IEC 61158-6-13

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 6-13: Application layer protocol specification – Type 13 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-13: Spécification du protocole de la couche application – Éléments de type 13

IEC 61158-6-13:2014-08(en-fr)

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

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

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

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

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

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

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

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

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

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

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

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

IEC 61158-6-13

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 6-13: Application layer protocol specification – Type 13 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-13: Spécification du protocole de la couche application – Éléments de type 13

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XB
CODE PRIX

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1763-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-13: Application layer protocol specification – Type 13 elements

FOREWORD

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

Attention is drawn to the fact that the use of the associated protocol type is restricted by its intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a layer protocol Type to be used with other layer protocols of the same Type, or in other Type combinations explicitly authorized by its intellectual-property-right holders.

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

International Standard IEC 61158-6-13 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2007. This edition constitutes a technical revision. The main changes with respect to the previous edition are listed below:

- addition of synchronization feature,
- corrections and
- editorial improvements.

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

FDIS	Report on voting
65C/764/FDIS	65C/774/RVD

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

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

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

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

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

INTRODUCTION

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

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

– as a guide for implementors and designers;

– for use in the testing and procurement of equipment;

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-13: Application layer protocol specification – Type 13 elements

1 Scope

1.1 General

The fieldbus application layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a “window between corresponding application programs.”

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 13 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 13 fieldbus application layer in terms of

a) the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
b) the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
c) the application context state machine defining the application service behavior visible between communicating application entities;
d) the application relationship state machines defining the communication behavior visible between communicating application entities.

The purpose of this standard is to define the protocol provided to

1) define the wire-representation of the service primitives defined in IEC 61158-5-13, and
2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 13 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-13.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in IEC 61158-6.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

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

IEC 61158-3-13, *Industrial communication networks – Fieldbus specifications – Part 3-13: Data-link layer service definition – Type 13 elements*

IEC 61158-4-13, *Industrial communication networks – Fieldbus specifications – Part 4-13: Data-link layer protocol specification – Type 13 elements*

IEC 61158-5-13, *Industrial communication networks – Fieldbus specifications – Part 5-13: Application layer service definition – Type 13 elements*

ISO/IEC 7498 (all parts), *Information technology – Open Systems Interconnection – Basic Reference Model*

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

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

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 9899, *Information technology – Programming languages – C*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 ISO/IEC 7498-1 terms

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

3.1.1 application entity
3.1.2 application process
3.1.3 application protocol data unit
3.1.4 application service element
3.1.5 application entity invocation
3.1.6 application transaction
3.1.7 transfer syntax

3.2 ISO/IEC 8822 terms

For the purposes of this document, the following term as defined in ISO/IEC 8822 applies:

3.2.1 abstract syntax

3.3 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

3.3.1 application-context
3.3.2 application-process-type
3.3.3 application-service-element
3.3.4 application control service element

3.4 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply:

3.4.1 any type
3.4.2 bitstring type
3.4.3 boolean type
3.4.4 choice type
3.4.5 false
3.4.6 integer type
3.4.7 module
3.4.8 null type
3.4.9 object identifier
3.4.10 octetstring type
3.4.11 production
3.4.12 simple type
3.4.13 sequence of type
3.4.14 sequence type
3.4.15 structured type
3.4.16 tag
3.4.17 tagged type

3.4.18 true

3.4.19 type

3.5 Terms and definitions from IEC 61158-5-13

3.5.1 application relationship

3.5.2 client

3.5.3 error class

3.5.4 publisher

3.5.5 server

3.5.6 subscriber

3.6 Other terms and definitions

The following terms and definitions are used in this standard:

3.6.1
receiving
service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU

3.6.2
resource
processing or information capability of a subsystem

3.6.3
sending
service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU

3.6.4
managing node
node that can manage the SCNM mechanism

3.6.5
controlled node
node without the ability to manage the SCNM mechanism

3.7 Abbreviations and symbols

AE	Application entity
AL	Application layer
AP	Application process
APDU	Application protocol data unit
AR	Application relationship
AREP	Application relationship end point
ARPM	Application relationship protocol machine
ASnd	Asynchronous Send (Type 13 frame type)
BNB-PEC	Buffered network-scheduled bi-directional pre-established connection
BNU-PEC	Buffered network-scheduled uni-directional pre-established connection
CmdL	Command layer
CN	Controlled node

cnf	confirmation
DL-	(as a prefix) data-link-
DLCEP	Data-link connection end point
DLL	Data-link layer
DLME	Data-link-management entity
DLSAP	Data-link service access point
DLSDU	DL-service-data-unit
DMPM	DLL mapping protocol machine
DNS	Domain name service
FAL	Fieldbus application layer
ind	indication
IP	Internet protocol (see RFC 791)
MAC	Media access controll
MN	Managing node
NMT	Network management
OD	Object dictionary
PDO	Process data object
PDU	Process data unit
QUB-CL	Queued user-triggered bi-directional connectionless
QUB-COS	Queued user-triggered bi-directional connection-oriented with segmentation
QUU	Queued user-triggered uni-directional
req	request
rsp	response
SDO	Service data object
SeqL	Sequence layer
UDP	User datagram protocol

4 FAL syntax description

4.1 General

This description of the Type 13 abstract syntax uses formalisms similar to ASN.1, although the encoding rules differ from that standard.

4.2 FAL-AR PDU abstract syntax

4.2.1 Top level definition

```
APDU ::= CHOICE {
      [3] Isoc1
      [4] Isoc2
      [5] Asyn1
      [6] Asyn2
}
```

4.2.2 Isoc1

```
Isoc1 ::= SEQUENCE {
      message-type
      destination
      source
      reserved
      signaling-flags
      PDO-version
      reserved8
      size
      PDO-payload
}
```

4.2.3 Isoc2

```
Isoc2 ::= SEQUENCE {
      message-type
      destination
      source
      NMT-status
      signaling-flags
      PDO-version
      reserved8
      size
      PDO-payload
}
```

4.2.4 Asyn1

```
Asyn1 ::= SEQUENCE {
      message-type
      destination
      source
      NMT-status
      signaling-flags
      requested-service-ID
      requested-service-target
      fieldbus-version
      reserved8
      pduBody     CHOICE{
         [1h...5h] reserved
         [6h] Sync-request
         [7h...FFh] reserved
      }
}
```

4.2.5 Asyn2

```
Asyn2 ::= SEQUENCE {
   message-type
   destination
   source
   service-ID
   pduBody        CHOICE {
      [1h] ident-response
      [2h] status-response
      [3h] NMT-request
      [4h] NMT-command
      [5h] SDO
      [6h] Sync-response
      [A0h...FEh] manufacturer-specific
      [FFh] reserved
   }
}
```

4.2.6 Message-type

message-type ::= Unsigned8 — Contains the context specific APDU tags

4.2.7 Addresses

destination ::= Unsigned8 — Node address (1...255)
source ::= Unsigned8 — Node address (1...250, 253, 254)

4.2.8 Service-ID

service-ID ::= Unsigned8 — Contains the context specific tags for the pduBody

4.2.9 Reserved8

reserved8 ::= Unsigned8

4.2.10 Reserved16

reserved16 ::= Unsigned16

4.2.11 Reserved24

reserved24 ::= Unsigned24

4.2.12 Signaling-flags

```
signaling-flags ::= BitString {
   RD          (0)
   ER          (1)
   EA          (2)
   EC          (3)
   EN          (4)
   MS          (5)
   PS          (6)
   MC          (7)
   RS_bit1     (8)
   RS_bit2     (9)
   RS_bit3     (10)
   PR_bit1     (11)
   PR_bit2     (12)
   PR_bit3     (13)
   reserved    (14)
   reserved    (15)
}
```

The different APDU types use the flags as listed in Table 1. In all cases without "x" the flags are present but not written resp. interpreted.

Table 1 – Use of signaling-flags

	Isoc1	Isoc2	Asyn1	IdentResponse	StatusResponse	SyncResponse
RD	x	x				
ER			x			
EA	x		x			
EC					x	
EN		x			x	
MS	x	x				
PS						
MC						
RS	x	x		x	x	
PR	x	x		x	x	

The usage of these flags is specified in IEC 61158-3-13 and IEC 61158-4-13.

4.2.13 PDO-version

PDO-version ::= Unsigned8 — High nibble: main version; low nibble: sub version

4.2.14 Size

size ::= Unsigned8 — Size of PDO payload; max. 1490 octets due to Ethernet restrictions and protocol overhead

4.2.15 PDO-payload

PDO-payload ::= Any

4.2.16 NMT-status

```
NMT-status ::= CHOICE {
   NMT_GS_OFF                    Unsigned8 ::= 0000 0000b
   NMT_xS_NOT_ACTIVE             Unsigned8 ::= 0001 1100b
   NMT_xS_PRE_OPERATIONAL_1      Unsigned8 ::= 0001 1101b
   NMT_xS_PRE_OPERATIONAL_2      Unsigned8 ::= 0101 1101b
   NMT_xS_READY_TO_OPERATE       Unsigned8 ::= 0110 1101b
   NMT_xS_OPERATIONAL            Unsigned8 ::= 1111 1101b
   NMT_xS_STOPPED                Unsigned8 ::= 0100 1101b
   NMT_xS_BASIC_ETHERNET         Unsigned8 ::= 0001 1110b
}
```

NOTE If sender = MN: "x" := "M"; if sender = CN: "x" := "C".

4.2.17 Requested-service-ID

```
requested-service-ID ::= CHOICE {
   no-service                    [0h] IMPLICIT Unsigned8
   ident-request                 [1h] IMPLICIT Unsigned8
   status-request                [2h] IMPLICIT Unsigned8
   NMT-req-inv                   [3h] IMPLICIT Unsigned8
   manufacturer-specific         [A0h]... [FEh] IMPLICIT Unsigned8
   unspecified-invite            [FFh] IMPLICIT Unsigned8
}
```

4.2.18 Requested-service-target

requested-service-target ::= Unsigned8 — Node address (1...255); not assigned (0)

4.2.19 Fieldbus-version

fieldbus-version ::= Unsigned8 — High nibble: main version; low nibble: sub version

4.3 Abstract syntax of Asyn1 pduBody

4.3.1 Sync-request

4.3.1.1 Overview

```
Sync-request ::= SEQUENCE {
   synchronization-control    Bitstring     — see 4.3.1.2
   PRes-time                  Unsigned32    — time delay between end of the reception of the PRes from MN
                                              and start of sending the own time-triggered PRes in ns
   reserved                   Unsigned32
   sync-MN-delay              Unsigned32    — propagation delay between MN and CN in ns
   reserved                   Unsigned32
   fallback-timeout           Unsigned32    — SoC timeout for deactivating the time-triggered sending of PRes
                                              in state NMT_CS_PRE_OPERATIONAL_2 in ns
   destination-MAC-address    Unsigned32    — destination MAC address of the node the Sync-request is sent to
}
```

NOTE The above listed elements are sometimes summarized as follows:
"synchronization-control" through "destination-MAC-address" are summarized under the term "sync-control".

4.3.1.2 Synchronization-control

Synchronization-control ::= Bitstring {

PRes-time-valid	(0)	— The parameter PRes-time is valid
Reserved bit1	(1)	
sync-MN-delay-valid	(2)	— The parameter sync-MN-delay is valid
reserved bit2	(3)	
fallback-timeout-valid	(4)	— The parameter fallback-timeout is valid
reserved bit3	(5)	
MAC-address-valid	(4)	— The parameter destination-MAC-address is valid
reserved bit4 through bit26	(7)...(29)	
PRes-mode-reset	(30)	— Deactivate the time-triggered sending of PRes
PRes-mode-set	(31)	— Activate the time-triggered sending of PRes. This bit overules bit30

}

4.4 Abstract syntax of Asyn2 pduBody

4.4.1 Ident-response

4.4.1.1 Overview

Ident-response ::= SEQUENCE {

signaling-flags		
NMT-status		
reserved8		
fieldbus-version		
reserved8		
feature-flags	BitString	— (see 4.4.1.2)
MTU	Unsigned16	— size of the largest possible IP frame incl. header
poll-in-size	Unsigned16	— actual CN setting for Isoc1 data block size
poll-out-size	Unsigned16	— actual CN setting for Isoc2 data block size
response-time	Unsigned32	— time required by the CN to respond to Isoc1
reserved16		
device-type	Unsigned32	— CN's device type
vendor-ID	Unsigned32	— CN's vendor ID
product-code	Unsigned32	— CN's product code
revision-number	Unsigned32	— CN's revision number
serial-number	Unsigned32	— CN's serial number
vendor-specific-extension-1	Unsigned64	— for vendor specific purpose, to be filled with zeros if not used
verify-configuration-date	Unsigned32	— CN's configuration date
verify-configuration-time	Unsigned32	— CN's configuration time
application-sw-date	Unsigned32	— CN's application software date
application-sw-time	Unsigned32	— CN's application software time
IP-address	Unsigned32	— current IP address value of the CN
subnet-mask	Unsigned32	— current IP subnet mask of the CN
default-gateway	Unsigned32	— current IP default gateway of the CN
host-name	VisibleString32	— current DNS host name of the CN
vendor-specific-extension-2	SEQUENCE SIZE(48) OF Unsigned8	—for vendor specific purpose, to be filled with zeros if not in use

}

NOTE Some of the above listed elements are sometimes summarized as follows:

"poll-in-size" through "response-time" are summarized under the term "cycle-timing",

"device-type" through "serial-number" under "identity",

"verify-configuration-date" and "verify-configuration-time" under "verify-configuration",

"application-sw-date" and "application-sw-time" under "application-software-version",

"vendor-specific-extension-1" and "vendor-specific-extension-2" under "vendor-specific-extensions",

"IP-address" through "default-gateway" under "IP-address".

4.4.1.2 Feature-flags

```
feature-flags ::= BitString {
   Isochronous                    (0)        — device may be isochronously accessed via Isoc1
   SDO by UDP/IP                  (1)        — device supports SDO communication via UDP/IP
   SDO by ASnd                    (2)        — device supports SDO communication via ASnd
   reserved for future use        (3)
   NMT-info services              (4)        — device supports NMT Info Services
   Extended NMT-state-commands    (5)        — device supports Extended NMT State Commands
   Dynamic PDO mapping            (6)        — device supports dynamic PDO Mapping
   NMT services by UDP/IP         (7)        — device supports NMT Services by UDP/IP
   Configuration manager          (8)        — device supports Configuration Manager functions
   Multiplexed access             (9)        — CN device supports multiplexed isochronous access.
   Node-ID setup by SW            (10)       — device supports NodeID setup by software
   MN basic ethernet mode         (11)       — MN device supports Basic Ethernet Mode
   Routing Type 1 support         (12)       — device supports Routing Type 1 functions
   Routing Type 2 support         (13)       — device supports Routing Type 2 functions
   WriteMultipleByIndex           (14)       — device supports WriteMultipleByIndex SDO service
   ReadMultipleByIndex            (15)       — device supports ReadMultipleByIndex SDO service
   reserved bit1 through bit2     (16)...(17)
   Time-triggered PRes            (18)       — device supports time-triggered sending of PRes
   reserved bit3 through bit16    (19)...(31)
}
```

4.4.2 Status-response

4.4.2.1 Overview

```
Status-response ::= SEQUENCE {
   signaling-flags
   NMT-status
   reserved24
   static-error-bit-field
   List-of-errors
}
```

4.4.2.2 Static-error-bitfield

```
static-error-bit-field ::= SEQUENCE {
   error-register          BitString                          — (see 4.4.2.3)
   reserved8
   specific-errors         SEQUENCE SIZE (6) OF Unsigned8     — Device profile or vendor specific errors
}
```

4.4.2.3 Error-register

```
error-register ::= BitString {
   Generic error           (0)                                — 0, if no static error present, else 1
   Current                 (1)        OPTIONAL
   Voltage                 (2)        OPTIONAL
   Temperature             (3)        OPTIONAL
   Communication error     (4)        OPTIONAL
   Device profile specific (5)        OPTIONAL
   reserved                (6)        OPTIONAL        — always 0
   Manufacturer specific   (7)        OPTIONAL
}
```

4.4.2.4 List-of-errors

```
List-of-errors ::= SEQUENCE SIZE (k) OF ErrorEntry        — k :>= 2, depends on device configuration
```

4.4.2.5 ErrorEntry

```
ErrorEntry ::= SEQUENCE {
   error-type              Unsigned16         — (see 4.4.2.6)
   error-code              Unsigned16         — (see 4.4.2.6 and Clause A.3)
   time-stamp              Unsigned64
   additional-information  Unsigned64
}
```

4.4.2.6 Error-type

The possible values in error-type and their meaning are listed in Table 2.

Table 2 – Values of error-type

Octet	Bit	Value	Description
0 .. 1	15 (status)	0b	Error-history entry
		1b	Status entry in Status-response frame (Bit 14 shall be set to 0b)
	14 (send)	0b	Error-history entry only
		1b	Additional to the error-history entry the entry shall also be entered in to the emergency queue of the error signaling
	13 .. 12 (mode)	0h	Not allowed in error-history entry. Entries with this mode may only be used by the error signaling itself to indicate the termination of the history entries in the Status-response frame
		1h	An error has occurred and is active (e.g. short circuit of output detected)
		2h	An active error was cleared (e.g. no short circuit anymore) (not allowed for status entries)
		3h	An error / event occurred (not allowed for status entries)
	11 .. 0 (profile)	000h	Reserved
		001h	error-code contains a vendor specific error code
		002h	error-code contains Type 13 network specific communication profile errors (see Clause A.3)
		003h .. FFFh	error-code contains device profile specific errors

4.4.3 NMT-request

```
NMT-request ::= SEQUENCE {
   NMT-requested-command-ID          Unsigned8      — value range see Clause A.2
   NMT-requested-command-target      Unsigned8      — target node address
   NMT-requested-command-data        Any
}
```

4.4.4 NMT-command

```
NMT-command ::= SEQUENCE {
   NMT-command-ID CHOICE {                          — value range see Clause A.2
      NMT-state-command              Unsigned8
      NMT-info                       Unsigned8
   }
   reserved8
   NMT-command-data   CHOICE {
      date-time       TimeOfDay                     — only if NMT-command-ID = B0h;
                                                      see Clause A.4
      node-states     SEQUENCE SIZE (255) OF Unsigned8 — if NMT-command-ID = 96h;
                                                      see Clause A.4;
                                                      reports each node state individually
      node-list                                     — if NMT-command-ID = 40h...95h, A0h;
                                                      see Clause A.4
      NULL                                          — else
   }
}
```

4.4.5 SDO

4.4.5.1 Overview

```
SDO ::= SEQUENCE {
   SequenceLayer
   CommandLayer
}
```

4.4.5.2 SequenceLayer

```
SequenceLayer ::= SEQUENCE {
   SequenceFlags     Bitstring {
      rcon_bit1                        (0)      — 0: no connection; 1: initialisation; 2: connection valid
      rcon_bit2                        (1)         3: error response (retransmission requested)
      rsnr_bit1                        (2)      — 0..63
      rsnr_bit2                        (3)
      rsnr_bit3                        (4)
      rsnr_bit4                        (5)
      rsnr_bit5                        (6)
      rsnr_bit6                        (7)
      scon_bit1                        (8)      — 0: no connection; 1: initialisation; 2: connection valid
      scon_bit2                        (9)         3: connection valid with acknowledge request
      ssnr_bit1                        (10)     — 0..63
      ssnr_bit2                        (11)
      ssnr_bit3                        (12)
      ssnr_bit4                        (13)
      ssnr_bit5                        (14)
      ssnr_bit6     (15)
   }
   reserved16
}
```

4.4.5.3 CommandLayer

```
CommandLayer ::= SEQUENCE {
   SDOCommandHeader          SEQUENCE {
      reserved8
      invoke-ID
      segmentation_abort_response
      command-ID
      segment-size
      reserved16
   }
   SDO-command      CHOICE {
      write-by-index-request               [1h] IMPLICIT WriteByIndex-RequestPDU
      read-by-index-request                [2h] IMPLICIT ReadByIndex-RequestPDU
      write-all-by-index-request           [3h] IMPLICIT WriteAllByIndex-RequestPDU
      read-all-by-index-request            [4h] IMPLICIT ReadAllByIndex-RequestPDU
      write-multiple-by-index-request      [31h] IMPLICIT WriteMultipleByIndex-RequestPDU
      read-multiple-by-index-request       [32h] IMPLICIT ReadMultipleByIndex-RequestPDU
      abort                                IMPLICIT AbortPDU
      write-by-index-response              [1h] IMPLICIT WriteByIndex-ResponsePDU
      read-by-index-response               [2h] IMPLICIT ReadByIndex-ResponsePDU
      write-all-by-index-response          [3h] IMPLICIT WriteAllByIndex-ResponsePDU
      read-all-by-index-response           [4h] IMPLICIT ReadAllByIndex-ResponsePDU
      write-multiple-by-index-response     [31h] IMPLICIT WriteMultipleByIndex-ResponsePDU
      read-multiple-by-index-response      [32h] IMPLICIT ReadMultipleByIndex-ResponsePDU
   }
}
```

4.4.5.4 Invoke ID

```
invoke-ID ::= Unsigned8
```

4.4.5.5 Segmentation_abort_response

```
segmentation_abort_response ::= BitString {
   reserved                            (0)
   reserved                            (1)
   reserved                            (2)
   reserved                            (3)
   segmentation_bit1                   (4)      — 0: Expedited transfer; 1: initiate segment transfer
   segmentation_bit2                   (5)         2: segment; 3: segment transfer complete
   abort                               (6)      — 0: transfer ok; 1: abort
   response                            (7)      — 0: request; 1: response
}
```

4.4.5.6 Command-ID

```
command-ID ::= Unsigned8                    —  Contains context specific tag for SDO-command
```

4.4.5.7 Segment-size

segment-size::= Unsigned16 — Length of segment data. Counting from the beginning of the "SDO-command". Valid value range: 0...1458

4.4.5.8 WriteByIndex services

```
WriteByIndex-RequestPDU ::= SEQUENCE {
   data-size                    —  only present if in SDOCommandHeader segmentation = "initiate"
   index
   sub-index
   reserved8
   payload-data
}

WriteByIndex-ResponsePDU ::= NULL
```

4.4.5.9 ReadByIndex services

```
ReadByIndex-RequestPDU ::= SEQUENCE {
   index
   sub-index
   reserved16
}

ReadByIndex-ResponsePDU ::= SEQUENCE {
   data-size                    —  only present if in SDOCommandHeader segmentation = "initiate"
   payload-data
}
```

4.4.5.10 WriteAllByIndex services

```
WriteAllByIndex-RequestPDU ::= SEQUENCE {
   data-size                    —  only present if in SDOCommandHeader segmentation = "initiate"
   index
   reserved16
   payload-data
}

WriteAllByIndex-ResponsePDU ::= NULL
```

4.4.5.11 ReadAllByIndex services

```
ReadAllByIndex-RequestPDU ::= SEQUENCE {
   index
   reserved16
}

ReadAllByIndex-ResponsePDU ::= SEQUENCE {
   data-size                    —  only present if in SDOCommandHeader segmentation = "initiate"
   payload-data
}

WriteByName-ResponsePDU ::= NULL
```

4.4.5.12 WriteMultipleByIndex services

```
WriteMultipleByIndex-RequestPDU ::= SEQUENCE {
   data-size                          —  only present if in SDOCommandHeader segmentation = "initiate"
   offset (k)                         —  Byte offset of next payload data set
   index
   sub-index
   padding-length
   payload-data
   offset (m)                         —  Byte offset of next payload data set
   index
   sub-index
   padding-length
   payload-data
   ...                                —  (further write requests)
}

WriteMultipleByIndex-ResponsePDU ::= SEQUENCE {
   data-size                          —  only present if in SDOCommandHeader segmentation = "initiate"
   index                              —  only present if write acces failed
   sub-index
   sub-abort
   sub-abort-code
   index                              —  only present if write acces failed
   sub-index
   sub-abort
   sub-abort-code
   ...                                —  (further write responses)
}
```

4.4.5.13 ReadMultipleByIndex services

```
ReadMultipleByIndex-RequestPDU ::= SEQUENCE {
   data-size                          —  only present if in SDOCommandHeader segmentation = "initiate"
   index
   sub-index
   reserved8
   index
   sub-index
   reserved8
   ...                                —  (further read requests)
}

ReadMultipleByIndex-ResponsePDU ::= SEQUENCE {
   data-size                          —  only present if in SDOCommandHeader segmentation = "initiate"
   offset (k)                         —  offset of the next payload data set
   index
   sub-index
   sub-abort -padding-length
   payload-data / sub-abort-code
   offset (m)                         —  offset of the next payload data set
   index
   sub-index
   sub-abort-padding-length
   payload-data / sub-abort-code
   ...                                —  (further read requests)
}
```

4.4.5.14 AbortPDU

```
AbortPDU ::= {
   abort-code     Unsigned32          —  see Clause A.1 for valid values
}
```

4.4.5.15 Data size

data-size ::= Unsigned32 — Length of transferred data block. Counting from the beginning of the "SDO-command". Valid value range: $0...2^{32}-1$.

4.4.5.16 Index

index ::= Unsigned16 — Specifies an entry of the device object dictionary; 0.. 65.535

4.4.5.17 Sub-index

sub-index ::= Unsigned8 — Specifies a component of a device object dictionary entry; 0..254

4.4.5.18 Payload-data

payload-data ::= Any — application dependent type and length;
total frame length must comply with Ethernet rules

4.4.5.19 Offset

offset (i) ::= Unsigned8 — offset of specified data; i is 4-aligned

4.4.5.20 Padding-length

padding-length ::= Unsigned8 — Number of padding bytes in the last quadlet (4-byte word) of the payload data; coded in the two least significant bits

4.4.5.21 Sub-abort

sub-abort ::= Unsigned8 — 0: transfer ok; 1: abort; coded in the most significant bit

4.4.5.22 Sub-abort-padding-length

sub-abort-padding-length ::= Unsigned8 — sub-abort (see 4.4.5.21) and padding-length (see 4.4.5.20) merged in one octet.

4.4.5.23 Sub-abort-code

sub-abort-code ::= Unsigned32 — Values and meaning identically equal to abort-code, see Clause A.1.

4.4.6 Sync-response

4.4.6.1 Overview

Sync-response ::= SEQUENCE {

synchronization-status	Bitstring	— see 4.4.6.2
latency	Unsigned32	— PRes latency in ns
sync-node-number	Unsigned32	— node number received last inside SyncRequest/SyncResponse
sync-delay	Unsigned32	— time difference between the end of receiving SyncRequest and the beginning of receiving the SyncResponse in ns
PRes-time	Unsigned32	— time delay between reception of PRes from MN and time-triggered sending of the own PRes in ns

}

NOTE The above listed elements are sometimes summarized as follows:
"synchronization-status" through "destination-MAC-address" are summarized under the term "sync-status".

4.4.6.2 Synchronization-status

Synchronization-status ::= Bitstring {

PRes-time-valid	(0)	— The parameter PRes-time is valid.
reserved bit1 through bit30	(1)...(30)	
PRes-mode-status	(31)	— The time-triggered sending of PRes is active.

}

4.4.7 Manufacturer-specific

These parts are reserved for manufacturer specific purpose. Their specification is not in the scope of this international standard.

5 Transfer syntax

5.1 Encoding of data types

5.1.1 General description of data types and encoding rules

To be able to exchange meaningful data, the format of this data and its meaning have to be known by the producer and consumer(s). This specification models this by the concept of data types.

The encoding rules define the representation of values of data types and the transfer syntax for the representations. Values are represented as bit sequences. Bit sequences are transferred in sequences of octets (bytes). For numerical data types the encoding is little endian style as shown in Table 3.

5.1.2 Transfer syntax for bit sequences

For transmission a bit sequence is reordered into a sequence of octets. Hexadecimal notation is used for octets as specified in ISO/IEC 9899. Let $b = b_0 \ldots b_{n-1}$ be a bit sequence. Denote k a non-negative integer such that $8(k - 1) < n \leq 8k$. Then b is transferred in k octets assembled as shown in Table 3. The bits b_i, $i \geq n$ of the highest numbered octet are do not care bits.

Table 3 – Transfer syntax for bit sequences

octet number	1.	2.	k.
	$b_7 .. b_0$	$b_{15} .. b_8$	b8k –1 .. b8k -8

Octet 1 is transmitted first and octet *k* is transmitted last. The bit sequence is transferred as follows across the network (transmission order within an octet is determined by ISO/IEC 8802-3):

$b_7, b_6, \ldots, b_0, b_{15}, \ldots, b_8, \ldots$

EXAMPLE

Bit 9	…	Bit 0
10b	0001b	1100b
2h	1h	Ch
		= 21Ch

The bit sequence $b = b_0 .. b_9 = 0011\ 1000\ 01_b$ represents an Unsigned10 with the value 21Ch and is transferred in two octets: First 1Ch and then 02h.

5.1.3 Encoding of a Boolean value

Data of basic data type BOOLEAN attains the values TRUE or FALSE.

The values are represented as bit sequences of length 1. The value TRUE is represented by the bit sequence 1, and FALSE by 0.

A BOOLEAN shall be transferred over the network as UNSIGNED8 of value 1 (TRUE) resp. 0 (FALSE). Sequent BOOLEANs may be packed to one UNSIGNED8. Sequences of BOOLEAN and BIT type items may be also packed to one UNSIGNED8.

5.1.4 Encoding of an Unsigned Integer value

Data of basic data type UNSIGNEDn has values in the non-negative integers. The value range is 0, …, 2^n-1. The data is represented as bit sequences of length n.

The bit sequence

$$b = b_0 \ldots b_{n-1}$$

is assigned the value

$$\text{UNSIGNEDn}(b) = b_{n-1}\times 2^{n-1} + \ldots + b_1\times 2^1 + b_0\times 2^0$$

Note that the bit sequence starts on the left with the least significant byte.

Example: The value $266_d = 10A_h$ with data type UNSIGNED16 is transferred in two octets across the bus, first $0A_h$ and then 01_h.

The following UNSIGNEDn data types are transferred as shown in Table 4.

Table 4 – Transfer syntax for data type UNSIGNEDn

octet number	0	1	2	3	4	5	6	7
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
UNSIGNED48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
UNSIGNED56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

The data types UNSIGNED24, UNSIGNED40, UNSIGNED48 and UNSIGNED56 should not be applied by new applications.

UNSIGNEDn data types of length deviating from the values listed above may be applied by compound data types only.

5.1.5 Encoding of a Signed Integer

Data of basic data type INTEGERn has values in the integers. The value range is from -2^{n-1} to $2^{n-1}-1$. The data is represented as bit sequences of length n. The bit sequence

$$b = b_0 \,..\, b_{n-1}$$

is assigned the value

$$\text{INTEGERn}(b) = b_{n-2}\times 2^{n-2} + \ldots + b_1\times 2^1 + b_0\times 2^0 \text{ if } b_{n-1} = 0$$

and, performing two's complement arithmetic,

$$\text{INTEGERn}(b) = -\ \text{INTEGERn}(\hat{}b) - 1 \quad \text{if } b_{n-1} = 1$$

Note that the bit sequence starts on the left with the least significant bit.

Example: The value $-266_d = 0xFEF6_h$ with data type Integer16 is transferred in two octets, first 0xF6 and then 0xFE.

The INTEGERn data types are transferred as specified in Table 5.

Table 5 – Transfer syntax for data type INTEGERn

octet number	0	1	2	3	4	5	6	7
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

The data types INTEGER24, INTEGER40, INTEGER48 and INTEGER56 should not be applied by new applications.

INTEGERn data types of length deviating from the values listed above may be applied by compound data types only.

5.1.6 Encoding of a Floating point value

Data of basic data types REAL32 and REAL64 have values in the real numbers.

The data type REAL32 is represented as bit sequence of length 32. The encoding of values follows the IEEE 754 Standard for single precision floating-point.

The data type REAL64 is represented as bit sequence of length 64. The encoding of values follows the IEEE 754 Standard for double precision floating-point numbers.

A bit sequence of length 32 either has a value (finite non-zero real number, ± 0, ± _) or is NaN (not-a-number).

The bit sequence

$$b = b_0 \ldots b_{31}$$

is assigned the value (finite non-zero number)

$$REAL32(b) = (-1)^S \times 2^{E-127} \times (1+ F)$$

Here

$S = b_{31}$ is the sign.

$E = b_{30} \times 2^7 + \ldots + b_{23} \times 2^0$, $0 < E < 255$, is the un-biased exponent.

$F = 2^{-23} \times (b_{22} \times 2^{22} + \ldots + b_1 \times 2^1 + b_0 \times 2^0)$ is the fractional part of the number.

E = 0 is used to represent ± 0. E = 255 is used to represent infinities and NaN's.

Note that the bit sequence starts on the left with the least significant bit.

5.1.7 Encoding of an Octet String value

The data type OCTET_STRINGlength is defined as follows; "length" is the length of the octet string.

ARRAY [length] OF UNSIGNED8 OCTET_STRINGlength

5.1.8 Encoding of a Visible String value

VISIBLE_CHAR are 0_h and the range from 20_h to $7E_h$. The data are interpreted as ISO 646-1973(E) 7- bit coded characters. "length" is the length of the visible string.

UNSIGNED8 VISIBLE_CHAR

ARRAY [length] OF VISIBLE_CHAR VISIBLE_STRINGlength

There is no 0_h necessary to terminate the string.

5.1.9 Encoding of a Unicode String Value

The data type UNICODE_STRINGlength is defined below; "length" is the length of the unicode string.

ARRAY [length] OF UNSIGNED16 UNICODE_STRINGlength

5.1.10 Encoding of a Time of Day value

The data type TimeOfDay represents absolute time. It follows from the definition and the encoding rules that TimeOfDay is represented as bit sequence of length 48.

Component "ms" is the time in milliseconds after midnight. Component "days" is the number of days since January 1, 1984.

```
STRUCT OF
   UNSIGNED28     ms,
   VOID4          reserved,
   UNSIGNED16     days
TIME_OF_DAY
```

The encoding is as shown in Figure 1.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	number of milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	number of days since 1984-01-01
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 1 – Encoding of Time of Day value

5.1.11 Encoding of a Time Difference value

The data type TimeDifference represents a time difference. It follows from the definition and the encoding rules that TimeDifference is represented as bit sequence of length 48.

Time differences are sums of numbers of days and milliseconds. Component "ms" is the number milliseconds. Component "days" is the number of days.

```
STRUCT OF
   UNSIGNED28      ms,
   VOID4           reserved,
   UNSIGNED16      days
TIME_DIFFERENCE
```

The encoding is as shown in Figure 2.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	milliseconds
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	days
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 2 – Encoding of Time Difference value

6 FAL protocol state machines

Interface to FAL services and protocol machines are specified in Clause 6.

NOTE The state machines specified in Clause 6 and ARPMs defined in the following clauses only define the valid events for each. It is a local matter to handle invalid events.

The behavior of the FAL is described by the protocol machines shown in Figure 3. Specific sets of these protocol machines are defined for different AREP types. Figure 3 also shows the primitives exchanged between the protocol machines.

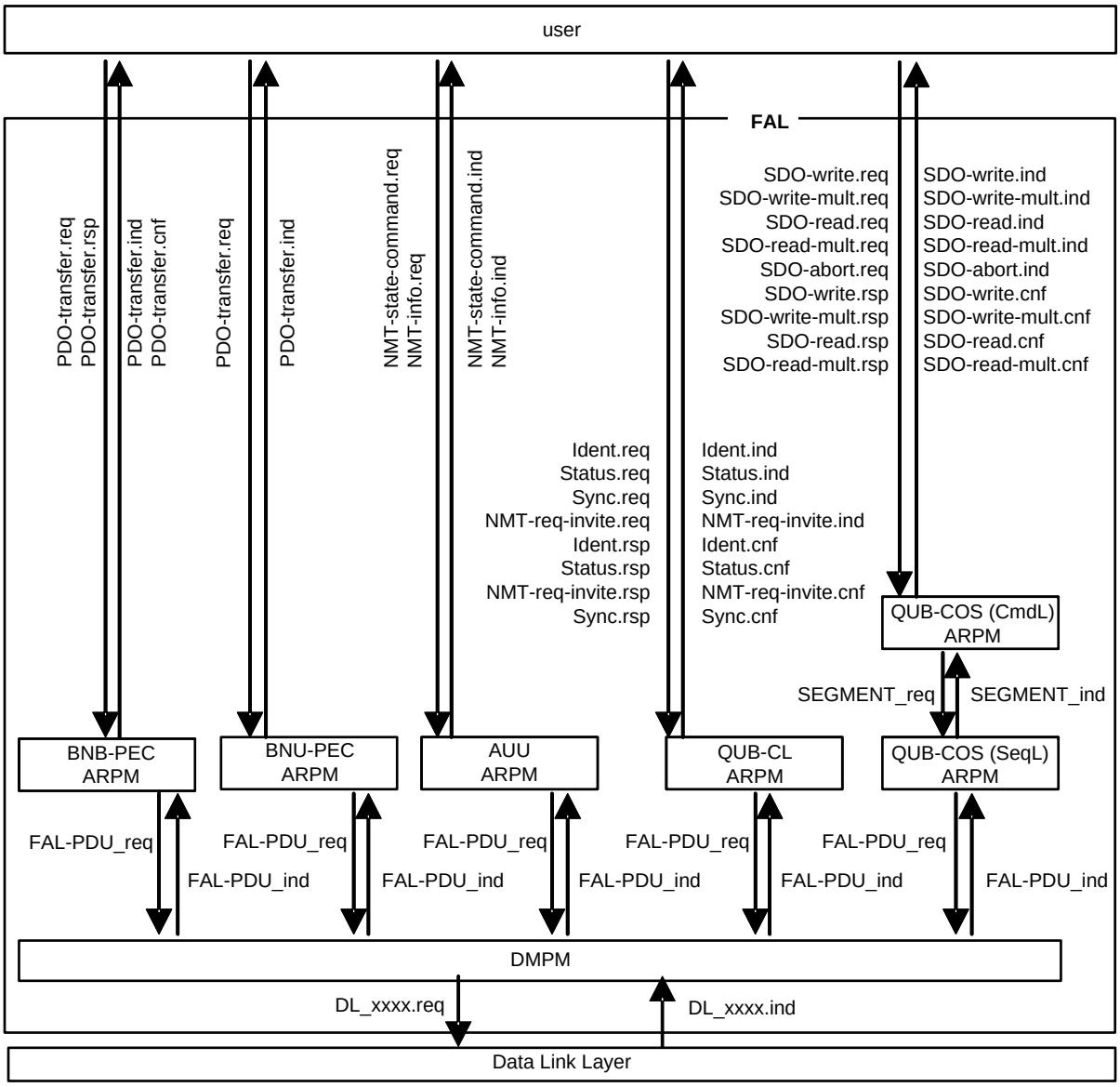


Figure 3 – Primitives exchanged between protocol machines

7 AP context state machine

There is no AP-context state machine defined for this protocol.

8 FAL service protocol machine

There is no FAL service protocol state machine defined for this protocol.

9 AR protocol machine

9.1 Buffered-network-scheduled bi-directional pre-established connection (BNB-PEC) ARPM

9.1.1 BNB-PEC primitive definitions

9.1.1.1 Primitives exchanged between BNB-PEC ARPM and user

Table 6 and Table 7 list the primitives exchanged between the ARPM and the user.

Table 6 – Primitives issued by user to BNB-PEC ARPM

Primitive name	Source	Associated parameters	Functions
PDO-transfer.req	user	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13
PDO-transfer.rsp	user	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13

Table 7 – Primitives issued by BNB-PEC ARPM to user

Primitive name	Source	Associated parameters	Functions
PDO-transfer.ind	ARPM	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13
PDO-transfer.cnf	ARPM	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13

9.1.1.2 Parameters of primitives

The parameters of the primitives are described in IEC 61158-5-13.

9.1.2 DLL mapping of BNB-PEC class

9.1.2.1 Formal model

Subclause 9.1.2 describes the mapping of the BNB-PEC AREP class to the Type 13 data link layer defined in IEC 61158-3-13 and IEC 61158-4-13. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data link layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is not in the scope of this International Standard.

The DLL mapping attributes and their permitted values and the DLL services used with the BNB-PEC AREP class are defined in 9.1.2.

CLASS: **Type 13 BNB-PEC**
PARENT CLASS: **Buffered network-scheduled bi-directional pre-established connection AREP**
ATTRIBUTES:
1 (m) KeyAttribute: LocalDlcepAddress
2 (m) Attribute: RemoteDlcepAddress
DLL SERVICES:
1 (m) OpsService: DL-PDO

9.1.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

9.1.2.3 DLL services

Refer to IEC 61158-3-13 for DLL service descriptions.

9.1.3 BNB-PEC ARPM state machine

9.1.3.1 BNB-PEC ARPM states

The BNB-PEC ARPM state machine has only one state called "ACTIVE", see Figure 4.

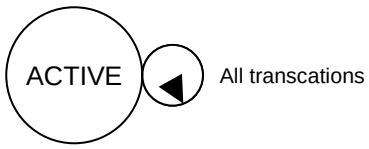


Figure 4 – State transition diagram of BNB-PEC ARPM

9.1.3.2 BNB-PEC ARPM state table

Table 8 and Table 9 define the state machine of the BNB-PEC ARPM.

Table 8 – BNB-PEC ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	ACTIVE	PDO-transfer.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc1"
		data := PDO-transfer.req)
		}
			ACTIVE

S2	ACTIVE	PDO-transfer.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc2"
		data := PDO-transfer.rsp)
		}
			ACTIVE

NOTE Transaction S1 is executed by the MN only, transaction S2 is executed by the addressed CN only.			

Table 9 – BNB-PEC ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	FAL-PDU_ind && message-type = "Isoc1" ⇒ PDO-transfer.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && message-type = "Isoc2" ⇒ PDO-transfer.cnf	ACTIVE
NOTE Transaction R1 is executed by the CNs only, transaction R2 is executed by the MN and may be executed by CNs depending on their configuration.			

9.1.3.3 Functions used by BNB-PEC ARPM

The receipt of a FAL-PDU_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 10 defines the other function used by this state machine.

Table 10 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
message-type data additional information		dlsdu	
Function			
Builds a FAL-PDU out of the parameters given as input variables.			

9.2 Buffered-network-scheduled uni-directional pre-established connection (BNU-PEC) ARPM

9.2.1 BNU-PEC primitive definitions

9.2.1.1 Primitives exchanged between BNU-PEC ARPM and user

Table 11 and Table 12 list the primitives exchanged between the ARPM and the user.

Table 11 – Primitives issued by user to BNU-PEC ARPM

Primitive name	Source	Associated parameters	Functions
PDO-transfer.req	user	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13

Table 12 – Primitives issued by BNU-PEC ARPM to user

Primitive name	Source	Associated parameters	Functions
PDO-transfer.ind	ARPM	AREP PDO PDO-version	Refer to service data definitions in IEC 61158-5-13

9.2.1.2 Parameters of primitives

The parameters of the primitives are described in IEC 61158-5-13.

9.2.2 DLL mapping of BNU-PEC class

9.2.2.1 Formal model

Subclause 9.2.2 describes the mapping of the BNU AREP class to the Type 13 data link layer defined in IEC 61158-3-13 and IEC 61158-4-13. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data link layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is not in the scope of this International Standard.

The DLL mapping attributes and their permitted values and the DLL services used with the BNU AREP class are defined in 9.2.2.

CLASS:			**Type 13 BNU-PEC**
PARENT CLASS:			**Buffered network-scheduled uni-directional pre-established connection AREP**
ATTRIBUTES:			
1	(m)	KeyAttribute:	PublisherDlcepAddress
DLL SERVICES:			
1	(m)	OpsService:	DL-PDO

9.2.2.2 Attributes

PublisherDlcepAddress
This attribute specifies the publisher's DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

9.2.2.3 DLL services

Refer to IEC 61158-3-13 for DLL service descriptions.

9.2.3 BNU-PEC ARPM state machine

9.2.3.1 BNU-PEC ARPM states

The BNU-PEC ARPM state machine has only one state called "ACTIVE", see Figure 5.

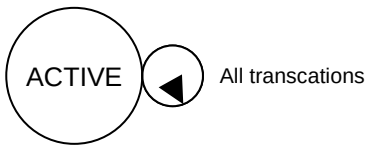


Figure 5 – State transition diagram of BNU-PEC ARPM

9.2.3.2 BNU-PEC ARPM state table

Table 13 and Table 14 define the state machine of the BNU-PEC ARPM.

Table 13 – BNU-PEC ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	ACTIVE	PDO-transfer.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc2"
		data := PDO-transfer.req)
		}
			ACTIVE

NOTE This transaction is executed by the MN only.			

Table 14 – BNU-PEC ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	FAL-PDU_ind && message-type = "Isoc2" ⇒ PDO-transfer.ind	ACTIVE
NOTE This transaction is executed by the CNs only.			

9.2.3.3 Functions used by BNU ARPM

The receipt of a FAL-PDU_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 15 defines the other function used by this state machine.

Table 15 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
message-type data additional information		dlsdu	
Function			
Builds a FAL-PDU out of the parameters given as input variables.			

9.3 Queued user-triggered uni-directional (QUU) ARPM

9.3.1 QUU primitive definitions

9.3.1.1 Primitives exchanged between QUU ARPM and user

Table 16 and Table 17 list the primitives exchanged between the ARPM and the user.

Table 16 – Primitives issued by user to QUU ARPM

Primitive name	Source	Associated parameters	Functions
NMT-state-command.req	user	AREP command-ID node-list	Refer to service data definitions in IEC 61158-5-13.
NMT-info.req	user	AREP publish-node-list publish-time	Refer to service data definitions in IEC 61158-5-13.

Table 17 – Primitives issued by QUU ARPM to user

Primitive name	Source	Associated parameters	Functions
NMT-state-command.ind	ARPM	AREP command-ID node-list	Refer to service data definitions in IEC 61158-5-13.
NMT-info.ind	ARPM	AREP publish-node-list publish-time	Refer to service data definitions in IEC 61158-5-13.

9.3.1.2 Parameters of primitives

The parameters of the primitives are described in IEC 61158-5-13.

9.3.2 DLL mapping of QUU AREP class

9.3.2.1 Formal model

Subclause 9.3.2 describes the mapping of the QUU AREP class to the Type 13 data link layer defined in IEC 61158-3-13 and IEC 61158-4-13. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data link layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is not in the scope of this International Standard.

The DLL Mapping attributes and their permitted values and the DLL services used with the QUU AREP class are defined in 9.3.2.

CLASS: **Type 13 QUU**
PARENT CLASS: **Queued User-triggered uni-directional AREP**
ATTRIBUTES:
1 (m) KeyAttribute: LocalDlcepAddress
2 (m) Attribute: RemoteDlcepAddress
DLL SERVICES:
1 (m) OpsService: DL-CMD

9.3.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

RemoteDlcepAddress

This attribute specifies the remote DLCEP address and identifies the DLCEP

9.3.2.3 DLL services

Refer to IEC 61158-3-13 for DLL service descriptions.

9.3.3 QUU ARPM state machine

9.3.3.1 QUU ARPM states

The QUU ARPM state machine has only one state called "ACTIVE", see Figure 6.

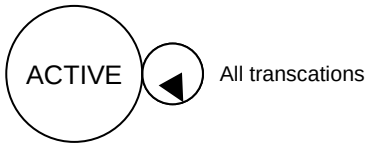


Figure 6 – State transition diagram of QUU ARPM

9.3.3.2 QUU ARPM state table

Table 18 and Table 19 define the state machine of the QUU ARPM.

Table 18 – QUU ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	ACTIVE	NMT-state-command.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 4h — "NMT-command"
		data := NMT-state-command.req)
		}
			ACTIVE

S2	ACTIVE	NMT-info.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 4h — "NMT-command"
		data := NMT-info.req)
		}
			ACTIVE

NOTE These transactions are executed by the MN only.			

Table 19 – QUU ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	FAL-PDU_ind && service-id = 4h — "NMT-command" && 20h <= command-ID <= 5Fh — (see Clause A.2) ⇒ NMT-state-command.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && service-id = 4h — "NMT-command" && 80h <= command-ID <= BFh — (see A.2) ⇒ NMT-info.ind	ACTIVE
NOTE These transactions are executed by the CNs only.			

9.3.3.3 Functions used by QUU ARPM

The receipt of a FAL-PDU_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 20 defines the other functions used by this state machine.

Table 20 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
message-type service-id data additional information		dlsdu	
Function			
Builds a FAL-PDU out of the parameters given as input variables.			

9.4 Queued user-triggered bi-directional connectionless (QUB-CL) ARPM

9.4.1 QUB-CL Primitive definitions

9.4.1.1 Primitives exchanged between QUB-CL ARPM and user

Table 21 and Table 22 list the primitives exchanged between the ARPM and the user.

Table 21 – Primitives issued by user to QUB-CL ARPM

Primitive name	Source	Associated parameters	Functions
Ident.req	user	AREP	Refer to service data definitions in IEC 61158-5-13
Status.req	user	AREP	Refer to service data definitions in IEC 61158-5-13
NMT-req-invite.req	user	AREP	Refer to service data definitions in IEC 61158-5-13
Ident.rsp	user	AREP NMT-status fieldbus-version feature-flags cycle-timing Identity verify-configuration application-software-version IP-address host-name vendor-specific-extensions	Refer to service data definitions in IEC 61158-5-13
Status.rsp	user	AREP NMT-status static-error error-history	Refer to service data definitions in IEC 61158-5-13
NMT-req-invite.rsp	user	AREP command-ID target-node data	Refer to service data definitions in IEC 61158-5-13
Sync.req	user	AREP sync-control	Refer to service data definitions in IEC 61158-5-13
Sync.rsp	user	AREP sync-status	Refer to service data definitions in IEC 61158-5-13

Table 22 – Primitives issued by QUB-CL ARPM to user

Primitive name	Source	Associated parameters	Functions
Ident.ind	ARPM	AREP	Refer to service data definitions in IEC 61158-5-13
Status.ind	ARPM	AREP	Refer to service data definitions in IEC 61158-5-13
NMT-req-invite.ind	ARPM	AREP	Refer to service data definitions in IEC 61158-5-13
Sync.ind	ARPM	AREP sync-control	Refer to service data definitions in IEC 61158-5-13
Ident.cnf	ARPM	AREP NMT-status fieldbus-version feature-flags cycle-timing Identity verify-configuration application-software-version IP-address host-name vendor-specific-extensions	Refer to service data definitions in IEC 61158-5-13
Status.cnf	ARPM	AREP NMT-status static-error error-history	Refer to service data definitions in IEC 61158-5-13
NMT-req-invite.cnf	ARPM	AREP command-ID target-node data	Refer to service data definitions in IEC 61158-5-13
Sync.cnf	ARPM	AREP sync-status	Refer to service data definitions in IEC 61158-5-13

9.4.1.2 Parameters of primitives

The parameters of the primitives are described in IEC 61158-5-13.

9.4.2 DLL mapping of QUB-CL AREP class

9.4.2.1 Formal model

Subclause 9.4.2 describes the mapping of the QUB-CL AREP class to the Type 13 data link layer defined in IEC 61158-3-13 and IEC 61158-4-13. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data link layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is not in the scope of this standard.

The DLL Mapping attributes and their permitted values and the DLL services used with the QUB-CL AREP class are defined in 9.4.2.

CLASS:			**Type 13 QUB-CL**
PARENT CLASS:			**Queued User-triggered Bi-directional connectionless AREP**
ATTRIBUTES:			
1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
DLL SERVICES:			
1	(m)	OpsService:	DL-IDE
2	(m)	OpsService:	DL-STA
2	(m)	OpsService:	DL-REQ

9.4.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

9.4.2.3 DLL services

Refer to IEC 61158-3-13 for DLL service descriptions.

9.4.3 QUB-CL ARPM state machine

9.4.3.1 QUB-CL ARPM states

The QUB-CL ARPM state machine has only one state called "ACTIVE", see Figure 7.

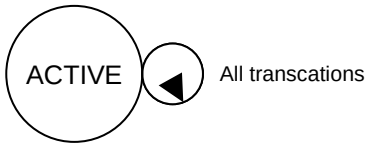


Figure 7 – State transition diagram of QUB-CL ARPM

9.4.3.2 QUB-CL ARPM state table

Table 23 and Table 24 define the state machine of the QUB-CL ARPM.

Table 23 – QUB-CL ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	ACTIVE	Ident.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 1h — "ident-request"
		data := Ident.req)
		}
			ACTIVE

S2	ACTIVE	Status.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 2h — "status-request"
		data := Status.req)
		}
			ACTIVE

S3	ACTIVE	NMT-req-invite.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 3h — "NMT-req-invite"
		data := NMT-req-invite.req)
		}
			ACTIVE

S4	ACTIVE	Ident.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 1h — "ident-response"
		data := Ident.rsp)
		}
			ACTIVE

S5	ACTIVE	Status.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 2h — "status-response"
		data := Status.rsp)
		}
			ACTIVE

S6	ACTIVE	NMT-req-invite.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 3h — "NMT-request"
		data := NMT-req-invite.rsp)
		}
			ACTIVE

S7	ACTIVE	Sync.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 6h — "sync-request"
		data := Sync.req)
		}
			ACTIVE

S8	ACTIVE	Sync.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 6h — "sync-response"
		data := Sync.rsp)
		}
			ACTIVE

NOTE Transactions S1 through S3 and S7 are executed by the MN only, transactions S4 through S6 and S8 are executed by CNs only.			

Table 24 – QUB-CL ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	FAL-PDU_ind && requested-service-id := 1h — "ident-request" ⇒ Ident.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && requested-service-id := 2h — "status-request" ⇒ Status.ind	ACTIVE
R3	ACTIVE	FAL-PDU_ind && requested-service-id := 3h — "NMT-req-invite" ⇒ NMT-req-invite.ind	ACTIVE
R4	ACTIVE	FAL-PDU_ind && service-id := 1h — "ident-response" ⇒ Ident.cnf	ACTIVE
R5	ACTIVE	FAL-PDU_ind && service-id := 2h — "status-response" ⇒ Status.cnf	ACTIVE
R6	ACTIVE	FAL-PDU_ind && service-id := 3h — "NMT-request" ⇒ NMT-req-invite.cnf	ACTIVE
R7	ACTIVE	FAL-PDU_ind && requested-service-id := 6h — "sync-request" ⇒ Sync.ind	ACTIVE
R8	ACTIVE	FAL-PDU_ind && service-id := 6h — "sync-response" ⇒ Sync.cnf	ACTIVE
NOTE Transactions R1 through R3 and R7 are executed by CNs only, transactions R4 through R6 are executed by the MN and may be executed by CNs depending on their configuration, transaction R8 is executed by the MN and CNs.			

9.4.3.3 Functions used by QUB-CL ARPM

The receipt of a FAL-PDU_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 25 defines the other function used by this state machine.

Table 25 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
message-type service-id data additional information		dlsdu	
Function			
Builds a FAL-PDU out of the parameters given as input variables.			

9.5 Queued user-triggered bi-directional connection-oriented with segmentation (QUB-COS) ARPM

9.5.1 Overview

The QUB-COS ARPM is divided in two sub-sections, so called layers:

- Sequence layer, QUB-COS (SeqL)
- Command layer, QUB-COS (CmdL)

The sequence layer provides the service of a reliable bidirectional connection that guarantees that no messages are lost or duplicated and that all messages arrive in the correct order. There shall be a sequence number for each sent frame, and an acknowledgement for the sequence number of the opposite node, as well a connection state and a connection acknowledge.

The command layer has to decide whether a large block of data can be transferred in one frame (expedited transfer) or if it must be segmented in several frames (segmented transfer).

9.5.2 QUB-COS (CmdL) primitive definitions

9.5.2.1 Primitives exchanged between QUB-COS (CmdL) ARPM and user

Table 26 and Table 27 list the primitives exchanged between the ARPM and the user.

Table 26 – Primitives issued by user to QUB-COS (CmdL) ARPM

Primitive name	Source	Associated parameters	Functions
SDO-write.req	user	AREP invoke-ID command-ID segment-size data-size OD-identifier payload-data	Refer to service data definitions in IEC 61158-5-13
SDO-write-mult.req	user	AREP invoke-ID command-ID segment-size OD-identifier (n) payload-data (n)	Refer to service data definitions in IEC 61158-5-13
SDO-read.req	user	AREP invoke-ID command-ID OD-identifier	Refer to service data definitions in IEC 61158-5-13
SDO-read-mult.req	user	AREP invoke-ID command-ID OD-identifier (n)	Refer to service data definitions in IEC 61158-5-13
SDO-abort.req	user	AREp invoke-ID error-info	Refer to service data definitions in IEC 61158-5-13
SDO-write.rsp	user	AREP invoke-ID error-info	Refer to service data definitions in IEC 61158-5-13
SDO-write-mult.rsp	user	AREP invoke-ID error-info (n)	Refer to service data definitions in IEC 61158-5-13

Primitive name	Source	Associated parameters	Functions
SDO-read.rsp	user	AREP invoke-ID segment-size data-size payload-data error-info	Refer to service data definitions in IEC 61158-5-13
SDO-read-mult.rsp	user	AREP invoke-ID segment-size data-size payload-data / error-info (n)	Refer to service data definitions in IEC 61158-5-13

Table 27 – Primitives issued by QUB-COS (CmdL) ARPM to user

Primitive name	Source	Associated parameters	Functions
SDO-write.ind	ARPM	AREP invoke-ID command-ID segment-size data-size OD-identifier payload-data	Refer to service data definitions in IEC 61158-5-13
SDO-write-mult.ind	ARPM	AREP invoke-ID command-ID segment-size data-size OD-identifier (n) payload-data (n)	Refer to service data definitions in IEC 61158-5-13
SDO-read.ind	ARPM	AREP invoke-ID command-ID OD-identifier	Refer to service data definitions in IEC 61158-5-13
SDO-read-mult.ind	ARPM	AREP invoke-ID command-ID OD-identifier (n)	Refer to service data definitions in IEC 61158-5-13
SDO-abort.ind	user	AREP invoke-ID error-info	Refer to service data definitions in IEC 61158-5-13
SDO-write.cnf	ARPM	AREP invoke-ID error-info	Refer to service data definitions in IEC 61158-5-13

Primitive name	Source	Associated parameters	Functions
SDO-write-mult.cnf	ARPM	AREP invoke-ID error-info (n)	Refer to service data definitions in IEC 61158-5-13
SDO-read.cnf	ARPM	AREP invoke-ID segment-size data-size payload-data error-info	Refer to service data definitions in IEC 61158-5-13
SDO-read-mult.cnf	ARPM	AREP invoke-ID segment-size data-size payload-data / error-info (n)	Refer to service data definitions in IEC 61158-5-13

9.5.2.2 Parameters of primitives

The parameters of the primitives are described in IEC 61158-5-13.

9.5.3 QUB-COS (CmdL) ARPM state machine

9.5.3.1 QUB-COS (CmdL) ARPM states

The QUB-COS (CmdL) ARPM state machine has only one state called "ACTIVE", see Figure 8.

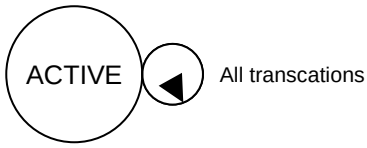


Figure 8 – State transition diagram of QUB-COS (CmdL) ARPM

9.5.3.2 QUB-COS (CmdL) ARPM state table

Table 28 and Table 29 define the state machine of the QUB-COS (CmdL) ARPM.

Table 28 – QUB-COS (CmdL) ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state
S1	ACTIVE	SDO-write.req || SDO-write-mult.req && segment-size <= max-segment-size ⇒ response := 0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data)	ACTIVE

S2	ACTIVE	SDO-write.req || SDO-write-mult.req
		&& segment-size > max-segment-size
		⇒
		response := 0
		abort := 0
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(see Notes)
			ACTIVE

S3	ACTIVE	SDO-write.rsp ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE

#	Current state	Event or condition ⇒ action	Next state
S4	ACTIVE	SDO-write-mult.rsp && all data successfully written ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE
S5	ACTIVE	SDO-write-mult.rsp && at least one data transfer failed && segment-size <= max-segment-size ⇒ response := 1 abort := 1 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := “null”)	ACTIVE

S6	ACTIVE	SDO-write-mult.rsp
		&& at least one data transfer failed
		&& segment-size > max-segment-size
		⇒
		response := 1
		abort := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(see Notes)
			ACTIVE

#	Current state	Event or condition ⇒ action	Next state
S7	ACTIVE	SDO-read.req ⇒ response :=0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE
S8	ACTIVE	SDO-read.rsp && segment-size <= max-segment-size ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT _req := BuildSegment (header segment-data)	ACTIVE

S9	ACTIVE	SDO-read.rsp
		&& segment-size > max-segment-size
		⇒
		response := 1
		abort := 0
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(see Notes)
			ACTIVE

#	Current state	Event or condition ⇒ action	Next state
S10	ACTIVE	SDO-read-mult.req && segment-size <= max-segment-size ⇒ response := 0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data)	ACTIVE

S11	ACTIVE	SDO-read-mult.req
		&& segment-size > max-segment-size
		⇒
		response := 0
		abort := 0
		segmentation := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(see Notes)
			ACTIVE

S12	ACTIVE	SDO-read-mult.rsp
		&& segment-size <= max-segment-size
		⇒
		response := 1
		if all data were successfully read
		abort := 0
		else
		abort := 1
		endif
		segmentation := 0
		data-size := "null"
		SEGMENT_req := BuildSegment (
		header
		segment-data)
			ACTIVE

#	Current state	Event or condition ⇒ action	Next state

S13	ACTIVE	SDO-read-mult.rsp
		&& segment-size > max-segment-size
		⇒
		response := 1
		if all data were successfully read
		abort := 0
		else
		abort := 1
		endif
		segmentation := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(see Notes)
			ACTIVE

S14	ACTIVE	SDO-abort.req ⇒ response := 0 abort := 1 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "error-info")	ACTIVE

NOTE 1 When the length of the data exceeds the value of the "max-segment-size" parameter the QUB_COS (CmdL) protocol splits the payload data into N segment-data.

NOTE 2 For each segment-data, the function “BuildSegment” builds a Segment := Header with Command Layer parameters followed with segment-data without any gap.

NOTE 3 The segments reach the receiver AREP in the same order as they were created. This is guaranteed by the Sequence layer. Thus an additional numbering of the segments is not necessary.

Table 29 – QUB-COS (CmdL) ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && (command-ID = 1h — "write-by-index" || command-ID = 3h — "write-all-by-index" ⇒ SDO-write.ind	ACTIVE

R2	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& (command-ID = 1h — "write-by-index"
		|| command-ID = 3h — "write-all-by-index")
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write.ind
		endif
		
		(see Note)
			ACTIVE

R3	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 && (command-ID = 1h — "write-by-index" || command-ID = 3h — "write-all-by-index" ⇒ SDO-write.cnf	ACTIVE
R4	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && command-ID = 31h — "write-multiple-by-index" ⇒ SDO-write-mult.ind	ACTIVE

R5	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& command-ID = 31h — "write-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write-mult.ind
		endif
		
		(see Note)
			ACTIVE

R6	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 || abort = 1 && segmentation = 0 && command-ID = 31h — "write-multiple-by-index" ⇒ SDO-write-mult.cnf	ACTIVE

R7	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 1
		&& segmentation <> 0
		&& command-ID = 31h — "write-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write-mult.cnf
		endif
		
		(see Note)
			ACTIVE

#	Current state	Event or condition ⇒ action	Next state
R8	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && (command-ID = 2h — "read-by-index" || command-ID = 4h — "read-all-by-index" ⇒ SDO-read.ind	ACTIVE
R9	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 && segmentation = 0 && (command-ID = 2h — "read-by-index" || command-ID = 4h — "read-all-by-index" ⇒ SDO-read.cnf	ACTIVE

R10	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 0
		&& segmentation <> 0
		&& (command-ID = 2h — "read-by-index"
		|| command-ID = 4h — "read-all-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read.cnf
		endif
		
		(see Note)
			ACTIVE

R11	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && command-ID = 32h — "read-multiple-by-index" ⇒ SDO-read-mult.ind	ACTIVE

R12	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& command-ID = 32h — "read-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read-mult.ind
		endif
		
		(see Note)
			ACTIVE

R13	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 || abort = 1 && segmentation = 0 && command-ID = 32h — "read-multiple-by-index" ⇒ SDO-read-mult.cnf	ACTIVE

R14	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 0 || abort = 1
		&& segmentation <> 0
		&& command-ID = 32h — "read-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read-mult.cnf
		endif
		
		(see Note)
			ACTIVE

R15	ACTIVE	SEGMENT_ind && abort = 1 ⇒ SDO-abort.ind	ACTIVE

NOTE When the length of the data exceeds the value of the "max-segment-size" parameter the payload data are split into N segment-data. The segments are delivered in the order as they were created. Each segment contains a header with additional information. On the receiver end of the AR the function “AddSegment” removes this header (including "data-size" in the first segment) and appends the segment-data to the previous received segment-data. Once the N Segment-data are appended together without any gap, the function “GetintermediatePDU” gives the original OD entry identifier with their related payload data.

9.5.3.3 Functions used by QUB-COS (CmdL) ARPM

The receipt of a SEGMENT_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 30 through Table 34 define the other functions used by this state machine.

Table 30 – Function BuildSegment

Name	BuildSegment	Used in	ARPM
Input		Output	
header := AREP invoke-ID response abort segmentation command-ID segment-size segment-data := data-size (only for the first segment) data containing OD identifier(s) and related payload data, or error info (if applicable)		SEGMENT_req	
Function			
Builds a SEGMENT out of the parameters given as input variables. There is no additional segment number necessary as the correct order of segments is already guaranteed by the Sequence Layer already. This function adds the specified header to each segment. These headers are identical for all coherent segments with the exception that the parameter "segmentation" will have the value 1 for the first segment, 3 for the last and 2 for all segments in between. The parameter "data-size" will be provided only with the first segment.			

Table 31 – Function RoundUp

Name	BuildSegment	Used in	ARPM
Input		Output	
data-size max-segment-size		integer	
Function			
divides "data-size" by "max-segment-size" and rounds up the result to the next higher integer			

Table 32 – Function MoreFollows

Name	AddSegment	Used in	ARPM
Input		Output	
segmentation		Boolean	
Function			
This function inspects the "segmentation" parameter of the SEGMENT_ind header. If its value is 3, the function sets its output to "False"			

NOTE The following two functions make use of a persistent variable "IntermediatePDU", in which all segments of the current user data are stored by the receiver of this data.

Table 33 – Function AddSegment

Name	AddSegment	Used in	ARPM
Input		Output	
header := invoke-ID response abort segmentation command-ID segment-size data-size (first segment only) segment-data		Error code	
Function			
This function removes the header of the received SEGMENT_ind and appends the Segment_data to the previous received Segment_data. Once the N Segment_data are appended together without any gap, the function "GetintermediatePDU" gives the original OD entry identifier(s) with their related payload data.			

Table 34 – Function GetIntermediatePDU

Name	GetIntermediatePDU	Used in	ARPM
Input		Output	
(none)		OD entry identifier(s) with their related payload data	
Function			
This function returns the original data, which was received in multiple segments. After this function call the variable IntermediatePDU is reset and does not contain any segments.			

9.5.4 QUB-COS (SeqL) primitive definitions

9.5.4.1 Primitives exchanged between QUB-COS (SeqL) and QUB-COS (CmdL)

Table 35 shows the primitives issued by QUB-COS (CmdL) to QUB-COS (SeqL).

Table 35 – Primitives issued by QUB-COS (CmdL) to QUB-COS (SeqL)

Primitive names	Source	Associated parameters	Functions
SEGMENT_req	QUB-COS (CmdL)	AREP invoke-ID response abort segmentation command-ID segment-size data-size OD-identifier(s) data	This primitive is used to request the QUB-COS (SeqL) to transfer a data segment, It also carries information about the segmentation which will be needed at the destination AREP to reconstruct the complete message

Table 36 shows the primitives issued by QUB-COS (SeqL) to QUB-COS (CmdL).

Table 36 – Primitives issued by QUB-COS (SeqL) to QUB-COS (CmdL)

Primitive names	Source	Associated parameters	Functions
SEGMENT_ind	QUB-COS (SeqL)	AREP invoke-ID response abort segmentation command-ID segment-size data-size OD-identifier(s) data	This primitive is used to transmit a data segment from QUB-COS (SeqL) to QUB-COS (CmdL). In case of segmentation it contains the data segment itself and additional information about the segmentation from the remote AREP to reconstruct the complete message in the local QUB-COS (CmdL)

9.5.4.2 Parameters of primitives

The parameters used with the primitives exchanged between the QUB-COS (SeqL) and the QUB-COS (CmdL) are described in Table 37.

Table 37 – Parameters used with primitives exchanged between QUB-COS (SeqL) and QUB-COS (CmdL)

Parameter name	Description
AREP, invoke-ID, error-info	These parameters are as defined in IEC 61158-1.
response	This parameter indicates whether the SEGMENT contains a request or a response.
abort	This parameter indicates that the requested transfer could not be completed.
segmentation	This parameter indicates whether the SEGMENT is part of a segmented transfer or not.
command-ID	Specifies the command.
segment-size	In case of a segmented transfer this parameter contains the length of segment data.
data size	In case of a segmented transfer this parameter contains the total length of the transferred data block.
OD-identifier(s)	Identifies the OD entries to be affected.
data	Payload data to /from the OD entries.

9.5.5 DLL mapping of QUB-COS AREP Class

9.5.5.1 Formal model

Subclause 9.5.5 describes the mapping of the QUB-COS AREP class to the Type 13 data link layer defined in IEC 61158-3-13 and IEC 61158-4-13. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data link layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is not in the scope of this International Standard.

The DLL Mapping attributes and their permitted values and the DLL services used with the QUB-COS AREP class are defined in 9.5.5.

CLASS: **Type 13 QUB-COS**
PARENT CLASS: **Queued user-triggered bi directional-connection-oriented AREP**
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	RecSeqNr
4	(m)	Attribute:	RecCon
5	(m)	Attribute:	SndSeqNr
6	(m)	Attribute:	SndCon

DLL SERVICES:

1	(m)	OpsService:	DL-SDO

9.5.5.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

RecSeqNr
This attribute is a local buffer for the sequence number of the last correctly received frame.

RecConNr
This attribute is a local buffer for the acknowledge of connection code to the receiver.

SndSeqNr
This attribute is a local buffer for the sequence number of the frame.

SndCon
This attribute is a local buffer for the connection code of the sender.

9.5.5.3 DLL services

Refer to IEC 61158-3-13 for DLL service descriptions.

9.5.6 QUB-COS (SeqL) ARPM state machine

9.5.6.1 QUB-COS (SeqL) ARPM states

The defined states and their descriptions of the QUB-COS (SeqL) ARPM are shown in Table 38 and Figure 9.

Table 38 – QUB-COS (SeqL) ARPM states

State	Description
CLOSED	The AREP is defined, but not capable of sending or receiving FAL-PDUs. It only may send or receive FAL-PDUs with the parameter scon set to 1 to indicate the wish to initialize a connection.
REQ	The AREP acts as a client. It has sent a FAL-PDU with the parameter scon set to 1, and is waiting for a response from the remote server AREP.
RSP	The AREP acts as a server. It has received a FAL-PDU from the remote client AREP with the parameter scon set to 1, has returned a response Fal-PDU with the parameters scon := 1 and rcon := 1, and is waiting for a response from its user.
OPEN	The AREP is defined and capable of sending or receiving FAL-PDUs.

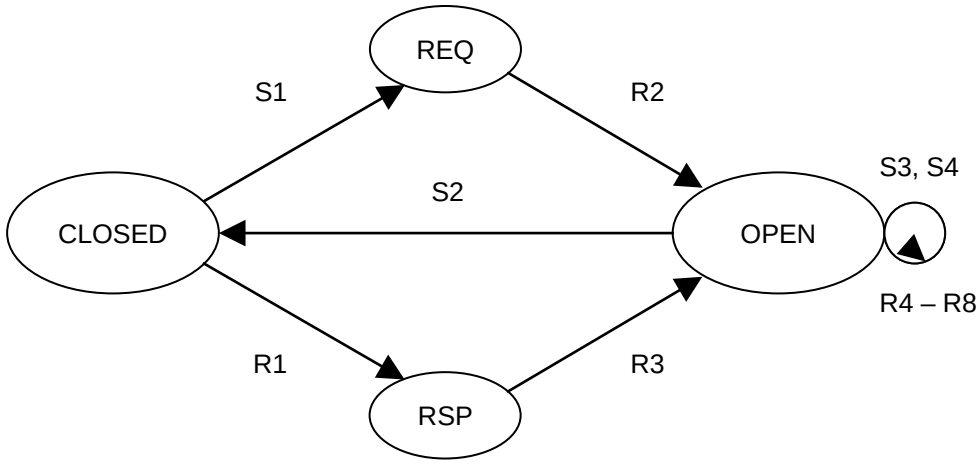


Figure 9 – State transition diagram of QUB-COS (SeqL) ARPM

9.5.6.2 QUB-COS (SeqL) ARPM state table

Table 39 and Table 40 define the state machine of the QUB-COS (SeqL) ARPM. In the comments of these tables "Node 1" indicates the initiator of a connection establishment process, "Node 2" indicates the addressed connection partner.

Table 39 – QUB-COS (SeqL) ARPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	CLOSED	SEGMENT_req — Node 1
		&& RecCon = 0, SndCon = 0
		&& RecSeqNr = ?, SndSeqNr = i
		⇒
		SndCon := 1
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			REQ

S2	OPEN	The connection is not needed any longer
		|| Delay of an expected response > SDO_SequLayerTimeout_U32
		⇒
		RecSeqNr := ?
		RecCon := 0
		SndSeqNr := ?
		SndCon := 0
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
		
		NOTE The necessity for a connection is evaluated by means beyond the scope of this standard.
			CLOSED

S3	OPEN	SEGMENT_req — regular data transfer, sender
		&& RecCon = 2, SndCon = 2
		⇒
		IncrementCounter(SndSeqNr)
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req)
		}
		
		History(SndSecNr) = AddToHistoryBuffer()
			OPEN

#	Current state	Event or condition ⇒ action	Next state

S4	OPEN	SEGMENT_req — sender history full → ack request
		&& RecCon = 2, SndCon = 2
		&& SndSeqNr + 1 = D_SDO_SeqLayerTxHistorySize_U16
		⇒
		RecCon = 2
		RecSeqNr
		SndCon = 3
		IncrementCounter(SndSeqNr)
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req)
		}
			OPEN

Table 40 – QUB-COS (SeqL) ARPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state

R1	CLOSED	FAL-PDU_ind — Node 2
		&& rcon = 0, scon = 1
		&& ssnr = i
		&& SndSeqNr = j
		&& RecCon = 0, SndCon = 0
		⇒
		RecSeqNr := ssnr
		RecCon := 1
		SndCon := 1
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			RSP

R2	REQ	FAL-PDU_ind — Node 1
		&& rcon = 1, scon = 1
		&& ssnr = j, rsnr = i
		&& RecCon = 0, SndCon = 1
		⇒
		RecSeqNr := ssnr
		RecCon := 1
		SndSeqNr := rsnr
		SndCon := 2
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			OPEN

R3	RSP	FAL-PDU_ind — Node 2
		&& rcon = 1, scon = 2
		&& ssnr = i
		&& RecCon = 1, SndCon = 1
		⇒
		RecSeqNr := ssnr
		RecCon := 2
		SndSeqNr := rsnr
		SndCon := 2
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			OPEN

#	Current state	Event or condition ⇒ action	Next state
R4	OPEN	FAL-PDU_ind — regular data transfer, receiver && RecCon = 2, SndCon = 2 && scon = 2, rcon = 2 && ssnr = RecSeqNr +1 && rsnr = SndSeqNr ⇒ RecSeqNr := ssnr RecCon := 2 SndSeqNr := rsnr + 1 SndCon := 2 SEGMENT_ind	OPEN

R5	OPEN	FAL-PDU-ind — Loss of Frame detected
		&& RecCon = 2, SndCon = 2
		&& scon = 2, rcon = 2
		&& ssnr > RecSeqNr +1
		&& rsnr = SndSeqNr
		⇒
		RecCon := 3
		RecSeqNr remains unchanged
		SndCon := 2
		SndSeqNr := rsnr
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req ! if present || "null")
		}
			OPEN

R6	OPEN	FAL-PDU_ind — data completion after loss of frame && RecCon = 2, SndCon = 2 && scon= 2, rcon =3 && rsnr < SndSeqNr ⇒ for i := rsnr+1 to SndSeqNr FAL-PDU_req := History(i) end for	OPEN

R7	OPEN	FAL-PDU-ind — Response to an ack request
		&& scon = 3, rcon = 2
		&& ssnr = RecSeqNr +1
		&& rsnr = SndSeqNr
		⇒
		RecCon := 2
		RecSeqNr := ssnr
		SndSeqCon := 2
		SndSeqNr := rsnr
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req ! if present || "null")
		}
			OPEN

R8	OPEN	FAL-PDU_ind — Duplication of frames detected && scon = 2 && ssnr <= RecSeqNr. ⇒ drop FAL-PDU_ind, no further action	OPEN

9.5.6.3 Functions used by QUB-COS (SeqL) ARPM

The receipt of a FAL-PDU_ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Table 41 through Table 43 define the other functions used by this state machine.

Table 41 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
default-parameters := message-type := 6h ("Asyn2") service-ID = 5h ("SDO") rsnr := RecSeqNr rcon := RecCon ssnr := SndSecNr scon := SndCon		dlsdu	
Function			
Builds a FAL-PDU out of the parameters given as input variables. The specified default values are valid for the use of this function in the QUB-COS (SeqL) ARPM only.			

Table 42 – Function IncrementCounter

Name	IncrementCounter	Used in	ARPM
Input		Output	
identifier			
Function			
This function increments the selected counter.			

Table 43 – Function AddToHistoryBuffer

Name	AddToHistoryBuffer	Used in	ARPM
Input		Output	
FAL-PDU_req SndSeqNr		History	
Function			
This function adds a sent FAL-PDU_req primitive to the history buffer of the state machine. The entry is referenced by the related SndSeqNr.			

10 DLL mapping protocol machine

10.1 Primitive definitions

10.1.1 Primitives exchanged between DMPM and ARPM

Table 44 and Table 45 list the primitives exchanged between DMPM and ARPM.

Table 44 – Primitives issued by ARPM to DMPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_req	ARPM	dlsdu	This primitive is used to request the DMPM to transfer an FAL-PDU, or to request an abort without transferring an FAL-PDU. It passes the FAL-PDU to the DMPM as a DLSDU.

Table 45 – Primitives issued by DMPM to ARPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_ind	DMPM	dlsdu	This primitive is used to pass an FAL-PDU received as a data link layer service data unit to a designated ARPM.

10.1.2 Parameters of ARPM / DMPM primitives

The "dlsdu" parameter contains the data of the application process and all relevant information for the state machine. The DMPM state machine is able to extract these informations.

10.1.3 Primitives exchanged between data-link layer and DMPM

Table 46 and Table 47 list the primitives exchanged between data-link layer and DMPM.

Table 46 – Primitives issued by DMPM to data-link layer

Primitive name	Source	Associated parameters
DL-PDO.req	DMPM	dl_dls_user_data
DL-CMD.req	DMPM	dl_ dls_user_data
DL-IDE.req	DMPM	dl_dls_user_data
DL-STA.req	DMPM	dl_dls_user_data
DL-REQ.req	DMPM	dl_dls_user_data
DL-SDO.req	DMPM	dl_dls_user_data

Table 47 – Primitives issued by data-link layer to DMPM

Primitive name	Source	Associated parameters
DL-PDO.ind	data-link layer	dl_dls_user_data
DL-CMD.ind	data-link layer	dl_ dls_user_data
DL-IDE.ind	data-link layer	dl_dls_user_data
DL-STA.ind	data-link layer	dl_dls_user_data
DL-REQ.ind	data-link layer	dl_dls_user_data
DL-SDO.ind	data-link layer	dl_dls_user_data

10.1.4 Parameters of DMPM / data-link layer primitives

The parameters used with the primitives exchanged between the DMPM and the data link layer are defined in the DLL Service definition (see IEC 61158-3-13). They are prefixed by “dl_” to indicate that they are used by the FAL.

10.2 DMPM state machine

10.2.1 DMPM states

The DMPM state machine has only one state called "ACTIVE", see Figure 10.

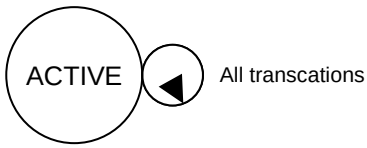


Figure 10 – State transition diagram of DMPM

10.2.2 DMPM state table

Table 48 and Table 49 define the state machine of the DMPM.

Table 48 – DMPM state table – sender transactions

#	Current state	Event or condition ⇒ action	Next state

S1	ACTIVE	FAL-PDU_req
		&& message-type = 3h
		|| message-type = 4h
		⇒
		DL-PDO.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S2	ACTIVE	FAL-PDU_req
		&& service-ID = 4h
		⇒
		DL-CMD.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S3	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 1h || service-ID = 1h
		⇒
		DL-IDE.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S4	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 2h || service-ID = 2h
		⇒
		DL-STA.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S5	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 3h || service-ID = 3h
		⇒
		DL-REQ.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S6	ACTIVE	FAL-PDU_req
		&& service-ID = 5h
		⇒
		DL-SDO.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

Table 49 – DMPM state table – receiver transactions

#	Current state	Event or condition ⇒ action	Next state
R1	ACTIVE	DL-xxx.ind ⇒ FAL_PDU_ind	ACTIVE

10.2.3 Functions used by DMPM

The receipt of a DL_Put.ind or a DL_Data.ind primitive is always followed by its decoding to derive its relevant parameters for the state machine. Thus this implicit function is not listed separately.

Annex A
(normative)

Constant value assignments

A.1 Values of abort-code

The valid values of the abort-code and their meaning are listed in Table A.1.

Table A.1 – Values of abort-code

Abort code	Description
0503 0000h	reserved
0504 0000h	SDO protocol timed out.
0504 0001h	Client/server command-ID not valid or unknown.
0504 0002h	Invalid block size.
0504 0003h	Invalid sequence number.
0504 0004h	reserved
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write-only object.
0601 0002h	Attempt to write a read-only object.
0602 0000h	Object does not exist in the object dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility.
0604 0044h	Invalid heartbeat declaration
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to an hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).
0800 0024h	EDS, DCF or Concise DCF Data set empty.

A.2 NMT-command-ID

The valid values of the NMT-command-ID and their meaning are listed in Table A.2.

Table A.2 – Values of NMTCommandID

Name	ID Value
Plain NMT State Commands	20h .. 3Fh
NMTStartNode	21h
NMTStopNode	22h
NMTEnterPreOperational2	23h
NMTEnableReadyToOperate	24h
NMTResetNode	28h
NMTResetCommunication	29h
NMTResetConfiguration	2Ah
NMTSwReset	2Bh
Extended NMT State Commands	40h .. 5Fh
NMTStartNodeEx	41h
NMTStopNodeEx	42h
NMTEnterPreOperational2Ex	43h
NMTEnableReadyToOperateEx	44h
NMTResetNodeEx	48h
NMTResetCommunicationEx	49h
NMTResetConfigurationEx	4Ah
NMTSwResetEx	4Bh
NMT Info services	80h .. BFh
NMTPublishConfiguredNodes	80h
NMTPublishActiveNodes	90h
NMTPublishPreOperational1	91h
NMTPublishPreOperational2	92h
NMTPublishReadyToOperate	93h
NMTPublishOperational	94h
NMTPublishStopped	95h
NMTPublishNodeStates	96h
NMTPublishEmergencyNew	A0h
NMTPublishTime	B0h
NMTInvalidService	FFh

A.3 Type 13 specific error-code constants

The error-code constants specific for Type 13 are listed in Table A.3.

Table A.3 – Type 13 specific error-code constants

Name	Value	Description
	816xh	HW errors
E_DLL_BAD_PHYS_MODE	8161h	
E_DLL_COLLISION	8162h	
E_DLL_COLLISION_TH	8163h	
E_DLL_CRC_TH	8164h	
E_DLL_LOSS_OF_LINK	8165h	
E_DLL_MAC_BUFFER	8166h	
	82xxh	Protocol errors
E_DLL_ADDRESS_CONFLICT	8201h	
E_DLL_MULTIPLE_MN	8202h	
	821xh	Frame size errors
E_PDO_SHORT_RX	8210h	
E_PDO_MAP_VERS	8211h	
E_NMT_ASND_MTU_DIF	8212h	
E_NMT_ASND_MTU_LIM	8213h	
E_NMT_ASND_TX_LIM	8214h	
	823xh	Timing errors
E_NMT_CYCLE_LEN	8231h	
E_DLL_CYCLE_EXCEED	8232h	
E_DLL_CYCLE_EXCEED_TH	8233h	
E_NMT_IDLE_LIM	8234h	
E_DLL_JITTER_TH	8235h	
E_DLL_LATE_PRES_TH	8236h	
E_NMT_PREQ_CN	8237h	
E_NMT_PREQ_LIM	8238h	
E_NMT_PRES_CN	8239h	
E_NMT_PRES_RX_LIM	823Ah	
E_NMT_PRES_TX_LIM	823Bh	
	824xh	Frame errors
E_DLL_INVALID_FORMAT	8241h	
E_DLL_LOSS_PREQ_TH	8242h	
E_DLL_LOSS_PRES_TH	8243h	
E_DLL_LOSS_SOA_TH	8244h	
E_DLL_LOSS_SOC_TH	8245h	
E_DLL_LOSS_STATUSRES_TH	8246h	
	84xxh	BootUp errors
E_NMT_BA1	8410h	
E_NMT_BA1_NO_MN_SUPPORT	8411h	
E_NMT_BPO1	8420h	
E_NMT_BPO1_GET_IDENT	8421h	
E_NMT_BPO1_DEVICE_TYPE	8422h	
E_NMT_BPO1_VENDOR_ID	8423h	
E_NMT_BPO1_PRODUCT_CODE	8424h	
E_NMT_BPO1_REVISION_NO	8425h	
E_NMT_BPO1_SERIAL_NO	8426	
E_NMT_BPO1_CONFIGURATION	8428h	
E_NMT_BPO2	8430h	
E_NMT_BRO	8440h	
E_NMT_WRONG_STATE	8480h	

A.4 Node-list

The node-list assigns one bit for each Type 13 fieldbus node ID. The node ID assignment is given in Table A.4.

Table A.4 – Node-list format

Octet offset	Bit offset							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	7	6	5	4	3	2	1	-
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	39	38	37	36	35	34	33	32
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56
8	71	70	69	68	67	66	65	64
9	79	78	77	76	75	74	73	72
10	87	86	85	84	83	82	81	80
11	95	94	93	92	91	90	89	88
12	103	102	101	100	99	98	97	96
13	111	110	109	108	107	106	105	104
14	119	118	117	116	115	114	113	112
15	127	126	125	124	123	122	121	120
16	135	135	133	132	131	130	129	128
17	143	142	141	140	139	138	137	136
18	151	150	149	148	147	146	145	144
19	159	158	157	156	155	154	153	152
20	167	166	165	164	163	162	161	160
21	175	174	173	172	171	170	169	168
22	183	182	181	180	179	178	177	176
23	191	190	189	188	187	186	185	184
24	199	198	197	196	195	194	193	192
25	207	206	205	204	203	202	201	200
26	215	214	213	212	211	210	209	208
27	223	222	221	220	219	218	217	216
28	231	230	229	228	227	226	225	224
29	239	238	237	236	235	234	233	232
30	247	246	245	244	243	242	241	240
31	-	254	253	252	251	250	249	248

Bibliography

IEC 61158-1, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-6 (all parts), *Industrial communication networks – Fieldbus specifications – Part 6: Application layer protocol specification*

IEC 61784-1, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

EPSG DS 301 V1.2.0, *Ethernet POWERLINK Communication Profile Specification, Draft Standard Version 1.2.0, EPSG 2013,* available from http://www.ethernet-powerlink.com

_____

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-13: Spécification du protocole de la couche application – Éléments de type 13

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.

L'attention est attirée sur le fait que l'utilisation de certains types de protocole associés est limitée par leurs détenteurs de droit à la propriété intellectuelle. Dans tous les cas, l'engagement visant à limiter l'abandon des droits à la propriété intellectuelle prévus par les détenteurs de ces droits permet d'utiliser un type de protocole de couche avec les autres protocoles de même type, ou dans d'autres combinaisons de type explicitement autorisées par leurs détenteurs de droits à la propriété intellectuelle respectifs.

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

La Norme internationale CEI 61158-6-13 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette deuxième édition annule et remplace la première édition parue en 2007. Cette édition constitue une révision technique.

Les modifications majeures par rapport à l'édition antérieure sont énumérées ci-dessous:

- ajout de la caractéristique de synchronisation,
- corrections et
- améliorations éditoriales.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65C/764/FDIS	65C/774/RVD

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

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

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

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

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

INTRODUCTION

La présente partie de la CEI 61158 est l'une d'une série produite pour faciliter l'interconnexion de composants de systèmes d'automation. Elle est liée à d'autres normes dans l'ensemble tel que défini par le modèle de référence des bus de terrain "à trois couches" décrit dans la CEI 61158-1.

Le protocole d'application fournit le service d'application en utilisant les services disponibles issus de la couche liaison de données ou d'une autre couche immédiatement supérieure. Le but principal de la présente norme est de fournir un ensemble de règles pour la communication exprimées en termes des procédures devant être accomplies par des entités d'application (AE) d'homologues au moment de la communication. Ces règles pour la communication visent à fournir une base solide pour le développement et de servir une diversité de besoins:

– comme un guide pour les réalisateurs et les concepteurs;
– pour une utilisation dans les essais et achats d'équipements;
– comme partie intégrante d'un accord pour l'admission de systèmes dans l'environnement de systèmes ouverts;
– comme affinement pour la compréhension de communications prioritaires au sein de l'OSI (Open Systems Interconnexion, c'est-à-dire Interconnexion des systèmes ouverts).

La présente norme est concernée, en particulier, par la communication et l'interfonctionnement des capteurs, des effecteurs et autres appareils d'automatisation. L'utilisation de la présente norme conjointement à d'autres normes positionnées dans les modèles de référence de l'OSI ou de bus de terrain permet à n'importe quelle combinaison de systèmes autrement incompatibles de fonctionner.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-13: Spécification du protocole de la couche application – Éléments de type 13

1 Domaine d'application

1.1 Généralités

La couche application de bus de terrain (FAL «Fieldbus Application Layer») fournit aux programmes d'utilisateur un moyen d'accéder à l'environnement de communication du bus de terrain. À cet égard, la FAL peut être vue comme une «fenêtre entre des programmes d'application correspondants».

La présente norme fournit les éléments communs pour les communications de messagerie de base prioritaire et non prioritaire entre des programmes d'application dans un environnement d'automation et dans un matériel spécifiques au bus de terrain du Type 13. Le terme «prioritaire» sert à représenter la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un certain niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, l'installation et éventuellement pour la vie humaine.

La présente norme spécifie les interactions entre les applications distantes et définit le comportement visible de l'extérieur fourni par la couche application de bus de terrain de Type 13 en termes

a) de la syntaxe abstraite formelle définissant les unités de données de protocole de couche application acheminées entre les entités d'application engagées dans une communication;

b) de la syntaxe de transfert définissant les règles de codage qui sont appliquées aux unités de données de protocole de couche application;

c) du diagramme d'états de contexte application définissant le comportement de service application visible entre des entités d'application engagées dans une communication;

d) des diagrammes d'états de relation d'applications définissant le comportement de communication visible entre des entités d'application engagées dans une communication.

Le but de la présente norme est de définir le protocole fourni pour

1) définir la représentation câblée des primitives de service définies dans la CEI 61158-5-13, et

2) définir le comportement visible de l'extérieur qui est associé à leur transfert.

La présente norme spécifie le protocole de la couche application des réseaux de terrain de Type 13, en conformité avec le Modèle de référence de base de l'OSI (ISO/CEI 7498) et la Structure de la couche application de l'OSI (ISO/CEI 9545).

1.2 Spécifications

L'objet principal de la présente norme est de spécifier la syntaxe et le comportement du protocole de couche application qui achemine les services de couche application définis dans la CEI 61158-5-13.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. C'est ce dernier objectif qui donne naissance à la diversité des protocoles normalisés dans la CEI 61158-6.

1.3 Conformité

La présente norme ni ne spécifie de mises en œuvre individuelles ou de produits individuels ni ne contraint les mises en œuvre d'entités de couche application au sein des systèmes d'automatisation industriels. La conformité est obtenue par la mise en œuvre de cette spécification de protocole de couche application.

2 Références normatives

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

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

CEI 61158-3-13, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-13: Définition des services de la couche liaison de données – Eléments de type 13*

CEI 61158-4-13, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-13: Spécification du protocole de la couche liaison de données – Eléments de type 13*

CEI 61158-5-13, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-13: Définition des services de la couche application – Eléments de type 13*

ISO/CEI 7498 (toutes les parties), *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base*

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le modèle de base*

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

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts – Définition du service de présentation*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation* (disponible en anglais seulement)

ISO/CEI 9545, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Structure de la couche Application*

ISO/IEC 9899*, Information technology – Programming languages – C* (disponible en anglais seulement)

IEEE 754, *IEEE Standard for Floating-Point Arithmetic* (disponible en anglais seulement)

3 Termes, définitions, symboles, abréviations et conventions

Pour les besoins du présent document, les termes, définitions, symboles, abréviations et conventions suivants s'appliquent.

3.1 Termes de l'ISO/CEI 7498-1

La présente norme est partiellement basée sur les concepts développés dans l'ISO/CEI 7498-1 et utilise les termes suivants y définis:

3.1.1 entité d'application

3.1.2 processus d'application

3.1.3 unité de donnée de protocole application

3.1.4 élément de service application

3.1.5 invocation d'entité d'application

3.1.6 transaction d'application

3.1.7 syntaxe de transfert

3.2 Termes de l'ISO/CEI 8822

Pour les besoins du présent document, le terme suivant tel que défini dans l'ISO/CEI 8822 s'applique:

3.2.1 syntaxe abstraite

3.3 Termes de l'ISO/CEI 9545

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 9545 s'appliquent:

3.3.1 contexte d'application

3.3.2 type de processus d'application

3.3.3 élément de service application

3.3.4 élément de service de contrôle d'application

3.4 Termes de l'ISO/CEI 8824-1

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 8824 s'appliquent:

3.4.1 type any ("n'importe quel type")

3.4.2 type bitstring (type "chaîne de bits")

3.4.3 type Boolean (type booléen)

3.4.4 type choice (type "choix")

3.4.5 false (faux)

3.4.6 type integer (type "entier")

3.4.7 module

3.4.8 null type (type "null")

3.4.9 identificateur d'objet

3.4.10 type octetstring (type "chaîne d'octets")

3.4.11 production

3.4.12 type simple

3.4.13 séquence de type

3.4.14 sequence type (type "séquence")

3.4.15 type structuré

3.4.16 indicateur

3.4.17 type marqué

3.4.18 true (vrai)

3.4.19 type

3.5 Termes et définitions issus de la CEI 61158-5-13

3.5.1 relation entre applications

3.5.2 client

3.5.3 classe d'erreurs

3.5.4 éditeur

3.5.5 serveur

3.5.6 abonné

3.6 Autres termes et définitions

Les termes et définitions suivants sont utilisés dans la présente norme:

3.6.1
récepteur, destinataire
utilisateur de service qui reçoit une primitive confirmée ou une primitive non conformée, ou fournisseur de service qui reçoit une APDU confirmée ou une APDU non confirmée

3.6.2
ressource
capacité de traitement ou d'informations d'un sous-système

3.6.3
expéditeur
utilisateur de service qui envoie une primitive confirmée ou une primitive non conformée, ou fournisseur de service qui envoie une APDU confirmée ou une APDU non confirmée

3.6.4
nœud gérant
nœud qui peut gérer le mécanisme SCNM

3.6.5
nœud commandé
nœud sans la capacité de gérer le mécanisme SCNM

3.7 Abréviations et symboles

AE	Application Entity (Entité d'application)
AL	Application Layer (Couche application)
AP	Application Process (Processus d'application)
APDU	Application Protocol Data Unit (Unité de données de protocole application)
AR	Application Relationship (Relation entre applications)
AREP	Application Relationship End Point (Point d'extrémité de relation

	d'applications)
ARPM	Application Relationship Protocol Machine (Machine protocolaire de relations entre applications)
ASnd	"Asynchronous Send" (Envoi de l'asynchrone) (type d'une trame de Type 13)
BNB-PEC	Buffered network-scheduled bi-directional pre-established connection (Connexion préétablie bidirectionnelle programmée par un réseau et mise en tampon)
BNU-PEC	Buffered network-scheduled uni-directional pre-established connection (Connexion préétablie unidirectionnelle programmée par un réseau et mise en tampon)
CmdL	Command Layer (Couche de commande)
CN	Controlled Node (Nœud commandé)
cnf	confirmation
DL-	Préfixe désignant la couche Liaison de données
DLCEP	Data-Link Connection End Point (Point d'extrémité de connexion de liaison de données)
DLL	Data-Link Layer (Couche liaison de données)
DLME	Data-Link-Management Entity (Entité de gestion de liaison de données)
DLSAP	Data-Link Service Access Point (Point d'accès au service liaison de données)
DLSDU	DL-Service-Data-Unit (Unité de données de service de DL)
DMPM	DLL Mapping Protocol machine (Machine protocolaire de mapping à la couche Liaison de données)
DNS	Domain Name Service (Service de noms de domaine)
FAL	Fieldbus Application Layer (Couche application de bus de terrain)
ind	indication
IP	Internet Protocol (protocole internet, voir RFC 791)
MAC	Media Access Control (Commande d'accès au support physique)
MN	Managing Node (Nœud gérant)
NMT	Network Management (Gestion de réseau)
OD	Object Dictionary (Dictionnaire d'objets)
PDO	Process Data Object (Objet de données de processus)
PDU	Process Data Unit (Unité de données de processus)
QUB-CL	Queued user-triggered bi-directional connectionless (En file d'attente, déclenché par un utilisateur, bidirectionnel et sans connexion)
QUB-COS	Queued user-triggered bi-directional connection-oriented with segmentation (Orienté connexion bidirectionnel déclenché par un utilisateur et mis en file d'attente avec segmentation)
QUU	Queued user-triggered uni-directional (Unidirectionnel déclenché par un utilisateur et mis en file d'attente)
req	request (Demande)
rsp	response (Réponse)
SDO	Service Data Object (Objet de données de service)
SeqL	Sequence Layer (Couche Séquence)
UDP	User Datagram Protocol (Protocole datagramme d'utilisateur)

4 Description de la syntaxe de FAL

4.1 Généralités

La présente description de la syntaxe abstraite du Type 13 utilise des formalismes semblables à ASN.1, bien que les règles de codage diffèrent par rapport à la norme en question.

4.2 Syntaxe abstraite des PDU des AR de la FAL

4.2.1 Définition de haut niveau

```
APDU ::= CHOICE {
      [3] Isoc1
      [4] Isoc2
      [5] Asyn1
      [6] Asyn2
}
```

4.2.2 Isoc1

```
Isoc1 ::= SEQUENCE {
      message-type
      destination
      source
      reserved
      signaling-flags
      PDO-version
      reserved8
      size
      PDO-payload
}
```

4.2.3 Isoc2

```
Isoc2 ::= SEQUENCE {
      message-type
      destination
      source
      NMT-status
      signaling-flags
      PDO-version
      reserved8
      size
      PDO-payload
}
```

4.2.4 Asyn1

```
Asyn1 ::= SEQUENCE {
      message-type
      destination
      source
      NMT-status
      signaling-flags
      requested-service-ID
      requested-service-target
      fieldbus-version
      reserved8
      pduBody     CHOICE{
         [1h…5h] reserved
         [6h] Sync-request
         [7h…FFh] reserved
      }

}
```

4.2.5 Asyn2

```
Asyn2 ::= SEQUENCE {
   message-type
   destination
   source
   service-ID
   pduBody        CHOICE {
      [1h] ident-response
      [2h] status-response
      [3h] NMT-request
      [4h] NMT-command
      [5h] SDO
      [6h] Sync-response
      [A0h...FEh] manufacturer-specific
      [FFh] reserved
   }
}
```

4.2.6 **Message-type** (Type de message)

message-type ::= Unsigned8 — Contient les indicateurs APDU spécifiques à un contexte.

4.2.7 Adresses

destination ::= Unsigned8 — Adresse de nœud (1...255)
source ::= Unsigned8 — Adresse de nœud (1...250, 253, 254)

4.2.8 **Service-ID** (Identificateur du service)

service-ID ::= Unsigned8 — Contient les indicateurs spécifiques à un contexte pour le pduBody

4.2.9 Reserved8

reserved8 ::= Unsigned8

4.2.10 Reserved16

reserved16 ::= Unsigned16

4.2.11 Reserved24

reserved24 ::= Unsigned24

4.2.12 **Signaling-flags** (Fanions de signalisation)

```
signaling-flags ::= BitString {
   RD             (0)
   ER             (1)
   EA             (2)
   EC             (3)
   EN             (4)
   MS             (5)
   PS             (6)
   MC             (7)
   RS_bit1        (8)
   RS_bit2        (9)
   RS_bit3        (10)
   PR_bit1        (11)
   PR_bit2        (12)
   PR_bit3        (13)
   reserved       (14)
   reserved       (15)
}
```

Les différents types d'APDU utilisent des fanions tels qu'énumérés dans le Tableau 1. Dans tous les cas ne comportant de "x", les fanions sont présents, mais ne sont pas écrits respectivement interprétés.

Tableau 1 – Utilisation de fanions de signalisation

	Isoc1	Isoc2	Asyn1	IdentResponse	StatusResponse	SyncResponse
RD	x	x				
ER			x			
EA	x		x			
EC					x	
EN		x			x	
MS	x	x				
PS						
MC						
RS	x	x		x	x	
PR	x	x		x	x	

L'utilisation de ces fanions est spécifiée dans la CEI 61158-3-13 et dans la CEI 61158-4-13.

4.2.13 **PDO-version** (Version du PDO)

PDO-version ::= Unsigned8 — Quartet de poids fort: version principale; quartet de poids faible: version secondaire

4.2.14 **Size** (Taille)

size ::= Unsigned8 — Taille de la charge utile du PDO; 1 490 au maximum en raison — de restrictions Ethernet et de la surcharge de protocole

4.2.15 **PDO-payload** (Charge utile de PDO)

PDO-payload ::= Any

4.2.16 **NMT-status** (Statut de NMT)

```
NMT-status ::= CHOICE {
   NMT_GS_OFF                    Unsigned8 ::= 0000 0000b
   NMT_xS_NOT_ACTIVE             Unsigned8 ::= 0001 1100b
   NMT_xS_PRE_OPERATIONAL_1      Unsigned8 ::= 0001 1101b
   NMT_xS_PRE_OPERATIONAL_2      Unsigned8 ::= 0101 1101b
   NMT_xS_READY_TO_OPERATE       Unsigned8 ::= 0110 1101b
   NMT_xS_OPERATIONAL            Unsigned8 ::= 1111 1101b
   NMT_xS_STOPPED                Unsigned8 ::= 0100 1101b
   NMT_xS_BASIC_ETHERNET         Unsigned8 ::= 0001 1110b
}
```

NOTE si expéditeur = MN: "x" := "M"; si expéditeur = CN: "x" := "C".

4.2.17 **Requested-service-ID** (ID du service demandé)

```
requested-service-ID ::= CHOICE {
   no-service                    [0h] IMPLICIT Unsigned8
   ident-request                 [1h] IMPLICIT Unsigned8
   status-request                [2h] IMPLICIT Unsigned8
   NMT-req-inv                   [3h] IMPLICIT Unsigned8
   manufacturer-specific         [A0h]... [FEh] IMPLICIT Unsigned8
   unspecified-invite            [FFh] IMPLICIT Unsigned8
}
```

4.2.18 **Requested-service-target** (Cible du service demandé)

requested-service-target ::= Unsigned8 — Adresse de nœud (1…255); non attribuée (0)

4.2.19 **Fieldbus-version** (Version du bus de terrain)

fieldbus-version ::= Unsigned8 — Quartet de poids fort: version principale; quartet de poids — faible: version secondaire

4.3 Syntaxe abstraite de l'Asyn1 pduBody

4.3.1 **Sync-request** (Demande de synchronisation)

4.3.1.1 Vue d'ensemble

Sync-request ::= SEQUENCE {			
synchronization-control	Bitstring	—	voir 4.3.1.2.
PRes-time	Unsigned32	—	retard temporel entre la fin de la réception du PRes en provenance du MN et le début de l'envoi de son propre PRes à déclenchement temporel, en ns
reserved	Unsigned32		
sync-MN-delay	Unsigned32	—	temps de propagation entre MN et CN, en ns
reserved	Unsigned32		
fallback-timeout	Unsigned32	—	délai d'expiration de SoC pour désactiver l'envoi à déclenchement temporel du PRes dans l'état NMT_CS_PRE_OPERATIONAL_2, en ns
destination-MAC-address	Unsigned32	—	adresse MAC de destination un nœud vers lequel la demande de synchronisation est envoyée
}			

NOTE Les éléments énumérés ci-dessus sont parfois récapitulés comme suit : "synchronization-control" à " destination-MAC-address " sont récapitulés sous le terme "sync-control" (commande de synchronisation).

4.3.1.2 **Synchronization-control** (Commande de synchronisation)

Synchronization-control ::= Bitstring {			
PRes-time-valid	(0)	—	Le paramètre PRes-time est valide.
Reserved bit1	(1)		
sync-MN-delay-valid	(2)	—	Le paramètre sync-MN-delay est valide.
Reserved bit2	(3)		
fallback-timeout-valid	(4)	—	Le paramètre fallback-timeout est valide.
Reserved bit3	(5)		
MAC-address-valid	(4)	—	Le paramètre destination-MAC-address est valide.
reserved_bit4 à bit26	(7)…(29)		
PRes-mode-reset	(30)	—	Désactiver l'envoi à déclenchement temporel de PRes
PRes-mode-set	(31)	—	Activer l'envoi à déclenchement temporel de PRes. Ce bit annule le bit30
}			

4.4 Syntaxe abstraite de l'Asyn2 pduBody

4.4.1 **Ident-response** (Réponse d'identification)

4.4.1.1 Vue d'ensemble

Ident-response ::= SEQUENCE {

signaling-flags		
NMT-status		
reserved8		
fieldbus-version		
reserved8		
feature-flags	BitString	— (voir 4.4.1.2)
MTU	Unsigned16	— taille de la trame IP la plus grande possible y compris l'en-tête
poll-in-size	Unsigned16	— Valeur de réglage réelle du CN pour le bloc de données Isoc1
poll-out-size	Unsigned16	— Valeur de réglage réelle du CN pour le bloc de données Isoc2
response-time	Unsigned32	— temps requis par le CN pour répondre à Isoc1
reserved16		
device-type	Unsigned32	— type d'appareil du CN
vendor-ID	Unsigned32	— Identificateur du vendeur de CN
product-code	Unsigned32	— code-produit du CN
revision-number	Unsigned32	— numéro de révision du CN
serial-number	Unsigned32	— numéro de série du CN
vendor-specific-extension-1	Unsigned64	— à des fins spécifiques à un vendeur, à remplir de zéros s'il n'est pas utilisé
verify-configuration-date	Unsigned32	— date de configuration du CN
verify-configuration-time	Unsigned32	— heure de configuration du CN
application-sw-date	Unsigned32	— date du logiciel d'application du CN
application-sw-time	Unsigned32	— heure du logiciel d'application du CN
IP-address	Unsigned32	— valeur actuelle de l'adresse IP du CN
subnet-mask	Unsigned32	— masque de sous-réseau IP courant du CN
default-gateway	Unsigned32	—passerelle IP par défaut courante du CN
host-name	VisibleString32	— nom d'hôte DNS courant du CN
vendor-specific-extension-2	SEQUENCE SIZE(48) OF Unsigned8	— à des fins spécifiques à un vendeur, à remplir de zéros s'il n'est pas utilisé

}

NOTE Certains des éléments énumérés ci-dessus sont parfois récapitulés comme suit:

"poll-in-size" à "response-time" sont récapitulés sous le terme "cycle-timing" (durée de cycle),

"device-type" à "serial-number" sous "identity" (identité),

"verify-configuration-date" et "verify-configuration-time" sous "verify-configuration" («vérifier la configuration»),

"application-sw-date" et "application-sw-time" sous "application-software-version" («version du logiciel d'application»),

"vendor-specific-extension-1" et "vendor-specific-extension-2" sous "vendor-specific-extensions" (extensions spécifiques à un vendeur),

"IP-address" à "default-gateway" sous "IP-address" (adresse IP).

4.4.1.2 **Feature-flags** (Fanions caractéristiques)

feature-flags ::= BitString {

Isochronous	(0)	— l'appareil peut être accessible de manière isochrone par l'intermédiaire de Isoc1.
SDO by UDP/IP	(1)	— l'appareil prend en charge la communication de SDO par UDP/IP.
SDO by ASnd	(2)	— l'appareil prend en charge la communication de SDO par l'intermédiaire d'ASnd.
réservé pour un usage ultérieur	(3)	
NMT-info services	(4)	— l'appareil prend en charge les Services Info NMT.
Extended NMT-state-commands	(5)	— l'appareil prend en charge les commandes d'état étendues NMT.
Dynamic PDO mapping	(6)	— l'appareil prend en charge le mapping dynamique de PDO.
NMT services by UDP/IP	(7)	— l'appareil prend en charge les services NMT par UDP/IP.
Configuration manager	(8)	— l'appareil prend en charge les fonctions de Gestionnaire de la Configuration.
Multiplexed access	(9)	— l'appareil CN prend en charge l'accès isochrone multiplexé.
Node-ID setup by SW	(10)	— l'appareil prend en charge l'installation de NodeID par logiciel
MN basic ethernet mode	(11)	— l'appareil MN prend en charge le mode Ethernet de base.
Routing Type 1 support	(12)	— l'appareil prend en charge les fonctions de routage du Type 1
Routing Type 2 support	(13)	— l'appareil prend en charge les fonctions de routage du Type 2
WriteMultipleByIndex	(14)	— l'appareil prend en charge le service SDO WriteMultipleByIndex.
ReadMultipleByIndex	(15)	— l'appareil prend en charge le service SDO ReadMultipleByIndex.
Reserved bit1 à bit2	(16)...(17)	
Time-triggered PRes	(18)	— l'appareil prend en charge l'envoi à déclenchement temporel de PRes
Reserved bit3 à bit16	(19)...(31)	

}

4.4.2 **Status-response** (Réponse de statut)

4.4.2.1 **Vue d'ensemble**

Status-response ::= SEQUENCE {
signaling-flags
NMT-status
reserved24
static-error-bit-field
List-of-errors
}

4.4.2.2 **Static-error-bitfield** (Champ de bits d'erreur statique)

static-error-bit-field ::= SEQUENCE {

error-register	BitString	— (voir 4.4.2.3)
reserved8		
specific-errors	SEQUENCE SIZE (6) OF Unsigned8	— Erreurs spécifiques à un profil ou à un vendeur d'appareil.

}

4.4.2.3 **Error-register** (Registre d'erreurs)

error-register ::= BitString {

Generic error	(0)		— 0, si absence d'erreur statique, sinon 1
Current	(1)	OPTIONAL	
Voltage	(2)	OPTIONAL	
Temperature	(3)	OPTIONAL	
Communication error	(4)	OPTIONAL	
Device profile specific	(5)	OPTIONAL	
reserved	(6)	OPTIONAL	— toujours 0
Manufacturer specific	(7)	OPTIONAL	

}

4.4.2.4 **List-of-errors** (Liste d'erreurs)

List-of-errors ::= SEQUENCE SIZE (k) OF ErrorEntry — k :>= 2, dépend de la configuration de l'appareil

4.4.2.5 **ErrorEntry** (Entrée d'erreur)

```
ErrorEntry ::= SEQUENCE {
   error-type              Unsigned16      — (voir 4.4.2.6)
   error-code              Unsigned16      — (voir 4.4.2.6 et Article A.3)
   time-stamp              Unsigned64
   additional-information  Unsigned64
}
```

4.4.2.6 **Error-type** (Type d'erreur)

Les valeurs possibles dans error-type et leurs significations sont énumérées dans le Tableau 2.

Tableau 2 – Valeurs de error-type

Octet	Bit	Valeur	Description
0 .. 1	15 (statut)	0b	Entrée d'historique des erreurs
		1b	Entrée de statut dans la trame de réponse de statut (le Bit 14 doit être mis à 0b)
	14 (envoi)	0b	Entrée d'historique des erreurs seulement
		1b	En plus de l'entrée d'historique des erreurs, l'entrée doit aussi être introduite dans la file d'attente d'urgence de la signalisation d'erreur.
	13 .. 12 (mode)	0h	Interdite dans l'entrée d'historique des erreurs. Les entrées avec ce mode peuvent seulement être utilisées par la signalisation d'erreur elle-même pour indiquer la fin des entrées d'historique dans la trame de réponse de Statut.
		1h	Une erreur s'est produite et est active (par exemple: court-circuit de sortie détecté)
		2h	Une erreur active a été éliminée (plus de court-circuit, par exemple) (interdite pour les entrées de statut)
		3h	Une erreur / un événement a eu lieu (interdite pour les entrées de statut)
	11 .. 0 (profil)	000h	Réservés
		001h	error-code contient un code d'erreur spécifique à un vendeur
		002h	error-code contient des erreurs de profil de communication spécifiques à un réseau de Type 13 (voir Article A.3)
		003h .. FFFh	error-code contient des erreurs spécifiques à un profil d'appareil

4.4.3 **NMT-request** (Demande NMT)

```
NMT-request ::= SEQUENCE {
   NMT-requested-command-ID      Unsigned8   — plage de valeurs voir Article A.2
   NMT-requested-command-target  Unsigned8   — adresse de nœud cible
   NMT-requested-command-data    Any
}
```

4.4.4 **NMT-command** (Commande NMT)

```
NMT-command ::= SEQUENCE {
   NMT-command-ID CHOICE {                                  — plage de valeurs, voir Article A.2
      NMT-state-command              Unsigned8
      NMT-info                       Unsigned8
   }
   reserved8
   NMT-command-data   CHOICE {
      date-time          TimeOfDay                          — seulement si NMT-command-ID = B0h; voir
                                                              Article A.4
      node-states        SEQUENCE SIZE (255) OF Unsigned8    — si NMT-command-ID = 96h; voir Article A.4;
                                                              rapporte individuellement chaque état de
                                                              nœud
      node-list                                             — si NMT-command-ID = 40h...95h, A0h;
                                                              voir Article A.4
      NULL                                                  — sinon
   }
}
```

4.4.5 SDO

4.4.5.1 Vue d'ensemble

```
SDO ::= SEQUENCE {
   SequenceLayer
   CommandLayer
}
```

4.4.5.2 SequenceLayer

```
SequenceLayer ::= SEQUENCE {
   SequenceFlags     Bitstring {
      rcon_bit1                        (0)      — 0: aucune connexion; 1: initialisation; 2: connexion valide
      rcon_bit2                        (1)          3: réponse d'erreur (réémission demandée)
      rsnr_bit1                        (2)      — 0..63
      rsnr_bit2                        (3)
      rsnr_bit3                        (4)
      rsnr_bit4                        (5)
      rsnr_bit5                        (6)
      rsnr_bit6                        (7)
      scon_bit1                        (8)      — 0: aucune connexion; 1: initialisation; 2: connexion valide
      scon_bit2                        (9)          3: connexion valide avec demande d'acquittement
      ssnr_bit1                        (10)     — 0..63
      ssnr_bit2                        (11)
      ssnr_bit3                        (12)
      ssnr_bit4                        (13)
      ssnr_bit5                        (14)
      ssnr_bit6     (15)
   }
   reserved16
}
```

4.4.5.3 CommandLayer

```
CommandLayer ::= SEQUENCE {
   SDOCommandHeader          SEQUENCE {
      reserved8
      invoke-ID
      segmentation_abort_response
      command-ID
      segment-size
      reserved16
   }
```

```
   SDO-command      CHOICE {
      write-by-index-request              [1h] IMPLICIT WriteByIndex-RequestPDU
      read-by-index-request               [2h] IMPLICIT ReadByIndex-RequestPDU
      write-all-by-index-request          [3h] IMPLICIT WriteAllByIndex-RequestPDU
      read-all-by-index-request           [4h] IMPLICIT ReadAllByIndex-RequestPDU
      write-multiple-by-index-request     [31h] IMPLICIT WriteMultipleByIndex-RequestPDU
      read-multiple-by-index-request      [32h] IMPLICIT ReadMultipleByIndex-RequestPDU
      abort                               IMPLICIT AbortPDU
      write-by-index-response             [1h] IMPLICIT WriteByIndex-ResponsePDU
      read-by-index-response              [2h] IMPLICIT ReadByIndex-ResponsePDU
      write-all-by-index-response         [3h] IMPLICIT WriteAllByIndex-ResponsePDU
      read-all-by-index-response          [4h] IMPLICIT ReadAllByIndex-ResponsePDU
      write-multiple-by-index-response    [31h] IMPLICIT WriteMultipleByIndex-ResponsePDU
      read-multiple-by-index-response     [32h] IMPLICIT ReadMultipleByIndex-ResponsePDU
   }
}
```

4.4.5.4 Invoke ID

invoke-ID ::= Unsigned8

4.4.5.5 Segmentation_abort_response

```
segmentation_abort_response ::= BitString {
   reserved                (0)
   reserved                (1)
   reserved                (2)
   reserved                (3)
   segmentation_bit1       (4)     — 0: Transfert accéléré; 1: initier le transfert de segment
   segmentation_bit2       (5)       2: segment; 3: transfert de segment achevé
   abort                   (6)     — 0: transfert ok; 1: arrêter prématurément
   response                (7)     — 0: demande; 1: réponse
}
```

4.4.5.6 Command-ID

command-ID ::= Unsigned8 — Contient un marqueur spécifique à un contexte pour "SDO-command" (commande SDO)

4.4.5.7 Segment-size

segment-size::= Unsigned16 — Longueur des données du segment. En comptant à partir du début de la "SDO-command". Plage de valeurs valide: 0...1458

4.4.5.8 Services WriteByIndex

```
WriteByIndex-RequestPDU ::= SEQUENCE {
   data-size               — seulement présent si dans SDOCommandHeader, la segmentation
                             = "initiate"
   index
   sub-index
   reserved8
   payload-data
}

WriteByIndex-ResponsePDU ::= NULL
```

4.4.5.9 Services ReadByIndex

```
ReadByIndex-RequestPDU ::= SEQUENCE {
   index
   sub-index
   reserved16
}

ReadByIndex-ResponsePDU ::= SEQUENCE {
   data-size               — seulement présent si dans SDOCommandHeader, la segmentation =
                             "initiate"
   payload-data
}
```

4.4.5.10 Services WriteAllByIndex

```
WriteAllByIndex-RequestPDU ::= SEQUENCE {
   data-size                    — seulement présent si dans SDOCommandHeader, la segmentation =
                                  "initiate"
   index
   reserved16
   payload-data
}

WriteAllByIndex-ResponsePDU ::= NULL
```

4.4.5.11 Services ReadAllByIndex

```
ReadAllByIndex-RequestPDU ::= SEQUENCE {
   index
   reserved16
}

ReadAllByIndex-ResponsePDU ::= SEQUENCE {
   data-size                    — seulement présent si dans SDOCommandHeader, la segmentation =
                                  "initiate"
   payload-data
}

WriteByName-ResponsePDU ::= NULL
```

4.4.5.12 Services WriteMultipleByIndex

```
WriteMultipleByIndex-RequestPDU ::= SEQUENCE {
   data-size                    — seulement présent si dans SDOCommandHeader, la segmentation =
                                  "initiate"
   offset (k)                   —  Décalage d'octet des prochaines de données de charge utile mis
   index
   sub-index
   padding-length
   payload-data
   offset (m)                   —  Décalage d'octet des prochaines de données de charge utile mis
   index
   sub-index
   padding-length
   payload-data
   ...                          —  (autres demandes d'écriture)
}

WriteMultipleByIndex-ResponsePDU ::= SEQUENCE {
   data-size                    — seulement présent si dans SDOCommandHeader, la segmentation =
                                  "initiate"
   index                        —  seulement présent si l'accès en écriture a échoué
   sub-index
   sub-abort
   sub-abort-code
   index                        —  seulement présent si l'accès en écriture a échoué
   sub-index
   sub-abort
   sub-abort-code
   ...                          —  (autres réponses d'écriture)
}
```

4.4.5.13 Services ReadMultipleByIndex

```
ReadMultipleByIndex-RequestPDU ::= SEQUENCE {
   data-size                          — seulement présent si dans SDOCommandHeader, la segmentation =
                                        "initiate"
   index
   sub-index
   reserved8
   index
   sub-index
   reserved8
   ...                                — (autres demandes de lecture)
}

ReadMultipleByIndex-ResponsePDU ::= SEQUENCE {
   data-size                          — seulement présent si dans SDOCommandHeader, la segmentation =
                                        "initiate"
   offset (k)                         — Décalage des prochaines de données de charge utile mis
   index
   sub-index
   sub-abort -padding-length
   payload-data / sub-abort-code
   offset (m)                         — Décalage des prochaines de données de charge utile mis
   index
   sub-index
   sub-abort -padding-length
   payload-data / sub-abort-code
   ...                                — (autres demandes de lecture)
}
```

4.4.5.14 AbortPDU

```
AbortPDU ::= {
   abort-code     Unsigned32          — voir Article A.1 pour les valeurs valides.
}
```

4.4.5.15 **Data size** (Taille de données)

data-size ::= Unsigned32 — Longueur du bloc de données transféré. En comptant à partir du début de la "SDO-command". Plage de valeurs valide: 0...2^{32}-1.

4.4.5.16 **Index** (Indice)

index ::= Unsigned16 — Spécifie une entrée du dictionnaire d'objets de l'appareil; 0.. 65.535

4.4.5.17 **Sub-index** (Sous-indice)

sub-index ::= Unsigned8 — Spécifie un composant d'une entrée du dictionnaire d'objets de l'appareil; 0..254

4.4.5.18 **Payload-data** (Données de charge utile)

payload-data ::= Any — type et longueur dépendant d'une application;
la longueur de trame totale doit se conformer aux règles Ethernet.

4.4.5.19 **Offset** (Décalage)

offset (i) ::= Unsigned8 — décalage des données spécifiées; *i* est aligné sur 4.

4.4.5.20 **Padding-length** (Longueur de bourrage)

padding-length ::= Unsigned8 — Nombre d'octets de bourrage dans le dernier quadlet (mot de 4 octets) des données de charge utile; codé dans les deux bits de poids faible

4.4.5.21 Sub-abort

sub-abort ::= Unsigned8 — 0: transfert ok; 1: arrêter prématurément; codé dans le bit de poids fort

4.4.5.22 Sub-abort-padding-length

sub-abort-padding-length ::= Unsigned8 — sub-abort (voir 4.4.5.21) et padding-length (voir 4.4.5.20) fusionnés en un seul octet.

4.4.5.23 Sub-abort-code

sub-abort-code ::= Unsigned32 — Valeurs et signification identiquement égales à abort-code, voir Article A.1.

4.4.6 **Sync-response** (Réponse de synchronisation)

4.4.6.1 Vue d'ensemble

Sync-response ::= SEQUENCE {

synchronization-status		Bitstring —	voir 4.4.6.2
latency	Unsigned32	—	latence de PRes en ns
sync-node-number	Unsigned32	—	numéro de nœud reçu le dernier dans SyncRequest/SyncResponse
sync-delay	Unsigned32	—	différence temporelle entre la fin de la réception de SyncRequest et le début de la réception de SyncResponse, en ns
PRes-time	Unsigned32	—	retard temporel entre la réception du PRes en provenance du MN et l'envoi à déclenchement temporel de son propre PRes, en ns

}

NOTE Les éléments énumérés ci-dessus sont parfois récapitulés comme suit: "synchronization-status" à " destination-MAC-address " sont récapitulés sous le terme "sync-status" (statut de synchronisation).

4.4.6.2 Synchronization-status

Synchronization-status ::= Bitstring {

PRes-time-valid	(0)	— Le paramètre PRes-time est valide.
Reserved bit1 à bit30		(1)…(30)
PRes-mode-status	(31)	— L'envoi à déclenchement temporel de PRes est actif.

}

4.4.7 **Manufacturer-specific** (spécifique constructeur)

Ces parties sont réservées à des fins spécifiques à un fabricant. Leur spécification ne s'inscrit pas dans le domaine d'application de la présente norme internationale.

5 Syntaxe de transfert

5.1 Codage des types de données (data types)

5.1.1 Description générale des data types et des règles de codage

Pour qu'ils puissent échanger des données qui aient du sens, le format et la signification de ces données nécessitent d'être connus du producteur et du/des consommateur(s). La présente spécification modélise cette exigence par le biais du concept de data types.

Les règles de codage définissent la représentation des valeurs des data types et la syntaxe de transfert des représentations. Les valeurs sont représentées par des suites de bits. Les suites de bits sont transférées sous la forme de suites d'octets. Pour les data types numériques, le codage est de style little endian (petit-boutiste), comme montré dans le Tableau 3.

5.1.2 Syntaxe de transfert des suites de bits

Dans une optique de transmission, une suite de bits est réordonnée en une suite d'octets. La notation hexadécimale est utilisée pour les octets, conformément à l'ISO/CEI 9899. Soit la suite de bits $b = b_0...b_{n-1}$. Soit k un entier naturel tel que $8(k - 1) < n \leq 8k$. La suite b est convertie en k octets assemblés comme montré dans le Tableau 3. Les bits b_i, $i \geq n$ de l'octet portant le numéro le plus élevé sont des bits sans signification.

Tableau 3 – Syntaxe de transfert des suites de bits

numéro d'octet	1.	2.	k.
	b_7 .. b_0	b_{15} .. b_8	b8k –1 .. b8k -8

L'octet 1 est transmis le premier et l'octet *k le* dernier. Par conséquent, la suite de bits est transférée comme suit sur le réseau (l'ordre de transmission au sein d'un octet est déterminé parl'ISO/CEI 8802-3):

$b_7, b_6, ..., b_0, b_{15}, ..., b_8, ...$

EXEMPLE

Bit 9	...	Bit 0
10b	0001b	1100b
2h	1h	Ch
		= 21Ch

La suite de bits $b = b_0 .. b_9 = 0011\ 1000\ 01_b$ représente un entier de type Unsigned10, de valeur 21Ch, et est transférée sous la forme de deux octets: D'abord 1Ch puis 02h.

5.1.3 Codage d'une valeur Boolean

Les données du data type de base BOOLEAN atteignent les valeurs TRUE ou FALSE.

Les valeurs sont représentées sous forme de suites de bits de longueur 1. La valeur TRUE est représentée par la suite de bits 1, et FALSE par 0.

Un BOOLEAN doit être transféré sur le réseau comme UNSIGNED8 de valeur 1 (TRUE) respectivement 0 (FALSE). Les BOOLEAN en séquence peuvent être condensés en un seul UNSIGNED8. Les suites d'articles de type BOOLEAN et BIT peuvent être condensées en un seul UNSIGNED8.

5.1.4 Codage d'une valeur entière Unsigned (non signée)

Les données du data type de base UNSIGNEDn ont des valeurs entières naturelles. La plage de valeurs est 0, ..., 2^n-1. Les données sont représentées sous forme de suites de bits de longueur n.

La suite de bits

$b = b_0 ... b_{n-1}$

se voit affecter la valeur

$$UNSIGNEDn(b) = b_{n-1}\times 2^{n-1} + \ldots + b_1\times 2^1 + b_0\times 2^0$$

Noter que la suite de bits commence (à gauche) par l'octet de poids faible.

Exemple: La valeur $266_d = 10A_h$ de data type UNSIGNED16 est transférée sous la forme de deux octets à travers le bus, d'abord $0A_h$ puis 01_h.

Le transfert des data types UNSIGNEDn suivants s'effectue comme montré dans le Tableau 4.

Tableau 4 – Syntaxe de transfert du data type UNSIGNEDn

numéro d'octet	0	1	2	3	4	5	6	7
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
UNSIGNED48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
UNSIGNED56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

Il convient que les data types UNSIGNED24, UNSIGNED40, UNSIGNED48 et UNSIGNED56 ne soient pas appliqués par de nouvelles applications.

Les data types UNSIGNEDn de longueur s'écartant des valeurs énumérées ci-dessus peuvent être appliqués par des data types composites seulement.

5.1.5 Codage d'une valeur entière signée

Les données du data type de base INTEGERn ont des valeurs entières. La plage de valeurs s'étend de -2^{n-1} à $2^{n-1}-1$. Les données sont représentées sous forme de suites de bits de longueur n. La suite de bits

$$b = b_0 .. b_{n-1}$$

se voit affecter la valeur

$$INTEGERn(b) = b_{n-2}\times 2^{n-2} + \ldots + b_1\times 2^1 + b_0\times 2^0 \text{ if } b_{n-1} = 0$$

puis, grâce au calcul du complément à deux,

$$INTEGERn(b) = - INTEGERn(\hat{}b) - 1 \quad \text{si } b_{n-1} = 1$$

Noter que la suite de bits commence (à gauche) par le bit de poids faible.

Exemple: La valeur $-266_d = 0xFEF6_{\mathbf{h}}$ de data type Integer16 est transférée sous la forme de deux octets, d'abord 0xF6 puis 0xFE.

Le transfert des data types INTEGERn s'effectue comme spécifié dans le Tableau 5.

Tableau 5 – Syntaxe de transfert du data type INTEGERn

numéro d'octet	0	1	2	3	4	5	6	7
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

Il convient que les data types INTEGER24, INTEGER40, INTEGER48 et INTEGER56 ne soient pas appliqués par de nouvelles applications.

Les data types INTEGERn de longueur s'écartant des valeurs énumérées ci-dessus peuvent être appliqués par des data types composites seulement.

5.1.6 Codage d'une valeur en virgule flottante (Floating point)

Les données des data types de base REAL32 et REAL64 ont des valeurs réelles.

Le data type REAL32 est représenté sous forme de suite de bits de longueur 32. Le codage des valeurs est conforme à l'IEEE 754-1985 pour la virgule flottante de simple précision.

Le data type REAL64 est représenté sous forme de suite de bits de longueur 64. Le codage des valeurs est conforme à l'IEEE 754-1985 pour les nombres en virgule flottante de double précision.

Une suite de bits de longueur 32 soit a une valeur (nombre réel fini non nul ± 0, ± _), soit est un NaN ("not-a-number", c'est-à-dire un «non-nombre»).

La suite de bits

$$b = b_0 \ldots b_{31}$$

reçoit en attribution la valeur (nombre fini non nul)

$$\text{REAL32}(b) = (-1)^S \times 2^{E-127} \times (1+ F)$$

Ici

$S = b_{31}$ est le signe.

$E = b_{30} \times 2^7 + \ldots + b_{23} \times 2^0$, $0 < E < 255$, est l'exposant non biaisé.

$F = 2^{-23} \times (b_{22} \times 2^{22} + \ldots + b_1 \times 2^1 + b_0 \times 2^0)$ est la partie fractionnaire du nombre.

$E = 0$ sert à représenter ± 0. $E = 255$ sert à représenter les infinis et les NaN.

Noter que la suite de bits commence (à gauche) par le bit de poids faible.

5.1.7 Codage d'une valeur Octet String

Le data type OCTET_STRINGlength est défini comme suit; "length" est la longueur de la chaîne d'octets.

ARRAY [length] OF UNSIGNED8 OCTET_STRINGlength

5.1.8 Codage d'une valeur Visible String

Les valeurs VISIBLE_CHAR sont 0_h et la plage 20_h à $7E_h$. Les données sont interprétées comme étant des caractères codées de 7 bits de l'ISO 646-1973(E). "length" est la longueur de la chaîne visible.

UNSIGNED8 VISIBLE_CHAR

ARRAY [length] OF VISIBLE_CHAR VISIBLE_STRINGlength

Il n'y a pas de 0_h nécessaire pour terminer la chaîne.

5.1.9 Codage d'une valeur Unicode String

Le data type UNICODE_STRINGlength est défini ci-dessous; "length" est la longueur de la chaîne unicode.

ARRAY [length] OF UNSIGNED16 UNICODE_STRINGlength

5.1.10 Codage d'une valeur TimeOfDay

Le data type TimeOfDay représente le temps absolu. Il découle de la définition et des règles de codage stipulant que TimeOfDay est représenté sous forme d'une suite de bits de longueur 48.

La composante "ms" est le temps en millisecondes après minuit. La composante "days" est le nombre de jours à partir du 1er janvier 1984.

```
STRUCT OF
   UNSIGNED28      ms,
   VOID4           reserved,
   UNSIGNED16      days
TIME_OF_DAY
```

Le codage est tel que montré à la Figure 1.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	nombre de millisecondes depuis minuit
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	nombre de jours depuis le 1984-01-01
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 1 – Codage d'une valeur TimeOfDay

5.1.11 Codage d'une valeur TimeDifference

Le data type TimeDifference représente une différence temporelle. Il découle de la définition et des règles de codage stipulant que TimeDifference est représenté sous forme d'une suite de bits de longueur 48.

Les différences temporelles sont les sommes des nombres de jours et de millisecondes. La composante "ms" est le nombre de millisecondes. La composante "days" est le nombre de jours.

```
STRUCT OF
   UNSIGNED28      ms,
   VOID4           reserved,
   UNSIGNED16      days
TIME_DIFFERENCE
```

Le codage est tel que montré à la Figure 2.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	millisecondes
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	jours
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

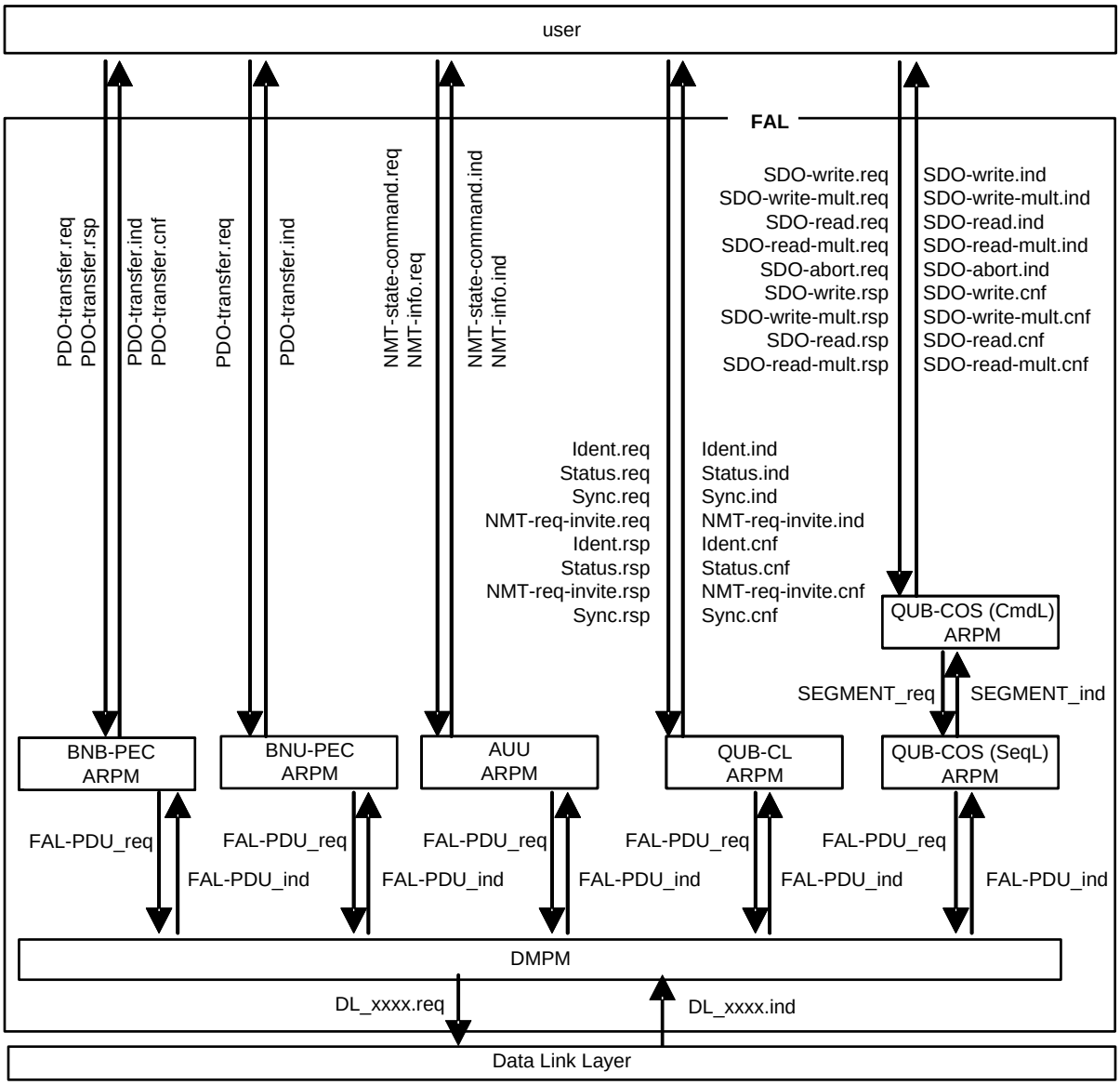
Figure 2 – Codage d'une valeur Time Difference

6 Diagrammes d'états de protocole FAL

Les diagrammes de protocole et de services d'interfaces à la FAL sont spécifiés dans l'Article 6.

NOTE Les diagrammes d'états spécifiés dans l'Article 6 et les machines ARPM définies dans les articles suivants définissent les événements valides pour chacun de ceux-ci. Le traitement des événements non valides relève d'une initiative locale.

Le comportement de la FAL est décrit par les machines protocolaires montrées à la Figure 3. Des ensembles spécifiques de ces machines protocolaires sont définis pour les différents types d'AREP. La Figure 3 montre également les primitives échangées entre les machines protocolaires.



Légende

Anglais	Français
User	Utilisateur
QUB-COS (CmdL) ARPM	ARPM QUB-COS (CmdL)
BNB-PEC ARPM	ARPM BNB-PEC
BNU-PEC ARPM	ARPM BNU-PEC
QUU ARPM	ARPM QUU
QUB-CL ARPM	ARPM QUB-CL
QUB-COS (SeqL) ARPM	ARPM QUB-COS (SeqL)
DMPM	DMPM
Data Link Layer	Couche Liaison de données

Figure 3 – Primitives échangées entre machines protocolaires

7 Diagramme d'états de contexte AP

Aucun diagramme d'états d'AP-Context n'est défini pour ce protocole.

8 Machine protocolaire de services FAL

Aucun diagramme d'états de protocole de services FAL n'est défini pour ce protocole.

9 Machine protocolaire d'AR

9.1 Machine ARPM BNB-PEC (Buffered-network-scheduled bi-directional pre-established connection, connexion préétablie bidirectionnelle programmée par le réseau et mise en tampon)

9.1.1 Définitions des primitives BNB-PEC

9.1.1.1 Primitives échangées entre l'ARPM BNB-PEC et l'utilisateur

Le Tableau 6 et le Tableau 7 énumèrent les primitives échangées entre l'ARPM et l'utilisateur.

Tableau 6 – Primitives émises par l'utilisateur vers l'ARPM BNB-PEC

Nom de primitive	Source	Paramètres associés	Fonctions
PDO-transfer.req	Utilisateur	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.
PDO-transfer.rsp	Utilisateur	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Tableau 7 – Primitives émises par l'ARPM BNB-PEC vers l'utilisateur

Nom de primitive	Source	Paramètres associés	Fonctions
PDO-transfer.ind	ARPM	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.
PDO-transfer.cnf	ARPM	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.

9.1.1.2 Paramètres des primitives

Les paramètres des primitives sont décrits dans la CEI 61158-5-13.

9.1.2 Mapping à la DLL de la classe BNB-PEC

9.1.2.1 Modèle formel

Le Paragraphe 9.1.2 décrit le mapping de la classe d'AREP BNB-PEC à la couche liaison de données de Type 13 définie dans la CEI 61158-3-13 et dans la CEI 61158-4-13. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou seront définis dans la norme

relative à la couche liaison de données; il définit plutôt comment ils sont utilisés par cette classe d'AR.

NOTE Le moyen de configurer et surveiller les valeurs de ces attributs ne s'inscrit pas dans le domaine d'application de la présente Norme internationale.

Les attributs de mapping à la DLL et leurs valeurs autorisées ainsi que les services de DLL utilisés avec la classe AREP BNB-PEC sont définis en 9.1.2.

CLASS:	**Type 13 BNB-PEC**
PARENT CLASS:	**Buffered network-scheduled bi-directional pre-established connection AREP**

ATTRIBUTES:

1	(m)	KeyAttribute («Attribut-clé»):	LocalDlcepAddress
2	(m)	Attribut:	RemoteDlcepAddress

SERVICES de DLL:

1	(m)	OpsService:	DL-PDO

9.1.2.2 Attributs

LocalDlcepAddress
Cet attribut spécifie l'adresse de DLCEP locale et identifie le DLCEP. La valeur de cet attribut est utilisée comme étant le paramètre “DLCEP-address” de la DLL.

RemoteDlcepAddress
Cet attribut spécifie l'adresse de DLCEP distante et identifie le DLCEP.

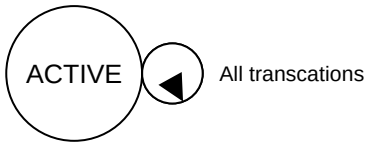
9.1.2.3 Services de DLL

Se référer à la CEI 61158-3-13 pour les descriptions des services de DLL.

9.1.3 Diagramme d'états de l'ARPM BNB-PEC

9.1.3.1 États de l'ARPM BNB-PEC

Le diagramme d'états de l'ARPM BNB-PEC ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 4).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 4 – Diagramme de transition d'états de l'ARPM BNB-PEC

9.1.3.2 Table d'états de l'ARPM BNB-PEC

Le Tableau 8 et le Tableau 9 définissent le diagramme d’états de l'ARPP BNB-PEC.

Tableau 8 – Table d'états de l'ARPM BNB-PEC – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	ACTIVE	PDO-transfer.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc1"
		data := PDO-transfer.req)
		}
			ACTIVE

S2	ACTIVE	PDO-transfer.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc2"
		data := PDO-transfer.rsp)
		}
			ACTIVE

NOTE La transaction S1 est exécutée par le MN seulement, la transaction S2 est exécutée par le CN adressé seulement.			

Tableau 9 – Table d'états de l'ARPM BNB-PEC – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	FAL-PDU_ind && message-type = "Isoc1" ⇒ PDO-transfer.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && message-type = "Isoc2" ⇒ PDO-transfer.cnf	ACTIVE
NOTE La transaction R1 est exécutée par les CN seulement, la transaction R2 est exécutée par MN et peut être exécutée par les CN selon leur configuration.			

9.1.3.3 Fonctions utilisées par l'ARPM BNB-PEC

La réception d'une primitive FAL-PDU_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Le Tableau 10 définit l'autre fonction utilisée par ce diagramme d'états.

Tableau 10 – Fonction BuildFAL-PDU

Nom	BuildFAL-PDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
message-type (type de message) data (données) additional information (informations complémentaires)		DLSDU	
Function (Fonction)			
Construit une PDU de la FAL à partir des paramètres donnés comme variables d'entrée.			

9.2 Machine ARPM BNU-PEC (Buffered-network-scheduled uni-directional pre-established connection, connexion préétablie unidirectionnelle programmée par le réseau et mise en tampon)

9.2.1 Définitions des primitives BNU-PEC

9.2.1.1 Primitives échangées entre l'ARPM BNU-PEC et l'utilisateur

Le Tableau 11 et le Tableau 12 énumèrent les primitives échangées entre l'ARPM et l'utilisateur.

Tableau 11 – Primitives émises par l'utilisateur vers l'ARPM BNU-PEC

Nom de primitive	Source	Paramètres associés	Fonctions
PDO-transfer.req	Utilisa-teur	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Tableau 12 – Primitives émises par l'ARPM BNU-PEC vers l'utilisateur

Nom de primitive	Source	Paramètres associés	Fonctions
PDO-transfer.ind	ARPM	AREP PDO PDO-version	Se référer aux définitions des données de services dans la CEI 61158-5-13.

9.2.1.2 Paramètres des primitives

Les paramètres des primitives sont décrits dans la CEI 61158-5-13.

9.2.2 Mapping à la DLL de la classe BNU-PEC

9.2.2.1 Modèle formel

Le Paragraphe 9.2.2 décrit le mapping de la classe d'AREP BNU à la couche liaison de données de Type 13 définie dans la CEI 61158-3-13 et dans la CEI 61158-4-13. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou seront définis dans la norme relative à la couche liaison de données; il définit plutôt comment ils sont utilisés par cette classe d'AR.

NOTE Le moyen de configurer et surveiller les valeurs de ces attributs ne s'inscrit pas dans le domaine d'application de la présente Norme internationale.

Les attributs de mapping à la DLL et leurs valeurs autorisées ainsi que les services de DLL utilisés avec la classe AREP BNU sont définis en 9.2.2.

CLASS: **Type 13 BNU-PEC**

PARENT CLASS: **Buffered network-scheduled uni-directional pre-established connection AREP**

ATTRIBUTES:

1 (m) KeyAttribute: PublisherDlcepAddress

SERVCES de DLL:

1 (m) OpsService: DL-PDO

9.2.2.2 Attributs

PublisherDlcepAddress
Cet attribut spécifie l'adresse de DLCEP d'éditeur et identifie le DLCEP. La valeur de cet attribut est utilisée comme étant le paramètre “DLCEP-address” de la DLL.

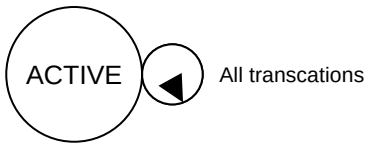
9.2.2.3 Services de DLL

Se référer à la CEI 61158-3-13 pour les descriptions des services de DLL.

9.2.3 Diagramme d'états de l'ARPM BNU-PEC

9.2.3.1 États de l'ARPM BNU-PEC

Le diagramme d'états de l'ARPM BNU-PEC ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 5).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 5 – Diagramme de transition d'états de l'ARPM BNU-PEC

9.2.3.2 Table d'états de l'ARPM BNU-PEC

Le Tableau 13 et le Tableau 14 définissent le diagramme d'états de l'ARPP BNU-PEC.

Tableau 13 – Table d'états de l'ARPM BNU-PEC – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	ACTIVE	PDO-transfer.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type :="Isoc2"
		data := PDO-transfer.req)
		}
			ACTIVE

NOTE	Cette transaction est exécutée par le MN seulement.		

Tableau 14 – Table d'états de l'ARPM BNU-PEC – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	FAL-PDU_ind && message-type = "Isoc2" ⇒ PDO-transfer.ind	ACTIVE
NOTE	Cette transaction est exécutée par les CN seulement.		

9.2.3.3 Fonctions utilisées par l'ARPM BNU

La réception d'une primitive FAL-PDU_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Le Tableau 15 définit l'autre fonction utilisée par ce diagramme d'états.

Tableau 15 – Fonction BuildFAL-PDU

Nom	BuildFAL-PDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
message-type data additional information		DLSDU	
Fonction			
Construit une PDU de la FAL à partir des paramètres donnés comme variables d'entrée.			

9.3 ARPM QUU (Queued user-triggered uni-directional, unidirectionnelle déclenchée par l'utilisateur et mise en tampon)

9.3.1 Définitions des primitives QUU

9.3.1.1 Primitives échangées entre l'ARPM QUU et l'utilisateur

Le Tableau 16 et le Tableau 17 énumèrent les primitives échangées entre l'ARPM et l'utilisateur.

Tableau 16 – Primitives émises par l'utilisateur vers l'ARPM QUU

Nom de primitive	Source	Paramètres associés	Fonctions
NMT-state-command.req	utilisateur	AREP command-ID node-list	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-info.req	utilisateur	AREP publish-node-list publish-time	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Tableau 17 – Primitives émises par l'ARPM QUU vers l'utilisateur

Nom de primitive	Source	Paramètres associés	Fonctions
NMT-state-command.ind	ARPM	AREP command-ID node-list	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-info.ind	ARPM	AREP publish-node-list publish-time	Se référer aux définitions des données de services dans la CEI 61158-5-13.

9.3.1.2 Paramètres des primitives

Les paramètres des primitives sont décrits dans la CEI 61158-5-13.

9.3.2 Mapping à la DLL de la classe d'AREP QUU

9.3.2.1 Modèle formel

Le Paragraphe 9.3.2 décrit le mapping de la classe d'AREP QUU à la couche liaison de données de Type 13 définie dans la CEI 61158-3-13 et dans la CEI 61158-4-13. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou seront définis dans la norme relative à la couche liaison de données; il définit plutôt comment ils sont utilisés par cette classe d'AR.

NOTE Le moyen de configurer et surveiller les valeurs de ces attributs ne s'inscrit pas dans le domaine d'application de la présente Norme internationale.

Les attributs de mapping à la DLL et leurs valeurs autorisées ainsi que les services de DLL utilisés avec la classe AREP QUU sont définis en 9.3.2.

CLASS: **Type 13 QUU**
PARENT CLASS: **Queued User-triggered uni-directional AREP**
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribut:	RemoteDlcepAddress

SERVCES de DLL:

1	(m)	OpsService:	DL-CMD

9.3.2.2 Attributs

LocalDlcepAddress
Cet attribut spécifie l'adresse de DLCEP locale et identifie le DLCEP. La valeur de cet attribut est utilisée comme étant le paramètre “DLCEP-address” de la DLL.

RemoteDlcepAddress

Cet attribut spécifie l'adresse de DLCEP distante et identifie le DLCEP.

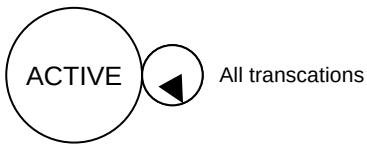
9.3.2.3 Services de DLL

Se référer à la CEI 61158-3-13 pour les descriptions des services de DLL.

9.3.3 Diagramme d'états de l'ARPM QUU

9.3.3.1 États de l'ARPM QUU

Le diagramme d'états de l'ARPM QUU ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 6).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 6 – Diagramme de transition d'états de l'ARPM QUU

9.3.3.2 Table d'états de l'ARPM QUU

Le Tableau 18 et le Tableau 19 définissent le diagramme d’états de l'ARPP QUU.

Tableau 18 – Table d'états de l'ARPM QUU – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	ACTIVE	NMT-state-command.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 4h — "commande NMT"
		data := NMT-state-command.req)
		}
			ACTIVE

S2	ACTIVE	NMT-info.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 4h — "commande NMT"
		data := NMT-info.req)
		}
			ACTIVE

NOTE Ces transactions sont exécutées par le MN seulement.			

Tableau 19 – Table d'états de l'ARPM QUU – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	FAL-PDU_ind && service-id = 4h — "commande NMT" && 20h <= command-ID <= 5Fh — (voir Article A.2) ⇒ NMT-state-command.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && service-id = 4h — "commande NMT" && 80h <= command-ID <= BFh — (voir Article A.2) ⇒ NMT-info.ind	ACTIVE
NOTE Ces transactions sont exécutées par les CN seulement.			

9.3.3.3 Fonctions utilisées par l'ARPM QUU

La réception d'une primitive FAL-PDU_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Le Tableau 20 définit les autres fonctions utilisées par ce diagramme d'états.

Tableau 20 – Fonction BuildFAL-PDU

Nom	BuildFAL-PDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
message-type service-id (identificateur de service) data additional information		DLSDU	
Function (Fonction)			
Construit une PDU de la FAL à partir des paramètres donnés comme variables d'entrée.			

9.4 ARPM QUB-CL (Queued user-triggered bi-directional connectionless, sans connexion bidirectionnelle déclenchée par l'utilisateur et mise en tampon)

9.4.1 Définitions des primitives QUB-CL

9.4.1.1 Primitives échangées entre l'ARPM QUB-CL et l'utilisateur

Le Tableau 21 et le Tableau 22 énumèrent les primitives échangées entre l'ARPM et l'utilisateur.

Tableau 21 – Primitives émises par l'utilisateur vers l'ARPM QUB-CL

Nom de primitive	Source	Paramètres associés	Fonctions
Ident.req	Utilisa-teur	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Status.req	Utilisa-teur	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-req-invite.req	Utilisa-teur	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Ident.rsp	Utilisa-teur	AREP NMT-status fieldbus-version feature-flags cycle-timing Identity verify-configuration application-software-version IP-address host-name vendor-specific-extensions	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Status.rsp	Utilisa-teur	AREP NMT-status static-error error-history	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-req-invite.rsp	Utilisa-teur	AREP command-ID target-node data	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Sync.req	Utilisa-teur	AREP synchronization-control	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Sync.rsp	Utilisa-teur	AREP sync-status	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Tableau 22 – Primitives émises par l'ARPM QUB-CL vers l'utilisateur

Nom de primitive	Source	Paramètres associés	Fonctions
Ident.ind	ARPM	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Status.ind	ARPM	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-req-invite.ind	ARPM	AREP	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Sync.ind	ARPM	AREP synchronization-control	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Ident.cnf	ARPM	AREP NMT-status fieldbus-version feature-flags cycle-timing Identity verify-configuration application-software-version IP-address host-name vendor-specific-extensions	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Status.cnf	ARPM	AREP NMT-status static-error error-history	Se référer aux définitions des données de services dans la CEI 61158-5-13.
NMT-req-invite.cnf	ARPM	AREP command-ID target-node data	Se référer aux définitions des données de services dans la CEI 61158-5-13.
Sync.cnf	ARPM	AREP sync-status	Se référer aux définitions des données de services dans la CEI 61158-5-13.

9.4.1.2 Paramètres des primitives

Les paramètres des primitives sont décrits dans la CEI 61158-5-13.

9.4.2 Mapping à la DLL de la classe d'AREP QUB-CL

9.4.2.1 Modèle formel

Le Paragraphe 9.4.2 décrit le mapping de la classe d'AREP QUB-CL à la couche liaison de données de Type 13 définie dans la CEI 61158-3-13 et dans la CEI 61158-4-13. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou seront définis dans la norme relative à la couche liaison de données; il définit plutôt comment ils sont utilisés par cette classe d'AR.

NOTE Le moyen de configurer et surveiller les valeurs de ces attributs ne s'inscrit pas dans le domaine d'application de la présente norme.

Les attributs de mapping à la DLL et leurs valeurs autorisées ainsi que les services de DLL utilisés avec la classe AREP QUB-CL sont définis en 9.4.2.

CLASS:			**Type 13 QUB-CL**
PARENT CLASS:			**Queued User-triggered Bi-directional connectionless AREP**
ATTRIBUTES:			
1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribut:	RemoteDlcepAddress
SERVCES de DLL:			
1	(m)	OpsService:	DL-IDE
2	(m)	OpsService:	DL-STA
2	(m)	OpsService:	DL-REQ

9.4.2.2 Attributs

LocalDlcepAddress
Cet attribut spécifie l'adresse de DLCEP locale et identifie le DLCEP. La valeur de cet attribut est utilisée comme étant le paramètre “DLCEP-address” de la DLL.

RemoteDlcepAddress
Cet attribut spécifie l'adresse de DLCEP distante et identifie le DLCEP.

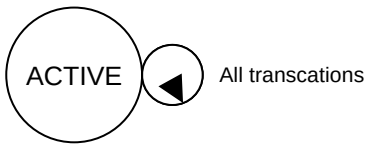
9.4.2.3 Services de DLL

Se référer à la CEI 61158-3-13 pour les descriptions des services de DLL.

9.4.3 Diagramme d'états de l'ARPM QUB-CL

9.4.3.1 États de l'ARPM QUB-CL

Le diagramme d'états de l'ARPM QUB-CL ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 7).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 7 – Diagramme de transition d'états de l'ARPM QUB-CL

9.4.3.2 Table d'états de l'ARPM QUB-CL

Le Tableau 23 et le Tableau 24 définissent le diagramme d’états de l'ARPP QUB-CL.

Tableau 23 – Table d'états de l'ARPM QUB-CL – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	ACTIVE	Ident.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 1h — "ident-request"
		data := Ident.req)
		}
			ACTIVE

S2	ACTIVE	Status.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 2h — "status-request"
		data := Status.req)
		}
			ACTIVE

S3	ACTIVE	NMT-req-invite.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 3h — "NMT-req-invite"
		data := NMT-req-invite.req)
		}
			ACTIVE

S4	ACTIVE	Ident.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 1h — "ident-response"
		data := Ident.rsp)
		}
			ACTIVE

S5	ACTIVE	Status.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 2h — "status-response"
		data := Status.rsp)
		}
			ACTIVE

S6	ACTIVE	NMT-req-invite.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 3h — "NMT-request"
		data := NMT-req-invite.rsp)
		}
			ACTIVE

S7	ACTIVE	Sync.req
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 5h — "Asyn1"
		requested-service-id := 6h — "sync-request"
		data := Sync.req)
		}
			ACTIVE

S8	ACTIVE	Sync.rsp
		⇒
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		message-type := 6h — "Asyn2"
		service-id := 6h — "sync-response"
		data := Sync.rsp)
		}
			ACTIVE

NOTE Les transactions S1 à S3 et S7 sont exécutées par le MN seulement, les transactions S4 à S6 et S8 sont exécutées par les CN seulement.			

Tableau 24 – Table d'états de l'ARPM QUB-CL – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	FAL-PDU_ind && requested-service-id := 1h — "ident-request" ⇒ Ident.ind	ACTIVE
R2	ACTIVE	FAL-PDU_ind && requested-service-id := 2h — "status-request" ⇒ Status.ind	ACTIVE
R3	ACTIVE	FAL-PDU_ind && requested-service-id := 3h — "NMT-req-invite" ⇒ NMT-req-invite.ind	ACTIVE
R4	ACTIVE	FAL-PDU_ind && service-id := 1h — "ident-response" ⇒ Ident.cnf	ACTIVE
R5	ACTIVE	FAL-PDU_ind && service-id := 2h — "status-response" ⇒ Status.cnf	ACTIVE
R6	ACTIVE	FAL-PDU_ind && service-id := 3h — "NMT-request" ⇒ NMT-req-invite.cnf	ACTIVE
R7	ACTIVE	FAL-PDU_ind && requested-service-id := 6h — "sync-request" ⇒ Sync.ind	ACTIVE
R8	ACTIVE	FAL-PDU_ind && service-id := 6h — "sync-response" ⇒ Sync.cnf	ACTIVE
NOTE Les transactions R1 à R3 et R7 sont exécutées par les CN seulement, les transactions R4 à R6 sont exécutées par le MN et peuvent être exécutées par les CN selon leur configuration, la transaction R8 est exécutée par le MN et les CN.			

9.4.3.3 Fonctions utilisées par l'ARPM QUB-CL

La réception d'une primitive FAL-PDU_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Le Tableau 25 définit l'autre fonction utilisée par ce diagramme d'états.

Tableau 25 – Fonction BuildFAL-PDU

Nom	BuildFAL-PDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
message-type service-id data additional information		DLSDU	
Function (Fonction)			
Construit une PDU de la FAL à partir des paramètres donnés comme variables d'entrée.			

9.5 ARPM QUB-COS (Queued user-triggered bi-directional connection-oriented with segmentation, orientée connexion bidirectionnelle déclenchée par l'utilisateur et mise en tampon, avec segmentation)

9.5.1 Vue d'ensemble

L'ARPM QUB-COS est divisée en deux sous-sections, appelées "couches".

- Couche Sequence, QUB-COS (SeqL)
- Couche Command, QUB-COS (CmdL)

La couche de séquence Sequence fournit le service d'une connexion bidirectionnelle fiable qui garantit qu'aucun message ne sera perdu ou dupliqué et que tous les messages arriveront dans l'ordre correct. Il doit y avoir un numéro de séquence pour chaque trame envoyée, et un acquittement pour le numéro de séquence du nœud opposé, ainsi que l'état de la connexion et un acquittement de la connexion.

La couche de commande Command est tenue de décider si un grand bloc de données peut être transféré dans une seule trame (transfert accéléré) ou s'il doit être segmenté en plusieurs trames (transfert segmenté).

9.5.2 Définitions des primitives de QUB-COS (CmdL)

9.5.2.1 Primitives échangées entre l'ARPM QUB-COS (CmdL) et l'utilisateur

Le Tableau 26 et le Tableau 27 énumèrent les primitives échangées entre l'ARPM et l'utilisateur.

Tableau 26 – Primitives émises par l'utilisateur vers l'ARPM QUB-COS (CmdL)

Nom de primitive	Source	Paramètres associés	Fonctions
SDO-write.req	utilisateur	AREP invoke-ID command-ID segment-size data-size OD-identifier payload-data	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write-mult.req	utilisateur	AREP invoke-ID command-ID segment-size OD-identifier (n) payload-data (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read.req	utilisateur	AREP invoke-ID command-ID OD-identifier	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read-mult.req	utilisateur	AREP invoke-ID command-ID OD-identifier (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Nom de primitive	Source	Paramètres associés	Fonctions
SDO-abort.req	utilisateur	AREp invoke-ID error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write.rsp	utilisateur	AREP invoke-ID error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write-mult.rsp	utilisateur	AREP invoke-ID error-info (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read.rsp	utilisateur	AREP invoke-ID segment-size data-size payload-data error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read-mult.rsp	utilisateur	AREP invoke-ID segment-size data-size payload-data / error-info (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Tableau 27 – Primitives émises par l'ARPM QUB-COS (CmdL) vers l'utilisateur

Nom de primitive	Source	Paramètres associés	Fonctions
SDO-write.ind	ARPM	AREP invoke-ID command-ID segment-size data-size OD-identifier payload-data	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write-mult.ind	ARPM	AREP invoke-ID command-ID segment-size data-size OD-identifier (n) payload-data (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read.ind	ARPM	AREP invoke-ID command-ID OD-identifier	Se référer aux définitions des données de services dans la CEI 61158-5-13.

Nom de primitive	Source	Paramètres associés	Fonctions
SDO-read-mult.ind	ARPM	AREP invoke-ID command-ID OD-identifier (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-abort.ind	utilisateur	AREP invoke-ID error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write.cnf	ARPM	AREP invoke-ID error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-write-mult.cnf	ARPM	AREP invoke-ID error-info (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read.cnf	ARPM	AREP invoke-ID segment-size data-size payload-data error-info	Se référer aux définitions des données de services dans la CEI 61158-5-13.
SDO-read-mult.cnf	ARPM	AREP invoke-ID segment-size data-size payload-data / error-info (n)	Se référer aux définitions des données de services dans la CEI 61158-5-13.

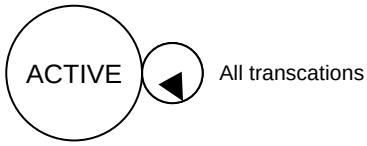
9.5.2.2 Paramètres des primitives

Les paramètres des primitives sont décrits dans la CEI 61158-5-13.

9.5.3 Diagramme d'états de l'ARPM QUB-COS (CmdL)

9.5.3.1 États de l'ARPM QUB-COS (CmdL)

Le diagramme d'états de l'ARPM QUB-COS (CmdL) ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 8).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 8 – Diagramme de transition d'états de l'ARPM QUB-COS (CmdL)

9.5.3.2 Table d'états de l'ARPM QUB-COS (CmdL)

Le Tableau 28 et le Tableau 29 définissent le diagramme d'états de l'ARPM QUB-COS (CmdL).

Tableau 28 – Table d'états de l'ARPM QUB-COS (CmdL) – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant
S1	ACTIVE	SDO-write.req || SDO-write-mult.req && segment-size <= max-segment-size ⇒ response := 0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data)	ACTIVE

S2	ACTIVE	SDO-write.req || SDO-write-mult.req
		&& segment-size > max-segment-size
		⇒
		response := 0
		abort := 0
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(voir les Notes)
			ACTIVE

S3	ACTIVE	SDO-write.rsp ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE

#	État courant	Événement ou condition ⇒ action	État suivant
S4	ACTIVE	SDO-write-mult.rsp && all data successfully written ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE
S5	ACTIVE	SDO-write-mult.rsp && at least one data transfer failed && segment-size <= max-segment-size ⇒ response := 1 abort := 1 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := “null”)	ACTIVE

S6	ACTIVE	SDO-write-mult.rsp
		&& at least one data transfer failed
		&& segment-size > max-segment-size
		⇒
		response := 1
		abort := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(voir les Notes)
			ACTIVE

#	État courant	Événement ou condition ⇒ action	État suivant
S7	ACTIVE	SDO-read.req ⇒ response :=0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "null")	ACTIVE
S8	ACTIVE	SDO-read.rsp && segment-size <= max-segment-size ⇒ response := 1 abort := 0 segmentation := 0 data-size := "null" SEGMENT _req := BuildSegment (header segment-data)	ACTIVE

S9	ACTIVE	SDO-read.rsp
		&& segment-size > max-segment-size
		⇒
		response := 1
		abort := 0
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(voir les Notes)
			ACTIVE

#	État courant	Événement ou condition ⇒ action	État suivant
S10	ACTIVE	SDO-read-mult.req && segment-size <= max-segment-size ⇒ response := 0 abort := 0 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data)	ACTIVE

S11	ACTIVE	SDO-read-mult.req
		&& segment-size > max-segment-size
		⇒
		response := 0
		abort := 0
		segmentation := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(voir les Notes)
			ACTIVE

S12	ACTIVE	SDO-read-mult.rsp
		&& segment-size <= max-segment-size
		⇒
		response := 1
		if all data were successfully read
		abort := 0
		else
		abort := 1
		endif
		segmentation := 0
		data-size := "null"
		SEGMENT_req := BuildSegment (
		header
		segment-data)
			ACTIVE

#	État courant	Événement ou condition ⇒ action	État suivant

S13	ACTIVE	SDO-read-mult.rsp
		&& segment-size > max-segment-size
		⇒
		response := 1
		if all data were successfully read
		abort := 0
		else
		abort := 1
		endif
		segmentation := 1
		for i := 1 to (N := RoundUp(data-size, max-segment-size))
		segmentation := 2
		if (i = 1)
		segmentation :=1
		endif
		if (i = N)
		segmentation := 3
		endif
		SEGMENT_req := BuildSegment (
		header
		segment-data, i)
		endfor
		
		(voir les Notes)
			ACTIVE

S14	ACTIVE	SDO-abort.req ⇒ response := 0 abort := 1 segmentation := 0 data-size := "null" SEGMENT_req := BuildSegment (header segment-data := "error-info")	ACTIVE

NOTE 1 Lorsque la longueur des données dépasse la valeur du paramètre "max-segment-size", le protocole QUB_COS (CmdL) partage les données de charge utile en *N* données de segments.

NOTE 2 Pour chaque donnée de segment, la fonction “BuildSegment” construit un Segment := Header avec les paramètres de Command Layer suivi des données de segments sans le moindre trou.

NOTE 3 Les segments parviennent à l'AREP destinataire dans le même ordre dans lequel ils ont été créés. Ceci est garanti par la couche Sequence. Par conséquent, un numérotage supplémentaire des segments n'est pas nécessaire.

Tableau 29 – Table d'états de l'ARPM QUB-COS (CmdL) – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && (command-ID = 1h — "write-by-index" || command-ID = 3h — "write-all-by-index" ⇒ SDO-write.ind	ACTIVE

R2	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& (command-ID = 1h — "write-by-index"
		|| command-ID = 3h — "write-all-by-index")
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write.ind
		endif
		
		(voir la Note)
			ACTIVE

R3	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 && (command-ID = 1h — "write-by-index" || command-ID = 3h — "write-all-by-index" ⇒ SDO-write.cnf	ACTIVE
R4	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && command-ID = 31h — "write-multiple-by-index" ⇒ SDO-write-mult.ind	ACTIVE

R5	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& command-ID = 31h — "write-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write-mult.ind
		endif
		
		(voir la Note)
			ACTIVE

R6	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 || abort = 1 && segmentation = 0 && command-ID = 31h — "write-multiple-by-index" ⇒ SDO-write-mult.cnf	ACTIVE

R7	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 1
		&& segmentation <> 0
		&& command-ID = 31h — "write-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-write-mult.cnf
		endif
		
		(voir la Note)
			ACTIVE

#	État courant	Événement ou condition ⇒ action	État suivant
R8	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && (command-ID = 2h — "read-by-index" || command-ID = 4h — "read-all-by-index" ⇒ SDO-read.ind	ACTIVE
R9	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 && segmentation = 0 && (command-ID = 2h — "read-by-index" || command-ID = 4h — "read-all-by-index" ⇒ SDO-read.cnf	ACTIVE

R10	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 0
		&& segmentation <> 0
		&& (command-ID = 2h — "read-by-index"
		|| command-ID = 4h — "read-all-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read.cnf
		endif
		
		(see Note)
			ACTIVE

R11	ACTIVE	SEGMENT_ind && response = 0 && abort = 0 && segmentation = 0 && command-ID = 32h — "read-multiple-by-index" ⇒ SDO-read-mult.ind	ACTIVE

R12	ACTIVE	SEGMENT_ind
		&& response = 0
		&& abort = 0
		&& segmentation <> 0
		&& command-ID = 32h — "read-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read-mult.ind
		endif
		
		(voir la Note)
			ACTIVE

R13	ACTIVE	SEGMENT_ind && response = 1 && abort = 0 || abort = 1 && segmentation = 0 && command-ID = 32h — "read-multiple-by-index" ⇒ SDO-read-mult.cnf	ACTIVE

R14	ACTIVE	SEGMENT_ind
		&& response = 1
		&& abort = 0 || abort = 1
		&& segmentation <> 0
		&& command-ID = 32h — "read-multiple-by-index"
		&& AddSegment(segment) = "OK"
		⇒
		if (MoreFollows(segment) = “False”
		SDO-read-mult.cnf
		endif
		
		(see Note)
			ACTIVE

R15	ACTIVE	SEGMENT_ind && abort = 1 ⇒ SDO-abort.ind	ACTIVE

NOTE Lorsque la longueur des données dépasse la valeur du paramètre "max-segment-size", les données de charge utile sont divisées en *N* données de segments. Les segments sont livrés dans l'ordre dans lequel ils ont été créés. Chaque contient un en-tête avec des informations complémentaires. Du côté destinataire de l'AR, la fonction “AddSegment” enlève cet en-tête (y compris le "data-size" dans le premier segment) et accole les données de segment aux précédentes données de segments reçues. Une fois que les *N* données de segment ont été accolées ensemble sans le moindre trou, la fonction “GetintermediatePDU” donne l'identificateur d'entrée d'OD d'origine avec leurs données de charge utile correspondantes.

9.5.3.3 Fonctions utilisées par l'ARPM QUB-COS (CmdL)

La réception d'une primitive SEGMENT_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d’états. Donc, cette fonction implicite n'est pas énumérée séparément.

Les Tableaux, Tableau 30 à Tableau 34, définissent les autres fonctions utilisées par ce diagramme d’états.

Tableau 30 – Fonction BuildSegment

Nom	BuildSegment	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
header := AREP invoke-ID response abort segmentation command-ID segment-size segment-data := data-size (seulement pour le premier segment) données contenant un ou plusieurs identificateurs d'OD et données de charge utile correspondantes, ou information d'erreur (le cas échéant)		SEGMENT_req	
Function (Fonction)			
Construit un SEGMENT à partir des paramètres donnés comme variables d'entrée. Aucun numérotage supplémentaire de segments n'est nécessaire, car l'ordre correct des segments est déjà garanti par la couche séquence (Sequence Layer). Cette fonction ajoute à chaque segment l'en-tête spécifié. Ces en-têtes sont identiques pour tous les segments cohérents, avec l'exception que le paramètre "segmentation" aura la valeur 1 pour le premier segment, la valeur 3 pour le dernier segment et la valeur 2 pour tous les segments intermédiaires. Le paramètre "data-size" ne sera fourni qu'au premier segment.			

Tableau 31 – Fonction RoundUp

Nom	BuildSegment	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
data-size max-segment-size		integer	
Function (Fonction)			
divise "data-size" par "max-segment-size" et arrondit le résultat à l'entier immédiatement supérieur			

Tableau 32 – Fonction MoreFollows

Nom	AddSegment	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
segmentation		Boolean	
Function (Fonction)			
Cette fonction inspecte le paramètre "segmentation" de l'en-tête de SEGMENT_ind. Si sa valeur est 3, la fonction met sa sortie à "False"			

NOTE Les deux fonctions suivantes utilisent une variable persistante "IntermediatePDU", dans laquelle tous les segments des données utilisateur courantes sont stockés par le destinataire de ces données.

Tableau 33 – Fonction AddSegment

Nom	AddSegment	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
header := invoke-ID response abort segmentation command-ID segment-size data-size (premier segment seulement) segment-data		Error code	
Function (Fonction)			
Cette fonction enlève l'en-tête de la SEGMENT_ind reçue et accole les Segment_data aux précédentes Segment_data reçues. Une fois que les *N* Segment_data ont été accolées ensemble sans le moindre trou, la fonction “GetintermediatePDU” donne l'/les identificateur(s) d'entrée d'OD d'origine avec leurs données de charge utile correspondantes.			

Tableau 34 – Fonction GetIntermediatePDU

Nom	GetIntermediatePDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
(aucun)		Identificateur(s) d'entrée d'OD avec ses/leurs données de charge utile correspondantes.	
Function (Fonction)			
Cette fonction retourne les données d'origine, qui avaient été reçues en plusieurs segments. Après cet appel de fonction, la variable IntermediatePDU est réinitialisée et ne contient aucun segment.			

9.5.4 Définitions des primitives de QUB-COS (SeqL)

9.5.4.1 Primitives échangées entre QUB-COS (SeqL) et QUB-COS (CmdL)

Le Tableau 35 montre les primitives émises par QUB-COS (CmdL) vers QUB-COS (SeqL).

Tableau 35 – Primitives émises par QUB-COS (CmdL) vers QUB-COS (SeqL)

Noms des primitives	Source	Paramètres associés	Fonctions
SEGMENT_req	QUB-COS (CmdL)	AREP invoke-ID response abort segmentation command-ID segment-size data-size OD-identifier(s) data	Cette primitive est utilisée pour demander à la QUB-COS (SeqL) de transférer un segment de données. Elle transporte aussi des informations relatives à la segmentation qui seront nécessaires à l'AREP de destination pour reconstruire le message complet.

Le Tableau 36 montre les primitives émises par QUB-COS (SeqL) vers QUB-COS (CmdL).

Tableau 36 – Primitives émises par QUB-COS (SeqL) vers QUB-COS (CmdL)

Noms des primitives	Source	Paramètres associés	Fonctions
SEGMENT_ind	QUB-COS (SeqL)	AREP invoke-ID response abort segmentation command-ID segment-size data-size OD-identifier(s) data	Cette primitive est utilisée pour émettre un segment de donnes de QUB-COS (SeqL) vers QUB-COS (CmdL). En cas de segmentation, elle contient le segment de données lui-même et des informations complémentaires relatives à la segmentation provenant de l'AREP distant pour reconstruire le message complet dans la QUB-COS (CmdL) locale.

9.5.4.2 Paramètres des primitives

Les paramètres utilisés avec les primitives échangées entre la QUB-COS (SeqL) et la QUB-COS (CmdL) sont décrits dans le Tableau 37.

Tableau 37 – Paramètres utilisés avec les primitives échangées entre QUB-COS (SeqL) et QUB-COS (CmdL)

Nom de paramètre	Description
AREP, invoke-ID, error-info	Ces paramètres sont tels que définis dans la CEI 61158-1.
response	Ce paramètre indique si le SEGMENT contient une demande ou une réponse.
abort	Ce paramètre indique que le transfert demandé n'a pas pu être mené à bien.
segmentation	Ce paramètre indique si le SEGMENT est, oui non, une partie intégrante d'un transfert segmenté.
command-ID	Spécifie la commande.
segment-size	En cas de transfert segmenté, ce paramètre contient la longueur des données de segment.

Nom de paramètre	Description
data size	En cas de transfert segmenté, ce paramètre contient la longueur totale du bloc de données transféré.
OD-identifier(s)	Identifie les entrées d'OD devant être affectées.
data	Données de charge utile à destination/en provenance des entrées d'OD.

9.5.5 Mapping à la DLL de la classe d'AREP QUB-COS

9.5.5.1 Modèle formel

Le Paragraphe 9.5.5 décrit le mapping de la classe d'AREP QUB-COS à la couche liaison de données de Type 13 définie dans la CEI 61158-3-13 et dans la CEI 61158-4-13. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou seront définis dans la norme relative à la couche liaison de données; il définit plutôt comment ils sont utilisés par cette classe d'AR.

NOTE Le moyen de configurer et surveiller les valeurs de ces attributs ne s'inscrit pas dans le domaine d'application de la présente Norme internationale.

Les attributs de mapping à la DLL et leurs valeurs autorisées ainsi que les services de DLL utilisés avec la classe AREP QUB-COS sont définis en 9.5.5.

CLASS:			**Type 13 QUB-COS**
PARENT CLASS:			**Queued user-triggered bi directional-connection-oriented AREP**
ATTRIBUTES:			
1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribut:	RemoteDlcepAddress
3	(m)	Attribut:	RecSeqNr
4	(m)	Attribut:	RecCon
5	(m)	Attribut:	SndSeqNr
6	(m)	Attribut:	SndCon
SERVCES de DLL:			
1	(m)	OpsService:	DL-SDO

9.5.5.2 Attributs

LocalDlcepAddress
Cet attribut spécifie l'adresse de DLCEP locale et identifie le DLCEP. La valeur de cet attribut est utilisée comme étant le paramètre “DLCEP-address” de la DLL.

RemoteDlcepAddress
Cet attribut spécifie l'adresse de DLCEP distante et identifie le DLCEP.

RecSeqNr
Cet attribut est un tampon local pour le numéro de séquence de la dernière trame correctement reçue.

RecConNr
Cet attribut est un tampon local pour l'acquittement de code de connexion à l'attention du destinataire.

SndSeqNr
Cet attribut est un tampon local pour le numéro de séquence de la trame.

SndCon
Cet attribut est un tampon local pour le code de connexion de l'expéditeur.

9.5.5.3 Services de DLL

Se référer à la CEI 61158-3-13 pour les descriptions des services de DLL.

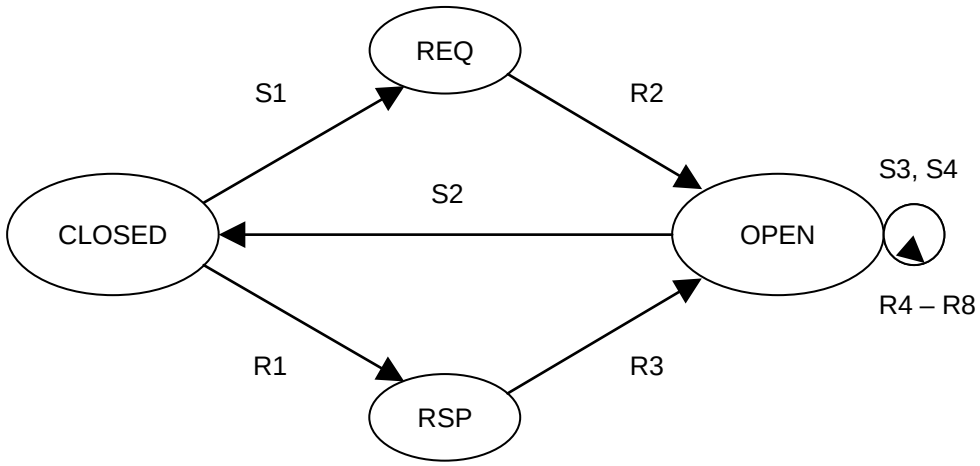
9.5.6 Diagramme d'états de l'ARPM QUB-COS (SeqL)

9.5.6.1 États de l'ARPM QUB-COS (SeqL)

Les états définis et leurs descriptions de l'ARPM QUB-COS (SeqL) sont montrés dans le Tableau 38 et à la Figure 9.

Tableau 38 – États de l'ARPM QUB-COS (SeqL)

État	Description
CLOSED	L'AREP est défini, mais il est incapable d'envoyer ou de recevoir des PDU de FAL. Il peut seulement envoyer ou recevoir des PDU de FAL dont le paramètre scon est mis à 1 pour indiquer le souhait d'initialiser une connexion.
req	L'AREP agit comme un client. Il a envoyé une PDU de FAL dont le paramètre scon est mis à 1 et il attend la réponse de l'AREP serveur distant.
RSP	L'AREP agit comme un client. Il a reçu une PDU de FAL issue de l'AREP client distant avec le paramètre scon mis à 1, a retourné une PDU de FAL de réponse avec les paramètres scon := 1 et rcon := 1 et il attend une réponse de son utilisateur.
OPEN	L'AREP est défini et il est capable d'envoyer ou de recevoir des PDU de FAL.



Légende

Anglais	Français
REQ	«REQUEST» (DEMANDE)
CLOSED	«CLOSED» (FERMÉE)
OPEN	«OPEN» (OUVERTE)
RSP	«RSP» (RÉPONSE)

Figure 9 – Diagramme de transition d'états de l'ARPM QUB-COS (SeqL)

9.5.6.2 Table d'états de l'ARPM QUB-COS (SeqL)

Le Tableau 39 et le Tableau 40 définissent le diagramme d'états de l'ARPM QUB-COS (SeqL). Dans les commentaires de ces tableaux, "Node 1" (c'est-à-dire Nœud 1) indique l'initiateur d'un processus d'établissement de connexion, tandis que "Node 2" (c'est-à-dire Nœud 2) indique le partenaire de connexion adressé.

Tableau 39 – Table d'états de l'ARPM QUB-COS (SeqL) – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	CLOSED	SEGMENT_req — Nœud 1
		&& RecCon = 0, SndCon = 0
		&& RecSeqNr = ?, SndSeqNr = i
		⇒
		SndCon := 1
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			req

S2	OPEN	The connection is not needed any longer
		|| Delay of an expected response > SDO_SequLayerTimeout_U32
		⇒
		RecSeqNr := ?
		RecCon := 0
		SndSeqNr := ?
		SndCon := 0
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
		
		NOTE La nécessité d'une connexion a été évaluée par des moyens ne s'inscrivant pas dans le domaine d'application de la présente norme.
			CLOSED

S3	OPEN	SEGMENT_req — transfert de données régulier, expéditeur
		&& RecCon = 2, SndCon = 2
		⇒
		IncrementCounter(SndSeqNr)
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req)
		}
		
		History(SndSecNr) = AddToHistoryBuffer()
			OPEN

S4	OPEN	SEGMENT_req — historique d'expéditeur plein → demande d'acquittement
		&& RecCon = 2, SndCon = 2
		&& SndSeqNr + 1 = D_SDO_SeqLayerTxHistorySize_U16
		⇒
		RecCon = 2
		RecSeqNr
		SndCon = 3
		IncrementCounter(SndSeqNr)
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req)
		}
			OPEN

Tableau 40 – Table d'états de l'ARPM QUB-COS (SeqL) – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant

R1	CLOSED	FAL-PDU_ind — Nœud 2
		&& rcon = 0, scon = 1
		&& ssnr = i
		&& SndSeqNr = j
		&& RecCon = 0, SndCon = 0
		⇒
		RecSeqNr := ssnr
		RecCon := 1
		SndCon := 1
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			RSP

R2	req	FAL-PDU_ind — Node 1
		&& rcon = 1, scon = 1
		&& ssnr = j, rsnr = i
		&& RecCon = 0, SndCon = 1
		⇒
		RecSeqNr := ssnr
		RecCon := 1
		SndSeqNr := rsnr
		SndCon := 2
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			OPEN

R3	RSP	FAL-PDU_ind — nœud 2
		&& rcon = 1, scon = 2
		&& ssnr = i
		&& RecCon = 1, SndCon = 1
		⇒
		RecSeqNr := ssnr
		RecCon := 2
		SndSeqNr := rsnr
		SndCon := 2
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := "null")
		}
			OPEN

R4	OPEN	FAL-PDU_ind — transfert de données régulier, destinataire && RecCon = 2, SndCon = 2 && scon = 2, rcon = 2 && ssnr = RecSeqNr +1 && rsnr = SndSeqNr ⇒ RecSeqNr := ssnr RecCon := 2 SndSeqNr := rsnr + 1 SndCon := 2 SEGMENT_ind	OPEN

#	État courant	Événement ou condition ⇒ action	État suivant

R5	OPEN	FAL-PDU-ind — Perte de trame détectée
		&& RecCon = 2, SndCon = 2
		&& scon = 2, rcon = 2
		&& ssnr > RecSeqNr +1
		&& rsnr = SndSeqNr
		⇒
		RecCon := 3
		RecSeqNr remains unchanged
		SndCon := 2
		SndSeqNr := rsnr
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req ! if present || "null")
		}
			OPEN

R6	OPEN	FAL-PDU_ind — parachèvement des données après perte de trame && RecCon = 2, SndCon = 2 && scon= 2, rcon =3 && rsnr < SndSeqNr ⇒ for i := rsnr+1 to SndSeqNr FAL-PDU_req := History(i) end for	OPEN

R7	OPEN	FAL-PDU-ind — Réponse à une demande d'acquittement
		&& scon = 3, rcon = 2
		&& ssnr = RecSeqNr +1
		&& rsnr = SndSeqNr
		⇒
		RecCon := 2
		RecSeqNr := ssnr
		SndSeqCon := 2
		SndSeqNr := rsnr
		
		FAL-PDU_req {
		dlsdu := BuildFAL-PDU (
		default-parameters,
		data := SEGMENT_req ! if present || "null")
		}
			OPEN

R8	OPEN	FAL-PDU_ind — Duplication de trames détectée && scon = 2 && ssnr <= RecSeqNr. ⇒ drop FAL-PDU_ind, no further action	OPEN

9.5.6.3 Fonctions utilisées par l'ARPM QUB-COS (SeqL)

La réception d'une primitive FAL-PDU_ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Les tableaux, Tableau 41 à Tableau 43, définissent les autres fonctions utilisées par ce diagramme d'états.

Tableau 41 – Fonction BuildFAL-PDU

Nom	BuildFAL-PDU	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
default-parameters := message-type := 6h ("Asyn2") service-ID = 5h ("SDO") rsnr := RecSeqNr rcon := RecCon ssnr := SndSecNr scon := SndCon		DLSDU	
Function (Fonction)			
Construit une PDU de la FAL à partir des paramètres donnés comme variables d'entrée. Les valeurs par défaut spécifiées sont valides pour l'utilisation de cette fonction dans l'ARPM QUB-COS (SeqL) seulement.			

Tableau 42 – Fonction IncrementCounter

Nom	IncrementCounter	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
identifier			
Function (Fonction)			
Cette fonction incrémente le compteur sélectionné.			

Tableau 43 – Fonction AddToHistoryBuffer

Nom	AddToHistoryBuffer	Utilisée dans	ARPM
Input (Entrée)		Output (Sortie)	
FAL-PDU_req SndSeqNr		History	
Function (Fonction)			
Cette fonction ajoute au tampon de l'historique du diagramme d'états une primitive FAL-PDU_req envoyée. L'entrée est référencée par la SndSeqNr connexe.			

10 Machine protocolaire de mapping à la DLL

10.1 Définitions des primitives

10.1.1 Primitives échangées entre la DMPM et l'ARPM

Le Tableau 44 et le Tableau 45 énumèrent les primitives échangées entre la DMPM et l'ARPM.

Tableau 44 – Primitives émises par l'ARPM vers la DMPM

Nom de primitive	Source	Paramètres associés	Fonctions
FAL-PDU_req	ARPM	DLSDU	Cette primitive est utilisée pour demander à la DMPM de transférer une PDU de FAL ou pour demander un arrêt prématuré sans transférer de PDU de FAL. Elle transmet la PDU de FAL à la DMPM comme une DLSDU.

Tableau 45 – Primitives émises par la DMPM vers l'ARPM

Nom de primitive	Source	Paramètres associés	Fonctions
FAL-PDU_ind	DMPM	DLSDU	Cette primitive est utilisée pour transmettre une PDU de FAL reçue comme unité de donnée de services de couche liaison de données vers l'ARPM désignée.

10.1.2 Paramètres de primitives d'ARPM/DMPM

Le paramètre "dlsdu" contient les données du processus d'application et toutes les informations pertinentes pour le diagramme d'états. Le diagramme d'états de la DMPM est capable d'extraire ces informations.

10.1.3 Primitives échangées entre la couche liaison de données et la DMPM

Le Tableau 46 et le Tableau 47 énumèrent les primitives échangées entre la couche liaison de données et la DMPM.

Tableau 46 – Primitives émises par la DMPM vers la couche liaison de données

Nom de primitive	Source	Paramètres associés
DL-PDO.req	DMPM	dl_dls_user_data
DL-CMD.req	DMPM	dl_ dls_user_data
DL-IDE.req	DMPM	dl_dls_user_data
DL-STA.req	DMPM	dl_dls_user_data
DL-REQ.req	DMPM	dl_dls_user_data
DL-SDO.req	DMPM	dl_dls_user_data

Tableau 47 – Primitives émises par la couche liaison de données vers la DMPM

Nom de primitive	Source	Paramètres associés
DL-PDO.ind	couche liaison de données	dl_dls_user_data
DL-CMD.ind	couche liaison de données	dl_ dls_user_data
DL-IDE.ind	couche liaison de données	dl_dls_user_data
DL-STA.ind	couche liaison de données	dl_dls_user_data
DL-REQ.ind	couche liaison de données	dl_dls_user_data
DL-SDO.ind	couche liaison de données	dl_dls_user_data

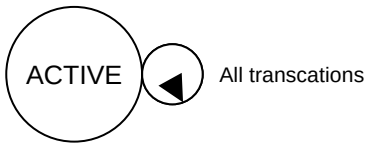
10.1.4 Paramètres des primitives de DMPM/couche liaison de données

Les paramètres utilisés avec les primitives échangées entre la DMPM et la couche liaison de données sont définis dans la définition des services de DLL (voir la CEI 61158-3-13). Ils sont préfixés par “dl_” pour indiquer qu'ils sont utilisés par la FAL.

10.2 Diagramme d'états de la DMPM

10.2.1 États de la DMPM

Le diagramme d'états de la DMPM ne comporte qu'un seul état, appelé "ACTIVE" (voir Figure 10).



Légende

Anglais	Français
ACTIVE	«ACTIVE» (Actif)
All transactions	Toutes transactions

Figure 10 – Diagramme de transition d'états de la DMPM

10.2.2 Table d'états de la DMPM

Le Tableau 48 et le Tableau 49 définissent le diagramme d'états de la DMPM.

Tableau 48 – Table d'états de la DMPM – transactions d'expéditeur

#	État courant	Événement ou condition ⇒ action	État suivant

S1	ACTIVE	FAL-PDU_req
		&& message-type = 3h
		|| message-type = 4h
		⇒
		DL-PDO.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

S2	ACTIVE	FAL-PDU_req
		&& service-ID = 4h
		⇒
		DL-CMD.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

S3	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 1h || service-ID = 1h
		⇒
		DL-IDE.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

S4	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 2h || service-ID = 2h
		⇒
		DL-STA.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

S5	ACTIVE	FAL-PDU_req
		&& requested-service-ID = 3h || service-ID = 3h
		⇒
		DL-REQ.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

S6	ACTIVE	FAL-PDU_req
		&& service-ID = 5h
		⇒
		DL-SDO.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

Tableau 49 – Table d'états de la DMPM – transactions de destinataire

#	État courant	Événement ou condition ⇒ action	État suivant
R1	ACTIVE	DL-xxx.ind ⇒ FAL_PDU_ind	ACTIVE

10.2.3 Fonctions utilisées par la DMPM

La réception d'une primitive DL_Put.ind ou DL_Data.ind est toujours suivie de son décodage afin de dériver ses paramètres pertinents pour le diagramme d'états. Donc, cette fonction implicite n'est pas énumérée séparément.

Annexe A
(normative)

Attributions de valeurs constantes

A.1 Valeurs de abort-code

Les valeurs valides de abort-code et leur signification sont énumérées dans le Tableau A.1.

Tableau A.1 – Valeurs de abort-code

Abort code	Description
0503 0000h	réservé
0504 0000h	Le délai du protocole SDO a expiré.
0504 0001h	Identificateur de commande client/serveur non valide ou inconnu.
0504 0002h	Taille de bloc non valide.
0504 0003h	Numéro de séquence non valide.
0504 0004h	réservé
0504 0005h	Hors mémoire.
0601 0000h	Accès non pris en charge à un objet.
0601 0001h	Tentative de lecture d'un objet en écriture seule.
0601 0002h	Tentative d'écriture d'un objet en lecture seule.
0602 0000h	L'objet n'existe pas dans le dictionnaire d'objets.
0604 0041h	L'objet ne peut pas être mappé au PDO.
0604 0042h	Le nombre et la longueur des objets à mapper dépasseraient la longueur de PDO.
0604 0043h	Incompatibilité générale des paramètres.
0604 0044h	Déclaration de heartbeat non valide
0604 0047h	Incompatibilité interne générale dans l'appareil.
0606 0000h	Échec d'accès en raison d'une erreur matérielle.
0607 0010h	Le data type ne concorde pas, la longueur du paramètre de service ne concorde pas.
0607 0012h	Le data type ne concorde pas, la longueur du paramètre de service est trop grande.
0607 0013h	Le data type ne concorde pas, la longueur du paramètre de service est trop faible.
0609 0011h	le sous-indice n'existe pas.
0609 0030h	Plage de valeurs du paramètre dépassée (seulement pour l'accès en écriture).
0609 0031h	Valeur du paramètre écrite trop élevée.
0609 0032h	Valeur du paramètre écrite trop faible.
0609 0036h	La valeur maximale est inférieure à la valeur minimale.
0800 0000h	Erreur générale
0800 0020h	Les données ne peuvent pas être transférées ou stockées dans l'application.
0800 0021h	Les données ne peuvent pas être transférées ou stockées dans l'application en raison d'une commande locale.
0800 0022h	Les données ne peuvent pas être transférées ou stockées dans l'application en raison du présent état de l'appareil.
0800 0023h	La production dynamique du dictionnaire d'objets échoue ou aucun dictionnaire d'objets n'est présent (par exemple: le dictionnaire d'objets est produit à partir d'un fichier et la production échoue en raison d'une erreur de fichier).
0800 0024h	Jeu de données vide pour l'EDS (Fiches de données électroniques), le DCF (Fichier de

Abort code	Description
	description d'appareil) ou le DCF Concis.

A.2 NMT-command-ID

Les valeurs valides de NMT-command-ID et leur signification sont énumérées dans le Tableau A.2.

Tableau A.2 – Valeurs de NMTCommandID

Nom		Valeur de l'ID
Commandes d'états NMT ordinaires (Plain NMT State)		20h .. 3Fh
	NMTStartNode	21h
	NMTStopNode	22h
	NMTEnterPreOperational2	23h
	NMTEnableReadyToOperate	24h
	NMTResetNode	28h
	NMTResetCommunication	29h
	NMTResetConfiguration	2Ah
	NMTSwReset	2Bh
Commandes d'états NMT étendues (Extended NMT State)		40h .. 5Fh
	NMTStartNodeEx	41h
	NMTStopNodeEx	42h
	NMTEnterPreOperational2Ex	43h
	NMTEnableReadyToOperateEx	44h
	NMTResetNodeEx	48h
	NMTResetCommunicationEx	49h
	NMTResetConfigurationEx	4Ah
	NMTSwResetEx	4Bh
Services informations de NMT		80h .. BFh
	NMTPublishConfiguredNodes	80h
	NMTPublishActiveNodes	90h
	NMTPublishPreOperational1	91h
	NMTPublishPreOperational2	92h
	NMTPublishReadyToOperate	93h
	NMTPublishOperational	94h
	NMTPublishStopped	95h
	NMTPublishNodeStates	96h
	NMTPublishEmergencyNew	A0h
	NMTPublishTime	B0h
NMTInvalidService		FFh

A.3 Constantes de codes d'erreur spécifiques au Type 13

Les constantes des codes d'erreur spécifiques au Type 13 sont énumérées dans le Tableau A.3.

Tableau A.3 – Constantes des codes d'erreur spécifiques au Type 13

Nom	Valeur	Description
	816xh	Erreurs de matériel
E_DLL_BAD_PHYS_MODE	8161h	
E_DLL_COLLISION	8162h	
E_DLL_COLLISION_TH	8163h	
E_DLL_CRC_TH	8164h	
E_DLL_LOSS_OF_LINK	8165h	
E_DLL_MAC_BUFFER	8166h	
	82xxh	Erreurs de protocole
E_DLL_ADDRESS_CONFLICT	8201h	
E_DLL_MULTIPLE_MN	8202h	
	821xh	Erreurs de taille de trame
E_PDO_SHORT_RX	8210h	
E_PDO_MAP_VERS	8211h	
E_NMT_ASND_MTU_DIF	8212h	
E_NMT_ASND_MTU_LIM	8213h	
E_NMT_ASND_TX_LIM	8214h	
	823xh	Erreurs de temporisation
E_NMT_CYCLE_LEN	8231h	
E_DLL_CYCLE_EXCEED	8232h	
E_DLL_CYCLE_EXCEED_TH	8233h	
E_NMT_IDLE_LIM	8234h	
E_DLL_JITTER_TH	8235h	
E_DLL_LATE_PRES_TH	8236h	
E_NMT_PREQ_CN	8237h	
E_NMT_PREQ_LIM	8238h	
E_NMT_PRES_CN	8239h	
E_NMT_PRES_RX_LIM	823Ah	
E_NMT_PRES_TX_LIM	823Bh	
	824xh	Erreurs de trame
E_DLL_INVALID_FORMAT	8241h	
E_DLL_LOSS_PREQ_TH	8242h	
E_DLL_LOSS_PRES_TH	8243h	
E_DLL_LOSS_SOA_TH	8244h	
E_DLL_LOSS_SOC_TH	8245h	
E_DLL_LOSS_STATUSRES_TH	8246h	
	84xxh	Erreurs d'amorçage (BootUp)
E_NMT_BA1	8410h	
E_NMT_BA1_NO_MN_SUPPORT	8411h	
E_NMT_BPO1	8420h	
E_NMT_BPO1_GET_IDENT	8421h	
E_NMT_BPO1_DEVICE_TYPE	8422h	
E_NMT_BPO1_VENDOR_ID	8423h	
E_NMT_BPO1_PRODUCT_CODE	8424h	
E_NMT_BPO1_REVISION_NO	8425h	
E_NMT_BPO1_SERIAL_NO	8426	
E_NMT_BPO1_CONFIGURATION	8428h	

Nom	Valeur	Description
E_NMT_BPO2	8430h	
E_NMT_BRO	8440h	
E_NMT_WRONG_STATE	8480h	

A.4 Node-list

La liste de nœuds (node-list) affecte un bit à chaque ID de nœud de bus de terrain du Type 13. L'attribution des ID de nœud est donnée dans le Tableau A.4.

Tableau A.4 – Format de node-list

	Décalage des bits							
Décalage des octets	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	-
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	39	38	37	36	35	34	33	32
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56
8	71	70	69	68	67	66	65	64
9	79	78	77	76	75	74	73	72
10	87	86	85	84	83	82	81	80
11	95	94	93	92	91	90	89	88
12	103	102	101	100	99	98	97	96
13	111	110	109	108	107	106	105	104
14	119	118	117	116	115	114	113	112
15	127	126	125	124	123	122	121	120
16	135	135	133	132	131	130	129	128
17	143	142	141	140	139	138	137	136
18	151	150	149	148	147	146	145	144
19	159	158	157	156	155	154	153	152
20	167	166	165	164	163	162	161	160
21	175	174	173	172	171	170	169	168
22	183	182	181	180	179	178	177	176
23	191	190	189	188	187	186	185	184
24	199	198	197	196	195	194	193	192
25	207	206	205	204	203	202	201	200
26	215	214	213	212	211	210	209	208
27	223	222	221	220	219	218	217	216

	Décalage des bits							
Décalage des octets	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
28	231	230	229	228	227	226	225	224
29	239	238	237	236	235	234	233	232
30	247	246	245	244	243	242	241	240
31	-	254	253	252	251	250	249	248

Bibliographie

CEI 61158-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61158-6 (toutes les parties), *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6: Spécification du protocole de la couche application*

CEI 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

CEI 61784-2, *Réseaux de communication industriels – Profils – Partie 2: Profils de bus de terrain supplémentaires pour les réseaux en temps réel basés sur l'ISO/CEI 8802-3*

EPSG DS 301 V1.2.0, *Ethernet POWERLINK Communication Profile Specification, Draft Standard Version 1.2.0, EPSG 2013,* disponible à l'adresse http://www.ethernet-powerlink.com

_____

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

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