service=ASO & not(Collision) & not(Flagabort)	Phy_ASO.ind(Frame)	*T.RSO*
Received	Service=ASO & Flagabort	$none()	*T.RSO*
T.RSO	(TypeAG=AGT) I (WinRSO>=MaxRSO) & (TypeAG=AGN)	stop_timer(TARSO)	*T.Session*
T.RSO	(WinRSO<MaxRSO) & (TypeAG=AGN)	Index=1 Frame=""	W.RSO
W.RSO	time_out(TARSO)	WinRSO=WinRSO+1 init_timer(TARSO) init_timer(TA1O)	M.Rec
W.RSO	Phy_ABORT.req()	Flagabort=TRUE	W.RSO

Tableau 5 – Transitions d'état de la gestion d'alimentation en énergie (Station Secondaire télé-alimentée seulement)

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	alarm_detection()	Flagalarm=TRUE FlagSendAlarm =FALSE station_power(ON)	Stopped
Initial	not(alarm_detection())	Flagalarm=FALSE	Stopped
Stopped	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB2
Stopped	occur(data_carrier_on)	init_timer(TOSEUIL) init_timer(TAGT)	W.TOSEUIL
W.TOSEUIL	time_out(TOSEUIL)& not(Flagalarm)	station_power(ON)	W.AGT
W.TOSEUIL	occur(data_carrier_off) & not(Flagalarm)	stop_timer(TOSEUIL) stop_timer(TAGT)	Stopped
W.TOSEUIL	time_out(TOSEUIL) & Flagalarm	station_signal(ON) Tend = TOAG	W.AGT
W.TOSEUIL	occur(data_carrier_off) & Flagalarm	stop_timer(TOSEUIL) stop_timer(TAGT) Tend = TOAGN init_timer(Tend)	Hide
W.TOSEUIL	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB1
W.AGT	occur(data_carrier_off)	stop_timer(TAGT) init_timer(TOAPPEL)	W.Sel
W.AGT	time_out(TAGT) & not(Flagalarm)	station_power(OFF)	Stopped
W.AGT	time_out(TAGT) & Flagalarm	init_timer(Tend)	Hide
W.Sel	occur(octet_received_event)	stop_timer(TOAPPEL) init_timer(TOBAVARD) init_timer(TAO)	Select
W.Sel	time_out(TOAPPEL) & not(Flagalarm)	station_power(OFF)	Stopped
W.Sel	time_out(TOAPPEL) & Flagalarm	station_signal(OFF)	Stopped
W.Sel	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Select	occur(octet_received_event)	stop_timer(TAO) init_timer(TAO)	Select
Select	time_out(TAO)	stop_timer(TOBAVARD) init_timer(TOPRE)	W.Answer
Select	time_out(TOBAVARD) & not(Flagalarm)	stop_timer(TAO) station_power(OFF)	Stopped
Select	time_out(TOBAVARD) & Flagalarm	stop_timer(TAO) init_timer(Tend)	Hide
W.Answer	occur(octet_sent_event)	stop_timer(TOPRE) init_timer(Tend)	Hide
W.Answer	time_out(TOPRE) & not(Flagalarm)	station_power(OFF)	Stopped
W.Answer	time_out(TOPRE) & Flagalarm	init_timer(Tend)	Hide
W.Answer	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Hide	occur(octet_received_event) | occur(octet_sent_event) I (occur(data_carrier_on) & not(FlagSendAlarm))	stop_timer(Tend) init_timer(Tend)	Hide
Hide	occur(data_carrier_on) & FlagSendAlarm	stop_timer(Tend)	W.AGend
Hide	time_out(Tend) & not(Flagalarm)	station_power(OFF)	Stopped
Hide	time_out(Tend) & Flagalarm & not(FlagSendAlarm)	station_signal(OFF)	Stopped
Hide	time_out(Tend) & Flagalarm & FlagSendAlarm	Send_AG(AGN)	W.AB
Hide	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
W.AGend	occur(data_carrier_off)	wait_time(TOALR) Send_AG(AGN)	W.AB
W.TVASB1	occur(cpt_carrier_off)	stop_timer(TVASB) init_timer(Tend)	Hide
W.TVASB1	time_out(TVASB)	FlagSendAlarm = TRUE init_timer(Tend)	Hide
W.TVASB1	time_out(Tend)	$none()	W.TVASB2
W.TVASB2	occur(cpt_carrier_off)	stop_timer(TVASB) station_signal(OFF)	Stopped
W.TVASB2	occur(data_carrier_on)	$none()	W.TVASB1
W.TVASB2	time_out(TVASB)	Send_AG(AGN)	W.AB
W.AB	AG_sent_event	FlagSendAlarm = FALSE station_signal(OFF)	Stopped

Tableau 6 – Transitions d'état de *Physique-62056-3-1*: Station Secondaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	energized()	MaxIndex=128 FlagRSO=FALSE FirstWinRSO=FALSE	Stopped
Initial	not(ener gized())	MaxIndex=18 FlagRSO=FALSE FirstWinRSO=TRUE	Stopped
Stopped	AG_received_event	Stop_timer(TOAG) init_timer(TA1O)	M.Rec
Stopped	Phy_ALARM.req()	TypeAG=ASB Send_AG(TypeAG)	W.ASB
Stopped	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	Stopped
M.Rec	octet_received_event	Stop_timer(TA1O) Index=2 Frame="" Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	Phy_ABORT(EP-1)	*WTOAG*
M.Rec	Phy_ABORT.req()	Stop_timer(TA1O)	*WTOAG*
Receiving	octet_received_event & Index<=MaxIndex	Stop_timer(TAO) Index=Index+1 Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Stop_timer(TAO) Phy_ABORT.ind(EP-4F)	*WTOAG*
Receiving	time_out(TAO)	Phy_DATA.ind(Frame) Init_timer(TOL)	M.Send
Receiving	Phy_ABORT.req()	Stop_timer(TAO)	*WTOAG*
M.Send	Phy_DATA.req(Frame)	Stop_timer(TOL) MaxIndex=128 Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 Init_timer(TOE)	Sending
M.Send	Phy_RSO.req(Frame, Window)	Stop_timer(TOL) MaxIndex=128 Wait_window(FirstWinRSO, Window) FirstWinRSO=FALSE Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 FlagRSO=TRUE Init_timer(TOE)	Sending
M.Send	time_out(TOL)	Init_timer(TA1O)	M.Rec
M.Send	Phy_ABORT.req()	Stop_timer(TOL)	*WTOAG*
Sending	octet_sent_event & Size>0	Index=Index+1 Send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0 & not(FlagRSO)	Stop_timer(TOE) init_timer(TA1O)	M.Rec
Sending	octet_sent_event & Size=0 & FlagRSO	Stop_timer(TOE) Wait_time(TAO) FlagRSO=FALSE	*WTOAG*
Sending	Phy_ABORT.req()	Stop_timer(TOE)	*WTOAG*
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F)	*WTOAG*
W.ASB	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	W.ASB
W.ASB	AG_sent_event	$none()	Stopped
WTOAG	Not(energized)	init_timer(TOAG)	Stopped
WTOAG	energized	$none()	Stopped

Tableau 7 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Initial	Initialisation des variables de la couche
Stopped	Attente d'un signal «Appel Général»
W.ETABS (Wait for end of " Alarm-Bus ")	Attente de la fin d'un signal «Alarme-Bus» reçu dans l'état Stopped
W.AG (Wait for end of " Wakeup Call ")	Attente de la fin de l'émission d'un signal «Appel Général»
W.TAB (Wait " Alarm-Bus ")	Attente d'un signal «Alarme-Bus» pendant la temporisation en fin d'émission d'un signal «Appel Général»
W.ETAB (Wait for end of " Alarm-Bus ")	Attente de la fin d'un signal «Alarme-Bus» reçu après l'émission d'un signal «Appel Général»
W.TASB	Attente du déclenchement du réveil TASB après le début de la réception d'un signal «Alarme-Bus»
W.TOL	Attente du déclenchement du réveil TOL
M.Send (Must Send)	Etat initial de l'émetteur, attente d'une trame à émettre
T.Session	Test du type de session (avec Station Secondaire alimentée ou télé-alimentée)
SendFirst	Envoi du premier octet de la trame à émettre
Sending	Etat récurrent de l'émetteur, émission octet par octet
Answer	Aiguillage en fonction du service demandé
M.Rec (Must Receive)	Etat initial du récepteur, attente du premier octet d'une trame
Receiving	Etat récurrent du récepteur, réception octet par octet
Received	Traitement de la trame reçue en fonction du service demandé
T.RSO (Test last RSO)	Test d'un intervalle de temps pour la réception d'une trame RSO
W.RSO (Wait for end of an RSO time slot)	Attente de la fin d'un intervalle de temps pour la réception d'une trame RSO
W.ASB	Attente de la fin d'émission d'un signal «Alarme Bus-Secondaire»
W.TOAG	Initialisation du délai de fin de session télé-alimentée TOAG si nécessaire
W.TOSEUIL	Attente du déclenchement du réveil TOSEUIL
W.AGT	Attente d'un signal «Appel Général» AGT
W.Sel (Wait for preSelection)	Attente d'une trame de présélection
Select	Réception d'une trame de présélection
W.Answer	Attente d'une trame de réponse en provenance d'une station sélectionnée
Hide	Attente de la fin d'une sélection
W.Agend	Attente de la fin de la réception d'un AG
W.TVASB1	Attente du déclenchement du réveil TVASB pour un signal «Alarme Bus-Secondaire» durant une communication
W.TVASB2	Attente du déclenchement du réveil TVASB pour un signal «Alarme Bus-Secondaire» à la fin d'une communication
W.AB	Attente de la fin d'émission d'un signal «Alarme-Bus»

Tableau 8 – Définition des procédures, des fonctions et des événements classés dans l'ordre alphabétique

Procédure, fonction ou événement	Définition
AG_received_event	Evénement issu du modem informant de la détection d'un signal «Appel Général» AGN
AG_sent_event	Evénement issu du modem informant de la fin de l'émission d'un signal «Appel Général»
alarm_detection()	Vérification que la station a son mode alarme à Actif
collision_detected_event	Evénement issu du modem informant de la détection d'une trame erronée sur réception d'un octet
concat(Frame, RecB)	Concaténation de l'octet RecB dans la trame Frame en cours de constitution
data_carrier_on, data_carrier_off	Occurrence de la détection de l'apparition ou de la disparition de la porteuse sur le bus
energized()	Vérification que la station est alimentée
init_timer(TOAPPEL), init_timer(TOSEUIL), init_timer(TAGT), init_timer(TOBAVARD), init_timer(TOPRE), init_timer(TOL), init_timer(TOE), init_timer(TAO), init_timer(TA1O), init_timer(TARSO) init_timer(TOAG), init_timer(TVASB) or init_timer(TAB)	Armement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB ou TAB
occur(cpt_carrier_on), occur(cpt_carrier_off), occur(data_carrier_on), occur(data_carrier_off), occur(octet_received_event) or occur(octet_sent_event)	Occurrence (report sans consommation) de la détection de l'apparition ou de la disparition de la porteuse sur le bus secondaire, de la détection de l'apparition de la porteuse sur le bus, de la disparition de la porteuse, de la réception d'un octet ou de l'émission d'un octet
octet_received_event	Evénement issu du modem informant qu'un octet a été reçu
octet_sent_event	Evénement issu du modem informant qu'un octet a été émis
read_data(RecB)	Traitement de l'événement octet_received_event par lecture de l'octet RecB reçu (les bits sont transmis dans un ordre croissant)
send_AG(TypeAG)	Demande au modem d'effectuer l'émission d'un signal «Appel Général» d'une durée TypeAG (AGN ou AGT)
send_octet(Frame, Index)	Emission de l'octet de rang Index dans la trame Frame (les bits sont transmis dans un ordre croissant)
size(Frame)	Calcul du nombre d'octets de la trame Frame
station_power(ON) or station_power(OFF)	Met en marche ou à l'arrêt l'alimentation en énergie de l'équipement
station_signal(ON) or station_signal(OFF)	Met en marche ou à l'arrêt l'émission de signal vers l'équipement sur le bus secondaire
stop_timer(TOAPPEL), stop_timer(TOSEUIL), stop_timer(TAGT), stop_timer(TOBAVARD), stop_timer(TOPRE), stop_timer(TOL), stop_timer(TOE), stop_timer(TAO), stop_timer(TA1O), stop_timer(TVASB) or stop_timer(TAB)	Désarmement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TVASB ou TAB
stop_timer(TOAG) or stop_timer(TARSO)	Désarmement du réveil TOAGT ou TARSO s'il a été armé au préalable

Procédure, fonction ou événement	Définition
time_out(TOAPPEL), time_out(TOSEUIL), time_out(TAGT), time_out(TOBAVARD), time_out(TOPRE), time_out(TOL), time_out(TOE), time_out(TAO), time_out(TA1O), time_out(TARSO), time_out(TOAG), time_out(TVASB) or stop_timer(TAB)	Déclenchement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB ou TAB
wait_time(TAO), wait_time(TICB), wait_time(TOL) or wait_time(TOALR)	Temporisation pendant le temps TAO, TICB, TOL ou TOALR
wait_window(FirstWinRSO, Window)	Temporisation calculée de la façon suivante: FirstWinRSO=VRAI ou Window=0 ==> 0 ms FirstWinRSO=FAUX et Window>0 ==> 40 ms + (TARSO*Window) ms (Le délai de 40 ms permet de garantir que l'émission a bien lieu dans l'intervalle de temps prévu)

5.1.6 Répertoire et traitement des erreurs

Les erreurs sont répertoriées par le codage suivant:

EP = erreur de la couche *Physique*

– = séparateur
N = numéro de l'erreur
F = erreur fatale

Tableau 9 – Tableau récapitulatif des erreurs

EP-1	*Délai du réveil TOL (Station Primaire) écoulé avant que la couche Liaison n'ait demandé l'envoi d'une trame, ou délai du réveil TA10 (Station Secondaire) écoulé avant la réception d'un caractère en provenance de la Station Primaire*
	Cette erreur conduit à l'attente d'un signal «Appel Général» après avoir informé la couche Liaison
EP-2	*Délai du réveil TOAG écoulé avant un signal «Appel Général»*
	Cette erreur conduit à l'attente d'un signal «Appel Général» après avoir informé la couche Liaison
EP-3	*Réception d'une alarme*
	Cette erreur conduit à réinitialiser la couche Physique après avoir informé la couche Liaison
EP-3F	*Durée anormale d'émission constatée après que le délai du réveil TOE est écoulé*
	Cette erreur conduit à réinitialiser la couche Physique après avoir informé la couche Liaison
EP-4F	*Nombre d'octets reçus supérieur à MaxIndex (Emetteur trop bavard)*
	Cette erreur conduit à réinitialiser la couche Physique après avoir informé la couche Liaison
EP-5F	*Délai du réveil TARSO écoulé en recevant une trame RSO (Station Primaire seulement)*
	Cette erreur conduit à réinitialiser la couche Physique après avoir informé la couche Liaison

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service Phy_ABORT.ind. La liste complète des numéros d'erreur fatale est fournie à l'Annexe C.

5.2 Couche Liaison

5.2.1 Protocole Liaison-62056-3-1

Le protocole *Liaison-62056-3-1* de la couche *Liaison* du profil d'échange de données par bus local sans DLMS adopte un comportement asymétrique. La machine d'état de la Station Primaire est donc différente de celle de la Station Secondaire.

La couche *Liaison* est chargée de transformer le canal physique exploité par la couche *Physique* en canal logique apte à transmettre de l'information fiable. Ses fonctions principales sont:

- effectuer une sérialisation et une désérialisation des données (dans la mesure où le canal physique fonctionne en série par bit);
- synchroniser les trames en émission et en réception;
- filtrer les trames en fonction des adresses primaire et secondaire;
- assurer une protection efficace contre les erreurs de transmission.

5.2.2 Gestion des échanges

Sur la Station Primaire, le protocole *Liaison-62056-3-1* prend en charge l'émission automatique des signaux «Appel Général» AGN ou AGT en fonction du type de la Station Secondaire. La détection d'une incompatibilité d'adresses dans un DSDU reçu de la couche supérieure entraîne une erreur fatale et l'arrêt du protocole *Liaison-62056-3-1*.

Sur la Station Secondaire, la réception d'une trame incorrecte ne donne lieu à aucun traitement, la récupération étant laissée à la charge de la Station Primaire.

Pour une station télé-alimentée, la détection d'un «Appel des Stations Oubliées» après un signal «Appel Général» AGT conduit à l'envoi d'une réponse RSO se situant toujours dans le premier intervalle de temps prévu à cet effet. La Station Primaire, lorsqu'elle détecte une collision au cours d'un tel échange, envoie un second «Appel des Stations Oubliées», mais cette fois-ci après un signal «Appel Général» AGN. Lorsqu'aucune collision n'est détectée après le premier appel, cela signifie qu'il n'y a au plus qu'une station oubliée, et il n'est pas nécessaire d'effectuer un deuxième appel.

5.2.3 Services et primitives de service de liaison

L'utilisateur du protocole *Liaison-62056-3-1* dispose des services et primitives de service donnés dans le Tableau 10.

Tableau 10 – Services et primitives de service de liaison

Service	Primitive de service
DL_DATA	DL_DATA.req(DSDU) DL_DATA.ind(DSDU)
DL_ALARM	DL_ALARM.req() DL_ALARM.ind()
DL_ABORT	DL_ABORT.req() DL_ABORT.ind(ErrorNb)

Le rôle attribué à chaque primitive est le suivant:

- DL_DATA.req(DSDU) permet à la couche *Application* de demander à la couche *Liaison* le transfert d'un paquet de données DSDU;
- DL_DATA.ind(DSDU) permet à la couche *Liaison* d'informer la couche *Application* de l'arrivée d'un paquet de données DSDU;

- DL_ALARM.req() permet à la couche *Application* de demander à la couche *Liaison* le transfert d'une alarme;
- DL_ALARM.ind() permet à la couche *Liaison* d'informer la couche *Application* de l'arrivée d'une alarme;
- DL_ABORT.req() permet à la couche *Application* de demander à la couche *Liaison* de mettre fin à son activité;
- DL_ABORT.ind(ErrorNb) permet à la couche Liaison d'informer la couche Application de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

5.2.4 Paramètres de liaison

Pour une Station Primaire, la valeur du nombre MaxRetry de réémissions d'une même trame avant déconnexion est fixée à 2.

La valeur du nombre MaxChain de séquences enchaînées sans signal «Appel Général» pour la télérelève et le télétransfert, est fixée à 5, de manière à assurer la compatibilité avec les Stations Secondaires utilisant une version antérieure du protocole.

La valeur du nombre maximal MaxRSO d'intervalles de temps RSC pour le traitement d'une trame «Appel des Stations Oubliées» est fixée à 3 après un «Appel Général» AGN, et à 1 après un «Appel Général» AGT.

La Station Secondaire doit connaître la liste des adresses de Stations Primaires et la liste des TABi auxquels elle a été sensibilisée.

Une station peut être sollicitée par l'intermédiaire d'une adresse primaire générale APG. Dans ce cas, elle répond avec la première adresse de la liste des adresses de Stations Primaires auxquelles elle a été sensibilisée.

5.2.5 Transitions d'état

Tableau 11 – Transitions d'état de Liaison-62056-3-1: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRetry=2 MaxChain=5	Stopped
Stopped	DL_DATA.req(DSDU) & check_req(DSDU)	NbChain=0 MaxRSO=3 PreSel=FALSE NoRetry=FALSE RepeatASO=FALSE EP-1=FALSE Context(ADS, ADP, TypeAG) Com=command(DSDU) init(Com, TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	DL_DATA.req(DSDU) & not(check_req(DSDU))	DL_ABORT.ind(EL-1F)	Stopped
Stopped	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
W.AG	Phy_APPG.ind() & not(PreSel) & not(NoRetry) & not(RepeatASO)	$none()	*T.Req*
W.AG	Phy_APPG.ind() & PreSel	Index=MaxRetry+1 Fr=PRE Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
W.AG	Phy_APPG.ind() & NoRetry	Index=MaxRetry+1 Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) NbChain=1 Phy_DATA.req(Fr) NoRetry=FALSE	M.Rec
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
W.EndS	Phy_ABORT.ind(ErrorNb)	$none()	*T.Error*
W.EndS	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
T.Error	((Error_Nb = EP-1 & TypeAG = AGN) I (Error_Nb = EP-2 & TypeAG = AGT)) & Com <>IB & Com <> TRB	$none()	Stopped
T.Error	((Error_Nb = EP-1 & TypeAG = AGN) I (Error_Nb = EP-2 & TypeAG = AGT)) & (Com=IB | Com=TRB)	DL_ABORT.ind(Error_Nb)	Stopped
T.Error	Error_Nb <> EP-1 & Error_Nb <> EP-2	DL_ABORT.ind(Error_Nb)	W.EndS
T.Error	(Error_Nb = EP-1) & TypeAG = AGT	$none()	W.EndS
T.Req	Com=IB | Com=TRB	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	Com=ASO & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	Com=ASO & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	((NbChain<MaxChain) & (Com=ENQ | Com=TRF)) | (NbChain=0 & Com=REC)	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Index=1 NbChain=NbChain+1 Phy_DATA.req(Fr)	M.Rec
T.Req	Com=AUT	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Index=1 NbChain=MaxChain Phy_DATA.req(Fr)	M.Rec
T.Req	(NbChain>=MaxChain) | (NbChain<>0 & Com=REC)	NbChain=0 Phy_APPG.req(AGN)	W.AG
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & not(PreSel)	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU)	M.Send
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & command(Frame)=SEL & PreSel	PreSel=FALSE	*T.Req*
M.Rec	Phy_DATA.ind(Frame) & check_frame(Frame) & command(Frame)<>SEL & PreSel	Phy_ABORT.req() DL_ABORT.ind(EL-2F)	W.EndS

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame)) & Index<=MaxRetry	Index=Index+1 Phy_DATA.req(Fr)	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame)) & Index>MaxRetry	Phy_ABORT.req() DL_ABORT.ind(EL-2F)	W.EndS
M.Rec	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & size(Frame)<>0 & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & size(Frame)<>0 & not(check_frame(Frame))	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req()	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.int(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DSDU=rso(RSO, Collision, ListRSO) DL_DATA.ind(DSDU)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	DL_DATA.req(DSDU) & check_req(DSDU) & not(EP-1)	Com=command(DSDU)	*T.Req*
M.Send	DL_DATA.req(DSDU) & check_req(DSDU) & EP-1	Com=command(DSDU) NbChain=0 EP-1=FALSE Phy_APPG.req(AGN)	W.AG
M.Send	DL_DATA.req(DSDU) & not(check_req(DSDU))	Phy_ABORT.req() DL_ABORT.ind(EL-1F)	W.EndS
M.Send	Phy_ABORT.ind(EP-1)	EP-1=TRUE	M.Send
M.Send	Phy_ABORT.ind(EP-2)	$none()	Stopped
M.Send	DL_ABORT.req() & not(EP_1 & TypAG=AGN)	Phy_ABORT.req()	W.EndS
M.Send	DL_ABORT.req() & EP_1 & TypAG=AGN	$none()	Stopped
M.Send	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS

Tableau 12 – Transitions d'état de Liaison-62056-3-1: Station Secondaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	FlagDSO=TRUE Discovered=FALSE Flag_alarm=FALSE	Stopped
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & check_address(Frame)	ADP=extract_ADP(Frame) Com=command(Frame)	*T.Com*
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & not(check_address(Frame))	Phy_ABORT.req()	Stopped
Stopped	Phy_DATA.ind(Frame) & not(check_frame(Frame))	$none()	Stopped
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	DL_ALARM.req()	Phy_ALARM.req() Flag_alarm=TRUE	Stopped
T.Com	Com=IB	FlagDSO=TRUE Discovered=FALSE Phy_ABORT.req()	Stopped
T.Com	Com=TRB	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU) Phy_ABORT.req()	Stopped
T.Com	Com=PRE	Fr=SEL Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	Stopped
T.Com	Com=ASO & test_TABi(Frame, TAB)	Fr=concat(RSO, TAB, ADS) Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_RSO.req(Fr, window_RSO())	Stopped
T.Com	Com=ASO & not(test_TABi(Frame, TAB))	Phy_ABORT.req()	Stopped
T.Com	Com=ENQ | Com=REC | Com=TRF | COM=AUT	DSDU=extract_DSDU(Frame) DL_DATA.ind(DSDU)	M.Send
M.Send	DL_DATA.req(DSDU)	Fr=DSDU Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) update_flag_DSO(command(Fr))	Stopped
M.Send	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped

Tableau 13 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Initial	Initialisation des variables de la couche
Stopped	Attente de la première demande de la couche supérieure ou de la première indication de la couche inférieure
W.AG (Wait for end of " Wakeup Call ")	Attente de la fin d'un signal «Appel Général» demandé
W.EndS (Wait for End of Session)	Attente de la fin d'une session
T.Req (Test Request)	Test de la nature d'une demande en provenance d'une couche supérieure
M.Rec (Must Receive)	Attente d'une réponse en provenance de la couche inférieure
M.RSO (Must receive RSO)	Attente d'une trame RSO après l'envoi d'une trame ASO
T.RSO (Test last RSO)	Test du dernier intervalle de temps pour la réception d'une trame RSO
M.Send (Must Send)	Attente d'une requête de la couche supérieure
T.Com (Test Command)	Test du code de commande d'une trame reçue

Tableau 14 – Définition des procédures et des fonctions classées dans l'ordre alphabétique

Procédure ou fonction	Définition
build_RSO(ListRSO, Frame)	Extraction des éléments (champs TAB et ADS) de la trame RSO reçue Frame et concaténation avec la précédente liste ListRSO
check_address(Frame)	Vérification que les adresses ADP et ADS sont reconnues selon les critères suivants: – la station est sensibilisée à l'ADP ou ADP=APG; – ADS=ADG si la commande est ASO, IB ou TRB, – ADS est l'adresse de la station secondaire si la commande n'est pas ASO, IB ou TRB
check_frame(Frame)	Vérification que la trame Frame reçue est correcte: – nombre d'octets supérieur ou égal à 11 et inférieur ou égal à 128; – CRC correct; – nombre d'octets compatible avec le champ N – commande connue et nombre d'octets compatible avec cette commande
check_req(DSDU)	Vérification que la commande du DSDU est compatible avec les adresses ADP et ADS du contexte de communication
command(DSDU) ou command(Frame)	Extraction de la valeur de la commande codée dans le DSDU à envoyer ou de la trame Frame reçue
concat(N, ADS, ADP, DSDU) ou concat(Frame, CRC)	Concaténation des champs N, ADS, ADP et du DSDU ou concaténation du CRC à la fin de la trame Frame
context(ADS, ADP, TypeAG)	Extraction des valeurs correspondantes du contexte de communication
crc(Frame)	Calcul du CRC de la trame Frame à émettre
extract_ADP(Frame)	Si l'adresse de la Station Primaire utilisée dans la trame est différente de APG ou bien s'il s'agit de APG mais que la liste des adresses primaires auxquelles est sensibilisée la Station Secondaire est vide, extraction de cette valeur, sinon extraction de la première valeur ADP de la liste des adresses primaires auxquelles est sensibilisée la Station Secondaire
extract_DSDU(Frame)	Extraction du DSDU (champs COM et DATA) de la trame Frame reçue

Procédure ou fonction	Définition
init(COM, TypeAG)	Met PreSel à VRAI si on a à la fois TypeAG égal à AGT et une trame de taille supérieure à 18 octets. Met NoRetry à VRAI si on a à la fois TypeAG égal à AGT et une trame de taille inférieure ou égale à 18 octets
rso(RSO, Collision, ListRSO)	Concaténation de la commande RSO, de l'indicateur de collision et de la liste d'éléments RSO (champs TAB et ADS)
size(Frame)	Calcul de la taille de la trame Frame reçue
size_frame(DSDU)	Calcul de la taille de la trame associée à un DSDU à émettre (taille du DSDU + 10)
test_TABi(Frame, TAB)	Si le premier des TABi contenus dans la trame ASO Frame vaut 00, vérification que la variable Discovered est à FAUX et vérification, après tirage d'un nombre aléatoire entre 0 et 100, que ce nombre est inférieur à la probabilité de réponse souhaitée au deuxième des TABi. Dans ce cas la valeur 00 est mémorisée dans la variable TAB; Si le premier TABi contenu dans la trame ASO vaut FF, vérification que la variable Flag_alarm est à VRAI et dans ce cas affectation de la valeur FAUX à la variable Flag_alarm et mémorisation de la valeur FF dans la variable TAB; sinon vérification que la variable FlagDSO est à VRAI, que la Station Secondaire est sensibilisée à l'un des TABi contenus dans la trame Frame de type ASO reçue et mémorisation de la première de ces valeurs dans la variable TAB
update_flag_DSO(Com)	Affectation de la valeur FAUX à la variable FlagDSO et de la valeur VRAI à la variable Discovered si COM est la commande ENQ
window_RSO()	Fourniture d'une valeur aléatoire entière comprise entre 0 et MaxRSO-1 utilisée comme numéro de l'intervalle de temps RSO dans laquelle la station doit répondre (voir l'Annexe F)

5.2.6 Répertoire et traitement des erreurs

Les erreurs sont répertoriées par le codage suivant:

EL = erreur de la couche *Liaison*
– = séparateur
N = numéro de l'erreur
F = erreur fatale

Tableau 15 – Tableau récapitulatif des erreurs

EL-1F	Réception d'une commande incompatible avec les adresses primaire et secondaire mémorisées dans le contexte de communication (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Liaison* après avoir informé la couche *Application*
EL-2F	Réponse incorrecte de la Station Secondaire après, soit une trame de présélection pour une station télé-alimentée, soit MaxRetry répétitions d'une requête (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Liaison* après avoir informé la couche *Application*

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service DL_ABORT.ind. La liste complète des numéros d'erreur fatale est fournie à l'Annexe C.

5.3 Couche Application

5.3.1 Protocole Application-62056-3-1

Le protocole *Application-62056-3-1* de la couche *Application* du profil d'échange de données par bus local sans DLMS adopte un comportement asymétrique. La machine d'état de la Station Primaire est donc différente de celle de la Station Secondaire.

Le protocole *Application-62056-3-1* de la couche *Application* de le profil d'échange de données par bus local sans DLMS est chargé de contrôler et d'enchaîner les messages successifs par analyse d'un code commande fourni par l'application utilisatrice.

5.3.2 Services et primitives de service d'Application

L'utilisateur du protocole *Application-62056-3-1* dispose des services et primitives de service donnés dans le Tableau 16.

Tableau 16 – Services et primitives de service d'Application

Service	Primitive de service
A_DATA	A_DATA.req(COM, ASDU) A_DATA.ind(ASDU)
A_ALARM	A_ALARM.req() A_ALARM.ind()
A_ABORT	A_ABORT.req() A_ABORT.ind(ErrorNb)

Le rôle attribué à chaque primitive est le suivant:

- A_DATA.req(COM, ASDU) permet à l'application de demander à la couche *Application* le transfert d'une commande COM (ENQ, REC, TRF, TRB, IB ou ASO pour une Station Primaire et DAT, DRJ, EOS ou TRA pour une Station Secondaire) associée à un ASDU;
- A_DATA.ind(ASDU) permet à la couche *Application* d'informer l'application de l'arrivée d'un ASDU;
- A_ALARM.req() permet à l'application de demander à la couche *Application* l'envoi d'une alarme;
- A_ALARM.ind() permet à la couche *Application* d'informer l'application de l'arrivée d'une alarme;
- A_ABORT.req() permet à l'application de demander à la couche *Application* de mettre fin à son activité;
- A_ABORT.ind(ErrorNb) permet à la couche *Application* d'informer l'application de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

5.3.3 Paramètres d'Application

La Station Primaire doit posséder les clefs de cryptage DES associées à chacune des Stations Secondaires sur lesquelles une téléprogrammation doit être réalisée.

Dans le cas d'une téléprogrammation, la Station Secondaire doit posséder la clef de cryptage DES utilisée par la Station Primaire.

5.3.4 Transitions d'état

Tableau 17 – Transitions d'état d'Application-62056-3-1: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Stopped	A_DATA.req(Com, ASDU) & (Com=ASO | Com=ENQ | Com=TRF)	APDU=concat(Com, _, _, ASDU) DL_DATA.req(APDU)	M.Rec
Stopped	A_DATA.req(Com, ASDU) & (Com=IB | Com=TRB)	APDU=concat(Com, _, _, ASDU) DL_DATA.req(APDU)	W.EndS
Stopped	A_DATA.req(Com, ASDU) & Com=REC	Na1=randomize() Zdt=zdt(ASDU) APDU=concat(REC, Na1, 0, Zdt) DL_DATA.req(APDU) Na1k=cipher(Na1)	M.Rec
Stopped	DL_ALARM.ind()	A_ALARM.ind()	Stopped
Stopped	A_ABORT.req()	DL_ABORT.req()	Stopped
M.Rec	DL_DATA.ind(DSDU)	Resp=command(DSDU)	*T.Resp*
M.Rec	A_ABORT.req()	DL_ABORT.req()	Stopped
M.Rec	DL_ABORT.ind(ErrorNb) & Error_Nb<> EP_1 & Error_Nb<> EP_2	A_ABORT.ind(ErrorNb)	Stopped
M.Rec	DL_ABORT.ind(ErrorNb) & (Error_Nb = EP_1 I Error_Nb = EP_2)	$none()	M.Rec
W.EndS	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped
W.EndS	A_ABORT.req()	DL_ABORT.req()	W.EndS
T.Resp	(Com=ASO & Resp=RSO) | (Com=ENQ & Resp=DAT) | (Com=ENQ & Resp=DRJ) | (Com=TRF & Resp=TRA) | (Com=TRF & Resp=DRJ) | (Com=AUT & Resp=EOS) | (Com=AUT & Resp=DRJ)	A_DATA.ind(DSDU)	Stopped
T.Resp	Com=REC & Resp=ECH & na1k(DSDU)=Na1k & Zdt=zdt(DSDU)	Na2=na2(DSDU) Na2k=cipher(Na2) Com=AUT APDU=concat(AUT, 0, Na2k, '''') DL_DATA.req(APDU)	M.Rec
T.Resp	(Com=ASO & Resp<>RSO) | (Com=ENQ & Resp<>DAT & Resp<>DRJ) | (Com=TRF & Resp<>TRA & Resp<>DRJ) | (Com=REC & Resp<>ECH) | (Com=AUT & Resp<>ARJ & Resp<>DRJ &Resp<>EOS)	A_ABORT.ind(EA-1F) DL_ABORT.req()	Stopped
T.Resp	Com=REC & (Resp<>ECH | (na1k(DSDU)<>Na1k) | (Zdt<>zdt(DSDU))	A_ABORT.ind(EA-2F) DL_ABORT.req()	Stopped
T.Resp	Com=AUT & Resp=ARJ	A_ABORT.ind(EA-3F) DL_ABORT.req()	Stopped

Tableau 18 – Transitions d'état d'Application-62056-3-1: Station Secondaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Stopped	DL_DATA.ind(DSDU) & (command(DSDU)=ENQ | Command(DSDU)=TRF)	A_DATA.ind(DSDU) Req=command(DSDU)	M.Send
Stopped	DL_DATA.ind(DSDU) & command(DSDU)=TRB	A_DATA.ind(DSDU)	Stopped
Stopped	DL_DATA.ind(DSDU) & command(DSDU)=REC	Zdt=zdt(DSDU) Na1=na1(DSDU) Na1k=cipher(Na1) Na2=randomize() APDU=concat(ECH, Na1k, Na2, Zdt) DL_DATA.req(APDU) Na2k=cipher(Na2) Req=REC	W.AUT
Stopped	A_ALARM.req() & alarm_detection()	DL_ALARM.req()	Stopped
M.Send	A_DATA.req(COM, ASDU) & (((COM=DAT | COM=DRJ) & Req=ENQ) | ((COM=TRA | COM=DRJ) & Req=TRF) | (COM=DRJ & Req=REC))	APDU=concat(COM, _, _, ASDU) DL_DATA.req(APDU)	Stopped
M.Send	A_DATA.req(COM, ASDU) & (COM=EOS & Req=REC)	APDU=concat(EOS, 0, 0, ‘’’’) DL_DATA.req(APDU)	Stopped
M.Send	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped
W.AUT	DL_DATA.ind(DSDU) & Command(DSDU)=AUT & na2k(DSDU)=Na2k	ASDU=concat(REC, _, _, Zdt) A_DATA.ind(ASDU)	M.Send
W.AUT	DL_DATA.ind(DSDU) & Command(DSDU)=AUT & na2k(DSDU)<>Na2k	APDU=concat(ARJ, _, _, ‘’’’) DL_DATA.req(APDU)	Stopped
W.AUT	DL_ABORT.ind(ErrorNb)	A_ABORT.ind(ErrorNb)	Stopped

Tableau 19 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Stopped	Attente de la première requête de la couche supérieure ou de la première indication de la couche inférieure
M.Rec (Must Receive)	Attente de la réponse à la requête émise
T.Resp (Test Response)	Traitement de la réponse reçue
M.Send (Must Send)	Attente d'une réponse à une requête reçue
W.AUT (Wait for AUT frame)	Attente de la trame AUT consécutive à une réponse ECH envoyée

Tableau 20 – Définition des procédures et des fonctions classées dans l'ordre alphabétique

Procédure ou fonction	Définition
alarm_detection()	Vérification que le mode Alarme est Actif
cipher(Na1) or cipher(Na2)	Cryptage du nombre aléatoire Na1 ou Na2 N par l'algorithme DES avec la clef du contexte de communication
command(DSDU)	Extraction du champ commande d'un DSDU reçu
concat(COM, _, _, ASDU), concat(COM, ZA1, ZA2, ZDT) or concat(COM, 0, 0, _)	Concaténation d'un champ commande COM et d'un ASDU ou concaténation d'un champ commande COM, d'une valeur cryptée ZA1, d'une valeur cryptée ZA2 et d'une zone de données ZDT (champs TAB et DATA) ou concaténation d'un champ commande COM et des champs ZA1 = 0 et ZA2 = 0
na1(DSDU)	Extraction de la valeur Na1 du champ ZA1 d'une trame REC reçue
na1k(DSDU)	Extraction de la valeur Na1k du champ ZA1 d'une trame ECH reçue
na2(DSDU)	Extraction de la valeur Na2 du champ ZA2 d'une trame ECH reçue
na2k(DSDU)	Extraction de la valeur Na2k du champ ZA2 d'une trame AUT reçue
randomize()	Génération d'un nombre aléatoire suivant la procédure définie à l'Annexe G
zdt(ASDU) or zdt(DSDU)	Extraction de données (champs TAB et DATA) d'une demande de type REC, d'une trame REC ou d'une trame ECH

5.3.5 Répertoire et traitement des erreurs

Les erreurs sont répertoriées par le codage suivant:

EA = erreur de la couche *Application*
– = séparateur
N = numéro de l'erreur
F = erreur fatale

Tableau 21 – Tableau récapitulatif des erreurs

EA-1F	Le code commande de la réponse reçue ne correspond pas à la requête émise (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Application* après avoir informé l'application
EA-2F	Erreur d'authentification détectée sur la réponse de la Station Secondaire (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Application* après avoir informé l'application
EA-3F	Erreur d'authentification détectée par la Station Secondaire (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Application* après avoir informé l'application

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service A_ABORT.ind. La liste complète des numéros d'erreur fatale est fournie à l'Annexe C.

6 Echange de données par bus local avec DLMS

6.1 Couche Physique

Le protocole *Physique-62056-3-1* de la couche *Physique* du profil d'échange de données par bus local avec DLMS est exactement le même que celui défini pour le profil sans DLMS.

6.2 Couche Liaison

6.2.1 Protocole Liaison-E/D

Le protocole *Liaison-E/D* de la couche *Liaison* du profil d'échange de données par bus local avec DLMS adopte un comportement asymétrique. La machine d'état de la Station Primaire est donc différente de celle de la Station Secondaire.

La couche *Liaison* est chargée de transformer le canal physique exploité par la couche *Physique* en canal logique apte à transmettre de l'information fiable. Ses fonctions principales sont:

- gérer directement les services d'Initialisation de Bus et de Réponse des Stations Oubliées;
- effectuer la sérialisation et la désérialisation des données (dans la mesure où le canal physique fonctionne en série par bit);
- synchroniser les trames en émission et en réception;
- filtrer les trames en fonction des adresses primaire et secondaire;
- assurer une protection efficace contre les erreurs de transmission.

6.2.2 Gestion des échanges

Les services d'Initialisation de Bus, Alarme et Appel des Stations Oubliées sont offerts par le protocole *Liaison-E/D* de la couche *Liaison* du profil d'échanges de données par bus local avec DLMS, mais les opérations sont réalisées en dehors de la couche *Application*. En particulier, le drapeau de station oubliée peut être mis à jour par l'interface de programmation de l'application alors qu'une télérelève a lieu.

Sauf pour l'ouverture d'une session, durant l'Initialisation de Bus, la remontée d'Alarme et la gestion de l'Appel des Stations Oubliées, le protocole est complètement symétrique, et les deux stations sont à tour de rôle Emetteur et Récepteur.

Après l'envoi d'une trame, la couche *Liaison* du côté Emetteur attend toujours une trame de la couche *Liaison* du Récepteur avant d'émettre à nouveau. Cette attente est contrôlée par le délai de garde T1, d'une valeur de 10 s.

Après l'envoi d'une trame et la réception de l'acquittement de l'envoi précédent, la trame courante est retransmise. Le nombre de retransmissions est limité à MaxRetry. Au-delà de ce nombre, la communication est interrompue au niveau *Liaison* et la couche *Application* en est informée.

Chaque fois qu'une trame est reçue, il y a émission immédiate d'une trame en réponse. Cette trame peut contenir des données de la couche *Application*. Elle contient toujours un numéro dans le champ *Send* et un numéro dans le champ *Confirm*, calculés en fonction des valeurs précédemment envoyées et reçues. L'algorithme de calcul de ces numéros est le suivant:

- si la dernière trame reçue est sans erreur et que son numéro Send est égal au complément à 1 du numéro Confirm précédent en émission, alors le paquet de données est transmis à la couche Application et la prochaine trame envoyée aura un numéro Confirm égal au numéro Send reçu. Sinon le numéro Confirm reste inchangé et le paquet de données n'est pas transmis à la couche Application;
- si la dernière trame reçue est sans erreur et que son numéro Confirm est identique au numéro Send précédent en émission, alors le numéro Send en émission est complété à 1 pour la prochaine trame dans l'hypothèse où un nouveau paquet de données doit être envoyé;
- si la dernière trame reçue est incorrecte ou si son numéro Confirm n'est pas identique au numéro Send précédent en émission, alors la même trame est de nouveau transmise sous réserve que le nombre de réémissions reste inférieur ou égal à MaxRetry.

6.2.3 Services et primitives de service de liaison

L'utilisateur du protocole *Liaison-E/D* dispose des services et primitives de service donnés dans le Tableau 22.

Tableau 22 – Services et primitives de services de liaison

Service	Primitive de service
DL_DATA	DL_DATA.req(Pr, DSDU) DL_DATA.ind(Pr, DSDU)
DL_IB	DL_IB.req()
DL_ASO	DL_ASO.req(DSDU) DL_ASO.ind(Collision, List)
DL_ALARM	DL_ALARM.req()
DL_ABORT	DL_ABORT.req(Strong) DL_ABORT.ind(ErrorNb)

Le rôle attribué à chaque primitive est le suivant:

- DL_DATA.req(Pr, DSDU) permet à la couche *Application* de demander à la couche *Liaison* le transfert d'un paquet de données DSDU avec la priorité Pr [3)];
- DL_DATA.ind(Pr, DSDU) permet à la couche *Liaison* d'informer la couche *Application* de l'arrivée d'un paquet de données DSDU avec la priorité Pr;
- DL_IB.req() permet à l'*interface de programmation de l'Application* de demander à la couche *Liaison* l'envoi d'une trame Initialisation du Bus;
- DL_ASO.req(DSDU) permet à l'*interface de programmation de l'Application* de demander à la couche *Liaison* l'envoi d'une trame Appel des Stations Oubliées en cohérence avec un paquet de données DSDU;
- DL_ASO.ind(Collision, List) permet à la couche *Liaison* d'informer l'*interface de programmation de l'Application* du résultat d'un Appel des Stations Oubliées;
- DL_ALARM.req() permet à l'*interface de programmation de l'Application* de demander à la couche *Liaison* l'envoi d'une alarme;
- DL_ABORT.req(Strong) permet à la couche *Application* de demander à la couche *Liaison* de mettre fin à son activité avec la priorité Strong [4)];
- DL_ABORT.ind(ErrorNb) permet à la couche *Liaison* d'informer la couche *Application* de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

6.2.4 Paramètres de liaison

La valeur du nombre MaxRetry de réémissions d'une même trame avant déconnexion est fixée à 2.

Pour une Station Primaire, la valeur du nombre maximal MaxRSO d'intervalles de temps RSO pour le traitement d'une trame «Appel des Stations Oubliées» est fixée à 3.

La Station Secondaire doit connaître la liste des adresses de Stations Primaires et la liste des TABi auxquelles elle a été sensibilisée.

---

3) Le paramètre de priorité Pr différencie le traitement d'un service urgent tel que InformationReport (niveau Pr=1) d'un autre service DLMS (niveau Pr=0).

4) Le paramètre de priorité Strong différencie le traitement des erreurs fatales (Strong=1) de celui d'une autre demande de déconnexion physique (Strong=0) initialisée par la sous-couche *Application*.

Une station peut aussi être sollicitée par l'intermédiaire d'une adresse primaire générale APG. Dans ce cas, elle répond avec la première adresse de la liste des adresses de Stations Primaires auxquelles elle a été sensibilisée.

6.2.5 Transitions d'état

Tableau 23 – Transitions d'état de Liaison-E/D: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRetry=2 MaxChain = 5 init_incrChain()	Stopped
Stopped	exist_dl_req()	NbChain = 0 Index=0 RepeatASO=FALSE context(ADP, ADS, TypeAG) init(TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	Phy_ALARM.ind()	create_alarm(TPDU) DL_DATA.ind(Pr = 1, TPDU)	Stopped
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain = 0	$none()	*T.Req*
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain <> 0	NbChain = 0	*M.Send*
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
W.EndS	(Phy_ABORT.ind(EP-2) & TypeAG=AGT) I (Phy_ABORT.ind(EP-1) & TypeAG=AGN)	$none()	Stopped
W.EndS	Phy_ALARM.ind	create_alarm(TPDU) DL_DATA.ind(Pr = 1, TPDU)	Stopped
W.EndS	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS
T.Req	exist_dl_req(DL_IB.req())	Fr="" Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, IB, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	exist_dl_req(DL_ASO.req(DSDU)) & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	exist_dl_req(DL_ASO.req(DSDU)) & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
T.Req	not (exist_dl_req(DL_IB.req()) | exist_dl_req(DL_ASO.req(_)))	Pr=0 Send="00"B Confirm="11"B Fr="" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & not(check_frame(Frame)) & size(Frame)<>0	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DL_ASO.ind(Collision, ListRSO)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, DSDU)) & NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, DSDU)) & NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(_, _)) & NbChain < MaxChain	Pr=0 Fr="" Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	NbChain >= MaxChain	Phy_APPG.req(AGN)	W.AG

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	DL_DATA.ind(extract_prty(Frame), extract_text(Frame)) Confirm=incr(Confirm) Ack_expected=FALSE Index = 0	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	Ack_expected=FALSE Index = 0	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	DL_ABORT.ind(EL-2F) Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=1)	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS

Tableau 24 – Transitions d'état de Liaison-E/D: Station Secondaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRetry=2 FlagDSO=TRUE Discovered=FALSE Flag_alarm = FALSE	Stopped
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & check_address(Frame)	ADP=extract_ADP(Frame) Com=command(Frame)	*T.Com*
Stopped	Phy_DATA.ind(Frame) & check_frame(Frame) & not(check_address(Frame))	Phy_ABORT.req()	Stopped
Stopped	Phy_DATA.ind(Frame) & not(check_frame(Frame))	$none()	Stopped
Stopped	DL_ALARM.req() & alarm_detection()	Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
T.Com	Com=IB	FlagDSO=TRUE Discovered=FALSE Phy_ABORT.req()	Stopped
T.Com	Com=ASO & test_TABi(Frame, TAB)	Fr=concat(RSO, TAB, ADS) Fr=concat(size_frame(Fr), ADS, ADP, Fr) Fr=concat(Fr, crc(Fr)) Phy_RSO.req(Fr, window_RSO())	Stopped
T.Com	Com=ASO & not(test_TABi(Frame, TAB))	Phy_ABORT.req()	Stopped
T.Com	is_data+(Frame) & is_text(Frame)	Ack_expected=FALSE Send="11"B Confirm="00"B DL_DATA.ind(extract_prty, extract_text(Frame))	*M.Send*
T.Com	is_data+(Frame) & not(is_text(Frame))	Ack_expected=FALSE Send="11"B Confirm="00"B	*M.Send*

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, DSDU))	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, DSDU))	Discovered = TRUE Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	not(DL_DATA.req(_, _))	Pr=0 Fr="" Index=1 Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	stop_timer(T1) Confirm=incr(Confirm) Ack_expected=FALSE DL_DATA.ind(extract_prty, extract_text(Frame))	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	stop_timer(T1) Ack_expected=FALSE	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	stop_timer(T1) Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	stop_timer(T1) DL_ABORT.ind(EL-2F) Phy_ABORT.req()	Stopped
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	stop_timer(T1) Phy_ABORT.req()	Stopped
M.Rec	DL_ALARM.req() & alarm_detection()	stop_timer(T1) DL_ABORT.ind(EL_1F) Phy_ABORT.req() Flag_alarm = TRUE Phy_ALARM.req()	Stopped
M.Rec	DL_ABORT.req(Strong=1)	stop_timer(T1) Phy_ABORT.req()	Stopped
M.Rec	Phy_ABORT.ind(EP-1)	init_timer(T1)	M.Rec
M.Rec	Phy_ABORT.ind(ErrorNb) & ErrorNb <> EP-1	stop_timer(T1) DL_ABORT.ind(ErrorNb)	Stopped
M.Rec	time_out(T1)	DL_ABORT.ind(EL_3F)	Stopped

Tableau 25 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Initial	Initialisation des variables de la couche
Stopped	Attente de la première demande de la couche supérieure ou de la première indication de la couche inférieure
W.AG (Wait for end of «Appel Général»)	Attente de la fin d'un signal «Appel Général»
W.EndS (Wait for end of Session)	Attente de la fin d'une session
T.Req (Test Request)	Test de la nature d'une requête provenant de la couche supérieure
M.RSO (Must Receive RSO)	Attente d'une trame de réponse RSO après l'envoi d'une trame ASO
T.RSO (Test last RSO)	Test du dernier intervalle de temps pour la réception d'une trame RSO
M.Send (Must Send)	Test de la trame à envoyer (le champ Text peut être vide)
M.Rec (Must Receive)	Attente d'une indication en provenance de la couche inférieure
T.Com (Test Command)	Test du champ COM d'une trame reçue

Tableau 26 – Définition des procédures et des fonctions classées dans l'ordre alphabétique

Procédure ou fonction	Définition
alarm_detection()	Vérification que le mode alarme est Actif
build_RSO(ListRSO, Frame)	Extraction des éléments (champs TAB et ADS) de la trame RSO reçue Frame et concaténation avec la précédente liste ListRSO
check_address(Frame)	Vérification que les adresses ADP et ADS sont reconnues selon les critères suivants: – la station est sensibilisée à l'ADP ou ADP = APG; – ADS=ADG si la commande est ASO ou IB, – ADS est l'adresse de la station secondaire si la commande n'est pas ASO ou IB
check_frame(Frame)	Vérification que la trame Frame reçue est correcte: – nombre d'octets supérieur ou égal à 11 et inférieur ou égal à 128; – CRC correct; – nombre d'octets compatible avec le champ Size; – commande connue
com(DATA+, Pr, Send, Confirm)	Concaténation des champs binaires correspondants pour obtenir une commande spécifique
command(Frame)	Extraction de la valeur de la commande codée dans le champ COM de la trame Frame reçue
concat(Size, ADS, ADP, COM, Text), ou concat(Frame, CRC)	Concaténation des champs Size, ADS, ADP, COM et Text ou concaténation du CRC à la fin de la trame Frame
context(ADS, ADP, TypeAG)	Extraction des valeurs correspondantes du contexte de communication
crc(Frame)	Calcul du CRC de la trame Frame à émettre
create_alarm(TPDU)	Calcul d'un TPDU avec STSAP = 0, DTSAP = 0 et un UnsolicitedReqPDU avec: client-type = FFFF, serveridentifier = 0, object-name = FFFF variable type = boolean, value = TRUE

Procédure ou fonction	Définition
exist_dl_data-req(DL_DATA.req(Pr, DSDU))	Consommation d'un événement DL_DATA.req(Pr, DSDU)
exist_dl_req()	Vérification de l'existence d'un événement DL_IB.req(), DL_ASO.req(DSDU) ou DL_DATA.req(Pr, DSDU) et vérification de la compatibilité avec les adresses ADS et ADP du contexte de communication
exist_dl_req(DL_IB.req()) ou exist_dl_req(DL_ASO.req(DSDU))	Consommation d'un événement DL_IB.req() ou DL_ASO.req(DSDU)
extract_ADP(Frame)	Si l'adresse de la Station Primaire utilisée dans la trame est différente de APG ou bien s'il s'agit de APG mais que la liste des adresses primaires auxquelles est sensibilisée la Station Secondaire est vide, extraction de cette valeur, sinon extraction de la première valeur ADP de la liste des adresses primaires auxquelles est sensibilisée la Station Secondaire
extract_prty(Frame)	Extraction du champ Priority de la trame Frame reçue
extract_text(Frame)	Extraction du champ Text de la trame Frame reçue
init(TypeAG)	Met Index à MaxRetry si TypeAG vaut AGT, à 0 sinon
init_incrChain()	Met IncrChain à 0 si les alarmes ne sont pas gérées, à 1 sinon
init_timer(T1)	Armement du réveil T1
is_ack (Frame)	Vérification que la trame Frame reçue contient un champ Confirm égal au champ Send de la dernière trame émise
is_data+ (Frame)	Vérification que la trame Frame reçue contient un champ DATA+ correct ("111"B)
is_text(Frame)	Vérification que la trame Frame reçue contient un champ de données Text non vide et que le champ Send est égal au complement à un du champ Confirm de la dernière trame émise.
size(Frame)	Calcul de la taille de la trame Frame reçue
size_frame(DSDU)	Calcul de la taille de la trame à construire à partir de l'unité de données DSDU (taille de DSDU+11)
stop_timer(T1)	Désarmement du réveil T1
test_TABi(Frame, TAB)	Si le premier des TABi contenus dans la trame Frame de type ASO vaut 00, vérification que la variable Discovered est à FAUX et vérification, après tirage d'un nombre aléatoire entre 0 et 100, que ce nombre est inférieur à la probabilité de réponse souhaitée au deuxième des TABi. Dans ce cas la valeur 00 est mémorisée dans la variable TAB; si le premier TABi contenu dans la trame Frame de type ASO vaut FF, vérification que la variable Flag_alarm est à VRAI et dans ce cas affectation de la valeur FAUX à la variable Flag_alarm et mémorisation de la valeur FF dans la variable TAB; sinon vérification que la variable FlagDSO est à VRAI, que la Station Secondaire est sensibilisée à l'un des TABi contenus dans la trame Frame de type ASO reçue et mémorisation de la première de ces valeurs dans la variable TAB
time_out(T1)	Armement du réveil T1
window_RSO()	Récupération d'un nombre aléatoire entre 0 et MaxRSO-1, qui donne le numéro de l'intervalle de temps RSO pendant lequel la station doit répondre (voir Annexe F)

6.2.6 Répertoire et traitement des erreurs

Les erreurs sont répertoriées par le codage suivant:

EL = erreur de la couche *Liaison*
– = séparateur
N = numéro de l'erreur
F = erreur fatale

Tableau 27 – Tableau récapitulatif des erreurs

EL-1F	DL_ALARM.req() reçue durant une association
	Cette erreur conduit à réinitialiser la couche *Liaison* après avoir demandé à la couche Physique de transmettre une alarme
EL-2F	Réponse incorrecte de la Station Secondaire après MaxRetry répétitions de la requête
	Cette erreur conduit à réinitialiser la couche *Liaison* après avoir informé la couche *Application* et provoqué l'arrêt de la couche *Physique*
EL-3F	Fin de communication après un silence d'une durée T1
	Cette erreur conduit à réinitialiser la couche *Liaison* après avoir informé la couche *Application* et provoqué l'arrêt de la couche *Physique*

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service DL_ABORT.ind. La liste complète des numéros d'erreur fatale est fournie à l'Annexe C.

6.3 Couche Application

6.3.1 Généralités

Les spécifications de la couche *Application* sont données dans la CEI 62056-51. Ce paragraphe clarifie simplement le profil utilisé pour le profil d'échange de données par bus local avec DLMS.

6.3.2 Sous-couche Transport

La valeur du nombre MaxPktSize (voir la CEI 62056-51) taille maximale du champ *Packet*, doit être initialisée à 114.

6.3.3 Sous-couche Application

Comme cela est indiqué dans la CEI 62056-51, les fonctions client_connect et server_connect doivent être précisées en fonction du support de communication utilisé. Pour le profil d'échange de données par bus local avec DLMS, le service non sollicité n'étant pas supporté, la fonction server_connect n'est pas acceptée. La fonction client_connect est définie dans le Tableau 28.

Tableau 28 – Définition de la fonction client_connect

Procédure ou fonction	Définition
client_connect(Ads, Adp)	S'il n'existe aucune association d'application active, la fonction établit les paramètres du contexte de communication: ADS, ADP et TypeAG, utilisés par les couches inférieures. La fonction est transparente s'il y a déjà une association d'application active avec les mêmes adresses (ADP, ADS). La fonction échoue s'il existe déjà une association d'application avec un couple d'adresse (ADP, ADS) différent.

7 Echange de données par bus local avec DLMS/COSEM

7.1 Modèle

Le profile Euridis sous DLMS/COSEM est different des deux autres par le fait que la couche d'application est la couche d'application DLMS/COSEM comme specifié dans la CEI 62056-5-3: Ed 1.0. Elle permet ainsi aux équipements qui ont implémenté le modèle DLMS/COSEM de participer au bus Euridis.

Une autre difference est la présence d'une couche *Transport* et d'une couche *Gestion du Support*.

7.2 Couche physique

7.2.1 Généralités

Le protocole de couche Physique sous DLMS/COSEM est en tout point identique au protocole de couche physique avec ou sans DLMS jusqu'à la fin de la négociation correcte de la vitesse de transmission incluse. Une fois la phase négociation vitesse effectuée correctement, la couche physique COSEM s'applique. Elle ne diffère de la couche Physique avec et sans DLMS, que par la vitesse de transmission, la valeur MaxIndex qui vaut 255 et la modification de la valeur des time out TOL et TA10.

Après un signal «Appel général», les communications se déroulent en asynchrone et en semi-duplex à 1 200 bauds, 8 bits sans parité sur le bus durant toute la période de communication sans négociation de vitesse, et en asynchrone semi-duplex à la vitesse négociée, 8 bits sans parité après une négociation de vitesse correcte.

Cette négociation de vitesse a lieu uniquement pour les équipements non téléalimentés. Pour les équipements téléalimentés, afin de respecter les contraintes de consommation, la vitesse de communication demeure à 1 200 bauds durant toute la durée de la communication et MaxIndex à 128 octets. Le fonctionnement DLMS/COSEM sous Euridis démeure possible néanmoins avec ces paramètres initiaux.

Pour les équipements non téléalimentés le fonctionnement avec une couche d'Application DLMS/COSEM est possible avec ou sans changement de vitesse. Dans le cas où l'application décide d'effectuer un changement de vitesse, il convient que le positionnement des paramètres relatifs au changement de vitesse ait lieu avant l'établissement des associations d'application.

7.2.2 Paramètre de Physique

La communication démarre toujours à la vitesse de 1 200 bauds sans parité un bit de stop. La longueur maximale de la trame est de 128 octets.

Après la négociation de vitesse la nouvelle vitesse est celle négociée. La longueur maximale de la trame MaxIndex devient 255.

La valeur du nombre maximum MaxRSO de fenêtres d'écoute RSO pour le traitement d'une trame «Appel de Stations Oubliées» est fixée à 3.

Le timeout TOL d'attente maximale d'une requête issue de la couche supérieure est portée à 1 s.

7.2.3 Négociation de la vitesse

La gestion de la négociation de vitesse de transmission est du domaine de la couche gestion du Support. La couche physique est informée de la nouvelle vitesse à appliquer, avec ses conséquences sur MaxIndex et TOL. Comme conséquence de la modification de la valeur de TOL, suite à la négociation de la vitesse de transmission, le temps d'attente maximal (TA10) du premier octet d'une trame en réception est porté à une valeur supérieure à TOL, dépendant de l'application. Par défaut cette valeur est fixée à 1 100 ms.

7.2.4 Services et primitives de service de PhysiqueE/COSEM

L'utilisateur du protocole *Physique-E/COSEM* dispose des services et primitives de service donnés dans le Tableau 29.

Tableau 29 – Services et primitives de services de Physique-E/COSEM

Service	**Primitive de service**
Phy_DATA	Phy_DATA.req(Frame) Phy_DATA.ind(Frame)
Phy_UNACK	Phy_UNACK.req(Frame)
Phy_APPG	Phy_APPG.req(TypeAG) Phy_APPG.ind()
Phy_ASO	Phy_ASO.req(Frame) Phy_ASO.ind(Frame)
Phy_RSO	Phy_RSO.req(Frame, Window)
Phy_COLL	Phy_COLL.ind()
Phy_ALARM	Phy_ALARM_req() Phy_ALARM_ind()
Phy_ABORT	Phy_ABORT.req() Phy_ABORT.ind(ErrorNb)
Phy_SETUP	PHY_SETUP.req(params)

Le rôle attribué à chaque primitive est le suivant:

- *Phy_DATA.req(Frame)* permet à la couche Liaison de demander à la couche Physique l'émission d'une trame Frame
- *Phy_DATA.ind(Frame)* permet à la couche Physique d'informer la couche Liaison qu'une trame Frame est disponible;
- *Phy_UNACK.req(Frame)* permet à la couche Liaison de demander à la couche Physique l'émission d'une trame Frame sans attendre d'acquittement;
- *Phy_APPG.req(TypeAG)* permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'un signal " Appel Général ". La durée TypeAG du signal est soit AGN soit AGT;
- *Phy_APPG.ind()* permet à la couche *Physique* d'informer la couche *Liaison* de la fin d'émission d'un signal " Appel Général ";
- *Phy_ASO.req(Frame)* permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame " Appel des Stations Oubliées ";
- *Phy_ASO.ind(Frame)* permet à la couche *Physique* d'informer la couche *Liaison* qu'une trame Frame a été reçue dans l'un des intervalles de temps réservés pour la réponse des stations oubliées;
- *Phy_RSO.req(Frame, Window)* permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'une trame Frame de Réponse de Stations Oubliées dans l'intervalle de temps de numéro Window;
- *Phy_COLL.ind()* permet à la couche *Physique* d'informer la couche *Liaison* qu'une collision a été détectée dans l'un des intervalles de temps de réponse des stations oubliées;
- *Phy_ALARM.req()* permet à la couche *Liaison* de demander à la couche *Physique* l'émission d'un signal " Alarme ";
- *Phy_ALARM.ind()* permet à la couche *Physique* d'informer la couche *Liaison* qu'une Alarme a été détectée;

- *Phy_ABORT.req()* permet à la couche *Liaison* de demander à la couche *Physique* de mettre fin à son activité;
- *Phy_ABORT.ind(ErrorNb)* permet à la couche Physique d'informer la couche Liaison de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.
- *PHY_SETUP.req(params)* permet à la couche Liaison de demander à la couche ¨Physique de se reconfigurer en fonction des paramètres liés à la négociation de vitesse.

7.2.5 Transitions d'état

Tableau 30 – Transitions d'état de *Physique-E/COSEM*: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRSO=3 MaxIndex=128 Collision=FALSE SessionAGT=FALSE wait_time(TICB) Set_baurate(1200) TOL=100ms TA10=120ms	Stopped
Stopped	Phy_APPG.req(AG) & AG=AGN	stop_timer(TOAG) FlagAbort=FALSE TypeAG=AGN send_AG(TypeAG)	W.AG
Stopped	Phy_APPG.req(AG) & AG=AGT	SessionAGT=TRUE FlagAbort=FALSE TypeAG=AGT send_AG(TypeAG)	W.AG
Stopped	time_out(TOAG)	Phy_ABORT.ind(EP-2) SessionAGT=FALSE	*Initial*
Stopped	Phy_ABORT.req()	$none()	*Initial*
Stopped	data-carrier_on	init_timer(TAB) init_timer (TASB)	W.ETABS
W.ETABS	data_carrier_off	stop_timer(TASB) stop_timer(TAB)	*Initial*
W.ETABS	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind()	W.TASB
W.AG	AG_sent_event	Phy_APPG.ind() init_timer(TEMPO)	W.TAB
W.AG	Phy_ABORT.req()	FlagAbort=TRUE	W.AG
W.TAB	data-carrier_on	Carrier = TRUE init_timer(TAB) init_timer (TASB)	W.TAB
W.TAB	data-carrier_off	Carrier = FALSE stop_timer(TAB) stop_timer (TASB)	W.TAB
W.TAB	time_out(TEMPO) & not(FlagAbort) & not(Carrier)	init_timer(TOL)	M.Send
W.TAB	time_out(TEMPO) & FlagAbort & not(Carrier)	wait_time (TOL)	*T.Session*
W.TAB	time_out(TEMPO) & Carrier	init_timer(TOL)	W.ETAB
W.TAB	Phy_ABORT.req()	FlagAbort=TRUE	W.TAB
W.ETAB	time_out(TAB)	Phy_ABORT.ind(EP-3) Phy_ALARM.ind() stop_timer(TOL)	W.TASB
W.ETAB	data_carrier_off & not(FlagAbort)	stop_timer(TAB) stop_timer (TASB)	M.Send
W.ETAB	data_carrier_off & FlagAbort	stop_timer(TAB) stop_timer (TASB)	W.TOL

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
W.ETAB	Phy_ABORT.req()	FlagAbort=TRUE	W.ETAB
W.TASB	time_out(TASB)	$none()	*Initial*
W.TOL	time_out(TOL)	$none()	*T.Session*
M.Send	Phy_DATA.req(Frame)	Service=NORMAL	*SendFirst*
M.Send	Phy_UNACK.req(Frame)	Service=UNACKNOWLEDGED	*SendFirst*
M.Send	Phy_ASO.req(Frame)	Service=ASO	*SendFirst*
M.Send	Phy_ABORT.req()	$none()	M.Send
M.Send	time_out(TOL)	$none()	*T.Session*
M.Send	PHY_SETUP(params)	Set_baudrate(new_baudrate) MaxIndex=255 TOL=1000ms TA10=1100ms	M.Send
T.Session	SessionAGT=TRUE	init_timer(TOAG) Phy_ABORT.ind(EP-1) wait_time(TEMPO)	*Initial*
T.Session	SessionAGT=FALSE	Phy_ABORT.ind(EP-1) wait_time(TEMPO)	*Initial*
SendFirst	$true()	stop_timer(TOL) Size=size(Frame) Index=1 send_octet(Frame, Index) Size=Size-1 init_timer(TOE)	Sending
Sending	octet_sent_event & Size>0	Index=Index+1 send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0	stop_timer(TOE) wait_time(TAO) Index=1 Frame=""	*Answer*
Sending	Phy_ABORT.req()	stop_timer(TOE) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F) wait_time(TAO) init_timer(TA1O) FlagAbort=TRUE	M.Rec
Answer	Service=NORMAL I Service=UNACKNOWLEDGED	init_timer(TA1O)	M.Rec
Answer	Service=ASO	WinRSO=1 init_timer(TARSO) init_timer(TA1O)	M.Rec

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Rec	octet_received_event	stop_timer(TA1O) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
M.Rec	collision_detected_event	stop_timer(TA1O) Collision=TRUE init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	$none()	*Received*
M.Rec	Phy_ABORT.req()	FlagAbort=TRUE	M.Rec
Receiving	octet_received_event & Index<=MaxIndex	stop_timer(TAO) Index=Index+1 read_data(RecB) concat(Frame, RecB) init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Phy_ABORT.ind(EP-4F) FlagAbort=TRUE	*Received*
Receiving	collision_detected_event	stop_timer(TAO) Collision=TRUE init_timer(TAO)	Receiving
Receiving	time_out(TAO)	$none()	*Received*
Receiving	time_out(TARSO)	Phy_ABORT.ind(EP-5F) FlagAbort=TRUE	*Received*
Receiving	Phy_ABORT.req()	Flagabort = TRUE	*Received*
Received	Service=NORMAL & not(Flagabort)	Phy_DATA.ind(Frame) init_timer(TOL)	M.Send
Received	(Service=NORMAL & Flagabort) | Service=UNACKNOWLEDGED	wait_time(TOL)	*T.Session*
Received	Service=ASO & Collision & not(Flagabort)	Phy_COLL.ind() Collision=FALSE	*T.RSO*
Received	Service=ASO & not(Collision) & not(Flagabort)	Phy_ASO.ind(Frame)	*T.RSO*
Received	Service=ASO & Flagabort	$none()	*T.RSO*
T.RSO	(TypeAG=AGT) I (WinRSO>=MaxRSO) & (TypeAG=AGN)	stop_timer(TARSO)	*T.Session*
T.RSO	(WinRSO<MaxRSO) & (TypeAG=AGN)	Index=1 Frame=""	W.RSO
W.RSO	time_out(TARSO)	WinRSO=WinRSO+1 init_timer(TARSO) init_timer(TA1O)	M.Rec
W.RSO	Phy_ABORT.req()	Flagabort=TRUE	W.RSO

Tableau 31 – Transitions d'état de la gestion d'alimentation en énergie (Station Secondaire télé-alimentée seulement)

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	alarm_detection()	Flagalarm=TRUE FlagSendAlarm =FALSE station_power(ON) Set_baudrate(1200) MaxIndex=128 TOL=100ms TA10=120ms	Stopped
Initial	not(alarm_detection())	Flagalarm=FALSE Set_baudrate(1200) MaxIndex=128 TOL=100ms TA10=120ms	Stopped
Stopped	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB2
Stopped	occur(data_carrier_on)	init_timer(TOSEUIL) init_timer(TAGT)	W.TOSEUIL
W.TOSEUIL	time_out(TOSEUIL)& not(Flagalarm)	station_power(ON)	W.AGT
W.TOSEUIL	occur(data_carrier_off) & not(Flagalarm)	stop_timer(TOSEUIL) stop_timer(TAGT)	*Initial*
W.TOSEUIL	time_out(TOSEUIL) & Flagalarm	station_signal(ON) Tend = TOAG	W.AGT
W.TOSEUIL	occur(data_carrier_off) & Flagalarm	stop_timer(TOSEUIL) stop_timer(TAGT) Tend = TOAGN init_timer(Tend)	Hide
W.TOSEUIL	occur(cpt_carrier_on) & Flagalarm	init_timer(TVASB)	W.TVASB1
W.AGT	occur(data_carrier_off)	stop_timer(TAGT) init_timer(TOAPPEL)	W.Sel
W.AGT	time_out(TAGT) & not(Flagalarm)	station_power(OFF)	*Initial*
W.AGT	time_out(TAGT) & Flagalarm	init_timer(Tend)	Hide
W.Sel	occur(octet_received_event)	stop_timer(TOAPPEL) init_timer(TOBAVARD) init_timer(TAO)	Select
W.Sel	time_out(TOAPPEL) & not(Flagalarm)	station_power(OFF)	*Initial*
W.Sel	time_out(TOAPPEL) & Flagalarm	station_signal(OFF)	*Initial*
W.Sel	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Select	occur(octet_received_event)	stop_timer(TAO) init_timer(TAO)	Select
Select	time_out(TAO)	stop_timer(TOBAVARD) init_timer(TOPRE)	W.Answer
Select	time_out(TOBAVARD) & not(Flagalarm)	stop_timer(TAO) station_power(OFF)	*Initial*
Select	time_out(TOBAVARD) & Flagalarm	stop_timer(TAO) init_timer(Tend)	Hide
W.Answer	occur(octet_sent_event)	stop_timer(TOPRE) init_timer(Tend)	Hide
W.Answer	time_out(TOPRE) & not(Flagalarm)	station_power(OFF)	*Initial*
W.Answer	time_out(TOPRE) & Flagalarm	init_timer(Tend)	Hide
W.Answer	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
Hide	occur(octet_received_event) | occur(octet_sent_event) I (occur(data_carrier_on) & not(FlagSendAlarm))	stop_timer(Tend) init_timer(Tend)	Hide
Hide	occur(data_carrier_on) & FlagSendAlarm	stop_timer(Tend)	W.AGend

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Hide	time_out(Tend) & not(Flagalarm)	station_power(OFF)	*Initial*
Hide	time_out(Tend) & Flagalarm & not(FlagSendAlarm)	station_signal(OFF)	*Initial*
Hide	time_out(Tend) & Flagalarm & FlagSendAlarm	Send_AG(AGN)	W.AB
Hide	occur(cpt_carrier_on) & Flagalarm & not(FlagSendalarm)	init_timer(TVASB)	W.TVASB1
W.AGend	occur(data_carrier_off)	wait_time(TOALR) Send_AG(AGN)	W.AB
W.TVASB1	occur(cpt_carrier_off)	stop_timer(TVASB) init_timer(Tend)	Hide
W.TVASB1	time_out(TVASB)	FlagSendAlarm = TRUE init_timer(Tend)	Hide
W.TVASB1	time_out(Tend)	$none()	W.TVASB2
W.TVASB2	occur(cpt_carrier_off)	stop_timer(TVASB) station_signal(OFF)	*Initial*
W.TVASB2	occur(data_carrier_on)	$none()	W.TVASB1
W.TVASB2	time_out(TVASB)	Send_AG(AGN)	W.AB
W.AB	AG_sent_event	FlagSendAlarm = FALSE station_signal(OFF)	*Initial*

Tableau 32 – Transitions d'état de *Physique-E/Cosem:* Station Secondaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	energized()	MaxIndex=128 FlagRSO=FALSE FirstWinRSO=FALSE Setup_params(1200, TOL=100ms, TA10=160ms)	Stopped
Initial	not(energized())	MaxIndex=18 FlagRSO=FALSE FirstWinRSO=TRUE Set_baudrate(1200) TOL=100ms TA10=120ms	Stopped
Stopped	AG_received_event	Stop_timer(TOAG) init_timer(TA1O)	M.Rec
Stopped	Phy_ALARM.req()	TypeAG=ASB Send_AG(TypeAG)	W.ASB
Stopped	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	Stopped
M.Rec	octet_received_event	Stop_timer(TA1O) Index=2 Frame="" Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
M.Rec	time_out(TA1O)	Phy_ABORT(EP-1)	*WTOAG*
M.Rec	Phy_ABORT.req()	Stop_timer(TA1O)	*WTOAG*
M.Rec	PHY_SETUP(params)	Setup_params(new_baudrate, TOL=1000ms, TA10=1100ms) MaxIndex=255	M.Rec

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Receiving	octet_received_event & Index<=MaxIndex	Stop_timer(TAO) Index=Index+1 Read_data(RecB) Concat(Frame, RecB) Init_timer(TAO)	Receiving
Receiving	octet_received_event & Index>MaxIndex	Stop_timer(TAO) Phy_ABORT.ind(EP-4F)	*WTOAG*
Receiving	time_out(TAO)	Phy_DATA.ind(Frame) Init_timer(TOL)	M.Send
Receiving	Phy_ABORT.req()	Stop_timer(TAO)	*WTOAG*
M.Send	Phy_DATA.req(Frame)	Stop_timer(TOL) Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 Init_timer(TOE)	Sending
M.Send	Phy_RSO.req(Frame, Window)	Stop_timer(TOL) Wait_window(FirstWinRSO, Window) FirstWinRSO=FALSE Size=size(Frame) Index=1 Send_octet(Frame, Index) Size=Size-1 FlagRSO=TRUE Init_timer(TOE)	Sending
M.Send	time_out(TOL)	Init_timer(TA1O)	M.Rec
M.Send	Phy_ABORT.req()	Stop_timer(TOL)	*WTOAG*
Sending	octet_sent_event & Size>0	Index=Index+1 Send_octet(Frame, Index) Size=Size-1	Sending
Sending	octet_sent_event & Size=0 & not(FlagRSO)	Stop_timer(TOE) init_timer(TA1O)	M.Rec
Sending	octet_sent_event & Size=0 & FlagRSO	Stop_timer(TOE) Wait_time(TAO) FlagRSO=FALSE	*WTOAG*
Sending	Phy_ABORT.req()	Stop_timer(TOE)	*WTOAG*
Sending	time_out(TOE)	Phy_ABORT.ind(EP-3F)	*WTOAG*
W.ASB	time_out(TOAG)	MaxIndex=18 FirstWinRSO=TRUE	W.ASB
W.ASB	AG_sent_event	$none()	*Initial*
WTOAG	Not(energized)	init_timer(TOAG)	*Initial*
WTOAG	energized	$none()	*Initial*

Tableau 33 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Initial	Initialisation des variables de la couche
Stopped	Etat d'attente d'un signal " Appel Général "
W.ETABS (Wait for end of " Alarm-Bus ")	Etat d'attente de la fin d'un signal " Alarme-Bus " reçu dans l'état Stopped
W.AG (Wait for end of " Wakeup Call ")	Etat d'attente de la fin de l'émission d'un signal " Appel Général "
W.TAB (Wait " Alarm-Bus ")	Etat d'attente d'un signal " Alarme-Bus " durant la temporisation en fin d'émission d'un signal " Appel Général "
W.ETAB (Wait for end of " Alarm-Bus ")	Etat d'attente de la fin d'un signal " Alarme-Bus " reçu après l'émission d'un signal " Appel Général "
W.TASB	Attente du déclenchement du réveil TASB après le début de la réception d'un signal " Alarme-Bus "
W.TOL	Attente du déclenchement du réveil TOL
M.Send (Must Send)	Etat initial de l'émetteur, attente d'une trame à émettre
T.Session	Sous-état de test du type de session (avec Station Secondaire alimentée ou télé-alimentée)
SendFirst	Sous-état d'envoi du premier octet de la trame à émettre
Sending	Etat récurrent de l'émetteur, émission octet par octet
Answer	Sous-état d'aiguillage en fonction du service demandé
M.Rec (Must Receive)	Etat initial du récepteur, attente du premier octet d'une trame
Receiving	Etat récurrent du récepteur, réception octet par octet
Received	Sous-état de traitement de la trame reçue
T.RSO (Test last RSO)	Sous-état de test du dernier intervalle de temps pour la réception d'une trame RSO
W.RSO (Wait for end of an RSO time slot)	Etat d'attente de la fin du dernier intervalle de temps pour la réception d'une trame RSO
W.ASB	Attente de la fin d'émission d'un signal " Alarme Bus-Secondaire"
W.TOAG	Etat d'initialisation du délai de fin de session télé-alimentée TOAG si nécessaire
W.TOSEUIL	Etat d'attente du déclenchement du réveil TOSEUIL
W.AGT	Etat d'attente d'un signal " Appel Général " AGT
W.Sel (Wait for preSelection)	Etat d'attente d'une trame de présélection
Select	Etat de réception d'une trame de présélection
W.Answer	Etat d'attente d'une trame de réponse en provenance d'une station sélectionnée
Hide	Etat d'attente de la fin d'une sélection
W.Agend	Etat d'attente de la fin de la réception d'un AG
W.TVASB1	Etat d'attente du déclenchement du réveil TVASB pour un signal " Alarme Bus-Secondaire " durant une communication
W.TVASB2	Etat d'attente du déclenchement du réveil TVASB pour un signal " Alarme Bus-Secondaire " à la fin d'une communication
W.AB	Attente de la fin d'émission d'un signal " Alarme-Bus"

Tableau 34 – Définition des procédures, des fonctions et des événements classés dans l'ordre alphabétique

Procédure, fonction ou événement	Définition
AG_received_event	Evénement issu du modem informant de la détection d'un signal " Appel Général " AGN
AG_sent_event	Evénement issu du modem informant de la fin de l'émission d'un signal " Appel Général "
alarm_detection()	Vérification que la station a son mode alarme à Actif
collision_detected_event	Evénement issu du modem informant de la détection d'une trame erronée sur réception d'un octet
concat(Frame, RecB)	Concaténation de l'octet RecB dans la trame Frame en cours de constitution
data_carrier_on, data_carrier_off	Occurrence de la détection de l'apparition ou de la disparition de la porteuse sur le bus
energized()	Vérification que la station est alimentée
init_timer(TOAPPEL), init_timer(TOSEUIL), init_timer(TAGT), init_timer(TOBAVARD), init_timer(TOPRE), init_timer(TOL), init_timer(TOE), init_timer(TAO), init_timer(TA1O), init_timer(TARSO) init_timer(TOAG), init_timer(TVASB) or init_timer(TAB)	Armement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB ou TAB
occur(cpt_carrier_on), occur(cpt_carrier_off), occur(data_carrier_on), occur(data_carrier_off), occur(octet_received_event) or occur(octet_sent_event)	Occurrence (report sans consommation) de la détection de l'apparition ou de la disparition de la porteuse sur le bus secondaire, de la détection de l'apparition de la porteuse sur le bus, de la disparition de la porteuse, de la réception d'un octet ou de l'émission d'un octet
octet_received_event	Evénement issu du modem informant qu'un octet a été reçu
octet_sent_event	Evénement issu du modem informant qu'un octet a été émis
read_data(RecB)	Traitement de l'événement octet_received_event par lecture de l'octet RecB reçu (les bits sont transmis dans un ordre croissant)
send_AG(TypeAG)	Demande au modem d'effectuer l'émission d'un signal " Appel Général " d'une durée TypeAG (AGN ou AGT)
send_octet(Frame, Index)	Emission de l'octet de rang Index dans la trame Frame (les bits sont transmis dans un ordre croissant)
Setup_params(baurate, TOL, TA10)	Positionnement des paramètres baudrate, TOL et TA10.
size(Frame)	Calcul du nombre d'octets de la trame Frame
station_power(ON) or station_power(OFF)	Met en marche ou à l'arrêt l'alimentation en énergie de l'équipement
station_signal(ON) or station_signal(OFF)	Met en marche ou à l'arrêt l'émission de signal vers l'équipement sur le bus secondaire
stop_timer(TOAPPEL), stop_timer(TOSEUIL), stop_timer(TAGT), stop_timer(TOBAVARD), stop_timer(TOPRE), stop_timer(TOL), stop_timer(TOE), stop_timer(TAO), stop_timer(TA1O), stop_timer(TVASB) or stop_timer(TAB)	Désarmement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TVASB ou TAB

Procédure, fonction ou événement	Définition
stop_timer(TOAG) or stop_timer(TARSO)	Désarmement du réveil TOAGT ou TARSO s'il a été armé au préalable
time_out(TOAPPEL), time_out(TOSEUIL), time_out(TAGT), time_out(TOBAVARD), time_out(TOPRE), time_out(TOL), time_out(TOE), time_out(TAO), time_out(TA1O), time_out(TARSO), time_out(TOAG), time_out(TVASB) or stop_timer(TAB)	Déclenchement du réveil TOAPPEL, TOSEUIL, TAGT, TOBAVARD, TOPRE, TOL, TOE, TAO, TA1O, TARSO, TOAG, TVASB ou TAB
wait_time(TAO), wait_time(TICB), wait_time(TOL) or wait_time(TOALR)	Temporisation pendant le temps TAO, TICB, TOL ou TOALR
wait_window(FirstWinRSO, Window)	Temporisation calculée de la façon suivante: FirstWinRSO=VRAI ou Window=0 ==> 0 ms FirstWinRSO=FAUX et Window>0 ==> 40 ms + (TARSO*Window) ms (Le délai de 40 ms permet de garantir que l'émission a bien lieu dans l'intervalle de temps prévu)

Tableau 35 – Tableau récapitulatif des erreurs

EP-1	Délai TOL (Station Primaire) écoulé avant que la couche *Liaison* n'ait demandé l'envoi d'une trame, ou délai TA10 (Station Secondaire) écoulé avant la réception d'un caractère en provenance de la Station Primaire
	Cette erreur conduit à l'attente d'un signal " Appel Général " après avoir informé la couche *Liaison*
EP-2	Délai TOAG (Station Primaire) écoulé sans demande d'envoi
	Cette erreur conduit à l'attente d'un signal " Appel Général " après avoir informé la couche *Liaison*
EP-3	Réception d'une alarme
	Cette erreur conduit à réinitialiser la couche *Physique* après avoir informé la couche *Liaison*
EP-3F	Durée anormale d'émission constatée après l'échéance du réveil TOE
	Cette erreur conduit à réinitialiser la couche *Physique* après avoir informé la couche *Liaison*
EP-4F	Nombre d'octets reçus supérieur à MaxIndex (Emetteur trop bavard)
	Cette erreur conduit à réinitialiser la couche *Physique* après avoir informé la couche *Liaison*
EP-5F	Délai TARSO écoulé en recevant une trame RSO (Station Primaire seulement)
	Cette erreur conduit à réinitialiser la couche *Physique* après avoir informé la couche *Liaison*

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service Phy_ABORT.ind. La liste complète des numéros d'erreur fatale est fournie à l'Annexe C.

7.3 Couche Liaison

7.3.1 Généralités

La couche Liaison utilisée dans les échanges de données par bus local avec DLMS/COSEM est basée sur le même principe que la couche liaison E/D utilisée lors des échanges de

données par bus avec DLMS. Elle n'en diffère que par son comportement vis à vis des couches supérieures et le traitement de la négociation de la vitesse de transmission.

Au niveau supérieur, la couche liaison interface donc la couche transport et la couche de gestion du support de communication.

7.3.2 Identification des unités de données

La couche Liaison remet à la couche Transport DLMS/COSEM, toutes les PDU identifiés DATA+. Pour toutes les autres PDU le destinataire est la couche de gestion du support. A elle de reconnaître les PDU connus et gérés par elle et de les traiter. Ceux qui ne sont pas connus sont simplement ignorés. Voir Tableau 41 pour les commandes de Gestion du Support.

7.3.3 Rôle de la couche Liaison

La couche Liaison a pour but de:

- gérer la sérialisation et la désérialisation des données,
- synchroniser les trames en émission et réception,
- filtrer les trames en fonction des adresses primaire et secondaire,
- assurer une protection efficace contre les erreurs de transmission.

7.3.4 Gestion des échanges

Coté Emetteur, toute trame de données émise doit faire l'objet par la station réceptrice, d'un acquittement positif avant l'émission de la trame de données suivante. Après l'envoi d'une trame et la réception de l'acquittement de la trame précédemment émise, la trame courante est transmise. Le nombre de retransmissions est limité à MaxRetry. Au delà de ce nombre, la communication est interrompue au niveau Liaison et la couche Transport en est informée de même que la couche gestion du support.

Chaque fois qu'une trame est reçue, il y a émission d'une trame de réponse dans un délai compatible avec le temps TOL géré par la couche physique. Dans le cas où la couche Transport n'aurait pas de données à émettre disponibles, une DSDU avec un champ Text vide est envoyée, notifiant l'acquittement ou le non-acquittement de la DPDU reçue.

Le principe de gestion des acquittements /non-acquittements est identique à celui spécifié en 6.2.2.

7.3.5 Services et primitives de service de liaison

Tableau 36 – Services et primitives de services de liaison

Service	Primitive de service
DL_DATA	DL_DATA.req(Pr, Service_class, DSDU) DL_DATA.ind(Pr, Service_class, DSDU)
DL_ALARM	DL_ALARM.req() DL_ALARM.ind()
DL_ABORT	DL_ABORT.req(Strong) DL_ABORT.ind(ErrorNb)
DL_IB	DL _IB.req() DL _IB.ind()
DL _Discover	DL _Discover.req() DL _ Discover.ind()
DL _ChangeBaudrate	DL _ChangeBaudrate.req() DL _ChangeBaudrate.ind()
DL_PhysicalSetupParameters	DL_PhysicalSetup.req(Params)

Le role de chaque primitive est décrit ci-dessous:

DL_DATA.req(Pr, Service_class, DSDU) permet à la couche supérieure (Transport ou Gestion du Support) de demander à la couche Liaison le transfert d'un paquet avec la priorité Pr,[5] avec un service class CONFIRME on NON CONFIRME. Dans la requête, le paramètre service class concerne uniquement la station primaire. Lorsque le service_class est NON CONFIRME, la couche liaison de la station primaire doit utiliser une adresse de destination en diffusion.

DL_DATA.ind(Pr, Service_class, DSDU) permet à la couche Liaison d'informer la couche supérieure concernée de l'arrivée d'un paquet de données DSDU avec la priorité Pr. Dans l'indication, le paramètre service class concerne uniquement la station secondaire. Lorsque l'adresse de destination est une adresse en diffusion, la couche liaison de la station secondaire doit positionner le paramètre Service_class à NON CONFIRME, pour la couche supérieure.

DL_ALARM.req() permet à la couche Gestion du Support de la station secondaire de demander à la couche Liaison, l'envoi d'une alarme.

DL_ALARM.ind() permet à la couche Liaison de la station primaire d'avertir la couche Gestion du Support de la présence d'une alarme.

DL_ABORT.req(Strong[6]) permet aux couches supérieures de demander à la couche Liaison de mettre fin à son activité avec la priorité Strong.

DL_ABORT.ind(ErrorNb) permet à la couche Liaison d'informer la couche Gestion du Support de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

DL_IB.req() permet à la couche Gestion du Support de la station primaire de demander à la couche Liaison l'initialisation du bus.

DL_IB.ind() permet à la couche Liaison de la station secondaire d'informer la couche Gestion du Support de la présence d'une initialisation de bus.

DL _Discover.req() permet à la couche Gestion du Support de demander à la couche Liaison, l'envoi d'une trame d'Appel de stations oubliées.

DL_Discover.ind() permet à la couche Liaison d'informer la couche Gestion du Support de la présence d'une trame d'appel de stations oubliées.

DL_ChangeBaudrate.req() permet à la couche Gestion du Support de demander à la couche Liaison l'envoi d'une trame de négociation de la vitesse.

DL_ChangeBaudrate.ind() permet à la couche Liaison d'informer la couche Gestion du Support de la présence d'une trame de négociation de la vitesse.

DL_PhysicalSetupParameters.req(Params) permet à la couche gestion du support de demander à Liaison d'effectuer les modifications des paramètres liés au changement de vitesse. Ces paramètres sont constitués du baudrate négocié, du time out TOL qui passe à une seconde, le time out TA1O de 1 100 ms et MaxIndex qui passe de 128 à 255 octets.

---

5 Le paramètre de priorité Pr différencie le traitement d'un service urgent tel que InformationReport (niveau Pr=1) d'un autre service DLMS (niveau Pr=0).

6 Le paramètre de priorité Strong différencie le traitement des erreurs fatales (Strong=1) de celui d'une autre demande de déconnexion physique (Strong=0) initialisée par la sous-couche *Application*.

7.3.6 Paramètres de Liaison de données

Les parametres Liaison de données sont les mêmes que pour la couche Liaison avec DLMS voir 6.2.4.

A la fin de la procédure de changement de vitesse, du coté de l'équipement primaire, la couche liaison gère un time out TES (time out End of Setup) d'une valeur de 50 ms, avant le traitement de toute nouvelle requête destinée à l'équipement secondaire. L'objectif de ce time out est de permettre aux deux équipements de positionner correctement leur UART.

7.3.7 Transitions d'état

Tableau 37 – Transitions d'état de DLMS/COSEM Liaison-E/D Cosem: Station Primaire

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Initial	$true()	MaxRetry=2 MaxChain = 5 init_incrChain()	Stopped
Stopped	exist_dl_req()	NbChain = 0 Index=0 RepeatASO=FALSE context(ADP, ADS, TypeAG) init(TypeAG) Phy_APPG.req(TypeAG)	W.AG
Stopped	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	Stopped
Stopped	Phy_ALARM.ind()	DL_ALARM.ind()	Stopped
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain = 0	$none()	*T.Req*
W.AG	Phy_APPG.ind() & not(RepeatASO) & NbChain <> 0	NbChain = 0	*M.Send*
W.AG	Phy_APPG.ind() & RepeatASO	RepeatASO=FALSE Phy_ASO.req(Fr)	M.RSO
W.AG	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
W.EndS	(Phy_ABORT.ind(EP-2) & TypeAG=AGT) I (Phy_ABORT.ind(EP-1) & TypeAG=AGN)	$none()	Stopped
W.EndS	Phy_ALARM.ind	DL_ALARM.ind()	Stopped
W.EndS	Phy_ABORT.ind(ErrorNb) & ErrorNb<>EP-1 & ErrorNb<>EP-2	DL_ABORT.ind(ErrorNb)	W.EndS
T.Req	exist_dl_req(DL_IB.req())	Fr="" Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, IB, Fr) Fr=concat(Fr, crc(Fr)) Phy_UNACK.req(Fr)	W.EndS
T.Req	exist_dl_req(DL_Discover.req(DSDU)) & TypeAG=AGN	MaxRSO=3 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
T.Req	exist_dl_req(DL_Discover.req(DSDU)) & TypeAG=AGT	MaxRSO=1 NbRSO=1 ListRSO="" Collision=FALSE Fr=DSDU Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, ASO, Fr) Fr=concat(Fr, crc(Fr)) Phy_ASO.req(Fr)	M.RSO
T.Req	not (exist_dl_req(DL_IB.req()) | exist_dl_req(DL_ Discover.req(_)) | exist_dl_change_baudrate_req())	Pr=0 Send="00"B Confirm="11"B Fr="" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
T.Req	exist_dl_req(DL_ChangeBaudrate.req(proposed_baudrate)	Fr="proposed_baudrate" Index=Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Fr=concat(Size, ADS, ADP, XBR, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.RSO	Phy_ASO.ind(Frame) & size(Frame)=0	$none()	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & check_frame(Frame) & command(Frame)=RSO	build_RSO(ListRSO, Frame)	*T.RSO*
M.RSO	Phy_ASO.ind(Frame) & not(check_frame(Frame)) & size(Frame)<>0	Collision=TRUE	*T.RSO*
M.RSO	Phy_COLL.ind()	Collision=TRUE	*T.RSO*
M.RSO	DL_ABORT.req(_)	Phy_ABORT.req()	W.EndS
M.RSO	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
T.RSO	MaxRSO=1 & Collision	MaxRSO=3 Collision=FALSE RepeatASO=TRUE Phy_APPG.req(AGN)	W.AG
T.RSO	(MaxRSO=1 & not(Collision)) | (MaxRSO<>1 & NbRSO>=MaxRSO)	DL_Discover.ind (Collision, ListRSO)	W.EndS
T.RSO	NbRSO<MaxRSO	NbRSO=NbRSO+1	M.RSO
M.Send	exist_dl_data_req(DL_DATA.req(Pr=1, Service_class, DSDU)) &)) & not(TreqTimeout()) NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec
M.Send	not(DL_DATA.req(Pr=1, _)) & exist_dl_data_req(DL_DATA.req(Pr=0, Service_class, DSDU)) &)) & not(TreqTimeout()) NbChain < MaxChain	Send=incr(Send) Ack_expected=TRUE Fr=DSDU Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr) Stop_timer(Treq)	M.Rec

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
M.Send	not(DL_DATA.req(_, _)) & TreqTimeout() & NbChain < MaxChain	Pr=0 Fr="" Index= Index + 1 NbChain = NbChain + IncrChain Size=size_frame(Fr) Com=com(DATA+,Pr,Send,Confirm) Fr=concat(Size, ADS, ADP, Com, Fr) Fr=concat(Fr, crc(Fr)) Phy_DATA.req(Fr)	M.Rec
M.Send	NbChain >= MaxChain	Phy_APPG.req(AGN)	W.AG
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & is_text(Frame)	DL_DATA.ind(extract_prty(Frame), extract_text(Frame)) Confirm=incr(Confirm) Ack_expected=FALSE Index = 0 initTimer(Treq)	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & not(is_text(Frame))	Ack_expected=FALSE Index = 0 initTimer(Treq)	*M.Send*
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index<=MaxRetry	Phy_DATA.req(Fr) Index=Index+1	M.Rec
M.Rec	Phy_DATA.ind(Frame) & not(check_frame(Frame) & check_address(Frame) & is_data+(Frame) & is_ack(Frame)) & Index>MaxRetry	DL_ABORT.ind(EL-2F) Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=0) & not(DL_DATA.req(_, _)) & Ack_expected=FALSE	Phy_ABORT.req()	W.EndS
M.Rec	DL_ABORT.req(Strong=1)	Phy_ABORT.req()	W.EndS
M.Rec	Phy_ABORT.ind(ErrorNb)	DL_ABORT.ind(ErrorNb)	W.EndS
M.Rec	Phy_DATA.ind(Frame) & (check_frame(Frame) & (check_address(Frame) & (Com =XBA)	DL_ChangeBaudrate.ind(accepted_baurate)	M.Rec
M.Rec	exist_dl_physical_setup_req(params)	Phy_SETUP(params) wait(TES)	T.Req