IEC 61158-6-12

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial communication networks – Fieldbus specifications – Part 6-12: Application layer protocol specification – Type 12 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-12: Spécification du protocole de la couche application – Éléments de type 12

IEC 61158-6-12:2014-08(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 It also gives information on projects, replaced and withdrawn publications.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 61158-6-12

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial communication networks – Fieldbus specifications – Part 6-12: Application layer protocol specification – Type 12 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-12: Spécification du protocole de la couche application – Éléments de type 12

-

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XG

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1762-7

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-12: Application layer protocol specification – Type 12 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 61158-6-12 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision. The main changes with respect to the previous edition are listed below:

- bug fixes;
- editorial improvements;
- support of Explicit Device Identification added in ESM (Clause 6).

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.

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

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

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- withdrawn;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

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

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 6-12: Application layer protocol specification – Type 12 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 12 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 defines in an abstract way the externally visible behavior provided by the different Types of the fieldbus Application Layer in terms of

a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
c) the application context state machine defining the application service behavior visible between communicating application entities; and
d) the application relationship state machines defining the communication behavior visible between communicating application entities; and.

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

a) define the wire-representation of the service primitives defined in IEC 61158-5-12, and
b) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can

send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this standard to provide access to the FAL to control certain aspects of its operation.

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 61158-5-12.

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 subparts of 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.

There is no conformance of equipment to the application layer service definition standard. Instead, 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 61158-3-12, *Industrial communication networks – Fieldbus specifications – Part 3-12: Data-link layer service definition – Type 12 elements*

IEC 61158-5-12, *Industrial communication networks – Fieldbus specifications – Part 5-12: Application layer service definition – Type 12 elements*

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

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

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8802-3, *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 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

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

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

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

IEEE 802.1D, *IEEE standard for Local and metropolitan area networks – Common specifications – Media access control (MAC) Bridges;* available at <http://www.ieee.org>

IEEE 802.1Q, *IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks Bridges;* available at <http://www.ieee.org>

IETF RFC 768, *User Datagram Protocol*; available at <http://www.ietf.org>

IETF RFC 791, *Internet Protocol darpa internet program protocol specification;* available at <http://www.ietf.org>

IETF RFC 826, *An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware;* available at <http://www.ietf.org>

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Reference model terms and definitions

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

3.1.1 correspondent (N)-entities
correspondent AL-entities (N=7) [ISO/IEC 7498-1]

3.1.2 (N)-entity
AL-entity (N=7) [ISO/IEC 7498-1]

3.1.3 (N)-layer
AL-layer (N=7) [ISO/IEC 7498-1]

3.1.4 layer-management [ISO/IEC 7498-1]

3.1.5 peer-entities [ISO/IEC 7498-1]

3.1.6 primitive name [ISO/IEC 7498-3]

3.1.7 AL-protocol [ISO/IEC 7498-1]

3.1.8 AL-protocol-data-unit [ISO/IEC 7498-1]

3.1.9 reset [ISO/IEC 7498-1]

3.1.10 routing [ISO/IEC 7498-1]

3.1.11 segmenting [ISO/IEC 7498-1]

3.1.12 (N)-service
AL-service (N=7) [ISO/IEC 7498-1]

3.1.13 AL-service-data-unit [ISO/IEC 7498-1]

3.1.14 AL-simplex-transmission [ISO/IEC 7498-1]

3.1.15 AL-subsystem [ISO/IEC 7498-1]

3.1.16 systems-management [ISO/IEC 7498-1]

3.1.17 AL-user-data [ISO/IEC 7498-1]

3.2 Service convention terms and definitions

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

3.2.1 acceptor

3.2.2 asymmetrical service

3.2.3 confirm (primitive);
requestor.deliver (primitive)

3.2.4 deliver (primitive)

3.2.5 AL-service-primitive;
primitive

3.2.6 AL-service-provider

3.2.7 AL-service-user

3.2.8 indication (primitive);
acceptor.deliver (primitive)

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

3.2.10 requestor

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

3.2.12 submit (primitive)

3.2.13 symmetrical service

3.3 Application layer definitions

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

3.3.1
application
function or data structure for which data is consumed or produced

3.3.2
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.3.3
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.3.4
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

Note 1 to entry: This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.

3.3.5
attribute
description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes.

3.3.6
basic slave
slave device that supports only physical addressing of data

3.3.7
behavior
indication of how an object responds to particular events

3.3.8
bit
unit of information consisting of a 1 or a 0

Note 1 to entry: This is the smallest data unit that can be transmitted.

3.3.9
channel
representation of a single physical or logical management object of a slave to control conveyance of data

3.3.10
client
1) object which uses the services of another (server) object to perform a task

2) initiator of a message to which a server reacts

3.3.11
clock synchroization
representation of a sequence of interactions to synchronize the clocks of all time receivers by a time master

3.3.12
communication object
component that manages and provides a run time exchange of messages across the network

3.3.13
connection
logical binding between two application objects within the same or different devices

3.3.14
consume
act of receiving data from a provider

3.3.15
consumer
node or sink receiving data from a provider

3.3.16
conveyance path
unidirectional flow of APDUs across an application relationship

3.3.17
cyclic
events which repeat in a regular and repetitive manner

3.3.18
data
generic term used to refer to any information carried over a fieldbus

3.3.19
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.3.20
data type
relation between values and encoding for data of that type according to the definitions of IEC 61131-3

3.3.21
data type object
entry in the object dictionary indicating a data type

3.3.22
default gateway
device with at least two interfaces in two different IP subnets acting as router for a subnet

3.3.23
device
physical entity connected to the fieldbus composed of at least one communication element (the network element) and which may have a control element and/or a final element (transducer, actuator, etc.)

3.3.24
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.3.25
diagnosis information
all data available at the server for maintenance purposes

3.3.26
distributed clocks
method to synchronize slaves and maintain a global time base

3.3.27
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.3.28
error class
general grouping for related error definitions and corresponding error codes

3.3.29
error code
identification of a specific type of error within an error class

3.3.30
event
instance of a change of conditions

3.3.31
fieldbus memory management unit
function that establishes one or several correspondences between logical addresses and physical memory

3.3.32
fieldbus memory management unit entity
single element of the fieldbus memory management unit: one correspondence between a coherent logical address space and a coherent physical memory location

3.3.33
frame
denigrated synonym for DLPDU

3.3.34
full slave
slave device that supports both physical and logical addressing of data

3.3.35
index
address of an object within an application process

3.3.36
interface
shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

3.3.37
little endian
data representation of multi-octet fields where the least significant octet is transmitted first.

3.3.38
master
device that controls the data transfer on the network and initiates the media access of the slaves by sending messages and that constitutes the interface to the control system

3.3.39
mapping
correspondence between two objects in that way that one object is part of the other object

3.3.40
mapping parameters
set of values defining the correspondence between application objects and process data objects

3.3.41
medium
cable, optical fibre or other means by which communication signals are transmitted between two or more points

Note 1 to entry: "media" is the plural of medium.

3.3.42
message
ordered series of octets intended to convey information

Note 1 to entry: Normally used to convey information between peers at the application layer.

3.3.43
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.3.44
node
a) single DL-entity as it appears on one local link

b) end-point of a link in a network or a point at which two or more links meet

[SOURCE derived from IEC 61158-2]

3.3.45
object
abstract representation of a particular component within a device

Note 1 to entry: An object can be

a) an abstract representation of the capabilities of a device. Objects can be composed of any or all of the following components:

   1) data (information which changes with time),

   2) configuration (parameters for behavior),

3) methods (things that can be done using data and configuration);

b) a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior.

3.3.46
object dictionary
data structure addressed by Index and Sub-index that contains description of data type objects, communication objects and application objects

3.3.47
process data
collection of application objects designated to be transferred cyclically or acyclically for the purpose of measurement and control

3.3.48
process data object
structure described by mapping parameters containing one or several process data entities

3.3.49
producer
node or source sending data to one or many consumers

3.3.50
resource
processing or information capability

3.3.51
segment
collection of one real master with one or more slaves

3.3.52
server
object which provides services to another (client) object

3.3.53
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.3.54
slave
DL-entity accessing the medium only after being initiated by the preceding slave or the master

3.3.55
subindex
sub-address of an object within the object dictionary

3.3.56
sync manager
collection of control elements to coordinate access to concurrently used objects

3.3.57
sync manager channel
single control elements to coordinate access to concurrently used objects

3.3.58
switch
MAC bridge as defined in IEEE 802.1D

3.4 Common symbols and abbreviations

AL-	Application layer (as a prefix)
ALE	AL-entity (the local active instance of the application layer)
AL	AL-layer
APDU	AL-protocol-data-unit
ALM	AL-management
ALME	AL-management Entity (the local active instance of AL-management)
ALMS	AL-management service
ALS	AL-service
ALSDU	AL-service-data-unit
DL	Data-link-layer
FIFO	First-in first-out (queuing method)
OSI	Open systems interconnection
PhL	Ph-layer
QoS	Quality of service

3.5 Additional symbols and abbreviations

ASE	Application service element
AR	Application relationship
CAN	Controller Area Network
CiA	CAN in Automation
CoE	CAN application protocol over Type 12 services
CSMA/CD	Carrier sense multiple access with collision detection
DC	Distributed clocks
DNS	Domain name system (server for name resolution in IP networks)
E^2PROM	Electrically erasable programmable read only memory
EoE	Ethernet tunneled over Type 12 services

ESC	Type 12 slave controller
FCS	Frame check sequence
FMMU	Fieldbus memory management unit
FoE	File access with Type 12 services
HDR	Header
ID	Identifier
IETF	Internet engineering rask force
IP	Internet protocol
LAN	Local area network
MAC	Medium access control
OD	Object dictionary
PDI	Physical device internal interface (a set of elements that allows access to DL services from the AL)
PDO	Process data object
RAM	Random access memory
Rx	Receive
SDO	Service data object
SII	slave information interface
SM	Synchronization manager
SoE	Servo drive profile with Type 12 services
SyncM	Synchronization manager
TCP	Transmission control protocol
Tx	Transmit
UDP	User datagram protocol (IETF RFC 768)
VoE	Profile specific services
WKC	Working counter

3.6 Conventions

3.6.1 General concept

The services are specified in IEC 61158-5-12 standard. The service specification defines the services that are provided by the Type 12 DL. The mapping of these services to ISO/IEC 8802-3 is described in this International Standard.

This standard uses the descriptive conventions given in ISO/IEC 10731.

3.6.2 Convention for the encoding of reserved bits and octets

The term "reserved" may be used to describe bits in octets or whole octets. All bits or octets that are reserved should be set to zero at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved bits or octets are checked by a state machine.

The term "reserved" may also be used to indicate that certain values within the range of a parameter are reserved for future extensions. In this case the reserved values should not be used at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved values are check by a state machine.

3.6.3 Conventions for the common codings of specific field octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order according to Figure 1.

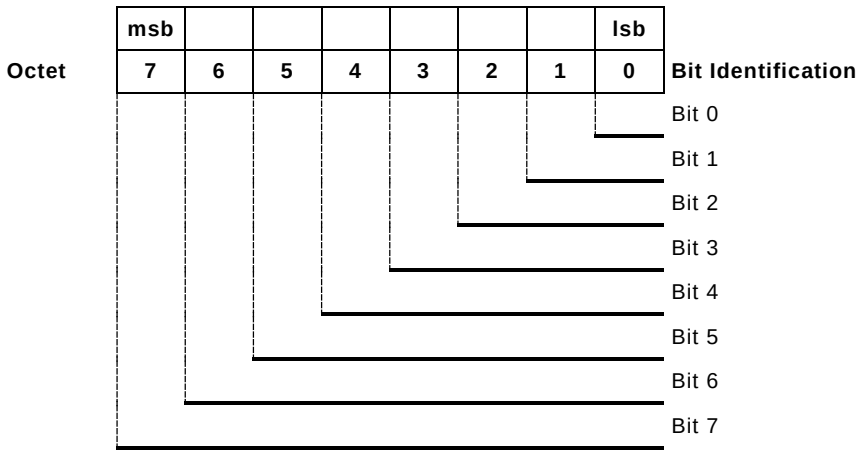


Figure 1 – Common structure of specific fields

Bits may be grouped as group of bits. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bits.

EXAMPLE Description and relation for the specific field octet

Bit 0: reserved.

Bit 1-3: Reason_Code The decimal value 2 for the Reason_Code means general error.

Bit 4-7: shall always set to one.

The octet that is constructed according to the description above looks as follows:

(msb) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(lsb) Bit 0 = 0.

This bit combination has an octet value representation of 0xf4.

3.6.4 Abstract syntax conventions

The AL syntax elements related to PDU structure are described as shown in the example of Table 1.

Frame part denotes the element that will be replaced by this reproduction.

Data field is the name of the elements.

Data Type denotes the type of the terminal symbol.

Value/Description contains the constant value or the meaning of the parameter.

Table 1 – PDU element description example

Frame part	Data Field	Data Type	Value/Description
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Size Indicator	Unsigned1	0x00: size of Data (1..4) unspecified 0x01: size of Data in Data Set Size specified
	Transfer Type	Unsigned1	0x01: Expedited transfer
	Data Set Size	Unsigned2	0x00: 4 Octet Data 0x01: 3 Octet Data 0x02: 2 Octet Data 0x03: 1 Octet Data
	Complete Access	Unsigned1	0x00
	Command Specifier	Unsigned3	0x01: Initiate Download Request

The informational attribute types are described in C language notations (ISO/IEC 9899) as shown in Figure 2. BYTE and WORD are elements of type unsigned char and unsigned short.

```
typedef struct
{
  Unsigned8     Type;
  Unsigned8     Revision;
  Unsigned16    Build;
  Unsigned8     NoOfSuppFmmuChannels;
  Unsigned8     NoOfSuppSyncManChannels;
  Unsigned8     RamSize;
  Unsigned8     Reserved1;
  unsigned      FmmuBitOperationNotSupp:  1;
  unsigned      Reserved2:                7;
  unsigned      Reserved3:                8;
} TDLINFORMATION;
```

Figure 2 – Type description example

The attributes of an object are described in a form as shown in Table 2.

Subindex describes a single element of the object.

Description denotes a name string for this attribute.

Data Type denotes the type of this element.

M/O/C indicates whether the attribute is mandatory (M), optional (O) or depends upon setting of other attributes (C).

Access type shows the access right to this element. R means read access right, W means write access right. It can be extended showing the AL state where the access right applies.

PDO Mapping denotes the possibility to map this attribute to TxPDO or RxPDO or to indicate that this parameter is not mappable.

Value contains the constant value and/or the meaning of the parameter.

Table 2 – Example attribute description

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of entries	UNSIGNED8	M	R	No	4
1	Vendor ID	UNSIGNED32	M	R	No	Assigned uniquely by ETG
2	Product Code	UNSIGNED32	M	R	No	Assigned uniquely by Vendor
3	Revision Number	UNSIGNED32	M	R	No	Assigned uniquely by Vendor
4	Serial Number	UNSIGNED32	M	R	No	assigned uniquely for this device by Vendor

3.6.5 State machine conventions

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes and state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The textual listing of the state transitions is structured as follows, see also Table 3.

– The first row contains the name of the transition.
– The second row defines the current state.
– The third row contains an optional event followed by Conditions starting with a "/" as first line character and finally followed by the Actions starting with a "=>" as first line character.
– The last row contains the next state.

If the event occurs and the conditions are fulfilled the transition fires, i.e. the actions are executed and the next state is entered.

The layout of a Machine description is shown in Table 3. The meaning of the elements of a State Machine Description is shown in Table 4.

Table 3 – State machine description elements

#	Current state	Event /Condition => Action	Next state

Table 4 – Description of state machine elements

Description element	Meaning
Current state Next state	name of the given states.
#	name or number of the state transition.
Event	name or description of the event.
/Condition	boolean expression. The preceding "\" is not part of the condition.
=> Action	list of assignments and service or function invocations. The preceding "=>" is not part of the action.

The conventions used in the state machines are shown in Table 5.

Table 5 – Conventions used in state machines

Convention	Meaning
=	value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.
axx	a parameter name if a is a letter. EXAMPLE Identifier = reason means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx"	indicates fixed visible string. EXAMPLE Identifier = "abc" means value "abc" is assigned to a parameter named 'Identifier.'
nnn	if all elements are digits, the item represents a numerical constant shown in decimal representation.
0xnn	if all elements nn are digits, the item represents a numerical constant shown in hexadecimal representation.
==	a logical condition to indicate an item on the left is equal to an item on the right.
<	a logical condition to indicate an item on the left is less than the item on the right.
>	a logical condition to indicate an item on the left is greater than the item on the right.
!=	a logical condition to indicate an item on the left is not equal to an item on the right.
&&	logical "AND"
||	logical "OR"
!	logical "NOT"
+ - * /	arithmetic operators
;	separator of expressions

Readers are strongly recommended to refer to the subclauses for the attribute definitions, the local functions and the FDL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions and they are used without further explanations.

Further constructs as defined in C language notation (ISO/IEC 9899) can be used to describe conditions and actions.

4 Application layer protocol specification

4.1 Operating principle

Type 12 is a Real Time Ethernet technology that aims to maximize the utilization of the full duplex Ethernet bandwidth. Medium access control employs the master/slave principle, where the master node (typically the control system) sends the Ethernet frames to the slave nodes, which extract data from and insert data into these frames.

From an Ethernet point of view, a Type 12 segment is a single Ethernet device, which receives and sends standard ISO/IEC 8802-3 Ethernet frames. However, this Ethernet device is not limited to a single Ethernet controller with a downstream microprocessor, but may consist of a large number of slave devices. These process the incoming frames directly and extract the relevant user data or insert data and transfer the frame to the next slave device. The last slave device within the segment sends the fully processed frame back, so that it is returned by the first slave device to the master as response frame.

This procedure utilizes the full duplex mode of Ethernet: both communication directions are operated independently. Direct communication without a switch between a master device and a Type 12 segment consisting of one or several slave devices may be established.

Industrial communication systems have to meet different requirements in terms of the data transmission characteristics. Parameter data is transferred acyclically and in large quantities, whereby the timing requirements are relatively non-critical, and the transmission is usually triggered by the control system. Diagnostic data is also transferred acyclically and event-driven, but the timing requirements are more demanding, and the transmission is usually triggered by a peripheral device.

Process data, on the other hand, is typically transferred cyclically with different cycle times. The timing requirements are most stringent for process data communication. Type 12 supports a variety of services and protocols to meet these differing requirements.

4.2 Node reference model

4.2.1 Mapping onto OSI basic reference model

Type 12 is described using the principles, methodology and model of ISO/IEC 7498-1. The OSI model provides a layered approach to communications standards, whereby the layers can be developed and modified independently. The Type 12 specification defines functionality from top to bottom of a full OSI stack, and some functions for the users of the stack. Functions of the intermediate OSI layers, layers 3 – 6, are consolidated into either the Type 12 Data Link layer or the Type 12 Application layer. Likewise, features common to users of the Fieldbus Application Layer may be provided by the Type 12 Application layer to simplify user operation, as noted in Figure 3.

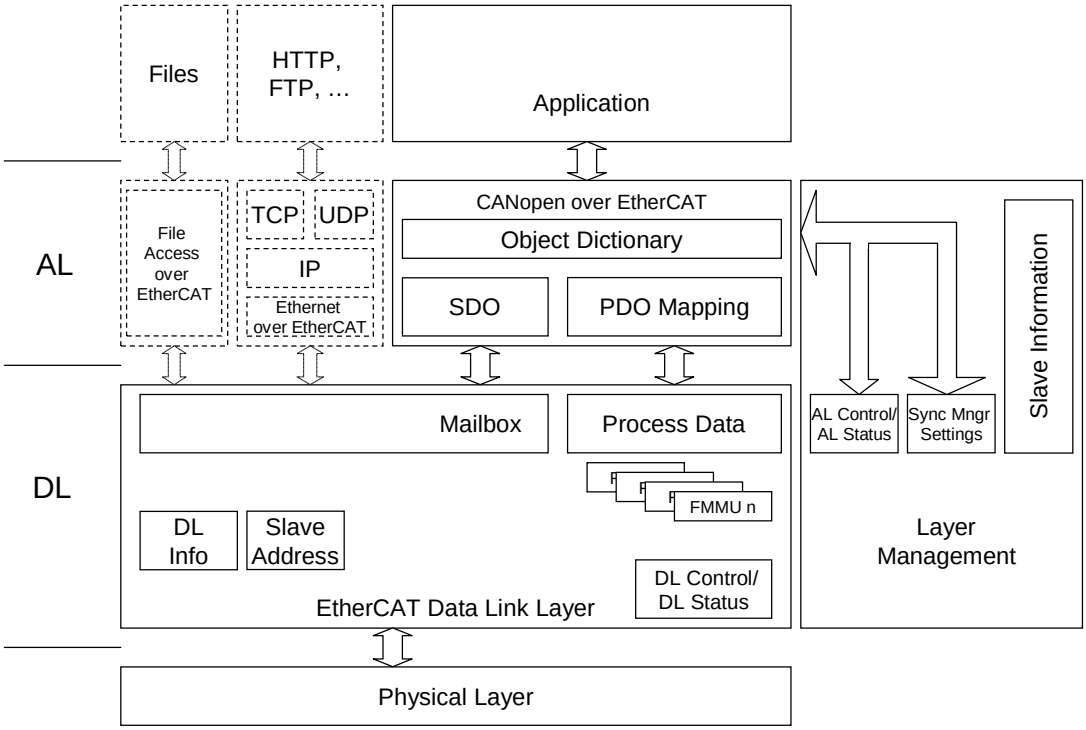


Figure 3 – Slave Node Reference Model

4.2.2 Data Link Layer features

The data link layer provides basic time critical support for data communications among devices. The term “time-critical” is used to describe applications having 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.

The data link layer has the task to compute, compare and generate the frame check sequence and provide communications by extracting data from and/or including data into the Ethernet frame. This is done depending on the data link layer parameters which are stored at pre-defined memory locations. The application data is made available to the application layer in physical memory, either in a mailbox configuration or within the process data section.

Additionally, some data structures in the Data Link layer will be used to allow a coordination of the interaction between master and slave such as AL Control, Status and Event and Sync manager settings.

4.2.3 Application Layer structure

The Application Layer consists of the following elements.

- A real time entity (mandatory)
- An entity that deals with TCP/UDP/IP and related protocols (optional)
- A file access utility (optional)
- A Management unit (mandatory)

The Application Layer uses the services provided by the Type 12 Data Link Layer to convey the Application Layer service data.

5 FAL syntax description

5.1 Coding principles

Application layer uses DL objects as defined in IEC 61158-4-12.

5.2 Data types and encoding rules

5.2.1 General description of data types and encoding rules

The format of this data and its meaning have to be known by the producer and consumer(s) to be able to exchange meaningful data. 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 6.

The data types and encoding rules shall be valid for the DL services and protocols as well as for the AL services and protocols specified. The encoding rules for the Ethernet frame are specified in ISO/IEC 8802-3. The DLSDU of Ethernet is an octet string. The transmission order within octets depends upon MAC and PhL encoding rules.

5.2.2 Encoding of a Boolean value

a) The encoding of a Boolean value shall be primitive. The ContentsOctets shall consist of a single octet.

b) If the Boolean value is FALSE, the ContentsOctets shall be 0 (zero). If the Boolean value is TRUE, the ContentsOctets shall be 0xff.

5.2.3 Encoding of a Time Of Day with and without date indication value

a) The encoding of a Time Of Day with and without date indication value shall be primitive.

b) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 4.

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
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}	since
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	midnight
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}	only with date indication
	msb								

Figure 4 – Encoding of Time of Day value

5.2.4 Encoding of a Time Difference with and without date indication value

a) The encoding of a Time Difference with and without date indication value shall be primitive.

b) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 5.

bits	7	6	5	4	3	2	1	0	
octets									
1	2^{31}	2^{30}	2^{29}	2^{28}	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}	only with date indication
	msb								

Figure 5 – Encoding of Time Difference value

5.2.5 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 6. The bits b_i, $i \geq n$ of the highest numbered octet are do not care bits.

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

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

Table 6 – Transfer Syntax for bit sequences

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

EXAMPLE

Bit 9	...	Bit 0
10b	0001b	1100b
0x2	0x1	0xC
		= 0x21C

The bit sequence b = b_0 .. b_9 = 0011 1000 01_b represents an Unsigned10 with the value 0x21C and is transferred in two octets: First 0x1C and then 0x02.

5.2.6 Encoding of a 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

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

is assigned the value

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

The bit sequence starts on the left with the least significant byte (Octet).

EXAMPLE The value 266 = 0x10A with data type Unsigned16 is transferred in two octets, first 0x0A and then 0x01.

The Unsignedn data types are transferred as specified in Table 7. Unsigned data types as Unsigned1 to Unsigned7 and Unsigned 9 to Unsigned15 will be used too. In this case the next element will start at the first free bit position as denoted in 3.6.3.

Table 7 – Transfer syntax for data type Unsignedn

octet number	1.	2.	3.	4.	5.	6.	7.	8.
Unsigned8	$b_7..b_0$							
Unsigned16	$b_7..b_0$	$b_{15}..b_8$						
Unsigned32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
Unsigned64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

BYTE is used as UNSIGNED8, WORD is used as UNSIGNED16.

5.2.7 Encoding of a Signed Integer value

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

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

and, performing two's complement arithmetic,

$$\text{Integern}(b) = -\ \text{Integern}(\text{^b}) - 1 \qquad \text{if } b_{n-1} = 1$$

NOTE The bit sequence starts on the left with the least significant bit.

EXAMPLE The value –266 = 0xFEF6 with data type Integer16 is transferred in two octets, first 0xF6 and then 0xFE.

The Integern data types are transferred as specified in Table 8. Integer data types as Integer1 to Integer7 and Integer9 to Integer15 will be used too. In this case the next element will start at the first free bit position as denoted in 3.6.3.

Table 8 – Transfer syntax for data type Integern

octet number	1.	2.	3.	4.	5.	6.	7.	8.
Integer8	$b_7..b_0$							
Integer16	$b_7..b_0$	$b_{15}..b_8$						
Integer32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
Integer64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

5.2.8 Encoding of a Floating Point value

Float32 ::= OCTET STRING SIZE (4) -- ISO/IEC/IEEE 60559 Single precision

Float64 ::= OCTET STRING SIZE (8) -- ISO/IEC/IEEE 60559 Double precision

5.2.9 Encoding of a Visible String value

a) The encoding of a variable length VisibleString value shall be primitive.

b) There is no Length field and no termination symbol; the length is encoded implicitly.

c) The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by the second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

5.2.10 Encoding of a Unicode String value

a) The encoding of a variable length UnicodeString value shall be primitive.

b) There is no Length field; the length is encoded implicitly.

c) The ContentsOctets shall be a sequence of unsigned integer. The leftmost string element is encoded in the first unsigned integer, followed by the second unsigned integer, followed by each unsigned integer in turn up to and including the last unsigned integer as rightmost of the ContentsOctets.

5.2.11 Encoding of an Octet String value

a) The encoding of a variable length OctetString value shall be primitive.

b) There is no Length field; the length is encoded implicitly.

c) The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

5.2.12 Encoding of GUID

Data of basic data type GUID (Globally Unique Identifier) is a unique reference number used as an identifier. The value of a GUID is stored as a 128-bit integer.

5.3 AR coding

5.3.1 AL Control Request (Indication)

The attribute types of AL Control Request are described in Figure 6.

```
typedef struct
{
  unsigned          State:            4;
  unsigned          Acknowledge:      1;
  unsigned          Reserved:         3;
  unsigned          ApplSpecific:     8;
} TALCONTROL;
```

Figure 6 – AL Control Request structure

The AL Control Request is mapped to a DL write service to the DL-user control register object and R2 as specified in IEC 61158-3-12. The AL Control Request coding is specified in Table 9.

Table 9 – AL Control Description

Parameter	DL-user Register	Data Type	Value
State	R1	Unsigned4	1: Init 2: Pre-Operational 3: Bootstrap 4: Safe-Operational 8: Operational
Acknowledge	R1	Unsigned1	0: Parameter Change of the AL Status Register will be unchanged 1: Parameter Change of the AL Status Register will be reset
Reserved	R1	Unsigned3	shall be zero
Application Specific	R2	Unsigned8	application specific

5.3.2 AL Control Response (Confirmation)

The AL Control Response is mapped to a DL read service to the DL-user status register object and register R4, R5 and R6 as specified in IEC 61158-3-12. The attribute types of AL Control Response are described in Figure 7.

```
typedef struct
{
  unsigned          State:            4;
  unsigned          Change:           1;
  unsigned          Reserved1:        3;
  unsigned          ApplSpecific:     8;
  unsigned          Reserved2:       16;
  unsigned          AlStatusCode:    16;
} TALSTATUSE;
```

Figure 7 – AL Control Response structure

The AL Control Response coding is specified in Table 10.

Table 10 – AL Control Response

Parameter	DL-user Register	Data Type	Value
State	R3	Unsigned4	1: Init 2: Pre-Operational 3: Bootstrap 4: Safe-Operational 8: Operational
Change	R3	Unsigned1	0: State transition successful 1: State transition not successful
Reserved	R3	Unsigned3	shall be zero
Application Specific	R4	Unsigned8	application specific
Reserved	R5	Unsigned16	shall be zero
Application Specific	R6	Unsigned16	see Table 11

Table 11 – AL Status Codes

Code	Description	Current state (or state change)	Resulting state
0x0000	No error	Any	Current state
0x0001	Unspecified error	Any	Any + E
0x0002	No Memory	Any	Any + E
0x0003	Invalid Device Setup	P -> S	P + E
0x0005	Reserved due to compatibility reasons		
0x0011	Invalid requested state change	I -> S, I -> O, P -> O O -> B, S -> B, P -> B	Current state + E
0x0012	Unknown requested state	Any	Current state + E
0x0013	Bootstrap not supported	I -> B	I + E
0x0014	No valid firmware	I -> P	I + E
0x0015	Invalid mailbox configuration	I -> B	I + E
0x0016	Invalid mailbox configuration	I -> P	I + E
0x0017	Invalid sync manager configuration	P -> S, S -> O	Current state + E
0x0018	No valid inputs available	O, S -> O	S + E
0x0019	No valid outputs	O, S -> O	S + E
0x001A	Synchronization error	O, S -> O	S + E
0x001B	Sync manager watchdog	O, S	S + E
0x001C	Invalid Sync Manager Types	O, S, P -> S	S + E
0x001D	Invalid Output Configuration	O, S, P -> S	S + E
0x001E	Invalid Input Configuration	O, S, P -> S	P + E
0x001F	Invalid Watchdog Configuration	O, S, P -> S	P + E
0x0020	Slave needs cold start	Any	Current state + E
0x0021	Slave needs INIT	B, P, S, O	Current state + E
0x0022	Slave needs PREOP	S, O	S + E, O + E

Code	Description	Current state (or state change)	Resulting state
0x0023	Slave needs SAFEOP	O	O + E
0x0024	Invalid Input Mapping	P -> S	P + E
0x0025	Invalid Output Mapping	P -> S	P + E
0x0026	Inconsistent Settings	P -> S	P + E
0x0027	FreeRun not supported	P -> S	P + E
0x0028	SyncMode not supported	P -> S	P + E
0x0029	FreeRun needs 3Buffer Mode	P -> S	P + E
0x002A	Background Watchdog	S, O	P + E
0x002B	No Valid Inputs and Outputs	O, S -> O	S + E
0x002C	Fatal Sync Error	O	S + E
0x002D	No Sync Error	O, S, P -> S	S + E
0x0030	Invalid DC SYNC Configuration	O, S -> O, P -> S	P + E, S + E
0x0031	Invalid DC Latch Configuration	O, S -> O, P -> S	P + E, S + E
0x0032	PLL Error	O, S -> O	S + E
0x0033	DC Sync IO Error	O, S -> O	S + E
0x0034	DC Sync Timeout Error	O, S -> O	S + E
0x0035	DC Invalid Sync Cycle Time	P -> S	P + E
0x0036	DC Sync0 Cycle Time	P -> S	P + E
0x0037	DC Sync1 Cycle Time	P -> S	P + E
0x0041	MBX_AOE	B, P, S, O	Current state + E
0x0042	MBX_EOE	B, P, S, O	Current state + E
0x0043	MBX_COE	B, P, S, O	Current state + E
0x0044	MBX_FOE	B, P, S, O	Current state + E
0x0045	MBX_SOE	B, P, S, O	Current state + E
0x004F	MBX_VOE	B, P, S, O	Current state + E
0x0050	EEPROM no access	Any	Any + E
0x0051	EEPROM Error	Any	Any + E
0x0060	Slave restarted locally	Any	I
0x0061	Device Identification value updated	P	P + E
0x0062 ...0 x00EF	Reserved		
0x00F0	Application controller available	I	I + E
other codes < 0x8000	reserved		
0x8000 – 0xFFFF	Vendor specific		

5.3.3 AL State Changed

The AL State Changed is mapped to a DL read service to the AL Status and AL Status Code object. The attribute types of AL State Changed are described in Figure 8.

```
typedef struct
{
  unsigned          State:            4;
  unsigned          Change:           1;
  unsigned          Reserved1:        3;
  unsigned          ApplSpecific:     8;
  unsigned          Reserved2:       16;
  unsigned          AlStatusCode:    16;
} TALSTATUSE;
```

Figure 8 – AL State Changed structure

The AL State Changed coding is specified in Table 12.

Table 12 – AL State Changed

Parameter	DL-user Register	Data Type	Value
State	R3	Unsigned4	1: Init 2: Pre-Operational 3: Bootstrap 4: Safe-Operational 8: Operational
Change	R3	Unsigned1	shall be one
Reserved	R3	Unsigned3	shall be zero
Application Specific	R4	Unsigned8	application specific
Reserved	R5	Unsigned16	shall be zero
AL Status Code	R6	Unsigned16	see Table 11

5.3.4 AL AR Attributes

AL AR attributes can be accessed by DL read or write services or by local read write services.

The attribute types of PDI Control are described in Figure 9.

```
typedef struct
{
  unsigned      PDIType:                  8;
  unsigned      StrictALControl:        1;
  unsigned      Reserved:               7;
} TPDICONTROL;
```

Figure 9 – PDI Control type description

The PDI Control coding is specified in Table 13. PDI Control will be loaded from SII at start-up.

Table 13 – PDI Control

Parameter	DL-user Register	Data Type	Access DL	Access local	Value/Description
PDI Type	R7	Unsigned8	R	R	type specific (see IEC 61158-3-12 DL information parameter)
Strict AL Control	Copy	Unsigned1	R	R	0x00: AL Management will be done by an application Controller 0x01: AL Management will be emulated (AL status follows directly AL control)

The PDI Configuration coding is controller specific as shown in Table 14 and set by SII at start-up.

Table 14 – PDI Configuration

Parameter	DL-user Register	Data Type	Access DL	Access local	Value/Description
Application Specific	R8	unsigned8	R	R	applicatgion specific

The attribute types of Sync Configuration are described in Figure 10.

```
typedef struct
{
  unsigned      SignalCondSync0:          2;
  unsigned      EnableSignalSync0:        1;
  unsigned      EnableInterruptSync0:     1;
  unsigned      SignalCondSync1:          2;
  unsigned      EnableSignalSync1:        1;
  unsigned      EnableInterruptSync1:     1;
} TSYNCCFG;
```

Figure 10 – Sync Configuration type description

The Sync Configuration coding is specified in Table 15. Sync Configuration will be loaded from SII at start-up.

Table 15 – Sync Configuration

Parameter	DL-user register	Data Type	Access DL	Access local	Value/Description
Signal Conditioning Sync0	R8	unsigned2	R	R	controller specific
Enable Signal Sync0	R8	unsigned1	R	R	0x00: disable 0x01: enable
Enable Interrupt Sync0	R8	unsigned1	R	R	0x00: disable 0x01: enable
Signal Conditioning Sync1	R8	unsigned2	R	R	controller specific
Enable Signal Sync1	R8	unsigned1	R	R	0x00: disable 0x01: enable
Enable Interrupt Sync1	R8	unsigned1	R	R	0x00: disable 0x01: enable

5.4 SII coding

The Slave Information Interface Area coding is specified in Table 16 and Table 17. Address means a word address (e.g. 0 is first word, 1 is second word).

Table 16 – Slave Information Interface Area

Parameter	Address	Data Type	Value/Description
PDI Control	0x0000	Unsigned16	initialization value for PDI Control register (0x140-0x141)
PDI Configuration	0x0001	Unsigned16	initialization value for PDI Configuration register (0x150-0x151)
SyncImpulseLen	0x0002	Unsigned16	sync Impulse in multiples of 10 ns
PDI Configuration2	0x0003	Unsigned16	initialization value for PDI Configuration register R8 most significant word (0x152-0x153)
Configured Station Alias	0x0004	Unsigned16	alias Address
Reserved	0x0005	BYTE[4]	shall be zero
Checksum	0x0007	Unsigned16	low byte contains remainder of division of word 0 to word 6 as unsigned number divided by the polynomial x^8+x^2+x+1(initial value 0xFF)
Vendor ID	0x0008	Unsigned32	CAN-Object 0x1018, Subindex 1
Product Code	0x000A	Unsigned32	CAN-Object 0x1018, Subindex 2
Revision Number	0x000C	Unsigned32	CAN-Object 0x1018, Subindex 3
Serial Number	0x000E	Unsigned32	CAN-Object 0x1018, Subindex 4
Reserved	0x0010	BYTE[8]	chall be zero
Bootstrap Receive Mailbox Offset	0x0014	Unsigned16	receive Mailbox Offset for Bootstrap state (master to slave)
Bootstrap Receive Mailbox Size	0x0015	Unsigned16	receive Mailbox Size for Bootstrap state (master to slave) Standard Mailbox size and Bootstrap Mailbox can differ. A bigger Mailbox size in Bootstrap mode can be used for optimiziation
Bootstrap Send Mailbox Offset	0x0016	Unsigned16	send Mailbox Offset for Bootstrap state (slave to master)
Bootstrap Send Mailbox Size	0x0017	Unsigned16	send Mailbox Size for Bootstrap state (slave to master) Standard Mailbox size and Bootstrap Mailbox can differ. A bigger Mailbox size in Bootstrap mode can be used for optimiziation
Standard Receive Mailbox Offset	0x0018	Unsigned16	receive Mailbox Offset for Standard state (master to slave)
Standard Receive Mailbox Size	0x0019	Unsigned16	receive Mailbox Size for Standard state (master to slave)
Standard Send Mailbox Offset	0x001A	Unsigned16	send Mailbox Offset for Standard state (slave to master)
Standard Send Mailbox Size	0x001B	Unsigned16	send Mailbox Size for Standard state (slave to master)
Mailbox Protocol	0x001C	Unsigned16	mailbox Protocols Supported as defined in Table 18
Reserved	0x001D	BYTE[66]	shall be zero
Size	0x003E	Unsigned16	size of E2PROM in [KiBit] + 1 NOTE: KiBit means 1024 Bit. NOTE: size = 0 means a EEPROM size of 1 KiBit
Version	0x003F	Unsigned16	this Version is 1

Table 17 – Slave Information Interface Categories

Parameter	Address	Data Type	Value/Description
First Category Header	0x0040	Unsigned15	category Type as defined in Table 19
	0x0040	Unsigned1	vendor Specific
	0x0041	Unsigned16	following Category Word Size x
First Category Data	0x0042	Category dependent	category Data
Second Category Header	0x0042 + x	Unsigned15	category Type as defined in Table 19
	0x0042 + x	Unsigned1	vendor Specific
	0x0043 + x	Unsigned16	following Category Word Size
Second Category Data	0x0044 + x	Category dependent	category Data
...			continuation of the scheme until last category

Table 18 – Mailbox Protocols Supported Types

Protocol	Value	Description
EoE	0x0002	Ethernet over Type 12 (tunnelling of Data Link services)
CoE	0x0004	CAN application protocol over Type 12(access to SDO)
FoE	0x0008	File Access over Type 12
SoE	0x0010	Servo Drive Profile over Type 12
VoE	0x0020	Vendor specific protocol over Type 12

Table 19 – Categories Types

Protocol	Value	Description
NOP	00	no info
Device specific	01 – 09	Device specific categories shall not be overwritten by Master or configuration tool. Might be used for calibration values)
STRINGS	10	string repository for other Categories structure of this category data see Table 20
DataTypes	20	data Types for future use
General	30	general information structure of this category data see Table 21
FMMU	40	FMMUs to be used structure of this category data see Table 22
SyncM	41	Sync Manager Configuration structure of this category data see Table 23
TXPDO	50	TxPDO description structure of this category data see Table 24
RXPDO	51	RxPDO description structure of this category data see Table 24
DC	60	Distributed Clock for future use
Reserved	60-2047	Reserved
Vendor specific	0x0800-0x0FFF	Vendor specific categories
Reserved	0x1000-0xFFFF	Reserved
End	0xffff	

Table 20 – Structure Category String

Parameter	Byte Address	Data Type	Value/Description
nStrings	0x0000	Unsigned8	number of Strings
str1_len	0x0001	Unsigned8	length String1
str_1	0x0002	BYTE [str1_len]	String1 Data
str2_len	0x0002+str1_len	Unsigned8	length String2
str_2	0x0003+str1_len	BYTE [str2_len]	String2 Data
....			
strn_len	0x000z	Unsigned8	length Stringn
str_n	0x000z+1	BYTE [strn_len]	Stringn Data
PAD_Byte	0x000y	BYTE	padding if Category length is odd

Table 21 – Structure Category General

Parameter	Byte Address	Data Type	Value/Description
GroupIdx	0x0000	Unsigned8	Group Information (Vendor specific) - Index to STRINGS
ImgIdx	0x0001	Unsigned8	Image Name (Vendor specific) - Index to STRINGS
OrderIdx	0x0002	Unsigned8	Device Order Number (Vendor specific) - Index to STRINGS
NameIdx	0x0003	Unsigned8	Device Name Information (Vendor specific) - Index to STRINGS
Reserved	0x0004	Unsigned8	reserved
CoE Details	0x0005	Unsigned8	Bit 0: Enable SDO Bit 1: Enable SDO Info Bit 2: Enable PDO Assign Bit 3: Enable PDO Configuration Bit 4: Enable Upload at startup Bit 5: Enable SDO complete acces
FoE Details	0x0006	Unsigned8	Bit 0: Enable Foe
EoE Details	0x0007	Unsigned8	Bit 0: Enable EoE
SoEChannels	0x0008	Unsigned8	reserved
DS402Channels	0x0009	Unsigned8	reserved
SysmanClass	0x000a	Unsigned8	reserved
Flags	0x000b	Unsigned8	Bit 0: Enable SafeOp Bit 1: Enable notLRW
CurrentOnEBus	0x000c	Signed16	EBus Current Consumption in mA, negative Values means feeding in current
PAD_Byte1	0x000b	BYTE[2]	reserved
Physical Port	0x0010	Unsigned16	Description of Physical Ports: 0x00: not use 0x01: MII 0x02: reserved 0x03: EBUS Each port is describe by 4 bits: 3:0: Port 0 7:4: Port 1 11:8: Port 2

Parameter	Byte Address	Data Type	Value/Description
			15:9: Port3
PAD_Byte2	0x0012	BYTE[14]	reserved

Table 22 – Structure Category FMMU

Parameter	Byte Address	Data Type	Value/Description
FMMU0	0x0000	Unsigned8	0x00: FMMU0 not used 0x01: FMMU0 used for Outputs 0x02: FMMU0 used for Inputs 0x03: FMMU0 used for SyncM Status(Read Mailbox) 0xFF: FMMU0 not used
FMMU1	0x0001	Unsigned8	0x00: FMMU1 not used 0x01: FMMU1 used for Outputs 0x02: FMMU1 used for Inputs 0x03: FMMU1 used for SyncM Status(Read Mailbox) 0xFF: FMMU1 not used
	...		continued if more than 2 FMMU used

Table 23 – Structure Category SyncM for each Element

Parameter	Byte Address	Data Type	Value/Description
Physical Start Address	0x0000	WORD	origin of Data (see Physical Start Address of SyncM)
Length	0x0002	WORD	
Control Register	0x0004	Unsigned8	defines Mode of Operation (see Control Register of SyncM)
Status Register	0x0005	BYTE	don't care
Enable Synch Manager	0x0006	Unsigned8	enable SynchM Bit 0: enable Bit 1: fixed content (info for config tool –SyncMan has fixed content) Bit 2: virtual SyncManager (virtual SyncMan – no hardware resource used) Bit 3: opOnly (SyncMan should be enabled only in OP state) Bit 7:4: reserved
Sync Manager Type	0x0007	BYTE	SyncManager Type 0x00 = not used or unknown 0x01 = used for mailbox out 0x02 = used for mailbox in 0x03 = used for process data outputs 0x04 = used for process data inputs

Table 24 – Structure Category TXPDO and RXPDO for each PDO

Parameter	Address	Data Type	Value/Description
PDO Index	0x0000	Unsigned16	ror RxPDO: 0x1600 to 17FF, ror TxPDO: 0x1A00 to 1BFF
nEntry	0x0002	Unsigned8	number of Entries
SyncM	0x0003	Unsigned8	related Sync Manager
Synchronization	0x0004	Unsigned8	reference to DC Synch
NameIdx	0x0005	Unsigned8	name of the Object - Index to STRINGS
Flags	0x0006	WORD	for future use
Entry 1	0x0008	8 BYTE	repeated for each entry as defined in Table 25
...			
Entry nEntry	nEntry*8	8 BYTE	repeated for each entry as defined in Table 25

Table 25 – Structure PDO Entry

Parameter	Offset within entry structure	Data Type	Value/Description
Entry Index	0x0000	Unsigned16	index of the entry
Subindex	0x0002	Unsigned8	subindex
Entry Name Idx	0x0003	Unsigned8	name of the Entry - Index to STRINGS
Data Type	0x0004	Unsigned8	Data Type of the entry (Index in CoE Object Dictionary)
BitLen	0x0005	Unsigned8	Data Length of the entry
Flags	0x0006	WORD	for future use

5.5 Isochronous PDI coding

The attribute types of Distributed Clock sync and latch are described in Figure 11.

```
typedef struct
{
  BYTE          Reserved1;
  unsigned      CyclicOperationEnable:   1;
  unsigned      SYNC0Activate:           1;
  unsigned      SYNC1Activate:           1;
  unsigned      Reserved2:               5;
  WORD          SYNCPulse;
  BYTE          Reserved5[10];
  unsigned      Interrupt1Status:        1;
  unsigned      Reserved2:               7;
  unsigned      Interrupt2Status:        1;
  unsigned      Reserved3:               7;
  DWORD         CyclicOperationStartTime;
  BYTE          Reserved4[12];
  DWORD         SYNC0CycleTime;
  DWORD         SYNC1CycleTime;
  unsigned      Latch0PosEdge:           1;
  unsigned      Latch0NegEdge:           1;
  unsigned      Reserved5:               14;
  unsigned      Latch1PosEdge:           1;
  unsigned      Latch1NegEdge:           1;
  unsigned      Reserved6:               14;
  BYTE          Reserved7[4];
  unsigned      Latch0PosEvnt:           1;
  unsigned      Latch0NegEvnt:           1;
  unsigned      Reserved8:               6;
  unsigned      Latch1PosEvnt:           1;
  unsigned      Latch1NegEvnt:           1;
  unsigned      Reserved9:               6;
  DWORD         Latch0PosEdgeValue;
  BYTE          Reserveda[4];
  DWORD         Latch0NegEdgeValue;
  BYTE          Reservedb[4];
  DWORD         Latch1PosEdgeValue;
  BYTE          Reservedc[4];
  DWORD         Latch1NegEdgeValue;
  BYTE          Reservedd[4];
} TDCISOCHRON;
```

Figure 11 – Distributed Clock sync and latch type description

The Distributed Clock sync parameter encoding is described in Table 26. The parameters are mapped to DL DC user parameter P1 to P6. The events SYNC0 and SYNC1 are mapped to the DL events.

Table 26 – Distributed Clock sync parameter

Parameter	DC user parameter	Data Type	Access Type Type 12 DL	Access Type PDI	Value/Description
Cyclic Operation Enable	P1	Unsigned1	RW	R	0: disabled 1: enabled
SYNC0 activate	P1	Unsigned1	RW	R	0: deactivated 1: SCNC0 pulse generated
SYNC1 activate	P1	Unsigned1	RW	R	0: deactivated 1: SCNC1 pulse generated
SYNC Pulse	P2	Unsigned16	R	R	Taken from SII
Interrupt 0 Status	P3	Unsigned1	R	R	0: not active 1: active
Reserved	P3	Unsigned7	R	R	
Interrupt 1 Status	P3	Unsigned1	R	R	0: not active 1: active
Reserved	P3	Unsigned7	R	R	
Cyclic Operation Start Time	P4	Unsigned32	RW	R	the interrupt generation will start when the lower 32 bits of the system time will reach this value (in ns)
SYNC0 Cycle Time	P5	DWORD	RW	R	cycle time of SYNC0
SYNC1 Cycle Time	P6	DWORD	RW	R	cycle time of SYNC1

The Distributed Clock latch data encoding is described in Table 27.

Table 27 – Distributed Clock latch data

Parameter	DC user parameter	Data Type	Access Type Type 12 DL	Access Type PDI	Value/Description
Latch0 positive Edge	P7	Unsigned1	RW	R	0: continuous 1: single
Latch0 negative Edge	P7	Unsigned1	RW	R	0: continuous 1: single
Reserved	P7	Unsigned6	R	R	
Latch1 positive Edge	P7	Unsigned1	RW	R	0: continuous 1: single
Latch1 negative Edge	P7	Unsigned1	RW	R	0: continuous 1: single
Reserved	P7	Unsigned6	R	R	
Latch0 positive Event	P8	Unsigned1	RW	R	0: no Event 1: Event stored
Latch0 negative Event	P8	Unsigned1	RW	R	0: no Event 1: Event stored
Reserved	P8	Unsigned6	R	R	
Latch1 positive Event	P8	Unsigned1	RW	R	0: no Event 1: Event stored
Latch1 negative Event	P8	Unsigned1	RW	R	0: no Event 1: Event stored
Reserved	P8	Unsigned6	R	R	
Latch 0 positive Edge Value	P9	DWORD	R	R	Latch0 Value positive Event
Latch 0 negative Edge Value	P10	DWORD	R	R	Latch0 Value negative Event
Latch 1 positive Edge Value	P11	DWORD	R	R	Latch1 Value positive Event
Latch 1 negative Edge Value	P12	DWORD	R	R	Latch1 Value negative Event

5.6 CoE coding

5.6.1 PDU structure

The general attribute types of CoE are described in Figure 12.

```
typedef struct
{
  unsigned          NumberLo:         8;
  unsigned          NumberHi:         1;
  unsigned          Reserved:         3;
  unsigned          Service:          4;
} TCOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  BYTE              Data[MBX_DATA_SIZE-2];
} TCOEMBX;
```

Figure 12 – CoE general structure

The CoE coding is specified in Table 28.

Table 28 – CoE elements

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	depending on the CoE service
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x01: Emergency 0x02: SDO Request 0x03: SDO Response 0x04: TxPDO 0x05: RxPDO 0x06: TxPDO remote request 0x07: RxPDO remote request 0x08: SDO Information

5.6.2 SDO

5.6.2.1 SDO Download Expedited

5.6.2.1.1 SDO Download Expedited Request

The attribute types of SDO Download Expedited Request are described in Figure 13.

```
typedef struct
{
  unsigned          SizeIndicator:    1;
  unsigned          TransferType:     1;
  unsigned          DataSetSize:      2;
  unsigned          CompleteAccess:   1;
  unsigned          Command:          3;
  BYTE              IndexLo;
  BYTE              IndexHi;
  BYTE              SubIndex;
} TINITSDOHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  BYTE              Data[4];
} TINITSDODOWNLOADEXPREQMBX;
```

Figure 13 – SDO Download Expedited Request structure

The SDO Download Expedited Request coding is specified in Table 29.

Table 29 – SDO Download Expedited Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Size Indicator	Unsigned1	0x01: size of Data in Data Set Size specified
	Transfer Type	Unsigned1	0x01: Expedited transfer
	Data Set Size	Unsigned2	0x00: 4 Octet Data 0x01: 3 Octet Data 0x02: 2 Octet Data 0x03: 1 Octet Data
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be downloaded 0x01: complete object will be downloaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x01: Download Request
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one if Complete Access = 0x01
	Data	BYTE[4]	data of the Object

5.6.2.1.2 SDO Download Expedited Response

The attribute types of SDO Download Expedited Response are described in Figure 14.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
} TINITSDODOWNLOADEXPRESMBX;
```

Figure 14 – SDO Download Expedited Response structure

The SDO Download Expedited Response coding is specified in Table 30.

Table 30 – SDO Download Expedited Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x03: SDO Response
SDO	Size Indicator	Unsigned1	0x00
	Transfer Type	Unsigned1	0x00
	Data Set Size	Unsigned2	0x00
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be downloaded 0x01: complete object will be uploaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x03: Download Response
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one if Complete Access = 0x01
	reserved	DWORD	

5.6.2.2 SDO Download Normal

5.6.2.2.1 SDO Download Normal Request

The attribute types of SDO Download Normal Request are described in Figure 15.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             CompleteSize;
  BYTE              Data[MBX_DATA_SIZE-10];
} TINITSDODOWNLOADNORMREQMBX;
```

Figure 15 – SDO Download Normal Request structure

The SDO Download Normal Request coding is specified in Table 31.

Table 31 – SDO Download Normal Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Size Indicator	Unsigned1	0x01
	Transfer Type	Unsigned1	0x00: Normal transfer
	Data Set Size	Unsigned2	0x00
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be downloaded 0x01: complete object will be uploaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x01: Download Request
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one if Complete Access = 0x01
	Complete Size	DWORD	complete Data Size of the Object
	Data	BYTE[n-10]	if ((Length-10) >= Complete Size): Data of the Object if ((Length-10) < Complete Size): First Data part of the Object, Download SDO Segment is following

5.6.2.2.2 SDO Download Normal Response

The attribute types and coding of SDO Download Normal Response are the same as of SDO Download Expedited Response (see 5.6.2.1.2).

5.6.2.3 Download SDO Segment

5.6.2.3.1 Download SDO Segment Request

The attribute types of Download SDO Segment Request are described in Figure 16.

```
typedef struct
{
  unsigned          MoreFollows:      1;
  unsigned          SegDataSize:      3;
  unsigned          Toggle:           1;
  unsigned          Command:          3;
} TSDOSEGHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
  BYTE              Data[MBX_DATA_SIZE-3];
} TDOWNLOADSDOSEGREQMBX;
```

Figure 16 – Download SDO Segment Request structure

The Download SDO Segment Request coding is specified in Table 32.

Table 32 – Download SDO Segment Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	More Follows	Unsigned1	0x00: Download SDO Segment is following 0x01: last Download SDO Segment
	SegData Size	Unsigned3	defines how much of the last 7 data Octets (which always has to be send) contain data (only applicable if Length is 0x0A, otherwise shall be 0x00): 0x00: 7 Octet Data 0x01: 6 Octet Data 0x02: 5 Octet Data 0x03: 4 Octet Data 0x04: 3 Octet Data 0x05: 2 Octet Data 0x06: 1 Octet Data 0x07: 0 Octet Data
	Toggle	Unsigned1	shall toggle with every Download SDO Segment Request, starting with 0x00
	Command specifier	Unsigned3	0x00: Download Segment Request
	Data	BYTE[n-3]	data part of the Object

5.6.2.3.2 Download SDO Segment Response

The attribute types of Download SDO Segment Response are described in Figure 17.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
} TDOWNLOADSDOSEGRESMBX;
```

Figure 17 – Download SDO Segment Response structure

The Download SDO Segment Response coding is specified in Table 33.

Table 33 – Download SDO Segment Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x03: SDO Response
SDO	Reserved	Unsigned4	0x00
	Toggle	Unsigned1	shall be the same as for the corresponding Download SDO Segment Request
	Command Specifier	Unsigned3	0x01: Download Segment Response
	Reserved	BYTE[7]	

5.6.2.4 SDO Upload Expedited

5.6.2.4.1 SDO Upload Expedited Request

The attribute types of SDO Upload Expedited Request are described in Figure 18.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
} TINITSDOUPLOADEXPREQMBX;
```

Figure 18 – SDO Upload Expedited Request structure

The SDO Upload Expedited Request coding is specified in Table 34.

Table 34 – SDO Upload Expedited Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Reserved	Unsigned4	0x00
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be uploaded 0x01: complete object will be uploaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x02: Upload Request
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one, if Complete Access = 0x01
	Reserved	DWORD	

5.6.2.4.2 SDO Upload Expedited Response

The attribute types of SDO Upload Expedited Response are described in Figure 19.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  BYTE              Data[4];
} TINITSDOUPLOADEXPREQMBX;
```

Figure 19 – SDO Upload Expedited Response structure

The SDO Upload Expedited Response coding is specified in Table 35.

Table 35 – SDO Upload Expedited Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x03: SDO Response
SDO	Size Indicator	Unsigned1	0x01: size of Data in Data Set Size specified
	Transfer Type	Unsigned1	0x01: Expedited transfer
	Data Set Size	Unsigned2	0x00: 4 Octet Data 0x01: 3 Octet Data 0x02: 2 Octet Data 0x03: 1 Octet Data
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be uploaded 0x01: complete object will be uploaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x02: Upload Response
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one, if Complete Access = 0x01
	Data	BYTE[4]	data of the Object

5.6.2.5 SDO Upload Normal

5.6.2.5.1 SDO Upload Normal Request

The attribute types and coding of SDO Upload Normal Request are the same as of SDO Upload Expedited Request (see 5.6.2.4.1).

5.6.2.5.2 SDO Upload Normal Response

The attribute types of SDO Upload Normal Response are described in Figure 20.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             CompleteSize;
  BYTE              Data[MBX_DATA_SIZE-10];
} TINITSDOUPLOADNORMRESMBX;
```

Figure 20 – SDO Upload Normal Response structure

The SDO Upload Normal Response coding is specified in Table 36. If the number of octets of the Data parameter is equal or less than 4 the response as specified in 5.6.2.4.2 can be used.

Table 36 – SDO Upload Normal Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x03: SDO Response
SDO	Size Indicator	Unsigned1	0x01
	Transfer Type	Unsigned1	0x00: Normal transfer
	Data Set Size	Unsigned2	0x00
	Complete Access	Unsigned1	0x00: entry addressed with index and subindex will be uploaded 0x01: complete object will be uploaded, subindex shall be zero (subindex 0 included) or one (subindex 0 excluded)
	Command Specifier	Unsigned3	0x02: Upload Response
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object, shall be zero or one, if Complete Access = 0x01
	Complete Size	DWORD	complete Data Size of the Object
	Data	BYTE[n-10]	if ((Length-10) >= Complete Size): Data of the Object if ((Length-10) < Complete Size): First Data part of the Object, Upload SDO Segment is following

5.6.2.6 Upload SDO Segment

5.6.2.6.1 Upload SDO Segment Request

The attribute types of Upload SDO Segment Request are described in Figure 21.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
} TUPLOADSDOSEGREQMBX;
```

Figure 21 – Upload SDO Segment Request structure

The Upload SDO Segment Request coding is specified in Table 37.

Table 37 – Upload SDO Segment Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Reserved	Unsigned4	0x00
	Toggle	Unsigned1	shall toggle with every Upload SDO Segment Request, starting with 0x00
	Command Specifier	Unsigned3	0x03: Upload Segment Request
	Reserved	BYTE[7]	

5.6.2.6.2 Upload SDO Segment Response

The attribute types of Upload SDO Segment Response are described in Figure 22.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
  BYTE              Data[MBX_DATA_SIZE-3];
} TUPLOADSDOSEGRESMBX;
```

Figure 22 – Upload SDO Segment Response structure

The Upload SDO Segment Response coding is specified in Table 38.

Table 38 – Upload SDO Segment Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x03: SDO Response
SDO	More Follows	Unsigned1	0x00: Upload SDO Segment is following 0x01: last Upload SDO Segment
	SegData Size	Unsigned3	defines how much of the last 7 data Octets (which always has to be send) contain data (only applicable if Length is 0x0A, otherwise shall be 0x00): 0x00: 7 Octet Data 0x01: 6 Octet Data 0x02: 5 Octet Data 0x03: 4 Octet Data 0x04: 3 Octet Data 0x05: 2 Octet Data 0x06: 1 Octet Data 0x07: 0 Octet Data
	Toggle	Unsigned1	shall be the same as for the corresponding Upload SDO Segment Request
	Command specifier	Unsigned3	0x00: Upload Segment Response
	Data	BYTE[n-3]	data part of the Object

5.6.2.7 Abort SDO Transfer

5.6.2.7.1 Abort SDO Transfer Request

The attribute types of Abort SDO Transfer Request are described in Figure 23.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             AbortCode;
} TABORTSDOTRANSFERREQMBX;
```

Figure 23 – Abort SDO Transfer Request structure

The Abort SDO Transfer Request coding is specified in Table 39.

Table 39 – Abort SDO Transfer Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Size Indicator	Unsigned1	0x00
	Transfer Type	Unsigned1	0x00
	Data Set Size	Unsigned2	0x00
	Reserved	Unsigned1	0x00
	Command Specifier	Unsigned3	0x04: Abort Transfer Request
	Index	WORD	index of the Object
	Subindex	BYTE	subindex of the Object
	Abort Code	DWORD	Abort Code as specified in Table 40

5.6.2.7.2 SDO Abort Codes

The SDO Abort Codes are specified in Table 40.

Table 40 – SDO Abort Codes

Value	Meaning
0x05 03 00 00	toggle bit not changed
0x05 04 00 00	SDO protocol timeout
0x05 04 00 01	Client/Server command specifier not valid or unknown
0x05 04 00 05	out of memory
0x06 01 00 00	unsupported access to an object
0x06 01 00 01	attempt to read to a write only object
0x06 01 00 02	attempt to write to a read only object
0x06 01 00 03	Subindex cannot be written, SI0 must be 0 for write access
0x06 01 00 04	SDO Complete access not supported for objects of variable length such as ENUM object types
0x06 01 00 05	Object length exceeds mailbox size
0x06 01 00 06	Object mapped to RxPDO, SDO Download blocked
0x06 02 00 00	the object does not exist in the object directory
0x06 04 00 41	the object can not be mapped into the PDO
0x06 04 00 42	the number and length of the objects to be mapped would exceed the PDO length
0x06 04 00 43	general parameter incompatibility reason
0x06 04 00 47	general internal incompatibility in the device
0x06 06 00 00	access failed due to a hardware error
0x06 07 00 10	data type does not match, length of service parameter does not match
0x06 07 00 12	data type does not match, length of service parameter too high
0x06 07 00 13	data type does not match, length of service parameter too low
0x06 09 00 11	subindex does not exist
0x06 09 00 30	value range of parameter exceeded (only for write access)
0x06 09 00 31	value of parameter written too high
0x06 09 00 32	value of parameter written too low
0x06 09 00 36	maximum value is less than minimum value
0x08 00 00 00	general error
0x08 00 00 20	data cannot be transferred or stored to the application
0x08 00 00 21	data cannot be transferred or stored to the application because of local control
0x08 00 00 22	data cannot be transferred or stored to the application because of the present device state
0x08 00 00 23	object dictionary dynamic generation fails or no object dictionary is present

5.6.3 SDO Information

5.6.3.1 General

Everything in the part "SDO Info Service Data" of the following services shall be handled as a block and be sent only once.

If the service have to be fragmented the Length field can be <= 0x0A for the last fragment

5.6.3.2 SDO Information Service

The attribute types of SDO Information Service are described in Figure 24.

```
typedef struct
{
  unsigned          OpCode:          7;
  unsigned          InComplete:      1;
  unsigned          Reserved:        8;
  WORD              FragmentsLeft;
} TSDOINFOHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
} TSDOINFOSERVICE;
```

Figure 24 – SDO Information Service structure

The SDO Information Service coding is specified in Table 41.

Table 41 – SDO Information Service

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n > 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x01: Get OD List Request 0x02: Get OD List Response 0x03: Get Object Description Request 0x04: Get Object Description Response 0x05: Get Entry Description Request 0x06: Get Entry Description Response 0x07: SDO Info Error Request
	Incomplete	Unsigned1	0x00: last SDO Information fragment 0x01: SDO Information fragments will follow
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	number of Fragments which will follow
SDO Info Service Data	Data	BYTE[n-6]	SDO Information Service Data

5.6.3.3 Get OD List

5.6.3.3.1 Get OD List Request

The attribute types of Get OD List Request are described in Figure 25.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              ListType;
} TGETODLISTREQ;
```

Figure 25 – Get OD List Request structure

The Get OD List Request coding is specified in Table 42.

Table 42 – Get OD List Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x01: Get OD List Request
	Incomplete	Unsigned1	shall be zero
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	shall be zero
SDO Info Service Data	List Type	WORD	0x00: get number of objects in the 5 different lists 0x01: all objects of the object dictionary shall be delivered in the response 0x02: only objects which are mappable in a RxPDO shall be delivered in the response 0x03: only objects which are mappable in a TxPDO shall be delivered in the response 0x04: only objects which has to stored for a device replacement shall be delivered in the response 0x05: only objects which can be used as startup parameter shall be delivered in the response

5.6.3.3.2 Get OD List Response

The attribute types of Get OD List Response are described in Figure 26.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              ListType;
} TGETODLISTRES;
```

Figure 26 – Get OD List Response structure

The Get OD List Response coding is specified in Table 43.

Table 43 – Get OD List Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x02: Get OD List Response
	Incomplete	Unsigned1	0x00: last SDO Information fragment 0x01: SDO Information fragments will follow
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	number of Fragments which will follow SDO Info Header will be sent with every fragment, SDO Info Data will be sent fragmented
SDO Info Service Data	List Type	WORD	0x00: list of length shall be delivered in the response 0x01: all objects of the object dictionary shall be delivered in the response 0x02: only objects which are mappable in a RxPDO shall be delivered in the response 0x03: only objects which are mappable in a TxPDO shall be delivered in the response 0x04: only objects which has to stored for a device replacement shall be delivered in the response 0x05: only objects which can be used as startup parameter shall be delivered in the response
	Index List	WORD [(n-8)/2]	list of object indexes or 5 words with the length of the list types if list type is 0 Order of object indexes is free. It is recommended to sort in ascending order

5.6.3.4 OD List Segment

If the SDO Info Service Data of the Get OD List Response must be segmented the SDO Info Service Data are fragmented and the OD List Segment service shall transfer the fragments.

5.6.3.5 Get Object Description

5.6.3.5.1 Get Object Description Request

The attribute types of Get Object Description Request are described in Figure 27.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
} TGETOBJDESCREQ;
```

Figure 27 – Get Object Description Request structure

The Get Object Description Request coding is specified in Table 44.

Table 44 – Get Object Description Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x03: Get Object Description Request
	Incomplete	Unsigned1	Shall be zero
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	shall be zero
SDO Info Service Data	Index	WORD	index of the requested object description

5.6.3.5.2 Get Object Description Response

The attribute types of Get Object Description Response are described in Figure 28.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  WORD              DataType;
  BYTE              MaxSubindex;
  BYTE              ObjCode;
} TGETOBJDESCRES;
```

Figure 28 – Get Object Description Response structure

The Get Object Description Response coding is specified in Table 45.

Table 45 – Get Object Description Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n > 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x04: Get Object Description Response
	Incomplete	Unsigned1	0x00: last SDO Information fragment
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	number of Fragments which will follow
SDO Info Service Data	Index	WORD	index of the object description
	Data Type	WORD	reference to data type list
	Max Subindex	BYTE	maximum number of subindexes of the object
	Object Code	BYTE	Object Code 7: Variable 8: Array 9: Record
	Name	char[n-12]	name of the object

5.6.3.6 Get Entry Description

5.6.3.6.1 Get Entry Description Request

The attribute types of Get Entry Description Request are described in Figure 29.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  BYTE              Subindex;
  BYTE              ValueInfo;
} TGETENTRYDESCREQ;
```

Figure 29 – Get Entry Description Request structure

The Get Entry Description Request coding is specified in Table 46.

Table 46 – Get Entry Description Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x05: Get Entry Description Request
	Incomplete	Unsigned1	shall be zero
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	shall be zero
SDO Info Service Data	Index	WORD	index of the requested object description
	Subindex	BYTE	subindex of the requested object description
	ValueInfo	BYTE	the value info includes which elements shall be in the response: Bit 0: reserved Bit 1: reserved Bit 2: reserved Bit 3: unit type Bit 4: default value Bit 5: minimum value Bit 6: maximum value

5.6.3.6.2 Get Entry Description Response

The attribute types of Get Entry Description Response are described in Figure 30.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  BYTE              Subindex;
  BYTE              ValueInfo;
  WORD              DataType;
  WORD              BitLength;
  WORD              ObjAccess;
} TGETENTRYDESCRES;
```

Figure 30 – Get Entry Description Response structure

The Get Object Description Response coding is specified in Table 47.

Table 47 – Get Entry Description Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n >= 0x10: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x06: Get Entry Description Response
	Incomplete	Unsigned1	0x00: last SDO Information fragment 0x01: SDO Information fragments will follow
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	number of Fragments which will follow
SDO Info Service Data	Index	WORD	index of the requested object description
	Subindex	BYTE	subindex of the requested object description
	ValueInfo	BYTE	the value info includes which elements are in the response: Bit 0: reserved Bit 1: reserved Bit 2: reserved Bit 3: unit type Bit 4: default value Bit 5: minimum value Bit 6: maximum value
	Data Type	WORD	index of the data type of the object
	Bit Length	WORD	bit length of the object If the length is = 0xFFFF: the length of the object is greater than 64 kBit or for objects with variable length. To get the length of the object this object has to be uploaded
	Object Access	WORD	Bit 0: read access in Pre-Operational state Bit 1: read access in Safe-Operational state Bit 2: read access in Operational state Bit 3: write access in Pre-Operational state Bit 4: write access in Safe-Operational state

Frame part	Data Field	Data Type	Value/Description
			Bit 5: write access in Operational state Bit 6: object is mappable in a RxPDO Bit 7: object is mappable in a TxPDO Bit 8: object can be used for backup Bit 9: object can be used for settings Bit 10-15: reserved
	Data	BYTE[n-16]	if the unit type is included in the response, the unit type of the object is following (DWORD) if the default value is included in the response, the default value of the object entry is following (same data type as the object value) if the minimum value is included in the response, the minimum value of the object entry is following (same data type as the object value) if the maximum value is included in the response, the maximum value of the object entry is following (same data type as the object value) if the Length is greater than the described response parameter, the description is following (array of char)

5.6.3.7 Entry Description Segment

The attribute types and coding of Entry Description Segment are the same as of Get Entry Description Response.

5.6.3.8 SDO Info Error

The attribute types of SDO Info Error Request are described in Figure 31.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  DWORD             AbortCode;
} TABORTSDOTRANSFERREQMBX;
```

Figure 31 – SDO Info Error Request structure

The SDO Info Error Request coding is specified in Table 48.

Table 48 – SDO Info Error Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x08: SDO Information
SDO Info Header	Opcode	Unsigned7	0x07: SDO Info Error Request
	Incomplete	Unsigned1	Shall be zero
	Reserved	Unsigned8	0x00
	Fragments Left	WORD	shall be zero
	Abort Code	DWORD	Abort Code as specified in Table 40

5.6.4 Emergency

5.6.4.1 Emergency Request

The Emergency Request coding is specified in Table 49.

Table 49 – Emergency Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n = 0x0A: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x01: Emergency
Emergency	Error Code	WORD	Error Code
	Error Register	BYTE	Error Register
	Data	BYTE[5]	Error Code 0000-9FFF: Manufacturer Specific Error Field Error Code A000-EFFF: Diagnostic Data Error Code F000-FFFF: Manufacturer Specific Error Field
	Reserved	BYTE[n-10]	

5.6.4.2 Emergency Error Codes

The Emergency Error Codes are specified in Table 50.

Table 50 – Emergency Error Codes

Error Code (hex)	Meaning
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current, device input side
22xx	Current inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains Voltage
32xx	Voltage inside the device
33xx	Output Voltage
40xx	Temperature
41xx	Ambient Temperature
42xx	Device Temperature
50xx	Device Hardware
60xx	Device Software
61xx	Internal Software
62xx	User Software
63xx	Data Set
70xx	Additional Modules
80xx	Monitoring
81xx	Communication
82xx	Protocol Error
8210	PDO not processed due to length error
8220	PDO length exceeded
90xx	External Error
A0xx	ESM Transition Error
F0xx	Additional Functions
FFxx	Device specific

5.6.4.3 ESM Transition Error

5.6.4.3.1 Error Code

The ESM Transition Error Codes are specified in Table 51.

Table 51 – Error Code

Error Code (hex)	Meaning
A000	transition PRE-OPERATIONAL to SAFE-OPERATIONAL not successful
A001	transition SAFE-OPERATIONAL to OPERATIONAL not successful

5.6.4.3.2 Diagnostic Data

The ESM Transition Diagnostic Data structure is specified in Table 52.

Table 52 – Diagnostic Data

Data[0]	Data[1..4]	Meaning
0x00 + channel*4	Sync Manager Length Error	length of the Sync Manager channel does not match
0x01 + channel*4	Sync Manager Address Error	Physical Start Address of the Sync Manager channel does not match
0x02 + channel*4	Sync Manager Settings Error	settings of the Sync Manager channel are not matching

5.6.4.3.3 Sync Manager Length Error

The Sync Manager Length Error coding is specified in Table 53.

Table 53 – Sync Manager Length Error

Data[1..4]	Data Type	Value/Description
Minimum Length	WORD	minimum value for the parameter Length of the Sync Manager channel
Maximum Length	WORD	maximum value for the parameter Length of the Sync Manager channel

5.6.4.3.4 Sync Manager Address Error

The Sync Manager Address Error coding is specified in Table 54.

Table 54 – Sync Manager Address Error

Data[1..4]	Data Type	Value/Description
Minimum Address	WORD	minimum value for the parameter Physical Start Address of the Sync Manager channel
Maximum Address	WORD	maximum value for the parameter Physical Start Address of the Sync Manager channel

5.6.4.3.5 Sync Manager Settings Error

The Sync Manager Settings Error coding is specified in Table 55.

Table 55 – Sync Manager Settings Error

Data[1..4]	Data Type	Value/Description
Expected Buffer Type	Unsigned2	expected value for the parameter Buffer Type of the Sync Manager channel
Expected Direction	Unsigned2	expected value for the parameter Direction of the Sync Manager channel
Reserved	Unsigned1	0x00 (Reserved for future)
Expected AL Event Enable	Unsigned1	expected value for the parameter AL Event Enable of the Sync Manager channel
Reserved	Unsigned10	0x00 (Reserved for future)
Expected Channel Enable	Unsigned1	expected value for the parameter Channel Enable of the Sync Manager channel
Reserved	Unsigned15	0x00 (Reserved for future)

5.6.5 Process Data

5.6.5.1 RxPDO

The protocol of the RxPDO Transmission via mailbox is specified in Table 56.

Table 56 – RxPDO Transmission via mailbox

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n > 0x02: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	related RxPDO-Number
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x05: RxPDO
PDO	Data	BYTE[n-2]	Process Output Data

5.6.5.2 TxPDO

The TxPDO Transmission via mailbox coding is specified in Table 57.

Table 57 – TxPDO Transmission via mailbox

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	n > 0x02: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	related TxPDO-Number
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x04: TxPDO
PDO	Data	BYTE[n-2]	Process Input Data

5.6.5.3 RxPDO Remote Transmission Request

The RxPDO Remote Transmission Request coding is specified in Table 58.

Table 58 – RxPDO Remote Transmission Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x02: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	related RxPDO-Number
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x07: RxPDO Remote Transmission Request

5.6.5.4 TxPDO Remote Transmission Request

The TxPDO Remote Transmission Request coding is specified in Table 59.

Table 59 – TxPDO Remote Transmission Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x02: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CoE Header	Number	Unsigned9	related TxPDO-Number
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x06: TxPDO Remote Transmission Request

5.6.6 Command

The Command object structure is specified in Table 60. Each command object shall have the data type 0x0025. The structure can be used by any object declared as command object.

Table 60 – Command object structure

Subindex	Description	Data Type	Value
0	Number of entries	UNSIGNED8	3
1	Command	OCTET_STRING	Byte 0-n: Request Data a write access to the command data will execute the command
2	Status	UNSIGNED8	0: last command completed, no errors, no reply 1: last command completed, no errors, reply there 2: last command completed, error, no reply 3: last command completed, error, reply there 4-99: reserved for future use 100-200: indicates how of the command has been executed (in %, 100 = 0 %, 200 = 100 %) 201-254: reserved for future use 255: command is executing (if the percentage display is not supported)
3	Reply	OCTET-STRING	octet 0-n: Response Data

5.6.7 Object Dictionary

5.6.7.1 Object Dictionary structure

The Object Dictionary is structured as noted in Table 61.

Table 61 – Object Dictionary Structure

Index (hex)	Object Dictionary Area
0x0000-0x0FFF	Data Type Area
0x1000-0x1FFF	CoE Communication Area
0x2000-0x5FFF	Manufacturer Specific Area
0x6000-0xFFFF	Profile Area

5.6.7.2 Object Code Definitions

The Object Code Definition entries are structured as noted in Table 62.

Table 62 – Object Code Definitions

Object Code	Object Name
0002	DOMAIN
0005	DEFTYPE
0006	DEFSTRUCT
0007	VAR
0008	ARRAY
0009	RECORD

5.6.7.3 Data Type Area

The Basic Data Type Area is specified in Table 63.

Table 63 – Basic Data Type Area

Index (hex)	Object Type	Name
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	INTEGER8
0003	DEFTYPE	INTEGER16
0004	DEFTYPE	INTEGER32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000A	DEFTYPE	OCTET_STRING
000B	DEFTYPE	UNICODE_STRING
000C	DEFTYPE	TIME_OF_DAY
000D	DEFTYPE	TIME_DIFFERENCE
000E		Reserved
000F	DEFTYPE	DOMAIN
0010	DEFTYPE	INTEGER24
0011	DEFTYPE	REAL64
0012	DEFTYPE	INTEGER40
0013	DEFTYPE	INTEGER48
0014	DEFTYPE	INTEGER56
0015	DEFTYPE	INTEGER64
0016	DEFTYPE	UNSIGNED24
0017		Reserved
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001A	DEFTYPE	UNSIGNED56
001B	DEFTYPE	UNSIGNED64
001C		Reserved
001D	DEFTYPE	GUID
001E	DEFTYPE	BYTE
001F-002C		Reserved
002D	DEFTYPE	BitARR8
002E	DEFTYPE	BITARR16
002F	DEFTYPE	BITARR32

The Extended Data Type Area is specified in Table 64.

Table 64 – Extended Data Type Area

Index (hex)	Object	Name
0020		reserved
0021	DEFSTRUCT	PDO_MAPPING
0022		reserved
0023	DEFSTRUCT	IDENTITY
0024		reserved
0025	DEFSTRUCT	COMMAND_PAR
0026-0028		reserved
0029	DEFSTRUCT	SYNC_PAR
002A-002F		reserved
0030	DEFTYPE	BIT1
0031	DEFTYPE	BIT2
0032	DEFTYPE	BIT3
0033	DEFTYPE	BIT4
0034	DEFTYPE	BIT5
0035	DEFTYPE	BIT6
0036	DEFTYPE	BIT7
0037	DEFTYPE	BIT8
0038-003F		reserved
0040-005F	DEFSTRUCT	manufacturer Specific Complex Data Types
0060-007F	DEFTYPE	Device Profile 0 Specific Standard Data Types
0080-009F	DEFSTRUCT	Device Profile 0 Specific Complex Data Types
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types
0260-07FF		reserved

The Enumerated Data Type Area occupies index 0x800 to 0xFFF. Each item has a data type that specifies the number of bits occupied (e.g. BIT3 or UNSIGNED16) and a list of entries that specifies integer value (data type is unsigned32) and the enumeration visible string as shown in Table 65.

Table 65 – Enumeration Definition

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of entries	UNSIGNED8	O	R	No	number of enumeration values n
	Padding	UNSIGNED8				0 padding to maintain even octets boundary for the following objects
1	Enum1	OCTET STRING	O	R	No	UNSIGNED32 as integer value VISIBLE STRING as enumeration string
	...					
1	Enumn	OCTET STRING	O	R	No	UNSIGNED32 as integer value VISIBLE STRING as enumeration string

5.6.7.4 CoE Communication Area

CoE Communication Object Dictionary Area consists of the elements described in Table 66.

Table 66 – CoE Communication Area

Index (hex)	Object type	Name	Type	M/O/C
1000	VAR	Device Type	UNSIGNED32	M
1001		Error Register	UNSIGNED8	O
1002		Reserved		
::::	::::	::::	::::	
1007		Reserved		
1008	VAR	Manufacturer Device Name	VisibleString	O
1009	VAR	Manufacturer Hardware Version	VisibleString	O
100A	VAR	Manufacturer Software Version	VisibleString	O
100B		Reserved		
::::	::::	::::	::::	
1017		Reserved		
1018	RECORD	Identity Object	Identity (0x23)	M
1019		Reserved		
::::	::::	::::	::::	
15FF		Reserved		
1600	RECORD	1st receive PDO Mapping	PDO Mapping (0x21)	C
1601	RECORD	2nd receive PDO Mapping	PDO Mapping	C
::::	::::	::::	::::	
17FF	RECORD	512th receive PDO Mapping	PDO Mapping	C
1800		Reserved		
::::	::::	::::	::::	
19FF		Reserved		
1A00	RECORD	1st transmit PDO Mapping	PDO Mapping (0x21)	C
1A01	RECORD	2nd transmit PDO Mapping	PDO Mapping	C

Index (hex)	Object type	Name	Type	M/O/C
::::	::::	::::	::::	
1BFF	RECORD	512th transmit PDO Mapping	PDO Mapping	C
1C00	ARRAY	Sync Manager Communication Type	UNSIGNED8	M
1C01		Reserved		
::::	::::	::::	::::	
1C0F		Reserved		
1C10	ARRAY	Sync Manager 0 PDO Assignment	UNSIGNED16	C
1C11	ARRAY	Sync Manager 1 PDO Assignment	UNSIGNED16	C
1C12	ARRAY	Sync Manager 2 PDO Assignment	UNSIGNED16	C
1C13	ARRAY	Sync Manager 3 PDO Assignment	UNSIGNED16	C
1C14	ARRAY	Sync Manager 4 PDO Assignment	UNSIGNED16	C
::::	::::	::::	::::	
1C2F	ARRAY	Sync Manager 31 PDO Assignment	UNSIGNED16	C
1C30	RECORD	Sync Manager 0 Synchronization		O
::::	::::	::::	::::	
1C4F	RECORD	Sync Manager 31 Synchronization		O
1C50		Reserved		
::::	::::	::::	::::	
1FFF		Reserved		

5.6.7.4.1 Device Type

The Device Type object dictionary entry (index 0x1000) is specified in Table 67.

Table 67 – Device Type

Attribute	Value
Name	Device Type
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory
Access	Ro
PDO Mapping	No
Value	Bit 0-15: used device profile, 0x0000 if no standardized device profile is used. Bit 16-31: additional information depending on the used device profile

5.6.7.4.2 Error Register

The Error Register object dictionary entry (index 0x1001) is specified in Table 68.

Table 68 – Error Register

Attribute	Value
Name	Error Register
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional
Access	Ro
PDO Mapping	
Value	Bit 0: generic error Bit 1: current error Bit 2: voltage error Bit 3: temperature error Bit 4: communication error Bit 5: device profile specific error Bit 6: reserved Bit 7: manufacturer specific error

5.6.7.4.3 Manufacturer Device Name

The Manufacturer Device Name object dictionary entry (index 0x1008) is specified in Table 69.

Table 69 – Manufacturer Device Name

Attribute	Value
Name	Manufacturer Device Name
Object Code	VAR
Data Type	VISIBLE_STRING
Category	Optional
Access	Ro
PDO Mapping	No
Value	name of the device as non zero terminated string

5.6.7.4.4 Manufacturer Hardware Version

The Manufacturer Hardware Version object dictionary entry (index 0x1009) is specified in Table 70.

Table 70 – Manufacturer Hardware Version

Attribute	Value
Name	Manufacturer Hardware Version
Object Code	VAR
Data Type	VISIBLE_STRING
Category	Optional
Access	Ro
PDO Mapping	No
Value	Hardware version of the device as non zero terminated string

5.6.7.4.5 Manufacturer Software Version

The Manufacturer Software Version object dictionary entry (index 0x100A) is specified in Table 71.

Table 71 – Manufacturer Software Version

Attribute	Value
Name	Manufacturer Software Version
Object Code	VAR
Data Type	VISIBLE_STRING
Category	Optional
Access	R
PDO Mapping	No
Value	Software version of the device as non zero terminated string

5.6.7.4.6 Identity Object

The Identity Object dictionary entry (index 0x1018) is specified in Table 72.

Table 72 – Identity Object

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of entries	UNSIGNED8	M	R	No	4
1	Vendor ID	UNSIGNED32	M	R	No	assigned uniquely by ETG
2	Product Code	UNSIGNED32	M	R	No	assigned uniquely by Vendor
3	Revision Number	UNSIGNED32	M	R	No	assigned uniquely by Vendor
4	Serial Number	UNSIGNED32	M	R	No	assigned uniquely for this device by Vendor 0 if there is no serial number given

5.6.7.4.7 Receive PDO Mapping

The Receive PDO Mapping object dictionary entry (index 0x1600 – 0x17FF) is specified in Table 73.

Table 73 – Receive PDO Mapping

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of objects in this PDO	UNSIGNED8	M	R/RW [a]	No	0 – 254 writeable if variable mapping is supported
1	First Output Object to be mapped	UNSIGNED32	C	R/RW [a]	No	Bit 0-7: length of the mapped objects in bits (for a gap in the PDO: shall have the bit length of the gap) Bit 8-15: subindex of the mapped object (0 in case of a gap in the PDO) Bit 16-31: index of the mapped object(for a gap in the PDO: shall be zero)
..						
n	Last Output Object to be mapped	UNSIGNED32	C	R/RW [a]	No	
[a] Writeable if variable PDO assign is supported.						

5.6.7.4.8 Transmit PDO Mapping

The Transmit PDO Mapping object dictionary entry (index 0x1A00 – 0x1BFF) is specified in Table 74.

Table 74 – Transmit PDO Mapping

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of objects in this PDO	UNSIGNED8	M	R/RW [a]	No	0 – 254 writeable if variable mapping is supported
1	First Output Object to be mapped	UNSIGNED32	C	R/RW [a]	No	Bit 0-7: length of the mapped objects in bits (for a gap in the PDO: shall have the bit length of the gap) Bit 8-15: subindex of the mapped object (0 in case of a gap in the PDO) Bit 16-31: index of the mapped object(for a gap in the PDO: shall be zero)
..						
n	Last Output Object to be mapped	UNSIGNED32	C	R/RW [a]	No	
[a] Writeable if variable PDO assign is supported.						

5.6.7.4.9 Sync Manager Communication Type

The Sync Manager Communication Type object dictionary entry (index 0x1C00) is specified in Table 75.

Table 75 – Sync Manager Communication Type

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of used Sync Manager channels	UNSIGNED8	M	R	No	4 – 32
1	Communication Type Sync Manager 0	UNSIGNED8	C	R	No	configurable [a] 0: unused 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output 4: process data input(slave to master)
2	Communication Type Sync Manager 1	UNSIGNED8	C	R	No	configurable [a] 0: unused 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output 4: process data input(slave to master)
3	Communication Type Sync Manager 2	UNSIGNED8	C	R	No	configurable [a] 0: unused 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output 4: process data input(slave to master)
4	Communication Type Sync Manager 3	UNSIGNED8	C	R	No	configurable [a] 0: unused 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output 4: process data input(slave to master)
5 – n	Communication Type Sync Manager 4 – (n-1)	UNSIGNED8	C	R	No	configurable [a] 0: unused 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output 4: process data input(slave to master)

[a] The Sync Manager communication type should be used in the following way:
Communication Type Sync Manager 0: 1 mailbox receive
Communication Type Sync Manager 1: 2 mailbox send
Communication Type Sync Manager 2: 3 process data output
Communication Type Sync Manager 3: 4 process data input

If not mailbox is supported, it should be used in the following way:
Communication Type Sync Manager 0: 3 process data output
Communication Type Sync Manager 1: 4 process data input

5.6.7.4.10 Sync Manager PDO Assignment

5.6.7.4.10.1 Sync Manager Channel

The Sync Manager Channel 0-31 object dictionary entry (index 0x1C10-0x1C2F) is specified in Table 76. If a Sync Manager Channel is used as a mailbox Sync Manager the corresponding Object shall not be available or Subindex 0 shall be 0.

Table 76 – Sync Manager Channel 0-31

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of assigned TxPDOs	UNSIGNED8	C	R/RW [a]	No	0 – 254
1 – n	PDO Mapping object index of assigned PDO	UNSIGNED16	C	R/RW [a]	No	Either 0x1600: RxPDO 1 0x1601: RxPDO 2 … 0x17FF: RxPDO 512 Or 0x1A00: TxPDO 1 0x1A01: TxPDO 2 … 0x1BFF: TxPDO 512
[a] Writeable if variable PDO assign is supported.						

5.6.7.4.11 Sync Manager Synchronization

The Sync Manager Synchronization object dictionary entry (index 0x1C30-0x1C4F) is specified in Table 77.

Table 77 – Sync Manager Synchronization

Sub-Index	Description	Data type	M/O/C	Access	PDO Mapping	Value
0	Number of Synchronization Parameters	UNSIGNED8	O	R	No	1 – 3
1	Synchronization type	UNSIGNED16	O	RW	No	• 0: not synchronized • 1: Synchron – synchronized with AL Event on this Sync Manager • 2: DC Sync0 – synchronized with AL Event Sync0 • 3: DC Sync1 – synchronized with AL Event Sync1 • 32: SyncSm0 – synchronized with AL Event of SM0 • 33: SyncSm1 – synchronized with AL Event of SM1 • … • 63: SyncSm31 – synchronized with AL Event of SM31
2	Cycle time	UNSIGNED32	O	RW	No	time between two events in ns
3	Shift time	UNSIGNED32	O	RW	No	time between related AL event and the associated action in ns

5.7 EoE coding

5.7.1 Initiate EoE

5.7.1.1 Initiate EoE Request

The attribute types of Initiate EoE Request are described in Figure 32.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          FragmentNumber:     6;
  unsigned          CompleteSize:       6;
  unsigned          FrameNumber:        4;
} TEOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TEOEHEADER        EoeHeader;
  BYTE              Data[MAX_EOE_DATA_SIZE];
} TINIEOEREQ;
```

Figure 32 – EoE general structure

The Initiate EoE Request coding is specified in Table 78.

Table 78 – Initiate EoE Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N = 0x24+ y*0x20: Length of the Mailbox Service Data (y = 0 to 0x2F)
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority ... 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x00
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x00: at least one EoE Fragment service is following 0x01: complete Ethernet frame is in the Data part
	Time Appended	Unsigned1	0x00: no time stamp value will be appended after the EoE Data in the last fragment 0x01: time stamp value will be appended after the EoE Data in the last fragment
	Time Request	Unsigned1	0x00: no time stamp valueof the send time requested 0x01: time stamp valueof the send time requested
	Reserved	Unsigned5	
	Fragment Number	Unsigned6	0x00
	Complete Size	Unsigned6	in 32-octet blocks Trunc ((complete size in octet of the Ethernet frame + 31)/32)
	Frame Number	Unsigned4	number of the Ethernet frame
	EoE Data	BYTE[N-4]	Ethernet frame (without Preamble, SFD, FCS) first portion (N-4) octets (4 Octets less if TimeStamp included)
(optional)	TimeStamp	Unsigned32	time of frame receipt, in ns starting at beginning of DA

5.7.1.2 Initiate EoE Response

The attribute types of Initiate EoE Response are described in Figure 33.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
  TimeStamp         Unsigned32;
} TINIEOERES;
```

Figure 33 – EoE Timestamp structure

The Initiate EoE Response coding is specified in Table 79.

Table 79 – Initiate EoE Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N = 0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x03
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x00
	Time Appended	Unsigned1	0x01: time stamp value will be appended
	Time Request	Unsigned1	0x00: no time stamp valueof the send time requested
	Reserved	Unsigned5	
	Fragment Number	Unsigned6	0x00
	Complete Size	Unsigned6	0x00
	Frame Number	Unsigned4	number of the Ethernet frame
	TimeStamp	Unsigned32	time of frame send, in ns starting at beginning of DA

5.7.2 EoE Fragment Data

The attribute types of EoE Fragment Data are described in Figure 34.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
  BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 34 – EoE Fragment Data structure

The EoE Fragment Data coding is specified in Table 80.

Table 80 – EoE Fragment Data

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x04: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority

Frame part	Data Field	Data Type	Value/Description
			… 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x00
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x00: at least one EoE Fragment service is following 0x01: last Data part of this Ethernet frame (including time stamp)
	Time Appended	Unsigned1	0x00: no time stamp value will be appended after the EoE Data in the last fragment 0x01: time stamp value will be appended after the EoE Data in the last fragment
	Time Request	Unsigned1	0x00: no time stamp valueof the send time requested 0x01: time stamp valueof the send time requested
	Reserved	Unsigned5	
	Fragment Number	Unsigned6	0x01-0x2F: fragment number of the Ethernet frame fragment
	Offset	Unsigned6	offset in 32-octet blocks of the Ethernet frame fragment
	Frame Number	Unsigned4	number of the Ethernet frame
	EoE Data	BYTE[N-4]	Ethernet frame (without Preamble, SFD, FCS) first portion (N-4) octets (4 Octets less if Timestamp included)
(optional)	TimeStamp	Unsigned32	time of frame receipt, in ns starting at beginning of DA

5.7.3 Data element for EoE

The EoE Data coding is specified in Table 81.

Table 81 – EoE Data

Frame part	Data Field	Data Type	Value/Description
Ethernet	Dest MAC	BYTE[6]	Destination MAC Address as specified in ISO/IEC 8802-3
	Src MAC	BYTE[6]	Source MAC Address as specified in ISO/IEC 8802-3
(optional)	VLAN Tag	BYTE[4]	0x81, 0x00 and two Octets Tag Control Information as specified in IEEE 802.1Q
	Ether Type	BYTE[2]	assigned by IEEE
User Frame	Data		user data (octet string) or EoE parameter
	Padding	BYTE[n]	shall be inserted if DL PDU is shorter than 64 octets as specified in ISO/IEC 8802-3

5.7.4 Set IP Parameter

5.7.4.1 Set IP Parameter Request

The attribute types of Set IP Parameter Request are described in Figure 35.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
  BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 35 – Set IP Parameter Request structure

The coding of Set IP Parameter Request is described in Table 82.

Table 82 – Set IP Parameter Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x02
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x01: last Data part
	Time Appended	Unsigned1	0x00: no time stamp value will be appended
	Time Request	Unsigned1	0x00: no time stamp value of the send time requested
	Reserved	Unsigned5	
	Fragment Number	Unsigned6	0x00
	Offset	Unsigned6	0x00
	Frame Number	Unsigned4	0x00
EoE Parameter	MAC included	Unsigned1	1, MAC address according to ISO/IEC 8802-3
	IP address included	Unsigned1	1, IP address according to IETF RFC 791
	Subnet Mask included	Unsigned1	1, Subnet mask according to IETF RFC 791
	Default Gateway included	Unsigned1	1, Default Gateway address according to IETF RFC 791
	DNS Server IP Address included	Unsigned1	1, IP address of DNS server according to IETF RFC 791
	DNS Name included	Unsigned1	1, DNS name according to IETF RFC 791
	reserved	Unsigned26	
	MAC	BYTE[6]	MAC address according to ISO/IEC 8802-3

Frame part	Data Field	Data Type	Value/Description
	IP address	BYTE[4]	IP address according to IETF RFC 791
	Subnet Mask	BYTE[4]	Subnet mask according to IETF RFC 791 and IETF RFC 826
	Default Gateway	BYTE[4]	Default Gateway address according to IETF RFC 791
	DNS Server IP Address	BYTE[4]	IP address of DNS server according to IETF RFC 791
	DNS Name	char[32]	DNS name according to IETF RFC 791

5.7.4.2 Set IP Parameter Response

The attribute types of Set IP Parameter Response are described in Figure 36.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          Result:            16;
} TEOEPARAHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEPARAHEADER    EoeHeader;
} TEOEFRAGREQ;
```

Figure 36 – Set IP Parameter Response structure

The coding of Set IP Parameter Response is described in Table 83.

Table 83 – Set IP Parameter Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N = 0x04: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x03
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x01: last Data part
	Time Appended	Unsigned1	0x00: no time stamp value will be appended
	Time Request	Unsigned1	0x00: no time stamp value of the send time requested
	Reserved	Unsigned5	
	Result	Unsigned16	see Table 84

Table 84 – EoE Result Parameter

result code	meaning
0x0000	success
0x0001	unspecified Error
0x0002	unsupported Frame Type
0x0201	no IP Support
0x0202	DHCP not supported If the Master sends Set IP Parameter Request with IP Address “0.0.0.0” the slave should send an DCHP_Discover to get an IP Address. If the slave does not support DHCP it should response with Result “DHCP not supported”
0x0401	no Filter Support

5.7.5 Set Address Filter

5.7.5.1 Set MAC Filter Request

The attribute types of Set MAC Filter Request are described in Figure 37.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
  BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 37 – Set MAC Filter Request structure

The coding of Set MAC Filter Request is described in Table 85.

Table 85 – Set MAC Filter Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x04
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x01: last Data part
	Time Appended	Unsigned1	0x00: no time stamp value will be appended
	Time Request	Unsigned1	0x00: no time stamp value of the send time requested
	Reserved	Unsigned5	

Frame part	Data Field	Data Type	Value/Description
	Fragment Number	Unsigned6	0x00
	Offset	Unsigned6	0x00
	Frame Number	Unsigned4	0x00
EoE Parameter	MAC filter count	Unsigned4	count of MAC address according to ISO/IEC 8802-3 which are accepted by Ethernet Ports on this slave
	MAC filter mask	Unsigned2	count of MAC address masks according to ISO/IEC 8802-3 which are combined with filter by Ethernet Ports on this slave
	Reserved	Unsigned1	subnet mask according to IETF RFC 791
	Inhibit Broadcast	Unsigned1	filter Broadcast messages
	Reserved	Unsigned8	
(conditional)	List of MAC Address	List of BYTE[6]	MAC address according to ISO/IEC 8802-3
(conditional)	List of MAC Address Filter	List of BYTE[6]	MAC address according to ISO/IEC 8802-3 a set bit means that this address bit of Destination MAC Address is compared with the corresponding entry in the List of MAC Address.

5.7.5.2 Set MAC Filter Response

The attribute types of Set MAC Filter Response are described in Figure 38.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          Result:            16;
} TEOEPARAHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEPARAHEADER    EoeHeader;
} TEOEFRAGREQ;
```

Figure 38 – Set MAC Filter Response structure

The coding of Set MAC Filter Response is described in Table 86.

Table 86 – Set MAC Filter Response

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N = 0x08: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
EoE Header	FrameType	Unsigned4	0x05
	Port	Unsigned4	0x00: send to no specific port 0x01-0x0F: specific ports selected
	Last Fragment	Unsigned1	0x01: last Data part
	Time Appended	Unsigned1	0x00: no time stamp value will be appended
	Time Request	Unsigned1	0x00: no time stamp value of the send time requested
	Reserved	Unsigned5	
	Result	Unsigned16	see Table 84

5.8 FoE Coding

5.8.1 Read Request

The attribute types of Read Request are described in Figure 39.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             Password;
  char              FileName[MAX_FILE_NAME_SIZE};
} TFOEREADREQ;
```

Figure 39 – Read Request structure

The FoE Read Request coding is specified in Table 87.

Table 87 – Read Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x01: Read Request
	Reserved	BYTE	shall be zero
Read Header	Password	DWORD	0: password unused 1-0xFFFFFFFF: password
	File Name	char[n-6]	name of the file to be read

5.8.2 Write Request

The attribute types of Write Request are described in Figure 40.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             Password;
  char              FileName[MAX_FILE_NAME_SIZE};
} TFOEWRITEREQ;
```

Figure 40 – Write Request structure

The FoE Write Request coding is specified in Table 88.

Table 88 – Write Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x02: Write Request
	Reserved	BYTE	shall be zero
Write Header	Password	DWORD	0: password unused 1-0xFFFFFFFF: password
	File Name	char[n-6]	name of the file to be written

5.8.3 Data Request

The attribute types of Data Request are described in Figure 41.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             PacketNo;
  BYTE              Data[MAX_DATA_SIZE};
} TFOEDATAREQ;
```

Figure 41 – Data Request structure

The FoE Data Request coding is specified in Table 89.

Table 89 – Data Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N > 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x03: Data Request
	Reserved	BYTE	shall be zero
Data Header	Packet Number	DWORD	1-0xFFFFFFFF
	Data	BYTE[n-6]	File data

5.8.4 Ack Request

The attribute types of Ack Request are described in Figure 42.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             PacketNo;
} TFOEACKREQ;
```

Figure 42 – Ack Request structure

The FoE Ack Request coding is specified in Table 90.

Table 90 – Ack Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x04: Ack Request
	Reserved	BYTE	shall be zero
Ack Header	Packet Number	DWORD	0: acknowledge of Write Request 1-0xFFFFFFFF: acknowledge of Data Request

5.8.5 Error Request

The attribute types of Error Request are described in Figure 43.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             ErrorCode;
  char              ErrorText[MAX_ERROR_TEXT_SIZE};
} TFOEERRORREQ;
```

Figure 43 – Error Request structure

The FoE Error Request coding is specified in Table 91.

Table 91 – Error Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N >= 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x05: Error Request
	Reserved	BYTE	shall be zero
Error Header	Error Code	DWORD	1-0xFFFFFFFF
	Error Text	char[n-6]	optional error description

The error codes of FoE are specified in Table 92.

Table 92 – Error codes of FoE

error code	meaning
0x8000	Not defined
0x8001	Not found
0x8002	Access denied
0x8003	Disk full
0x8004	Illegal
0x8005	Packet number wrong
0x8006	Already exists
0x8007	No user
0x8008	Bootstrap only
0x8009	Not Bootstrap
0x800A	No rights
0x800B	Program Error

5.8.6 Busy Request

The attribute types of Busy Request are described in Figure 44.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  WORD              Done;
  WORD              Entire;
  char              BusyText[MAX_BUSY_TEXT_SIZE};
} TFOEBUSYREQ;
```

Figure 44 – Busy Request structure

The FoE Busy Request coding is specified in Table 93.

Table 93 – Busy Request

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	N >= 0x06: Length of the Mailbox Service Data
	Address	WORD	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	counter of the mailbox services (0 reserved, 1 is start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
FoE Header	OpCode	BYTE	0x06: Busy Request
	Reserved	BYTE	shall be zero
Busy Header	Done	WORD	If entire is “0”, Done indicates the progress in percent. 0-100 (done in %) If entire is unequal to “0” the value indicates the size of the already transferred data in the same unit as the element “Entire”.
	Entire	WORD	If element is “0” the Done element indicates progress of the file transfer from 0% to 100%. If unequal to “0” Entire indicates the file size of the actual transfer in a specified unit. In this case Done indicates the size of date which already has been transferred in the same unit. With help of this two values a percentage quotation can be calculated: 100*Done/Entire
	Busy Text	char[n-6]	optional busy description

6 FAL protocol state machines

6.1 Overall structure

6.1.1 Overview

The FAL protocol state machine structure is as defined in Figure 45. The general structure is according to the IEC 61158-6 subseries protocol machine model.

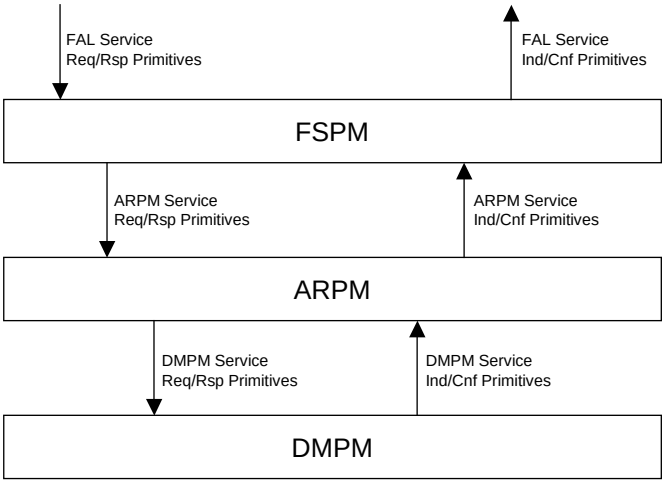


Figure 45 – Relationship among Protocol Machines

The behavior of the FAL is specified by three integrated protocol machines. The FSPM is the service interface between the FAL services that are part of the FAL Class specification and the particular AREP.

The Type 12 FAL provides a set of protocol machines for slave. The masters can anticipate the behavior of the slaves.

The FSPM is responsible for the following activities.

- To accept service primitives from the FAL service user and convert them into FAL internal primitives.
- To select the ARPM state machine based on the implicit addressing mechanism and send FAL internal primitives with the service parameters to the ARPM.
- To accept FAL internal primitives from the ARPM and convert them into service primitives for the FAL service user.
- To deliver the FAL service primitives to the FAL user.

The ARPM specifies the conveyance type for the application relation.

The DMPM specifies the mapping to the Data Link Layer. The DMPM defines therefore two protocol machines, the LMPM and the MAC protocol machines.

6.1.2 Fieldbus Service Protocol Machines (FSPM)

The FSPM State Machines co-ordinate the underlying state machines used for processing of the various services and application relations.

The FSPM basically is a mapping protocol machine. The main task is to pass the service to the protocol machine responsible for that service and to forward confirmations and responses to the user. In addition a basic redundancy control scheme is included in this machine that allows to collaborate two AR into a single entity with higher availability.

6.1.3 Application Relationship Protocol Machines (ARPM)

The ARPMs are responsible for the individual service procedures execution. The overall structure is shown in Figure 46. Process Data interaction is directly handled by DL and controlled by ESM. There are various ways for the application to run mailbox protocols.

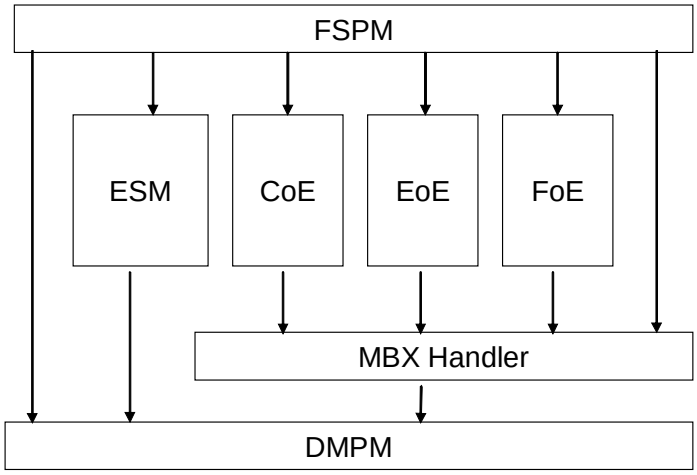


Figure 46 – AR Protocol machines

6.1.4 DLL Mapping Protocol Machines (DMPM)

The DL Mapping Protocol Machines (DMPM) connects the other State machines and Layer 2. DMPM provides the coordination of all state machines concerning the configuration and error handling of the Data Link Layer Usage. The functions are mapped by the DMPM to the DLL services of Layer 2. The DMPM generates the necessary Layer 2 parameters of the service, receives the confirmations and indications from Layer 2 and passes them to the appropriate DMPM-User.

6.2 AP-Context state machine

There is no AP-Context State Machine defined for this Protocol.

NOTE The AP Context state machine is part of the IEC 61158-6 model.

6.3 FAL service protocol machine (FSPM)

The services specified in IEC 61158-5-12 are directly mapped to services of the ARPMs.

6.4 Application Relationship Protocol Machines (ARPMs)

6.4.1 AL state machine

6.4.1.1 Description

ESM is responsible for the coordination of master and slave at start up and during operation. State changes are mainly caused by interactions between master and slave. They are primarily related to writes to AL Control word.

After Initialization of DL and AL the machine enters the INIT State. The ‘Init’ state defines the root of the communication relationship between the master and the slave in application layer. No direct communication between the master and the slave on application layer is possible. The master uses the ‘Init’ state to initialize a set of configuration register. If the slave supports a mailbox, the corresponding sync manager configurations are also done in the ‘Init’ state.

The ‘Pre-Operational’ state can be entered if the settings of the mailbox have been done if the slave supports the optional mailbox. Both the master and the slave can use the mailbox and the appropriate protocols to exchange application specific initializations and parameters. No process data communication is possible in this state.

The ‘Safe-Operational’ state can be entered if the settings of the input buffer have been done if the slave supports the inputs and the master requests inputs. The application of the slave shall deliver actual input data without processing the output data. The real outputs of the slave shall be set to their “safe state”.

The ‘Operational’ state can be entered if the settings of the output buffer have been done and actual outputs have been delivered to the slave (provides outputs of the slave will be used). The application of the slave shall deliver actual input data and the application of the master shall provide output data.

In the optional ‘Bootstrap’ state the application of the slave shall be able to accept persistent settings downloaded with the FoE protocol.

The ESM defines four states, which shall be supported:

- Init,
- Pre-Operational,
- Safe-Operational, and
- Operational.

All state changes are possible except for the ‘Init’ state, where only the transition to the ‘Pre-Operational’ state is possible and for the ‘Pre-Operational’ state, where no direct state change to ‘Operational’ exists.

State changes are normally requested by the master. The master requests a write to the AL Control register which results in a Register Event ‘AL Control’ indication in the slave. The slave shall respond to the change in AL Control through a local AL Status write service after a successful or a failed state change. If the requested state change failed, the slave shall respond with the error flag set.

EtherCAT devices can either be so called simple devices or complex devices. Their behaviour regarding AL control request and AL control response is different. While complex slaves reset the AL Error Flag if possible on receive of an AL Acknowledge flag (device emulation inactive) simple slaves will copy the Acknowledge flag to the AL Error flag (device emulation active).

Despite of the different behaviour a master should have the possibility to initialize the network by a broadcast command. Hence, it shall be allowed to reset all devices by sending a broadcast INIT request with Ack Flag set to false (AL Control register = 0x0001). Complex devices shall then reset the Error Flag.

In case of an error first the AL Status Code shall be set and then the Error Flag has to be set. After clearing the Error Flag the AL Status Code should be cleared, too. The master shall ignore the AL Status Code if the Error Flag is clear.

In case of a state transition from AL state Op to SafeOp with Error the output SyncManager shall be disabled. The input SyncManager shall only be disabled in case of an error which results in invalid inputs (input error or synchronization error).

The Output-SyncManager shall be re-enabled if the error was acknowledged and there is no output error remaining.

The Input-SyncManager shall be re-enabled if the error was acknowledged and there is no input error remaining.

The Bootstrap state is optional and there is only a transition from or to the Init state. The only purpose of this state is to download the device's firmware. In Bootstrap state the mailbox is active but restricted to the FoE protocol.

ESM is specified in Figure 47.

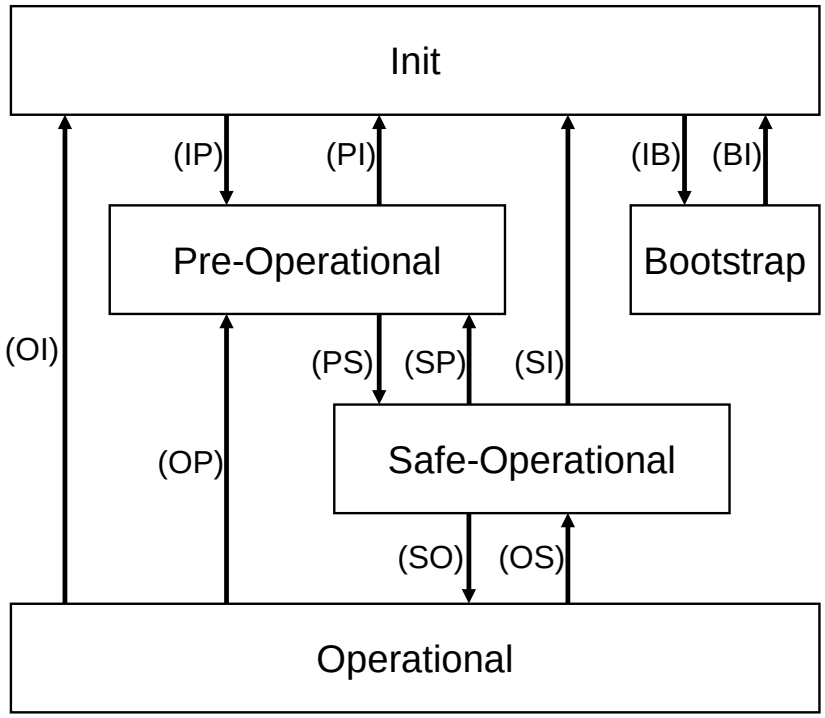


Figure 47 – ESM Diagramm

The local management services are related to the transitions in the ESM, as specified in Table 94. If there is more than one service related to the transition, the slave's application will process all of the related services.

Table 94 – State transitions and local management services

state transition	local management service
IP	Start Mailbox Communication
PI	Stop Mailbox Communication
PS	Start Input Update
SP	Stop Input Update
SO	Start Output Update
OS	Stop Output Update
OP	Stop Output Update, Stop Input Update
SI	Stop Input Update, Stop Mailbox Communication
OI	Stop Output Update, Stop Input Update, Stop Mailbox Communication
IB	Start Bootstrap Mode
BI	Restart Device

6.4.1.2 ESM States

6.4.1.2.1 Init

The ‘Init’ state defines the root of the communication relationship between the master and the slave in application layer. No direct communication between the master and the slave on application layer is possible. The master uses the ‘Init’ state to initialize a set of configuration register of the ESC. If the slave supports mailbox services, the corresponding sync manager configurations are also done in the ‘Init’ state.

6.4.1.2.2 Pre-Operational

In the ‘Pre-Operational’ state the mailbox is active if the slave supports the optional mailbox. Both the master and the slave can use the mailbox and the appropriate protocols to exchange application specific initializations and parameters. No process data communication is possible in this state.

6.4.1.2.3 Safe-Operational

In the ‘Safe-Operational’ state the application of the slave shall deliver actual input data without manipulating the output data. The outputs shall be set to their “safe state”.

6.4.1.2.4 Operational

In the ‘Operational’ state the application of the slave shall deliver actual input data and application of the master shall deliver actual output data.

6.4.1.2.5 Bootstrap

In the optional ‘Bootstrap’ state the application of the slave shall be able to accept a new firmware downloaded with the FoE protocol.

6.4.1.3 Primitive definitions

6.4.1.3.1 Primitives exchanged between DL and ESM

Table 95 shows the service primitives including their associated parameters issued by the ESM and received by the DL.

Table 95 – Primitives issued by ESM to DL

Primitive name	Associated parameters	Functions
AL State Change.req	AL Status Application Specific AL Status Code	refer to Service Definition Type 12 Fieldbus IEC 61158-5-12
AL Control.rsp(+)	AL State AL Status Code	refer to Service Definition Type 12 Fieldbus IEC 61158-5-12
AL Control.rsp(-)	AL State AL Status Code	refer to Service Definition Type 12 Fieldbus IEC 61158-5-12

Table 96 shows the service primitives including their associated parameters issued by the DL received by the ESM.

Table 96 – Primitives issued by DL to ESM

Primitive name	Associated parameters	Functions
AL Control.ind	AL Control State Ack Flag ID Request	refer to Service Definition Type 12 Fieldbus IEC 61158-5-12

6.4.1.3.2 Primitives exchanged between Application and ESM

Table 97 shows the service primitives including their associated parameters issued by the AL and received by the ESM.

Table 97 – Primitives issued by Application to ESM

Primitive name	Associated parameters	Functions
Stop Input		application stops update of process data
SM Change		Sync Manager Configuration change (can be enabled/disabled locally or issued by communication)

Table 98 shows the service primitives including their associated parameters issued by the ESM received by the AL.

Table 98 – Primitives issued by ESM to Application

Primitive name	Associated parameters	Functions
START_MBX_HANDLER		Start Mailbox Communication INIT->PREOP: Start Mailbox Communication by enabling SyncManager)
STOP_MBX_HANDLER		Stop Mailbox Communication PREOP->INIT: Stop Mailbox Communication by disabling SyncManager)
START_INPUT_HANDLER		Start Input Update PREOP->SAFEOP: Start Input Update to the ESC by enabling SyncManager) Start Output Update from the ESC by enabling SyncManager)
STOP_INPUT_HANDLER		Stop Input Update SAFEOP->PREOP: Stop Input Update to the ESC by disabling SyncManager) Stop Output Update from the ESC by disabling SyncManager) Stop watchdog
START_LOCAL_OUTPUT_HANDLER		Start local Output Update SAFEOP->OP: Set Outputs to received process values
STOP_LOCAL_OUTPUT_HANDLER		Stop local Output Update OP->SAFEOP: Set Outputs to safe values
ID_INFO		If ID Request AND IdSupported then AL_STATUS_CODE= ID value ID_FLAG = 1 else ID_FLAG = 0 end_if

6.4.1.3.3 ESM Variables

The Table 99 defines the variables used in the ESM.

Table 99 – ESM Variables

Vaiable name	Description
bootSupported	Bootstrap mode is supported by slave
dcNotAccepted	the device does not support DCs and does not accept DC settings, either. If DC settings are made the device will go into Error state. This means Sync Control register 0x0981 shall be 0.
dcRequired	device requires DC settings (Bits 0x0981:00 to 0x0981:02 shall be set accordingly). Other modes such as “Freerun” and “Synchronous with SM event” are not supported
dcRunning	DCs are running and used for synchronization between master and slave application.
dcSupported	Device supports at least one OpMode which uses Distributed Clocks
IdSupported	Device supports Explicit Device Identification using Register 0x0134
localErrorCode	original reason of a previous local error which resulted in disabeling SyncManager channels
localErrorFlag	flag of local application indicating a local error causing that SyncManagers were disabled. Used in case of an error resulting in a state change from Op to SafeOp
pllRunning	local application is synchronized with DC event and master application.
safeOp2OpTimeoutTimer	timeout for state change from SafeOp to Op
dcEventReceived	Slave has received the Sync0/Sync1 event at least once
waitForPllRunning	wait until local PLL has locked with master cycle (wait until pllRunning = TRUE). This time shall be smaller than the SafeOp2OpTimeout
wdEnabled	The watchdog behavior is disabled if at least the watchdog time (R400, R420) is equal zero, additionally the watchdog behavior can be disabled if the watchdog bit of the SyncManager is disabled (SyncManager register +0x04.06=0) (because when using the watchdog functionality of the ESC, this watchdog is only enabled if the watchdog bit in the Sync Manager is set) TRUE: watchdog behaviour is enabled FALSE: watchdog behaviour is disabled If a slave has only inputs wdEnabled may always be FALSE
readyForOP	Internal variable which is set when the device is ready to be switched to OP in Non-DC-Mode, this should happen when the outputs have been received within the watchdog time if enabled or during SafeOp state when the watchdog is disabled. However there can be another behavior due to legacy reasons.
smEventReceived	Internal variable which indicates that the SM-event was received

6.4.1.3.4 ESM Macros

The Table 100 defines the macros used in the ESM.

Table 100 – ESM macros

Macro name	Description
SM_SETTINGS_0_1_MATCH	local function that checks requested Mailbox-SyncManager settings against local settings
SM_SETTINGS_2_TO_n_MATCH	local function that checks requested Process data SyncManager

Macro name	Description
	settings against local settings For Slaves without Mailbox this can be SM 0 to n
DC_ACTIVATED	local application is synchronized. NOTE AssignActivate shall be used with a logical AND operation with 0x0701 and be either 0x0300 or 0x0700 (all other bits can be set as required)
DISABLE_SM_CHANNEL	SyncManager 2 shall be disabled. Only if there is no Output SyncManager or an input error occurs the Input SyncManager shall be disabled while SyncManager 2 remains enabled.
ENABLE_SM_CHANNEL	SyncManagers shall be enabled
RESTART_WD	If watchdog is enabled: Watchdog of ESC is started. Additionally, a local watchdog timer on the host controller may be started.

6.4.1.3.5 ESM Functions

The Table 101 defines the functions used in the ESM.

Table 101 – ESM functions

Function name	Description
Output Event	primitve issued from DL to ESM: Event that indicates arrival of new output data
WD Expired	(Local) WD is expired
SM_Chg	primitve issued from DL to ESM. Event service of DL to indicate a change in the SM settings
AL_State Change.req (AL Status, AL Status Code)	primitve issued from Application to ESM: Slave application indicates a state change
PLL Running Event	local event that indicates that the master is now synchronized to the local application (outputs are sent by the master and received by the slave before the DC Event)
SafeOp2OpTimeoutTimer Expired	event indicating that the local state change timeout has expired
Start SafeOp2OpTimeoutTimer	local timeout for SafeOp to Op timeout is started
AckSM_Chg	SyncManager Change Event shall be acknowledged by the slave
Stop Inp	Local Function of AL to disable Input Update
SM_SETTINGS_OBJECT_SYNC_TYPE_MATCH	local function that checks requested Synchronization settings with local properties

6.4.1.3.6 Parameters of primitives

The parameters used with the primitives exchanged between the DL, ESM and the Application are described in IEC 61158-5-12.

6.4.1.4 ESM State Table

Table 102 contains the complete description of the ESM state machine.

Table 102 – ESM state table

#	Current State	Event /Condition =>Action	Next State
1.1	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_CONTROL_STATE = STATE_INIT ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
1.2	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
2	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
3	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP and SM_SETTINGS_0_AND_1_MATCH => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_PREOP START_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
4	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
5	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and BOOT_SUPPORTED and SM_SETTINGS_0_AND_1_MATCH => AL_ERROR_FLAG = 0 AL_STATE = STATE_BOOT AL_STATUS_CODE = 0 START_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	BOOT
6	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)**	INIT

#	Current State	Event /Condition =>Action	Next State
		/(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and BOOT_SUPPORTED and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x15 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	
7	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and not BOOT_SUPPORTED => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x13 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
8	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = STATE_SAFEOP OR AL_CONTROL_STATE = STATE_OP) => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
9	INIT	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 L_STATE = STATE_INIT AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
10	INIT	**SM_Chg** => ignore	INIT
11.1	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER ID_INFO AL Control.rsp(=/+) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
11.2	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
12	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT	INIT

#	Current State	Event /Condition =>Action	Next State
		=> AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
13	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
14.1	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and ((not DC_ACTIVATED and not dcRequired) or (DC_ACTIVATED and not dcNotAccepted and not dcSupported)) => dcRunning = FALSE AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_SAFEOP START_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
14.2	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and DC_ACTIVATED and not dcNotAccepted and dcSupported => dcRunning = TRUE AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_SAFEOP START_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
14.3	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and ((DC_ACTIVATED and dcNotAccepted) or (not DC_ACTIVATED and dcRequired)) => ID_FLAG = 0 AL_STATUS_CODE = 0x30 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
17	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATE = STATE_PREOP	PREOP

#	Current State	Event /Condition =>Action	Next State
		AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	
18	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = STATE_BOOT or AL_CONTROL_STATE = STATE_OP) => ID_FLAG = 0 AL_STATE = STATE_PREOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
19	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 AL_STATE = STATE_PREOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
20.1	PREOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH => AckSM_Chg	PREOP
20.2	PREOP	**SM_Chg** / AL_ERROR_FLAG = 1 => ignore	PREOP
21	PREOP	**SM_Chg** /AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_MBX_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	INIT
22.1	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and .ACK_FLAG = 0) and AL_CONTROL = STATE_INIT => AL_ERROR_FLAG = 0 (AL_STATUS_CODE = 0) AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
22.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and .ACK_FLAG = 0) and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
23	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)**	INIT

#	Current State	Event /Condition =>Action	Next State
		/(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
24	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
25.1	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and (not localErrorFlag and .ACK_FLAG = 1)) and AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ENABLE_SM_CHANNEL ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
25.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /((AL_ERROR_FLAG = 0 or (localErrorFlag and .ACK_FLAG = 1)) and AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and readyForOP and (not dcRunning or pllRunning) and not localErrorFlag => AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_OP ENABLE_SM_CHANNEL START_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	OP
26.4	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and dcRunning and not pllRunning and not localErrorFlag => ENABLE_SM_CHANNEL waitForPllRunning = TRUE Start SafeOpT2OpTimeoutTimer ID_INFO	SAFEOP

#	Current State	Event /Condition =>Action	Next State
		AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
26.5	SAFEOP	**PLL Running Event** /waitForPllRunning => AL_STATE = STATE_OP AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 START_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	OP
26.6	SAFEOP	**PLL Running Event** /not waitForPllRunning => ignore	SAFEOP
26.7	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /waitForPllRunning and not dcEventReceived => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x2D AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.8	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /waitForPllRunning and dcEventReceived => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x32 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.9	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /not waitForPllRunning => ignore	SAFEOP
27.1	SAFEOP	AL_Control.ind (AL Control State, Ack Flag, ID Request) /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and not readyForOP and (not dcRunning or pllRunning) and not localErrorFlag => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x1B AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
27.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and localErrorFlag => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = localErrorCode AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
29	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)**	SAFEOP

#	Current State	Event /Condition =>Action	Next State
		/(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	
30	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
31.1	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and SM_SETTINGS_2_TO_n_MATCH => AckSM_Chg	SAFEOP
31.2	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 1 => ignore	SAFEOP
32	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	PREOP
33	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_INPUT_HANDLER STOP_MBX_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	INIT
34.1	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 DISABLE_SM_CHANNEL AL_STATE = STATE_SAFEOP AL_ERROR_FLAG = TRUE localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	SAFEOP
34.2	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 0 and localErrorFlag =>	SAFEOP

#	Current State	Event /Condition =>Action	Next State
		ENABLE_SM_CHANNEL localErrorFlag = FALSE localErrorCode = 0	
34.3	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_PREOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_INPUT_HANDLER AL_STATE = STATE_PREOP AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	PREOP
34.4	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_MBX_HANDLER STOP_INPUT_HANDLER AL_STATE = STATE_INIT AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	INIT
34.5	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_INIT or STATE_PREOP) and AL_ERROR_FLAG = 0) or AL_STATE = STATE_OP or STATE_BOOT or unknown => ignore	SAFEOP
35.1	SAFEOP	**Output Event** /wdEnabled => RESTART_WD	SAFEOP
37	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	INIT
38	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
39	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_SAFEOP STOP_LCOAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
40	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)**	OP

#	Current State	Event /Condition =>Action	Next State
		/AL_CONTROL_STATE = STATE_OP => ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
42	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_BOOT => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 DISABLE_SM_CHANNEL STOP_LOCAL_OUTPUT_HANDLER AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
43	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = unknownState => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 DISABLE_SM_CHANNEL STOP_LCOAL_OUTPUT_HANDLER AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
44	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and SM_SETTINGS_2_TO_n_MATCH => AckSM_Chg	OP
45	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP STOP_LCOAL_OUTPUT_HANDLER STOP_INPUT_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	PREOP
46	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_LOCAL_OUTPUT_HANDLER STOP_INPUT_HANDLER STOP_MBX_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	INIT
47.1	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 1 => STOP_LOCAL_OUTPUT_HANDLER DISABLE_SM_CHANNEL AL_STATE = STATE_SAFEOP localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE	SAFEOP

#	Current State	Event /Condition =>Action	Next State
		AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	
47.2	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_PREOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER AL_STATE = STATE_PREOP localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	PREOP
47.3	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER AL_STATE = STATE_INIT localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	INIT
47.4	OP	**AL State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_SAFEOP or STATE_PREOP or STATE_INIT) and AL_ERROR_FLAG = 0) or AL_CONTROL_STATE = STATE_BOOT or unknown => ignore	OP
48	OP	**Output event** => RESTART_WD Update Outputs	OP
49	OP	**WD expired** => ID_FLAG = 0 AL_STATUS_CODE = 0x1B AL_ERROR_FLAG = 1 AL_STATE = STATE_SAFEOP STOP_LOCAL_OUTPUT_HANDLER DISABLE_SM_CHANNEL AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	SAFEOP
51	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER AL Control.rsp(+) (AL State, AL Status Code)	INIT
52	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /AL_CONTROL_STATE = STATE_BOOT => AL_ERROR_FLAG = 0 AL Control.rsp(+) (AL State, AL Status Code)	BOOT
53.1	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /(AL_CONTROL_STATE = STATE_PREOP or AL_CONTROL_STATE = STATE_SAFEOP or AL_CONTROL_STATE = STATE_OP) => AL_STATUS_CODE = 0x11	INIT

#	Current State	Event /Condition =>Action	Next State
		AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code)	
53.2	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /(AL_CONTROL_STATE = STATE_PREOP or AL_CONTROL_STATE = STATE_SAFEOP or AL_CONTROL_STATE = STATE_OP) => ignore	BOOT
54.1	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /(AL_CONTROL_STATE = unknownState) => AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code)	INIT
54.2	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ignore	BOOT
55	BOOT	**SM_Chg** /SM_SETTINGS_0_AND_1_MATCH => ignore	BOOT
56.1	BOOT	**SM_Chg** /not SM_SETTINGS_0_AND_1_MATCH => AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_MBX_HANDLER AL Status Changed.req (AL state, AL staus code)	INIT
56.2	BOOT	**SM_Chg** /not SM_SETTINGS_0_AND_1_MATCH => ignore	BOOT
57	BOOT	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => STOP_MBX_HANDLER AL_STATE = STATE_INIT AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code)	INIT
58	BOOT	**AL_State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_INIT and AL_ERROR_FLAG = 0) or AL_STATE = STATE_PREOP or STATE_SAFEOP or STATE_OP or STATE_BOOT or unknown => ignore	BOOT

6.4.2 Mailbox handler state machine

6.4.2.1 Description

The mailbox handler is responsible for the coordination of master and slave regarding mailbox operation. Mailbox writes are forwarded to the specific machines and responses will be put to the read mailbox.

As there is no specific state orientated service, there is no state table.

The tasks or the mailbox handler are

– mapping of write mailbox services to protocol handler
– queuing of service request to read mailbox
– confirming transmittal of read mailbox.

The Protocol handler can be

– CoE state machine
– EoE state machine
– FoE state machine
– profile specific state machine.

6.4.2.2 Primitives exchanged between DL, Mailbox handler and Protocol Handler

Table 103 shows the service primitives including their associated parameters issued by the Mailbox handler and received by the DL.

Table 103 – Primitives issued by Mailbox handler to DL

Primitive name	Associated parameters	Functions
Mailbox Read Upd.req	Length Address Channel Priority Type Cnt Service Data	Refer to Service Definition Type 12 Fieldbus IEC 61158-3-12

Table 104 shows the service primitives including their associated parameters issued by the DL received by the Mailbox handler.

Table 104 – Primitives issued by DL to Mailbox handler

Primitive name	Associated parameters	Functions
Mailbox Write.ind	Length Address Channel Priority Type Cnt Service Data	Refer to Service Definition Type 12 Fieldbus IEC 61158-3-12
Mailbox Read Upd.cnf	success	Refer to Service Definition Type 12 Fieldbus IEC 61158-3-12

Table 105 shows the service primitives including their associated parameters issued by the Protocol handler and received by the Mailbox handler. The TYPE prefix is derived from type parameter of the corresponding DL service primitive.

Table 105 – Primitives issued by Protocol handler to Mailbox handler

Primitive name	Associated parameters	Functions
TYPE Mailbox Read Upd.req	Length Address Channel Priority Service Data	Refer to Mailbox Read Upd Service Definition Type 12 Fieldbus IEC 61158-3-12

Table 106 shows the service primitives including their associated parameters issued by the Mailbox handler received by the Protocol handler. The TYPE prefix is derived from type parameter of the corresponding DL service primitive.

Table 106 – Primitives issued by Mailbox handler to Protocol handler

Primitive name	Associated parameters	Functions
TYPE Mailbox Write.ind	Length Address Channel Priority Service Data	Refer to Service Definition Type 12 Fieldbus IEC 61158-3-12
TYPE Mailbox Read Upd.cnf	success	Refer to Mailbox Read Upd Service Definition Type 12 Fieldbus IEC 61158-3-12

6.4.3 CoE state machine

6.4.3.1 Description

The CoE state machine is responsible for processing CoE services.

The main task is to execute the service requests and provide the response either

– as single frame,
– or as sequence of frames (info services),
– or as single frame with more follows indication,
– or as abort for erroneous termination of the service.

6.4.3.2 Primitive definitions

6.4.3.2.1 Primitives exchanged between Mailbox Handler and CoESM

The primitives between CoESM and Mailbox Handler are described in 6.4.2.2.

6.4.3.2.2 Primitives exchanged between Application and CoESM

Table 107 shows the service primitives including their associated parameters issued by the AL and received by the CoESM. The SDO Info services respresents a group of services (Get OD list, Get object description, Get entry description) that are distinguished by the opcode parameter.

Table 107 – Primitives issued by Application to CoESM

Primitive name	Associated parameters	Functions
SDO Download Expedited.rsp	Success Address Index Subindex	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Download Normal.rsp	Success Address Index Subindex	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Download SDO Segment.rsp	Success Address Toggle	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Upload Expedited.rsp	Success Address Index Subindex Size	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Primitive name	Associated parameters	Functions
	Data	
SDO Upload Normal.rsp	Success Address Index Subindex Complete Size Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Upload SDO Segment.rsp	Success Address Toggle More Follows Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Abort SDO Transfer.req	Address Index Subindex Reason	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Info Service.rsp	Address Opcode Incomplete Fragments Left Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Info Seg Service.req	Address Opcode Incomplete Fragments Left Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Emergency Service.req	Address Error Code Error Register Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Table 108 shows the service primitives including their associated parameters issued by the CoESM received by the AL.

Table 108 – Primitives issued by CoESM to Application

Primitive name	Associated parameters	Functions
SDO Download Expedited.ind	Address Index Subindex Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Download Normal.ind	Address Index Subindex Complete Size Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Download SDO Segment.ind	Address Toggle More Follows Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Upload Expedited.ind	Address Index Subindex	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Upload Normal.ind	Address Index	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Primitive name	Associated parameters	Functions
	Subindex	
Upload SDO Segment.ind	Address Toggle	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Abort SDO Transfer.req	Address Index Subindex Reason	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
SDO Info Service.ind	Address Opcode Incomplete Fragments Left Size Data	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

6.4.3.2.3 Parameters of primitives

The parameters used with the primitives exchanged between CoESM and Application are described in IEC 61158-5-12.

6.4.3.3 CoESM State Table

Table 109 contains the complete description of the CoESM state machine.

Table 109 – CoESM state table

#	Current State	Event /Condition =>Action	Next State
1	OFF	**START MAILBOX** => Seg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** =>	OFF
5	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service != 2 && Service_Data.Service != 8 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service == 2 && Service_Data.Command_Specifier > 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 1 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
8	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 1 =>	DWLE

#	Current State	Event /Condition =>Action	Next State
		Size = 4 - Service_Data.Data_Set_Size Index = Service_Data.Index SubIndex = Service_Data.SubIndex Data = Service_Data.Data SDO Download Expedited.ind (Address, Index, Subindex , Size, Data)	
9	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
10	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 0 => Seg = (Service Data.Complete_Size - (Length - 10)) Toggle = FALSE Size = Length - 10 Complete Size = Service Data.Complete_Size Index = Service_Data.Index SubIndex = Service_Data.SubIndex Data = Service_Data.Data SDO Download Normal.ind (Address, Index, Subindex, Complete Size, Size, Data)	DWLN
11	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length < 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
12	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >=10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
13	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
14	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Index = Service_Data.Index SubIndex = Service_Data.SubIndex SDO Upload Expedited.ind (Address, Index, Subindex)	UPLE
15	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Length = 4	ERR

#	Current State	Event /Condition =>Action	Next State
		Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
16	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Index = Service_Data.Index SubIndex = Service_Data.SubIndex SDO Upload Normal.ind (Address, Index, Subindex)	UPLN
17	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 3 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
18	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 3 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
19	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 8 && Service_Data.Service == 8 && Service_Data.Opcode == 1,3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
20	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 8 && Service_Data.Service == 8 && Service_Data.Opcode == 1,3 => Size = Length -6 Opcode = Service_Data.Opcode Incomplete = Service_Data.Incomplete FragmentsLeft = Service_Data.FragmentsLeft SDO Info Service.ind (Address, Opcode, Incomplete, Fragments Left, Size, Data)	INF
21	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 10 && Service_Data.Service == 8 && Service_Data.Opcode == 5 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
22	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 10&& Service_Data.Service == 8 && Service_Data.Opcode == 5 => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ SDO Info Service.ind (Address, Opcode, Incomplete, Fragments Left, Size, Data)	INF
23	IDLE	**CoE Read Upd.cnf (success)** => ignore	IDLE
24	IDLE	**Emergency Service.req (Address, Error Code, Error Register, Size, Data)** => Length = 10	WUPD

#	Current State	Event /Condition =>Action	Next State
		Service_Data.Service = 1 Service_Data.Error Code = Error Code Service_Data.Error Register = Error Register Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
25	IDLE	**other application service primitives** => ignore	IDLE
26	DWLE	**START MAILBOX** => ignore	DWLE
27	DWLE	**STOP MAILBOX** => CANCEL SERVICE	OFF
28	DWLE	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
29	DWLE	**SDO Download Expedited.rsp (Success, Address, Index, Subindex)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 3 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
30	DWLE	**other application service primitives** => ignore	DWLE
31	DWLN	**START MAILBOX** => ignore	DWLN
32	DWLN	**STOP MAILBOX** => CANCEL SERVICE	OFF
33	DWLN	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
34	DWLN	**SDO Download Normal.rsp (Success, Address, Index, Subindex)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 3 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
35	DWLN	**other application service primitives** => ignore	DWLN

#	Current State	Event /Condition =>Action	Next State
36	DWLS	**START MAILBOX** => ignore	DWLS
37	DWLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
38	DWLS	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
39	DWLS	**Download SDO Segment.rsp (Success, Address, Toggle)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 1 Service_Data.Toggle, Toggle = !Toggle CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
40	DWLS	**other application service primitives** => ignore	DWLS
41	UPLE	**START MAILBOX** => ignore	UPLE
42	UPLE	**STOP MAILBOX** => CANCEL SERVICE	OFF
43	UPLE	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
44	UPLE	**SDO Upload Expedited.rsp (Success, Address, Index, Subindex, Size, Data)** => Length = 10 Service_Data.Service = 3 Service_Data.Command_Specifier = 2 Service_Data.Transfer Type = 1 Service_Data.Data Set Size = 4 - Size Service_Data.Index = Index Service_Data.Subindex = Subindex Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
45	UPLE	**other application service primitives** => ignore	UPLE
46	UPLN	**START MAILBOX** => ignore	UPLN
47	UPLN	**STOP MAILBOX** => CANCEL SERVICE	OFF
48	UPLN	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10	WUPD

#	Current State	Event /Condition =>Action	Next State
		Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
49	UPLN	**SDO Upload Normal.rsp (Success, Address, Index, Subindex, Complete Size, Size, Data)** => Seg = Size - Complete Size Toggle = FALSE Length = 10 + Size Service_Data.Service = 3 Service_Data.Command_Specifier = 2 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
50	UPLN	**other application service primitives** => ignore	UPLN
51	UPLS	**START MAILBOX** => ignore	UPLS
52	UPLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
53	UPLS	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
54	UPLS	**Upload SDO Segment.rsp (Success, Address, Toggle, More Follows, Size, Data)** => Seg = (Seg -Size) if (Size > 7) then Service_Data.SegDataSize = 0 else Service_Data.SegDataSize =7-Size Length = Size + 3 + Service_Data.SegDataSize Service_Data.Service = 3 Service_Data.Command_Specifier = 0 Service_Data.Toggle = Toggle Service_Data.Data = Data Service_Data.More Follows = More Follows CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
55	UPLS	**other application service primitives** => ignore	UPLS
56	INF	**START MAILBOX** => ignore	INF
57	INF	**STOP MAILBOX** => CANCEL SERVICE	OFF
58	INF	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason	WUPD

#	Current State	Event /Condition =>Action	Next State
		Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
59	INF	**SDO Info Service.rsp (Address, Opcode, Incomplete, Fragments Left, Size, Data)** => if FragmentsLeft >0) then Seg = 0x80000000 else Seg = 0 Length = Size + 6 Service_Data.Service = 8 Service_Data.Opcode = Opcode Service_Data. Incomplete = Incomplete Service_Data.Data = Data Service_Data. Fragments Left = Fragments Left CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
60	INF	**SDO Info Seg Service.req (Address, Opcode, Incomplete, Fragments Left, Size, Data)** => if FragmentsLeft >0) then Seg = 0x80000000 else Seg = 0 Length = Size + 6 Service_Data.Service = 8 Service_Data.Opcode = Opcode Service_Data. Incomplete = Incomplete Service_Data.Data = Data Service_Data. Fragments Left = Fragments Left CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
61	WUPD	**START MAILBOX** =>	WUPD
62	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
63	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0 =>	IDLE
64	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0x80000000 && (Fragments Left == 0) => Seg = 0	IDLE
65	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0x80000000 && (Fragments Left > 0) =>	INF
66	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg >0 && Seg < 0x80000000 =>	WSEG
67	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg < 0 && Seg > 0x80000000 =>	WSEG
69	ERR	**START MAILBOX** =>	ERR
70	ERR	**STOP MAILBOX** => RESET MAILBOX	OFF
71	ERR	**ERR Mailbox Read Upd.cnf (success)** =>	IDLE
73	WSEG	**START MAILBOX** =>	WSEG
74	WSEG	**STOP MAILBOX** =>	OFF
75	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service != 2 => Length = 4	ERR

#	Current State	Event /Condition =>Action	Next State
		Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
76	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service == 2 && Service_Data.Command_Specifier != 0,3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
77	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length < 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE Seg = 0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
78	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 && Seg < (Length - 3 - Service_Data.SegDataSize) => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
79	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 && Seg >= (Length - 3 - Service_Data.SegDataSize) && (Service_Data.More Follows != (0 < (Seg -(Length - 3 - Service_Data.SegDataSize))) || Toggle = Service_Data.Toggle) => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
80	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 && Seg >= (Length - 3 - Service_Data.SegDataSize) && Service_Data.More Follows == (0 < (Seg -(Length - 3 - Service_Data.SegDataSize))) && !Toggle = Service_Data.Toggle => Seg = (Seg -(Length - 3 - Service_Data.SegDataSize)) Size = Length - 3 - Service_Data.SegDataSize Toggle = Service_Data.Toggle More Follows = Service_Data.More Follows Data = Service_Data.Data Download SDO Segment.ind (Address, Toggle, More Follows, Size, Data)	DWLS
81	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE	ERR

#	Current State	Event /Condition =>Action	Next State
		ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
82	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 && Toggle = Service_Data.Toggle => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
83	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 && !Toggle = Service_Data.Toggle => Toggle = Service_Data.Toggle Upload SDO Segment.ind (Address, Toggle)	UPLS
84	WSEG	**Emergency Service.req (Address, Error Code, Error Register, Size, Data)** => Length = 10 Service_Data.Service = 1 Service_Data.Error Code = Error Code Service_Data.Error Register = Error Register Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
85	WSEG	**other application service primitives** => ignore	IDLE

6.4.4 EoE state machine

6.4.4.1 Description

The EoE state machine is responsible for conveying standard Ethernet frames over Type 12. The state machine receives and transmits Type 12 PDUs either with the complete Ethernet frame or a fragment of the Ethernet frame, which can be combined to a complete frame. After the reassembly (which is out of scope of this machine) the resulting Ethernet frame can be treated as if transmitted with Ethernet without Type 12 services.

The ingress and egress rules of an Ethernet port that may be associated with the EoE services are specified in IEEE 802.1D.

General Time Stamp definitions:

- 32 bit, 1 ns resolution
- DC System Time can be used
- Time Stamp trigger is the beginning of the destination address (DA)

Time Stamp appended (TA = 1):

- Slave -> master: Time Stamp contains exact receive time
- Master -> slave: Time Stamp contains desired send time
- Slave should always append a Time Stamp if it has this feature
- Time Stamp extends frame data by 32 bit
- TA bit allowed in last fragment only (LF=1)

- If Time Stamp does not fit into the last fragment -> add a fragment
  - fill the “last” fragment with parts of the Time Stamp (LF=0, TA=0) and send a very last fragment with the rest of the Time Stamp (LF=1, TA=1)

Time Stamp requested (TR = 1):

- Response with the exact send time and the same FrameNo requested
- Response should be send as soon as possible

6.4.4.2 Primitive definitions

6.4.4.2.1 Primitives exchanged between Mailbox Handler and EoESM

The primitives between EoESM and Mailbox Handler are described in 6.4.2.2.

6.4.4.2.2 Primitives exchanged between Application and EoESM

Table 110 shows the service primitives including their associated parameters issued by the AL and received by the EoESM.

Table 110 – Primitives issued by Application to EoESM

Primitive name	Associated parameters	Functions
Initiate_EoE.req	Address, Port Time Appended Time Requested Frame Number Complete Size Last Fragment Size Data Time Stamp	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Initiate_EoE.rsp	Address, Frame Number Time Stamp	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
EoE Fragment.req	Address, Port Time Appended Frame Number Offset Last Fragment Size Data Time Stamp	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Set IP Parameter.rsp	Address, Reason	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Set Address Filter.rsp	Address, Reason	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Table 111 shows the service primitives including their associated parameters issued by the EoESM received by the AL.

Table 111 – Primitives issued by EoESM to Application

Primitive name	Associated parameters	Functions
Initiate_EoE.ind	Address, Port Time Appended Time Requested Frame Number Complete Size Last Fragment	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Primitive name	Associated parameters	Functions
	Size Data Time Stamp	
EoE Fragment.ind	Address, Port Time Appended Frame Number Offset Last Fragment Size Data Time Stamp	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Set IP Parameter.ind	Address, MAC Address IP Address Subnet Mask Default Gateway DNS Server DNS Name	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12
Set Address Filter.ind	Address, Broadcast Fowarding MAC Address Filters MAC Filter Masks	Refer to CoE Service Definition Type 12 Fieldbus IEC 61158-5-12

6.4.4.2.3 Parameters of primitives

The parameters used with the primitives exchanged between EoESM and Application are described in IEC 61158-5-12.

6.4.4.3 EoESM State Table

Table 112 contains the complete description of the EoESM state machine. The Set IP Parameter and Set Address Filter are handled in any state in such a way that the requested parameter will be changed if possible.

Table 112 – EoESM state table

#	Current State	Event /Condition =>Action	Next State
1	OFF	**START MAILBOX** => SSeg, RSeg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** => Terminate Segmented Services	OFF
5	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length < 36 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && (Service_Data.Fragment > 0 || Service_Data.CompleteSize < 2) => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)**	IDLE

#	Current State	Event /Condition =>Action	Next State
		/ Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 1 => TimeAppended = Service_Data.TimeAppended TimeRequested= Service_Data.TimeRequested Size = Length-4-TimeAppended*4 if (TimeAppend) Timestamp = Service_Data[Length -4..Length-1] Frame Number = Service_Data.Frame Number Complete Size = Service_Data.CompleteSize Fragment Number = Service_Data.Fragment LastFragment = 1 Data = Service_Data.EoE Data Initiate_EoE.ind (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)	
8	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 0 && ((Length -4) mod 32) == 0 => TimeAppended = Service_Data.TimeAppended TimeRequested= Service_Data.TimeRequested Size,SSeg = Length-4 Frame Number = Service_Data.Frame Number Fragment Number = Service_Data.Fragment Complete Size = Service_Data.CompleteSize LastFragment = 0 Data = Service_Data.EoE Data Initiate_EoE.req (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)	IDLE
9	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 0 && ((Length -4) mod 32) != 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
10	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && (Service_Data.Fragment == 0 || Service_Data.Offset*32 != SSeg) => SSeg = 0 Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
11	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 1 => SSeg = 0 TimeAppended = Service_Data.TimeAppended Size = Length-4-TimeAppended*4 if (TimeAppend) Timestamp = Service_Data[Length -4..Length-1] Frame Number = Service_Data.Frame Number Offset = Service_Data.Offset Fragment Number = Service_Data.Fragment LastFragment = 1 Data = Service_Data.EoE Data EoE_Fragment.ind (Address, Port, Time Appended, Frame Number, Offset, Last Fragment, Size, Data, Timestamp)	IDLE
12	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 0 &&	IDLE

#	Current State	Event /Condition =>Action	Next State
		((Length -4) mod 32) == 0 => SSeg = SSEG + Length-4 Frame Number = Service_Data.Frame Number Offset = Service_Data.Offset Fragment Number = Service_Data.Fragment LastFragment = 0 Data = Service_Data.EoE Data EoE_Fragment.ind (Address, Port, Time Appended, Frame Number, Offset, Last Fragment, Size, Data, Timestamp)	
13	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 0 && ((Length -2) mod 32) != 0 => SSeg = 0 Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
14	IDLE	**Initiate_EoE.req (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)** => Service_Data.TimeAppended = TimeAppended Service_Data.TimeRequested = TimeRequested Length = Size+4+TimeAppended*4 if (TimeAppend) Service_Data[Length -4..Length-1] = Timestamp Service_Data.Frame Number = Frame Number Service_Data.Fragment Number = Fragment Service_Data.Complete Size = CompleteSize Service_Data.LastFragment = LastFragment Service_Data.Port = Port Service_Data.EoE Data = Data EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
15	IDLE	**EoE_Fragment.req(Address, Port, TimeAppended, Frame Number, Offset, Last Fragment, Size, Data,Timestamp)** => Service_Data.TimeAppended = TimeAppended Length = Size+4+TimeAppended*4 if (TimeAppend) Service_Data[Length -4..Length-1] = Timestamp Service_Data.Frame Number = Frame Number Service_Data.Fragment Number = Fragment Service_Data.Complete Size = CompleteSize Service_Data.LastFragment = LastFragment Service_Data.Port = Port Service_Data.EoE Data = Data EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
16	IDLE	**EOE Read Upd.cnf (success)** => ignore	IDLE
17	IDLE	**other application service primitives** => ignore	IDLE
18	IDLE	**Initiate_EoE.rsp(Address, Frame Number, Timestamp)** => Service_Data.FrameType = 1 Service_Data.TimeAppended = 1 Service_Data.TimeStamp = Timestamp EOE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
19	WUPD	**START MAILBOX** =>	WUPD
20	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF

#	Current State	Event /Condition =>Action	Next State
21	WUPD	**EOE Read Upd.cnf (success)** =>	WUPD
22	WUPD	**EOE Read.ind** =>	IDLE
23	WUPD	**Timeout** => RESET MAILBOX	OFF
24	ERR	**START MAILBOX**=>	ERR
25	ERR	**STOP MAILBOX** => RESET MAILBOX	OFF
26	ERR	**ERR Read Upd.cnf (success)** =>	ERR
27	ERR	**ERR Read.ind** =>	IDLE
28	ERR	**Timeout** => RESET MAILBOX	OFF
29	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 8 (fits request) && Service_Data.FrameType == 2 => Extract IP Parameter from Service Data. IP Parameter Set IP Parameter.ind(Address, Mac Address, IP Adress, Subnet Mask, Default Gateway, DNS Server, DNS Name)	IPPAR
30	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length not (fits request) && Service_Data.FrameType == 2 => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = 2 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
31	IPPAR	**START MAILBOX** => ignore	IPPAR
32	IPPAR	**STOP MAILBOX** => CANCEL SERVICE	OFF
33	IPPAR	**Set IP Parameter.rsp(+) (Adress)** => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = 0 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
34	IPPAR	**Set IP Parameter.rsp(-) (Adress, Reason)** => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = Reason Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
35	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 (fits request) && Service_Data.FrameType == 4 => Extract Address Fileter from Service Data SetAddress Filter.ind(Address, Broadcast forwarding, MAC Address Filter 1..16, MAC Filter Mask 1..4)	MFLT

#	Current State	Event /Condition =>Action	Next State
36	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length not (fits request) && Service_Data.FrameType == 4 => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = 2 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
37	MFLT	**START MAILBOX** => ignore	MFLT
38	MFLT	**STOP MAILBOX** => CANCEL SERVICE	OFF
39	MFLT	**SetAddress Filter.rsp(+) (Adress)** => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = 0 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
40	MFLT	**SetAddress Filter.rsp(-) (Adress, Reason)** => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = Reason Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
41	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.FrameType != 0,2,4 => ignore	IDLE

6.4.5 FoE state machine

6.4.5.1 Description

The FoE state machine is responsible for conveying Files over Type 12. The state machine receives and transmits Type 12 PDUseither with transfer command (Write means load file to slave, Read means load file to master).

6.4.5.2 Primitive definitions

6.4.5.2.1 Primitives exchanged between Mailbox Handler and FoESM

The primitives between FoESM and Mailbox Handler are described in 6.4.2.2.

6.4.5.2.2 Primitives exchanged between Application and FoESM

Table 113 shows the service primitives including their associated parameters issued by the AL and received by the FoESM.

Table 113 – Primitives issued by Application to FoESM

Primitive name	Associated parameters	Functions
FoE Data.req	Address, Packet Number Size Data	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Ack.req	Address, Packet Number	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Error.req	Address, Error Code, Error Text	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12

Table 114 shows the service primitives including their associated parameters issued by the FoESM received by the AL.

Table 114 – Primitives issued by FoESM to Application

Primitive name	Associated parameters	Functions
FoE Write.ind	Address, Password File Name	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Read.ind	Address, Password File Name	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Data.ind	Address, Packet Number Size Data	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Ack.ind	Address, Packet Number	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12
FoE Error.ind	Address, Error Code, Error Text	Refer to FoE Service Definition Type 12 Fieldbus IEC 61158-5-12

6.4.5.2.3 Parameters of primitives

The parameters used with the primitives exchanged between FoESM and Application are described in IEC 61158-5-12.

6.4.5.3 FoESM State Table

Table 115 contains the complete description of the FoESM state machine.

Table 115 – FoESM state table

#	Current State	Event /Condition =>Action	Next State
1	OFF	**START MAILBOX** => Seg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** =>	OFF

#	Current State	Event /Condition =>Action	Next State
5	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 0 || Service_Data.OpCode> 4 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 6 && Service_Data.OpCode == 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 && Service_Data.OpCode == 2 => Password = Service_Data.Password Filename = Service_Data.FileName FoE Write.ind (Address, Password, Filename)	DWLS
8	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 6 && Service_Data.OpCode == 1 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
9	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 && Service_Data.OpCode == 1 => Password = Service_Data.Password Filename = Service_Data.FileName FoE Read.ind (Address, Password, Filename)	UPLS
10	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 3 && (Service_Data.OpCode == 3 || Service_Data.OpCode == 4) => Length = 6 Service_Data.OpCode = 5 Service_Data.Error_Code = ABT_SEQ FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
11	IDLE	**FoE Read Upd.cnf (success)** => ignore	IDLE
12	IDLE	**other application service primitives** => ignore	IDLE
13	DWLS	**START MAILBOX** => ignore	DWLS
14	DWLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
15	DWLS	**FoE Error.req (Address, ErrorCode, ErrorText)** => Seg = 0 Length = 6 + size(Error_Text) Service_Data.OpCode = 5 Service_Data.Error_Code = Error_Code Service_Data.Error_Text= Error_Text FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
16	DWLS	**FoE Ack.req (Address, Packet Number)** => Length = 6 Service_Data.OpCode = 4 Service_Data.PacketNumber = Packet Number Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	DWUPD

#	Current State	Event /Condition =>Action	Next State
17	DWLS	**other application service primitives** => ignore	DWLS
18	UPLS	**START MAILBOX** => ignore	UPLS
19	UPLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
20	UPLS	**FoE Error.req (Address, ErrorCode, ErrorText)** => Seg = 0 Length = 6 + size(Error_Text) Service_Data.OpCode = 5 Service_Data.Error_Code = Error_Code Service_Data.Error_Text= Error_Text FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
11	IDLE	**FoE Read Upd.cnf (success)** => ignore	IDLE
21	UPLS	**FoE Data.req (Address, Packet Number, Size, Data)** /Size = FullSize => Length = 6 + SIze Service_Data.OpCode = 3 Service_Data.PacketNumber = Packet Number Service_Data.Data = Data Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	UWUPD
22	UPLS	**FoE Data.req (Address, Packet Number, Size, Data)** /Size< FullSize => Length = 6 + SIze Service_Data.OpCode = 3 Service_Data.PacketNumber = Packet Number Service_Data.Data = Data Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
23	UPLS	**other application service primitives** => ignore	UPLS
24	UWUPD	**START MAILBOX** =>	UWUPD
25	UWUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
26	UWUPD	**FoE Read Upd.cnf (success)** =>	UWSEG
27	UWUPD	**Timeout** => RESET MAILBOX	OFF
28	WUPD	**START MAILBOX** =>	WUPD
29	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
30	WUPD	**FoE Read Upd.cnf (success)** =>	IDLE
31	WUPD	**Timeout** => RESET MAILBOX	OFF

#	Current State	Event /Condition =>Action	Next State
32	DWUPD	**START MAILBOX** =>	DWUPD
33	DWUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
34	DWUPD	**FoE Read Upd.cnf (success)** =>	DWSEG
35	DWUPD	**Timeout** => RESET MAILBOX	OFF
36	ERR	**START MAILBOX** =>	ERR
37	ERR	**STOP MAILBOX** => RESET MAILBOX	OFF
38	ERR	**ERR Read Upd.cnf (success)** =>	IDLE
39	ERR	**Timeout** => RESET MAILBOX	OFF
40	DWSEG	**START MAILBOX** =>	DWSEG
41	DWSEG	**STOP MAILBOX** =>	OFF
42	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode != 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
43	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
44	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq && Length = FullSize + 6 => Size = Length - 6 Packet Number = Service_Data.Packet Number Data = Service_Data.Data FoE Data.ind (Address, Packet Number, Size, Data)	DWLS
45	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq && Length < FullSize + 6 => Size = Length - 6 Packet Number = Service_Data.Packet Number Data = Service_Data.Data FoE Data.ind (Address, Packet Number, Size, Data)	IDLE
46	DWSEG	**other application service primitives** => ignore	DWSEG
47	UWSEG	**START MAILBOX** =>	UWSEG

#	Current State	Event /Condition =>Action	Next State
48	UWSEG	**STOP MAILBOX** =>	OFF
49	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode != 4 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
50	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 4 && Service_Data.PackeNumber != Seq => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
51	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 4 && Service_Data.PackeNumber == Seq => Packet Number = Service_Data.Packet Number FoE Ack.req (Address, Packet Number)	UPLS
52	UWSEG	**other application service primitives** => ignore	UWSEG

6.5 DLL mapping protocol machine (DMPM)

The services specified in IEC 61158-3-12 are directly used in the ARPMs.

Bibliography

IEC 61131-3, *Programmable controllers – Programming languages*

IEC 61158-1:2014, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-4-12, *Industrial communication networks – Fieldbus specifications – Part 4-12: Data-link layer protocol specification – Type 12 elements*

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

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

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange*

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)*

ISO 8601, *Data elements and interchange formats – Information interchange – Representation of dates and times*

ISO 15745-1, *Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description*

IETF RFC 792, *Internet Control Message Protocol;* available at <http://www.ietf.org>

_____

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-12: Spécification du protocole de la couche application – Éléments de type 12

AVANT-PROPOS

1) La Commission Électrotechnique 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 du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type.

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

La Norme internationale CEI 61158-6-12 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition parue en 2010. Cette édition constitue une révision technique. Les principales modifications par rapport à l'édition précédente sont énumérées ci-dessous:

- réparations des erreurs;
- améliorations éditoriales;
- prise en charge de l'Identification Explicite d'Appareils ajoutée en ESM (Article 6).

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*, est disponible 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.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

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

Le protocole d'application fournit le service d'application au moyen des services disponibles au niveau de la couche Liaison de données ou de la couche immédiatement inférieure. Le principal objectif de la présente norme est de définir un ensemble de règles de communication, exprimées en termes de procédures que doivent suivre les entités d'application (Application Entity, AE) homologues au moment de la communication. Ces règles de communication visent à fournir une base saine pour le développement, dans divers buts:

- en tant que guide pour les développeurs et les concepteurs;
- dans une optique d'utilisation lors de l'essai et de l'achat de matériel;
- dans le cadre d'un accord pour l'admission de systèmes dans l'environnement de systèmes ouverts;
- en tant que précision apportée à la compréhension des communications en temps critique dans le modèle OSI.

Cette norme traite, en particulier, de la communication et de l'interfonctionnement des capteurs, effecteurs et autres appareils d'automatisation. L'utilisation conjointe de la présente norme avec d'autres normes entrant dans les modèles de référence OSI ou de bus de terrain permet à des systèmes qui ne pourraient pas, sans cela, fonctionner ensemble dans toute combinaison.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-12: Spécification du protocole de la couche application – Éléments de type 12

1 Domaine d'application

1.1 Généralités

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

La présente norme fournit des éléments communs pour les communications en temps critique ou non entre des programmes d'application dans un environnement et avec un matériel d'automatisation spécifiques aux bus de terrain de Type 12. Le terme "à temps critique" 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 définit de manière abstraite le comportement, visible par un observateur externe, assuré par les différents Types de la couche Application de bus de terrain, en termes

a) de syntaxe abstraite définissant les unités de données de protocole de la couche Application, transmises entre les entités d'application en communication,

b) de syntaxe de transfert définissant les unités de données de protocole de la couche Application, transmises entre les entités d'application en communication;

c) de diagramme d'états de contexte d'application définissant le comportement de service d'application observable entre les entités d'application en communication; et

d) de diagrammes d'états de relations entre applications définissant le comportement de communication visible entre les entités d'application en communication; et.

La présente norme vise à définir le protocole mis en place pour

a) définir la représentation filaire des primitives de service définies dans la CEI 61158-5-12, et

b) définir le comportement visible de l'extérieur associé à leur transfert.

La présente norme spécifie le protocole de la couche Application de bus de terrain de la CEI, en conformité avec le modèle de référence de base OSI (ISO/CEI 7498) et avec la structure de la couche Application OSI (ISO/CEI 9545).

Les services et protocoles de la FAL sont fournis par des entités d'application (application entity, AE) de la FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un jeu d'Éléments de service application (ASE, Application Service Element) orientés objet et d'une Entité de gestion de couche (LME, Layer Management Entity) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un jeu de classes d'objets de processus d'application (APO, application process object) connexes. L'un des ASE de la FAL est un ASE de gestion qui fournit un jeu commun de services pour la gestion des instances de classes de la FAL.

Bien que ces services spécifient, du point de vue des applications, la manière dont la demande et les réponses sont émises et délivrées, ils n'incluent pas une spécification de ce que les applications qui demandent et qui répondent doivent en faire. À savoir, les aspects comportementaux des applications ne sont pas spécifiés; seule une définition des demandes et réponses qu'elles peuvent envoyer/recevoir est spécifiée. Cela permet une plus grande flexibilité aux utilisateurs de la FAL pour normaliser un tel comportement d'objet. En plus de ces services, certains services d'appui sont également définis dans la présente norme pour fournir l'accès à la FAL afin de maîtriser certains aspects de son fonctionnement.

1.2 Spécifications

La présente norme a pour objectif principal de spécifier la syntaxe et le comportement du protocole de la couche Application qui véhicule les services de la couche Application définis dans la CEI 61158-5-12.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. Ce dernier objectif explique la diversité des protocoles normalisés dans les sous-parties de 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 la couche application au sein des systèmes d'automatisation industriels.

Il n'existe pas de conformité de l'équipement à la norme de définition de service de la couche Application. En revanche, la conformité est obtenue par le biais de la mise en œuvre de cette spécification de protocoles de la 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 61158-3-12, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-12 : Définition des services de la couche liaison de données – Éléments de type 12*

CEI 61158-5-12, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-12: Définition des services de la couche application – Éléments type 12*

CEI 61158-6 (toutes les parties), *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6: Spécification du protocole de la couche application*

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

ISO/CEI 7498-3, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Dénomination et adressage*

ISO/IEC 8802-3, *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 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)

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

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

IEEE 802.1D, *IEEE standard for Local and metropolitan area networks – Common specifications – Media access control (MAC) Bridges;* disponible à l'adresse <http://www.ieee.org>

IEEE 802.1Q, *IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks Bridges;* disponible à l'adresse <http://www.ieee.org>

IETF RFC 768, *User Datagram Protocol*; disponible à l'adresse <http://www.ietf.org>

IETF RFC 791, *Internet Protocol darpa internet program protocol specification;* disponible à l'adresse <http://www.ietf.org>

IETF RFC 826, *An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*; disponible à l'adresse <http://www.ietf.org>

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 et définitions du modèle de référence

La présente norme est basée en partie sur les concepts développés dans l'ISO/CEI 7498-1 et l'ISO/CEI 7498-3 et utilise les termes suivants y définis:

3.1.1 entités (N) correspondantes
entités d'AL correspondantes (N=7) [ISO/CEI 7498-1]

3.1.2 entité (N)
entité d'AL (N=7) [ISO/CEI 7498-1]

3.1.3 couche (N)
couche AL (N=7) [ISO/CEI 7498-1]

3.1.4 gestion de couche [ISO/CEI 7498-1]

3.1.5 entités homologues [ISO/CEI 7498-1]

3.1.6 nom primitif [ISO/CEI 7498-3]

3.1.7 protocole d'AL [ISO/CEI 7498-1]

3.1.8 unité de données de protocole d'AL [ISO/CEI 7498-1]

3.1.9 réinitialisation [ISO/CEI 7498-1]

3.1.10 acheminement [ISO/CEI 7498-1]

3.1.11 segmentation [ISO/CEI 7498-1]

3.1.12 service (N)
service d'AL (N=7) [ISO/CEI 7498-1]

3.1.13 unité de données de service d'AL [ISO/CEI 7498-1]

3.1.14 transmission simplex d'AL [ISO/CEI 7498-1]

3.1.15 sous-système d'AL [ISO/CEI 7498-1]

3.1.16 gestion-systèmes [ISO/CEI 7498-1]

3.1.17 données d'utilisateur d'AL [ISO/CEI 7498-1]

3.2 Termes et définitions de convention pour les services

La présente norme utilise également les termes suivants définis dans l'ISO/CEI 10731 tels qu'ils s'appliquent à la couche liaison de données:

3.2.1 acceptant

3.2.2 service asymétrique

3.2.3 (primitive) "confirm";
(primitive) "requestor.deliver"

3.2.4 (primitive) "deliver"

3.2.5 primitive de service d'AL;primitive

3.2.6 fournisseur de service d'AL

3.2.7 utilisateur de service d'AL

3.2.8 (primitive) "indication";
(primitive) "acceptor.deliver"

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

3.2.10 demandeur

3.2.11 réponse (primitive);
(primitive) "acceptor.submit"

3.2.12 (primitive) "submit"

3.2.13 service symétrique

3.3 Définitions relatives à la couche Application

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

3.3.1
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.3.2
objets d'applications
classes d'objets multiples qui gèrent et assurent un échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

3.3.3
processus d'application
partie d'une application distribuée sur un réseau, située sur un appareil et associée à une adresse non ambiguë

3.3.4
relation entre applications
association de type coopératif entre deux ou plus de deux invocations d'entités d'application, dans un but d'échange d'informations et de coordination de leur action conjointe

Note 1 à l'article: Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit à la suite d'activités de préconfiguration.

3.3.5
attribut
description d'une caractéristique, visible par un observateur externe, d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent des informations relatives aux portions variables d'un objet. En règle générale, ils fournissent des informations d'état ou régissent l'action d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.3.6
esclave de base
appareil esclave qui prend en charge uniquement l'adressage physique des données

3.3.7
comportement
indication de la manière dont un objet réagit à des événements particuliers

3.3.8
bit
unité d'information consistant en un 1 ou un 0.

Note 1 à l'article: il s'agit de la plus petite unité de données qui puisse être transmise.

3.3.9
canal
représentation d'un objet de gestion physique ou logique d'un esclave pour la commande du transport des données

3.3.10
client
1) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

2) initiateur d'un message auquel un serveur réagit

3.3.11
synchronisation des horloges
représentation d'une succession d'interactions visant à la synchronisation des horloges de tous les récepteurs de signaux d'horloge par une horloge maîtresse

3.3.12
objet de communication
composant gérant et assurant un échange de messages sur le réseau lors de l'exécution

3.3.13
connexion
liaison logique entre deux objets d'application au sein du même appareil ou dans des appareils différents

3.3.14
consommation
acte de recevoir des données provenant d'un fournisseur

3.3.15
consommateur
nœud ou récepteur qui reçoit des données provenant d'un fournisseur

3.3.16
chemin de transport
flux unidirectionnel d'unités APDU, d'un bout à l'autre d'une relation entre applications

3.3.17
cyclique
relatif à des événements qui se répètent d'une manière régulière et répétitive

3.3.18
donnée
terme générique servant à se référer à toute information transportée sur un bus de terrain

3.3.19
cohérence de données
moyen pour une émission et un accès cohérents de l'objet de donnée d'entrée ou de sortie entre client et serveur et au sein du client et du serveur

3.3.20
data type
relation entre les valeurs et l'encodage des données du type correspondant selon les définitions de la CEI 61131-3

3.3.21
objet «data type» (type de données)
entrée dans le dictionnaire d'objets indiquant un type de données

3.3.22
passerelle par défaut
appareil avec au moins deux interfaces dans deux sous-réseaux IP différents agissant comme routeur pour un sous-réseau

3.3.23
appareil
entité physique connectée au réseau de terrain composée d'au moins un élément de communication (l'élément de réseau) et qui peut avoir un élément de commande et/ou un élément final (transducteur, actionneur, etc.)

3.3.24
profil d'appareil
ensemble d'informations et de fonctionnalités dépendantes de l'appareil qui garantit la cohérence entre les appareils similaires appartenant au même type d'appareil

3.3.25
information de diagnostic
toute donnée disponible au niveau du serveur à des fins de maintenance

3.3.26
horloges réparties
méthode permettant de synchroniser les esclaves et maintenir une base de temps globale

3.3.27
erreur
discordance entre une valeur ou un état calculé(e), observé(e) ou mesuré(e) et la valeur ou l'état spécifié(e) ou théoriquement correct(e)

3.3.28
classe d'erreurs
regroupement général pour des définitions d'erreurs connexes et des codes d'erreurs correspondants

3.3.29
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.3.30
événement
instance d'un changement de conditions

3.3.31
unité de gestion de mémoire de bus de terrain
fonction qui établit une ou plusieurs correspondances entre les adresses logiques et la mémoire physique

3.3.32
entité d'unité de gestion de mémoire de réseau de terrain
élément simple de l'unité de gestion de mémoire de réseau de terrain: une correspondance entre un espace d'adresses logiques cohérent et un emplacement cohérent de la mémoire physique

3.3.33
trame
synonyme déconseillé de DLPDU

3.3.34
esclave complet
appareil esclave qui prend en charge à la fois l'adressage physique et l'adressage logique des données

3.3.35
indice
adresse d'un objet au sein d'un processus d'application

3.3.36
interface
frontière partagée entre deux unités fonctionnelles, définies par les caractéristiques fonctionnelles, caractéristiques de signal ou autres caractéristiques selon le cas approprié

3.3.37
little endian (petit boutiste)
style de représentation des données de champs multi-octet où l'octet de poids le plus faible est émis en premier

3.3.38
maître
appareil qui commande le transfert de données sur le réseau et lance l'accès des esclaves aux supports en envoyant des messages et qui constitue l'interface au système de commande

3.3.39
mapping
correspondance entre deux objets de telle manière que l'un des objets soit partie intégrante de l'autre objet

3.3.40
paramètres de mapping
ensemble de valeurs définissant la correspondance entre des objets d'application et des objets de données de processus

3.3.41
support
câble, fibre optique ou autre moyen par lequel les signaux de communication sont émis entre deux points ou plus

Note 1 à l'article: Le terme "media" est utilisé comme pluriel de "medium".

3.3.42
message
série ordonnée d'octets censés acheminer de l'information

Note 1 à l'article: Normalement utilisé pour acheminer de l'information entre homologues au niveau de la couche application.

3.3.43
réseau
ensemble de nœuds reliés par un certain type de support de communication, notamment des répétiteurs intermédiaires, ponts, routeurs et passerelles de couche inférieure

3.3.44
nœud
a) simple entité de DL telle qu'elle apparaît sur une liaison locale

b) point d'extrémité d'une liaison dans un réseau ou un point auquel deux ou plusieurs liaisons se rencontrent

[SOURCE dérivée de la CEI 61158-2]

3.3.45
objet
représentation abstraite d'un composant particulier au sein d'un appareil

Note 1 à l'article: Un objet peut être

a) une représentation abstraite des capacités d'un appareil. Les objets peuvent être constitués de tout ou partie des composants suivants:

1) données (informations qui changent dans le temps),

2) configuration (paramètres de comportement),

3) méthodes (actions qui peuvent être réalisables à l'aide des données et de la configuration);

b) un ensemble de données connexes (sous la forme de variables) et de méthodes (procédures) pour opérer sur les données en question qui ont une interface et un comportement clairement définis

3.3.46
dictionnaire d'objets
structure de données, avec adresses définies par indice et sous-indice, qui contient la description des objets de data type, des objets de communication et des objets applicatifs

3.3.47
donnée de processus
ensemble d'objets d'application destinés à être transférés de façon cyclique ou acyclique à des fins de mesure et de commande

3.3.48
objet de données de processus
structure décrite par des paramètres de mapping contenant une ou plusieurs entités de données de processus

3.3.49
producteur
nœud ou source qui envoie des données à un ou plusieurs consommateurs

3.3.50
ressource
capacité de traitement ou d'information

3.3.51
segment
ensemble d'un maître réel et d'un ou plusieurs esclaves

3.3.52
serveur
objet qui fournit des services à un autre objet (client)

3.3.53
service
opération ou fonction qu'un objet et/ou une classe d'objets réalise ou assure à la demande d'un autre objet et/ou d'une autre classe d'objets

3.3.54
esclave
entité de DL accédant au support uniquement après avoir été initiée par l'esclave précédent ou le maître

3.3.55
sous-indice
sous-adresse d'un objet dans le dictionnaire d'objets

3.3.56
gestionnaire de synchronisation
ensemble d'éléments de commande servant à coordonner l'accès à des objets utilisés simultanément

3.3.57
canal de gestionnaire de synchronisation
simples éléments de commande servant à coordonner l'accès à des objets utilisés simultanément

3.3.58
commutateur
pont MAC tel que défini dans l'IEEE 802.1D

3.4 Symboles et abréviations communs

AL-	Préfixe relatif à la couche application
ALE	AL-entity (Entité d'AL, l'instance locale active de la couche application)
AL	AL-layer (Couche application)
APDU	AL-protocol-data-unit (Unité de données de protocole d'AL)
ALM	AL-management (Gestion d'AL)
ALME	AL-management entity (Entité de gestion AL, l'instance active locale de la gestion d'AL)
ALMS	AL-management service (Service de gestion d'AL)
ALS	AL-service (Service d'AL)
ALSDU	Application Layer Service Data Unit (Unité de données de protocole d'application)
DL	Préfixe relatif à la couche liaison de données
FIFO	First-in first-out («Premier entré, premier sorti», méthode de mise en file d'attente)
OSI	Open Systems Interconnection (Interconnexion des systèmes ouverts)
PhL	Ph-layer (couche Physique)

QoS Quality of service (qualité de service)

3.5 Autres symboles et abréviations

ASE Application Service Element (Élément de service application)

AR Application Relationship (Relation d'applications)

CAN Controller Area Network (Gestionnaire de réseau de communication)

CiA CAN in Automation (CAN en automation)

CoE CAN application protocol over Type 12 services (Protocole d'application CAN sur services de Type 12)

CSMA/CD Carrier sense multiple access with collision detection (Accès multiple par surveillance du signal et détection de collision)

DC Distributed Clocks (Horloges distribuées)

DNS Domain name system (Système de nom de domaine) (serveur pour la résolution des noms dans les réseaux IP)

E²PROM Electrically erasable programmable read only memory (Mémoire morte à reprogrammation électrique)

EoE Ethernet tunneled over Type 12 services (Services Ethernet en tunnel sur Type 12)

ESC Type 12 slave controller (Appareil de commande d'esclave de Type 12)

FCS Frame Check Sequence (Séquence de contrôle de trame)

FMMU Fieldbus Memory Management Unit (Unité de gestion de mémoire de bus de terrain)

FoE File access with Type 12 services (Accès fichier avec les services de Type 12)

HDR Header (En-tête)

ID Identifier (Identificateur)

IETF Internet Engineering Task Force (Groupe de travail d'ingénierie Internet)

IP Internet Protocol (Protocole Internet)

LAN Local Area Network (Réseau local)

MAC Medium Access Control (Commande d'accès au support)

OD Object dictionary (Dictionnaire d'objets)

PDI Physical Device Internal Interface (Interface interne d'appareil physique), ensemble d'éléments permettant l'accès aux services DLL à partir de la couche AL

PDO Process Data Object (Objet de données de processus)

RAM	Random access memory (Mémoire vive)
Rx	Receive (recevoir)
SDO	Service Data Object (Objet de données de service)
SII	Slave information interface (Interface d'information esclave)
SM	Synchronization manager (Gestionnaire de synchronisation)
SoE	Servo drive profile with Type 12 services (Profil de servocommande avec services de Type 12)
SyncM	Synchronization manager (Gestionnaire de synchronisation)
TCP	Transmission Control Protocol (Protocole de commande de transport)
Tx	Transmit (émettre)
UDP	User Datagram Protocol (Protocole datagramme d'utilisateur) (IETF RFC 768)
VoE	Profile specific services (Services spécifiques à un profil)
WKC	Working counter (Compteur en fonction)

3.6 Conventions

3.6.1 Concept général

Les services sont spécifiés dans la CEI 61158-5-12. La spécification de services définit les services fournis par la couche DLL de Type 12. La présente norme internationale décrit le mapping de ces services avec l'ISO/CEI 8802-3.

La présente norme utilise les conventions descriptives données dans l'ISO/CEI 10731.

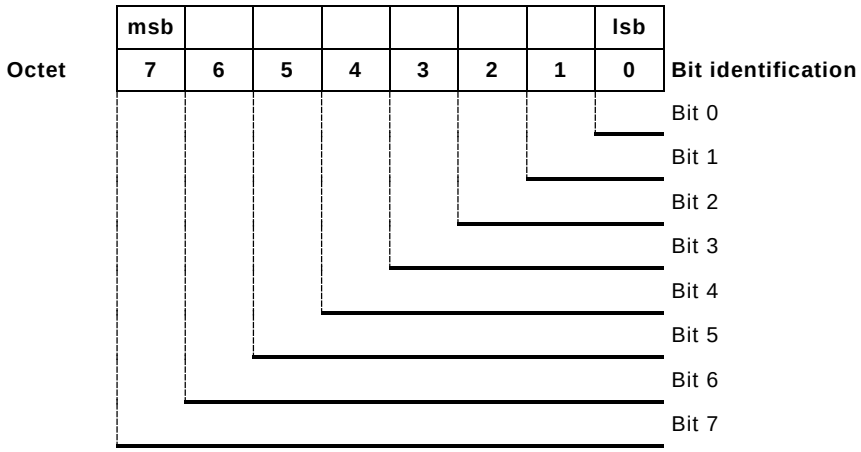
3.6.2 Conventions d'encodage des bits et octets réservés

Il est admis d'utiliser le terme "Reserved" (réservé) pour décrire des bits compris dans des octets ou des octets entiers. Il convient que tous les bits ou octets réservés soient mis à zéro du côté émetteur et ils ne doivent pas être soumis à essai du côté récepteur, sauf si cela est explicitement énoncé ou si les bits ou octets réservés sont vérifiés par un diagramme d'états.

Il est également permis d'utiliser le terme "Reserved" pour indiquer que certaines valeurs, comprises dans la plage d'un paramètre, sont réservées en vue d'extensions futures. Dans ce cas, il convient de ne pas utiliser les valeurs réservées du côté émetteur et elles ne doivent pas être soumises en essai du côté récepteur, sauf si cela est explicitement énoncé ou si les valeurs réservées sont vérifiées par un diagramme d'états.

3.6.3 Conventions de codages communs des octets de champs spécifiques

Il est admis que les unités APDU contiennent des champs spécifiques qui transmettent les informations de manière primitive et condensée. Ces champs doivent être codés dans l'ordre conformément à la Figure 1.



Légende

Anglais	Français
msb	bit de poids fort
lsb	bit de poids faible
Octet	Octet
Bit identification	Identification des bits

Figure 1 – Structure commune de champs particuliers

Les bits peuvent être réunis dans des groupes de bits. Chaque bit ou groupe de bits doit avoir son identification de bit pour adresse (par exemple, bit 0, bit 1 à bit 4). Sa position dans l'octet doit être conforme à la figure ci-dessus. Il est admis d'utiliser des alias pour chaque bit ou groupe de bits, ou de les marquer comme réservés. Le groupement des bits doit être réalisé par ordre croissant, sans espaces. Il est permis de représenter les valeurs d'un groupe de bits sous la forme d'une valeur binaire, décimale ou hexadécimale. Cette valeur ne doit désigner que les bits groupés et ne peut représenter que l'octet entier si les 8 bits sont groupés. Les valeurs décimales ou hexadécimales doivent être converties en valeurs binaires de manière que le bit portant le numéro le plus élevé au sein du groupe corresponde au bit de poids le plus fort parmi les bits groupés.

EXEMPLE Description et associations pour l'octet de champ spécifique

Bit 0: réservé.

Bit 1 à bit 3: Reason_Code La valeur décimale 2 de Reason_Code correspond à une erreur générale.

Bit 4 à bit 7: doit toujours être défini sur un.

L'octet construit selon la description ci-dessus se présente comme suit:

(msb) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(lsb) Bit 0 = 0.

Cette combinaison de bits est représentée par la valeur d'octet 0xf4.

3.6.4 Conventions de syntaxe abstraite

Les éléments de syntaxe de couche AL relatifs à la structure des unités de données de protocole (Process Data Unit, PDU) sont décrits tels qu'illustrés dans l'exemple dans le Tableau 1.

La colonne "Partie de la trame" indique l'élément qui sera remplacé par cette reproduction.

La colonne "Champ de données" indique le nom des éléments.

La colonne "Data type" correspond au type du symbole terminal.

La colonne "Valeur/Description" contient la valeur constante ou la signification du paramètre.

Tableau 1 – Exemple de description d'élément PDU

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête CoE	Numéro	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x00: taille des données (1..4) non spécifiée 0x01: taille des données dans la taille de l'ensemble de données spécifiée
	Type de transfert	Unsigned1	0x01: transfert express
	Taille de l'ensemble de données	Unsigned2	0x00: 4 octets de données 0x01: 3 octets de données 0x02: 2 octets de données 0x03: 1 octet de données
	Accès complet	Unsigned1	0x00
	Spécificateur de commande	Unsigned3	0x01: déclencher la demande de téléchargement

Les types d'attributs de nature informative sont décrits par des notations en langage C (ISO/CEI 9899) conformément à la Figure 2. BYTE et WORD sont des éléments de type caractère non signé et entier court non signé.

```
typedef struct
{
  Unsigned8     Type;
  Unsigned8     Revision;
  Unsigned16    Build;
  Unsigned8     NoOfSuppFmmuChannels;
  Unsigned8     NoOfSuppSyncManChannels;
  Unsigned8     RamSize;
  Unsigned8     Reserved1;
  unsigned      FmmuBitOperationNotSupp:  1;
  unsigned      Reserved2:                7;
  unsigned      Reserved3:                8;
} TDLINFORMATION;
```

Figure 2 – Exemple de description de type

Les attributs d'un objet sont décrits sous la forme illustrée dans le Tableau 2.

La colonne "Sous-indice" décrit un élément de l'objet.

La colonne "Description" contient une chaîne de nom correspondant à l'attribut pertinent.

La colonne "Data type" contient le type de l'élément.

"M/O/C" indique si l'attribut est obligatoire (M), facultatif (O) ou dépend du paramétrage d'autres attributs (C).

La colonne "Accès" décrit le droit d'accès à l'élément concerné. "R" indique un droit d'accès en lecture, "W", un droit d'accès en écriture. Ces données peuvent être complétées par l'état de la couche AL à l'emplacement où s'applique le droit d'accès.

La colonne "Mapping PDO" signale la possibilité de mapper l'attribut avec l'objet TxPDO ou RxPDO, à moins qu'elle n'indique que ce paramètre ne peut être mappé.

La colonne "Valeur" contient la valeur constante et/ou la signification du paramètre.

Tableau 2 – Exemple de description d'attribut

Sous-indice	Description	Data type	M/O/C	Accès	Mapping PDO	Valeur
0	Nombre d'entrées	UNSIGNED8	M	R	Non	4
1	ID du fournisseur	UNSIGNED32	M	R	Non	Attribué de manière non ambiguë par l'ETG
2	Code produit	UNSIGNED32	M	R	Non	Attribué de manière non ambiguë par le fournisseur
3	Numéro de révision	UNSIGNED32	M	R	Non	Attribué de manière non ambiguë par le fournisseur
4	Numéro de série	UNSIGNED32	M	R	Non	attribué de manière unique à cet appareil par le fournisseur

3.6.5 Conventions relatives aux diagrammes d'états

Les séquences de protocoles sont décrites au moyen de diagrammes d'états.

Dans les diagrammes d'états, les états sont représentés par des cadres et les changements d'état, par des flèches. Les noms d'états et de changements figurant dans le diagramme d'états correspondent aux noms qui apparaissent dans la liste (texte) des changements d'état.

La liste des transitions d'état est structurée comme suit (voir également le Tableau 3).

– La première ligne contient le nom du changement.
– La deuxième ligne définit l'état actuel.
– La troisième ligne contient un événement facultatif, suivi des conditions avec un "/" pour premier caractère de ligne, puis des actions, avec un "=>" pour premier caractère de ligne.
– La dernière ligne contient l'état suivant.

Si l'événement survient et que les conditions sont remplies, le changement se déclenche, à savoir que les actions sont réalisées et que le diagramme d'états passe à l'état suivant.

Le Tableau 3 présente une description d'un diagramme d'états. La signification des éléments d'une description de diagramme d'états est présentée dans le Tableau 4.

Tableau 3 – Éléments de la description d'un diagramme d'états

N	État actuel	Événement /Condition => Action	État suivant

Tableau 4 – Description des éléments d'un diagramme d'états

Élément de la description	Signification
État actuel État suivant	nom des états donnés
N	nom ou numéro du changement d'état
Événement	nom ou description de l'événement
/Condition	expression booléenne. Le caractère "\" de début de ligne ne fait pas partie de la condition.
=> Action	liste des tâches à effectuer et des invocations de services ou de fonctions. Le caractère "=>" de début de ligne ne fait pas partie de l'action.

Les conventions utilisées dans les diagrammes d'états sont présentées dans le Tableau 5.

Tableau 5 – Conventions utilisées dans les diagrammes d'états

Convention	Signification
=	la valeur d'un élément, à gauche, est remplacée par la valeur d'un élément, à droite. Si un élément à droite est un paramètre, il provient de la primitive indiquée comme événement d'entrée.
axx	nom d'un paramètre, si "a" est une lettre. EXEMPLE Identifier = reason signifie que la valeur d'un paramètre "reason" est attribuée à un paramètre appelé "Identifier".
"xxx"	indique une chaîne visible fixe. EXEMPLE Identifier = "abc" signifie que la valeur "abc" est attribuée à un paramètre appelé "Identifier".
nnn	si tous les éléments sont des chiffres, l'élément représente une constante numérique, exprimée au format décimal.
0xnn	si tous les éléments nn sont des chiffres, l'élément représente une constante numérique, exprimée au format hexadécimal.
==	condition logique indiquant qu'un élément à gauche est égal à un élément à droite.
<	condition logique indiquant qu'un élément à gauche est inférieur à l'élément à droite.
>	condition logique indiquant qu'un élément à gauche est supérieur à l'élément à droite.
!=	condition logique indiquant qu'un élément à gauche est différent d'un élément à droite.
&&	"ET" logique
||	"OU" logique
!	"SAUF" logique
+ - * /	opérateurs arithmétiques
;	séparateur d'expressions

Il est fortement recommandé de se référer aux paragraphes traitant des définitions d'attributs, des fonctions locales et des définitions d'unités FDL-PDU pour comprendre les machines de protocole. Nous supposons que les lecteurs possèdent une connaissance suffisante de ces définitions et savent les utiliser sans explications complémentaires.

D'autres constructions définies dans la notation du langage C (ISO/CEI 9899) peuvent être utilisées pour décrire les conditions et les actions.

4 Spécification de protocoles de la couche application

4.1 Principe de fonctionnement

Le Type 12 est une technologie Ethernet en temps réel visant à maximaliser l'utilisation de la bande passante Ethernet duplex. La commande d'accès au support utilise le principe de maître/esclave selon lequel le nœud maître (typiquement le système de commande) envoie les trames Ethernet aux nœuds esclaves, qui extraient des données de ces trames et insèrent des données dans ces trames.

Du point de vue de l'Ethernet, un segment de Type 12 est un appareil Ethernet pris séparément qui reçoit et envoie des trames Ethernet de la norme ISO/CEI 8802-3. Toutefois, cet appareil Ethernet ne se limite pas à un seul contrôleur Ethernet avec un microprocesseur en aval, mais peut comporter un grand nombre d'appareils esclaves. Ces appareils traitent directement les trames entrantes et en extraient les données d'utilisateur pertinentes ou y insèrent des données avant de transférer la trame à l'appareil esclave suivant. Le dernier appareil esclave au sein du segment renvoie la trame complètement traitée et, donc, elle est retournée par le premier appareil esclave vers le maître comme trame de réponse.

Cette procédure utilise le mode duplex d'Ethernet: les deux sens de communication sont exploités de façon indépendante. Il est admis d'établir une communication directe, sans commutateur, entre un appareil maître et un segment de Type 12 comprenant un ou plusieurs appareils esclaves.

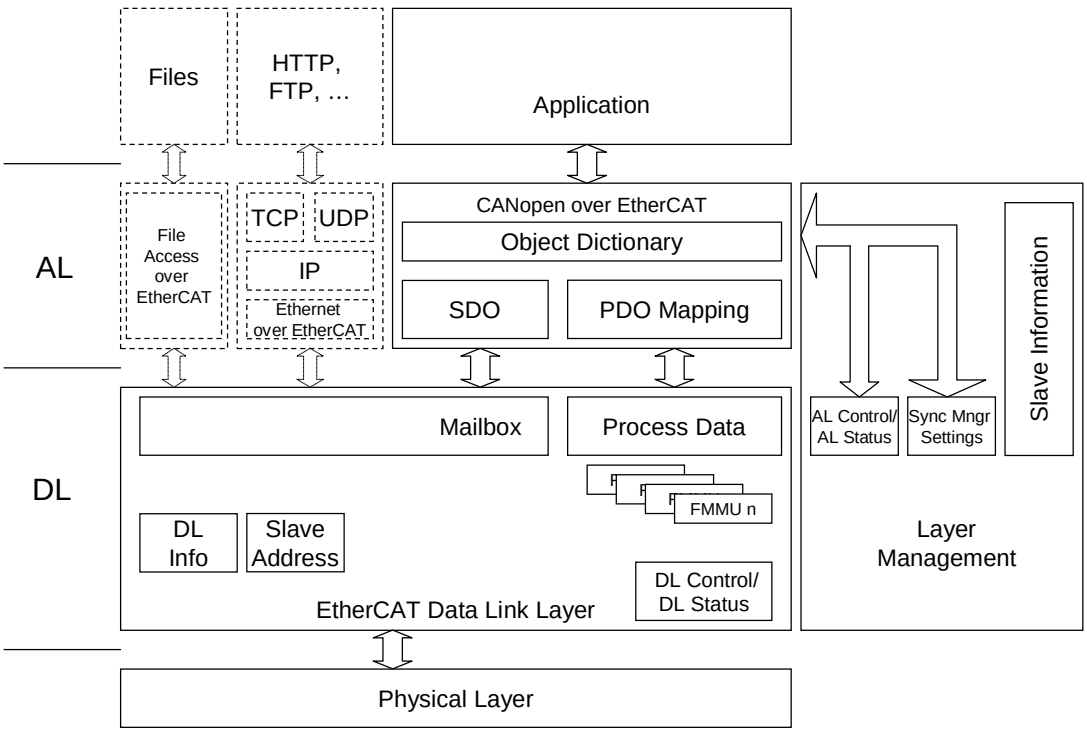
Les systèmes de communication industriels doivent satisfaire à différentes exigences en termes des caractéristiques de transmission de données. Les données de paramètres sont transférées de manière acyclique et en grandes quantités. De ce fait, les exigences de temps sont relativement non critiques et la transmission est habituellement déclenchée par le système de commande. Les données de diagnostic sont également transférées de manière acyclique et sont événementielles, mais les exigences de temps sont plus drastiques et la transmission est habituellement déclenchée par un appareil périphérique.

D'autre part, les données de processus sont typiquement transférées de manière cyclique avec des temps de cycle différents. Les exigences de temps sont le plus drastiques pour la communication de données de processus. Le Type 12 prend en charge des services et protocoles divers pour respecter ces exigences différentes.

4.2 Modèle de référence des nœuds

4.2.1 Mapping avec le modèle de référence de base OSI

Le Type 12 est décrit à l'aide des principes, de la méthodologie et du modèle de l'ISO/CEI 7498-1. Le modèle OSI fournit une approche stratifiée aux normes de communications et, par ce biais, des couches peuvent être mises au point et modifiées de manière indépendante. La spécification du Type 12 définit la fonctionnalité de haut en bas d'une pile OSI complète, ainsi que certaines fonctions destinées aux utilisateurs de la pile. Les fonctions des couches OSI intermédiaires, c'est-à-dire les couches 3 à 6, sont consolidées soit dans la couche DLL du Type 12, soit dans la couche AL du Type 12. De même, les fonctions communes aux utilisateurs de la couche d'application de bus de terrain peuvent être fournies par la couche d'application Type 12 afin de simplifier l'utilisation (voir Figure 3).



Légende

Anglais	Français
Files	Fichiers
HTTP, FTP, ...	HTTP, FTP, ...
Application	Application
AL	AL (couche application)
File Access over EtherCAT	Accès fichier sur EtherCAT
TCP	TCP
UDP	UDP
CANopen over EtherCAT	CANopen sur EtherCAT
Object Dictionary	Dictionnaire d'objets
IP	IP
Ethernet over EtherCAT	Ethernet sur EtherCAT
SDO	SDO
PDO Mapping	Mapping d'objets de données de processus
Mailbox	Boîte à lettres (Boîte d'échange)
Process Data	Données de processus
AL Control/AL Status	Commande de couche application/Statut de couche application
SyncM Settings	Paramètres du gestionnaire de synchronisation
Slave Information	Informations d'esclave
DL	Liaison de données
DL Info	Informations de DL
Slave Address	Adresse d'esclave
FMMU	Unité de gestion de mémoire de réseau de terrain
EtherCAT Data Link Layer	Couche liaison de données EtherCAT
DL Control/DL Status	Commande de DL/Statut de DL
Layer Management	Gestionnaire de couche
Physical Layer	Couche physique

Figure 3 – Modèle de référence du nœud esclave

4.2.2 Caractéristiques de la couche liaison de données

La couche DLL fournit une prise en charge élémentaire pour les communications en temps critique entre appareils. Le terme "à temps critique" sert à décrire des applications ayant 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 couche DLL a pour tâche de calculer, de comparer et de générer la séquence de contrôle de trame, ainsi que d'assurer des communications en extrayant des données de la trame Ethernet et/ou en intégrant des données à cette trame. Ces opérations sont réalisées en fonction des paramètres de couche DLL conservés dans des emplacements de mémoire prédéfinis. Les données d'application sont rendues disponibles à la couche application en mémoire physique, soit dans une configuration de boîte à lettres, soit au sein d'une section de données de processus.

Par ailleurs, certaines structures de données de couche DLL seront utilisées pour permettre une coordination des interactions entre maître et esclaves, comme les paramètres AL Control, Status, Event et Sync Manager Settings.

4.2.3 Structure de la couche Application

La couche AL se compose des éléments suivants.

- Une entité temps-réel (obligatoire)
- Une entité assurant le traitement des protocoles TCP/UDP/IP et des protocoles associés (facultative)
- Un utilitaire d'accès aux fichiers (facultatif)
- Une unité de gestion (obligatoire)

La couche AL utilise les services fournis par la couche DLL de Type 12 pour transporter les données de service de la couche Application.

5 Description de la syntaxe de la FAL

5.1 Principes de codage

La couche AL utilise des objets de couche DLL définis dans la 61158-4-12.

5.2 Data types et règles d'encodage

5.2.1 Description générale des data types et des règles d'encodage

Le format et la signification de ces données doivent être connus du producteur et du ou des consommateurs pour qu'ils puissent échanger des données qui aient du sens. La présente spécification modélise cette exigence par le biais du concept de data types.

Les règles d'encodage 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 types de données numériques, le codage est au format petit-boutiste (voir Tableau 6).

Les data types et les règles d'encodage doivent être valables pour les services et protocoles de couche DLL, ainsi que pour les services et protocoles de couche AL spécifiés. Les règles d'encodage de la trame Ethernet sont spécifiées dans l'ISO/CEI 8802-3. L'unité DLSDU d'Ethernet est une chaîne d'octets. L'ordre d'émission au sein des octets dépend des règles d'encodage des adresses MAC et de la couche PhL.

5.2.2 Encodage d'une valeur booléenne

a) L'encodage d'une valeur booléenne doit être de type primitif. La chaîne ContentsOctets doit être formée d'un seul octet.

b) Si la valeur booléenne est FALSE, la valeur de la chaîne ContentsOctets doit être 0 (zéro). Si la valeur booléenne est TRUE, la valeur de la chaîne ContentsOctets doit être 0xff.

5.2.3 Encodage d'une valeur Time Of Day avec et sans valeur d'indication de date

a) L'encodage d'une valeur TimeOfDay (heure), avec et sans valeur d'indication de date, doit être de type primitif.

b) La valeur de la chaîne ContentsOctets doit être égale aux octets de la valeur des données, comme montré à la Figure 4.

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
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}	since
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	midnight
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}	only with date indication
	msb								

Légende

Anglais	Français
bits	bits
octets	octets
number of milliseconds since midnight	nombre de millisecondes depuis minuit
number of days since 1984-01-01	nombre de jours depuis le 1984-01-01
only with date indication	seulement avec une indication de date
msb	bit de poids fort

Figure 4 – Encodage d'une valeur Time of Day

5.2.4 Encodage d'une valeur TimeDifference avec et sans valeur d'indication de date

a) L'encodage d'une valeur TimeDifference, avec et sans valeur d'indication de date, doit être de type primitif.

b) La valeur de la chaîne ContentsOctets doit être égale aux octets de la valeur des données, comme montré à la Figure 5.

bits	7	6	5	4	3	2	1	0	
octets									
1	2^{31}	2^{30}	2^{29}	2^{28}	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}	only with date indication
	msb								

Légende

Anglais	Français
bits	bits
octets	octets
milliseconds	millisecondes

Anglais	Français
days	jours
only with date indication	seulement avec une indication de date
msb	bit de poids fort

Figure 5 – Encodage d'une valeur Time Difference

5.2.5 Syntaxe de transfert des suites de bits

Dans une optique d'émission, 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 $8(k - 1) < n \leq 8k$. Alors, b est transféré dans k octets assemblés comme indiqué dans le Tableau 6. Les bits b_i, $i \geq n$ de l'octet portant le numéro le plus élevé sont des bits sans signification.

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

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

Tableau 6 – Syntaxe de transfert des séquences binaires

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

EXEMPLE

Bit 9	...	Bit 0
10b	0001b	1100b
0x2	0x1	0xC
		= 0x21C

La suite de bits $b = b_0$ .. b_9 = 0011 1000 01_b représente un entier de type Unsigned10, de valeur 0x21C, et est transférée sous la forme de deux octets: d'abord 0x1C, puis 0x02.

5.2.6 Encodage d'une valeur Unsigned Integer (entière non signée)

Les données du data type de base Unsignedn ont des valeurs entières naturelles. La plage de valeurs s'étend de 0 à 2^n-1. Les données sont représentées par des suites de bits de longueur n. La suite de bits

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

se voit affecter la valeur

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

La suite de bits commence (à gauche) par l'octet de poids le plus faible.

EXEMPLE La valeur 266 = 0x10A de type Unsigned16 est transférée sous la forme de deux octets: d'abord 0x0A, puis 0x01.

Les types de données Unsignedn sont spécifiés dans le Tableau 7. Des data types non signés tels que Unsigned1 à Unsigned7 et Unsigned9 à Unsigned15 seront également utilisés. Dans ce cas, l'élément suivant commence au premier bit libre (voir 3.6.3).

Tableau 7 – Syntaxe de transfert du type de données Unsignedn

Numéro d'octet	1.	2.	3.	4.	5.	6.	7.	8.
Unsigned8	$b_7..b_0$							
Unsigned16	$b_7..b_0$	$b_{15}..b_8$						
Unsigned32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
Unsigned64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

BYTE est utilisé comme UNSIGNED8, WORD est utilisé comme UNSIGNED16.

5.2.7 Encodage d'une valeur Signed Integer (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 la forme de suites de bits de longueur n. La suite de bits

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

se voit affecter la valeur

$$\text{Integern}(b) = b_{n-2}\times 2^{n-2} + ... + b_1\times 2^1 + b_0\times 2^0 \quad \text{si } b_{n-1} = 0$$

puis, grâce au calcul du complément à deux,

$$\text{Integern}(b) = -\ \text{Integern}(\text{^b}) - 1 \qquad \text{si } b_{n-1} = 1$$

NOTE La suite de bits commence (à gauche) par le bit de poids faible.

EXEMPLE La valeur –266 = 0xFEF6 de type Integer16 est transférée sous la forme de deux octets: d'abord 0xF6, ensuite 0xFE.

Le transfert des types de données Integern est tel que spécifié dans le Tableau 8. Des data types entiers tels que Integer1 à Integer7 et Integer9 à Integer15 seront également utilisés. Dans ce cas, l'élément suivant commence au premier bit libre (voir 3.6.3).

Tableau 8 – Syntaxe de transfert du type de données Integern

Numéro d'octet	1.	2.	3.	4.	5.	6.	7.	8.
Integer8	$b_7..b_0$							
Integer16	$b_7..b_0$	$b_{15}..b_8$						
Integer32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
Integer64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

5.2.8 Encodage d'une valeur Floating Point (virgule flottante)

Float32 ::= OCTET STRING SIZE (4) -- Simple précision selon l'ISO/IEC/IEEE 60559

Float64 ::= OCTET STRING SIZE (8) -- Double précision selon l'ISO/IEC/IEEE 60559

5.2.9 Encodage d'une valeur VisibleString

a) L'encodage d'une valeur VisibleString à longueur variable doit être de type primitif.

b) Il ne comprend ni champ de longueur, ni symbole de fin; la longueur est encodée de manière implicite.

c) La chaîne ContentsOctets doit être une suite d'octets. L'élément de chaîne de gauche est encodé dans le premier octet, puis l'on passe au deuxième octet, et ainsi de suite jusqu'au dernier octet (compris), qui encode l'octet de droite de la chaîne ContentsOctets.

5.2.10 Encodage d'une chaîne de valeur Unicode

a) L'encodage d'une valeur UnicodeString à longueur variable doit être de type primitif.

b) Il ne comprend pas de champ de longueur; la longueur est encodée de manière implicite.

c) La chaîne ContentsOctets doit être une suite d'entiers non signés. L'élément de chaîne de gauche est encodé dans le premier entier non signé, puis l'on passe au deuxième entier non signé, et ainsi de suite jusqu'au dernier entier non signé (compris), qui encode l'octet de droite de la chaîne ContentsOctets.

5.2.11 Encodage d'une valeur de chaîne d'octets

a) L'encodage d'une valeur OctetString à longueur variable doit être de type primitif.

b) Il ne comprend pas de champ de longueur; la longueur est encodée de manière implicite.

c) La chaîne ContentsOctets doit être une suite d'octets. L'élément de chaîne de gauche est encodé dans le premier octet, puis l'on passe au deuxième octet, et ainsi de suite jusqu'au dernier octet (compris), qui encode l'octet de droite de la chaîne ContentsOctets.

5.2.12 Encodage de GUID

La donnée du data type de base GUID (Globally Unique Identifier, c'est-à-dire identificateur globalement unique) sont un numéro de référence unique utilisé comme un identificateur. La valeur d'un GUID est stockée sous la forme d'un entier 128 bits.

5.3 Codage des relations AR

5.3.1 Demande AL Control (indication)

Les types d'attributs de la demande AL Control sont décrits à la Figure 6.

```
typedef struct
{
  unsigned          State:           4;
  unsigned          Acknowledge:     1;
  unsigned          Reserved:        3;
  unsigned          ApplSpecific:    8;
} TALCONTROL;
```

Figure 6 – Structure de la demande AL Control

La demande AL Control est mappée avec un service d'écriture de couche DLL dans l'objet de registre de commande de l'utilisateur de couche DLL, ainsi que dans le registre R2 défini par la CEI 61158-3-12. Le codage de la demande AL Control est spécifié dans le Tableau 9.

Tableau 9 – AL Control - Description

Paramètre	Registre de l'utilisateur DLL	Data type	Valeur
State (état)	R1	Unsigned4	1: initialisation (Initialization, INIT) 2: Pré-exploitation (Pre-Operational, PreOP) 3: amorçage (Bootstrap, Boot) 4: exploitation sûre (Safe-Operational, SafeOP) 8: exploitation (Operational, OP)
Acknowledge (acquittement)	R1	Unsigned1	0: pas de modification du paramètre de changement du registre d'état de la couche AL 1: réinitialisation du paramètre de changement du registre d'état de la couche AL
Réservé	R1	Unsigned3	doit être égal à zéro
Application-specific (Spécifique à une application)	R2	Unsigned8	spécifique à l'application

5.3.2 Réponse AL Control (confirmation)

La réponse AL Control est mappée avec un service de lecture de couche DLL dans l'objet de registre d'état de l'utilisateur de couche DLL, ainsi que dans les registres R4, R5 et R6 définis par la CEI 61158-3-12. Les types d'attributs de la réponse AL Control sont décrits à la Figure 7.

```
typedef struct
{
  unsigned          State:           4;
  unsigned          Change:          1;
  unsigned          Reserved1:       3;
  unsigned          ApplSpecific:    8;
  unsigned          Reserved2:      16;
  unsigned          AlStatusCode:   16;
} TALSTATUSE;
```

Figure 7 – Structure de la réponse AL Control

Le codage de la réponse AL Control est spécifié dans le Tableau 10.

Tableau 10 – Réponse AL Control

Paramètre	Registre de l'utilisateur DLL	Data type	Valeur
State (état)	R3	Unsigned4	1: initialisation (Initialization, INIT) 2: Pré-exploitation (Pre-Operational, PreOP) 3: amorçage (Bootstrap, Boot) 4: exploitation sûre (Safe-Operational, SafeOP) 8: exploitation (Operational, OP)
Change (changement)	R3	Unsigned1	0: changement d'état réussi 1: échec du changement d'état
Reserved (Réservé)	R3	Unsigned3	doit être égal à zéro

Application-specific	R4	Unsigned8	spécifique à l'application
Reserved (Réservé)	R5	Unsigned16	doit être égal à zéro
Application-specific (Spécifique à une application)	R6	Unsigned16	voir le Tableau 11

Tableau 11 – Codes du statut de la couche Application (AL StatusCodes)

Code	Description	État actuel (ou changement d'état)	État résultant
0x0000	Absence d'erreurs	Tous	État actuel
0x0001	Erreur non spécifiée	Tous	Tous + E
0x0002	Absence de mémoire	Tous	Tous + E
0x0003	Installation d'appareil non valide	P -> S	P + E
0x0005	Réservé, pour des raisons de compatibilité		
0x0011	Changement d'état demandé non valide	I -> S, I -> O, P -> O O -> B, S -> B, P -> B	État actuel + E
0x0012	État demandé inconnu	Tous	État actuel + E
0x0013	Amorçage non pris en charge	I -> B	I + E
0x0014	Absence de microprogramme valide	I -> P	I + E
0x0015	Configuration de boîte aux lettres non valide	I -> B	I + E
0x0016	Configuration de boîte aux lettres non valide	I -> P	I + E
0x0017	Configuration de gestionnaire de synchronisation non valide	P -> S, S -> O	État actuel + E
0x0018	Absence d'entrées valides disponibles	O, S -> O	S + E
0x0019	Absence de sorties valides	O, S -> O	S + E
0x001A	Erreur de synchronisation	O, S -> O	S + E
0x001B	Chien de garde du gestionnaire de synchronisation	O, S	S + E
0x001C	Types de gestionnaire de synchronisation non valides	O, S, P -> S	S + E
0x001D	Configuration de sortie non valide	O, S, P -> S	S + E
0x001E	Configuration d'entrée non valide	O, S, P -> S	P + E
0x001F	Configuration de chien de garde non valide	O, S, P -> S	P + E
0x0020	Démarrage à froid de l'esclave nécessaire	Tous	État actuel + E
0x0021	Initialisation de l'esclave nécessaire	B, P, S, O	État actuel + E
0x0022	Pré-exploitation de l'esclave nécessaire	S, O	S + E, O + E
0x0023	Exploitation sûre de l'esclave nécessaire	O	O + E
0x0024	Mapping d'entrée non valide	P -> S	P + E
0x0025	Mapping de sortie non valide	P -> S	P + E
0x0026	Paramètres incohérents	P -> S	P + E
0x0027	FreeRun non pris en charge	P -> S	P + E
0x0028	SyncMode non pris en charge	P -> S	P + E
0x0029	Mode 3Buffer nécessaire à FreeRun	P -> S	P + E

Code	Description	État actuel (ou changement d'état)	État résultant
0x002A	Chien de garde d'arrière-plan	S, O	P + E
0x002B	Absence d'entrées et de sorties valides	O, S -> O	S + E
0x002C	Erreur fatale de synchronisation	O	S + E
0x002D	Absence d'erreurs de synchronisation	O, S, P -> S	S + E
0x0030	Configuration de synchronisation DC non valide	O, S -> O, P -> S	P + E, S + E
0x0031	Configuration de verrouillage d'horloges DC non valide	O, S -> O, P -> S	P + E, S + E
0x0032	Erreur de PLL	O, S -> O	S + E
0x0033	Erreur d'E/S de synchronisation d'horloges DC	O, S -> O	S + E
0x0034	Erreur d'expiration de synchronisation d'horloges DC	O, S -> O	S + E
0x0035	Durée de cycle de synchronisation d'horloges DC non valide	P -> S	P + E
0x0036	Durée de cycle Sync0 DC	P -> S	P + E
0x0037	Durée de cycle Sync1 DC	P -> S	P + E
0x0041	MBX_AOE	B, P, S, O	État actuel + E
0x0042	MBX_EOE	B, P, S, O	État actuel + E
0x0043	MBX_COE	B, P, S, O	État actuel + E
0x0044	MBX_FOE	B, P, S, O	État actuel + E
0x0045	MBX_SOE	B, P, S, O	État actuel + E
0x004F	MBX_VOE	B, P, S, O	État actuel + E
0x0050	Absence d'accès à l'EEPROM	Tous	Tous + E
0x0051	Erreur d'EEPROM	Tous	Tous + E
0x0060	Esclave remis en marche localement	Tous	I
0x0061	Valeur d'Identification de l'appareil mise à jour	P	P + E
0x0062 ...0 x00EF	Réservé		
0x00F0	Contrôleur de l'application disponible	I	I + E
autres codes < 0x8000	réservé		
0x8000 – 0xFFFF	Spécifique au fournisseur		

5.3.3 Changements d'état de la couche AL

Le service AL State Changed est mappé avec un service de lecture de couche DLL dans les objets AL Status et AL Status Code. Les types d'attributs AL State Changed sont décrits à la Figure 8.

```
typedef struct
{
  unsigned          State:            4;
  unsigned          Change:           1;
  unsigned          Reserved1:        3;
  unsigned          ApplSpecific:     8;
  unsigned          Reserved2:       16;
  unsigned          AlStatusCode:    16;
} TALSTATUSE;
```

Figure 8 – Structure du service AL State Changed

Le codage du service AL State Changed est spécifié dans le Tableau 12.

Tableau 12 – Service "AL State Changed"

Paramètre	Registre de l'utilisateur DLL	Data type	Valeur
State (état)	R3	Unsigned4	1: initialisation (Initialization, INIT) 2: Pré-exploitation (Pre-Operational, PreOP) 3: amorçage (Bootstrap, Boot) 4: exploitation sûre (Safe-Operational, SafeOP) 8: exploitation (Operational, OP)
Change (changement)	R3	Unsigned1	doit être égal à un
Reserved (Réservé)	R3	Unsigned3	doit être égal à zéro
Application-specific	R4	Unsigned8	spécifique à l'application
Réservé	R5	Unsigned16	doit être égal à zéro
AL status code	R6	Unsigned16	voir le Tableau 11

5.3.4 Attributs des relations AR de couche AL

Les services de lecture ou d'écriture de couche DLL, ou encore les services de lecture-écriture locaux, peuvent accéder aux attributs des relations AR de couche AL.

Les types d'attributs de PDI Control (commande de l'interface PDI) sont décrits à la Figure 9.

```
typedef struct
{
  unsigned      PDIType:                  8;
  unsigned      StrictALControl:        1;
  unsigned      Reserved:               7;
} TPDICONTROL;
```

Figure 9 – Description des types de commande de l'interface PDI

Le codage de la commande de l'interface PDI est spécifié dans le Tableau 13. La commande de l'interface PDI sera chargée par l'interface SII au démarrage.

Tableau 13 – PDI Control ("Commande de l'interface PDI")

Paramètre	Registre de l'utilisateur DLL	Data type	Accès DL	Accès local	Valeur/Description
PDI type	R7	Unsigned8	R	R	spécifique au type (voir le paramètre d'information de couche DLL dans la CEI 61158-3-12)
Strict AL Control	Copie	Unsigned1	R	R	0x00: la gestion de la couche AL sera assurée par un contrôleur d'application 0x01: la gestion de la couche AL sera émulée (l'état de la couche AL suit directement la commande AL)

Le codage de la configuration de l'interface PDI est spécifique à un contrôleur, comme indiqué dans le Tableau 14. Il est défini par l'interface SII au démarrage.

Tableau 14 – Configuration de l'interface PDI

Paramètre	Registre de l'utilisateur DLL	Data type	Accès DL	Accès local	Valeur/Description
Application-specific	R8	unsigned8	R	R	spécifique à l'application

Les types d'attributs de la configuration de la synchronisation (Sync Configuration) sont décrits à la Figure 10.

```
typedef struct
{
  unsigned      SignalCondSync0:          2;
  unsigned      EnableSignalSync0:        1;
  unsigned      EnableInterruptSync0:     1;
  unsigned      SignalCondSync1:          2;
  unsigned      EnableSignalSync1:        1;
unsigned      EnableInterruptSync1:     1;
} TSYNCCFG;
```

Figure 10 – Description du type de configuration de synchronisation

Le codage de la configuration de la synchronisation est spécifié dans le Tableau 15. La configuration de la synchronisation sera chargée par l'interface SII au démarrage.

Tableau 15 –Sync Configuration (Configuration de la synchronisation)

Paramètre	Registre de l'utilisateur DLL	Data type	Accès DL	Accès local	Valeur/Description
Signal Conditioning Sync0	R8	unsigned2	R	R	spécifique au contrôleur
Enable Signal Sync0	R8	unsigned1	R	R	0x00: désactivation 0x01: activation
Enable Interrupt Sync0	R8	unsigned1	R	R	0x00: désactivation 0x01: activation
Signal Conditioning Sync1	R8	unsigned2	R	R	spécifique au contrôleur
Enable Signal Sync1	R8	unsigned1	R	R	0x00: désactivation 0x01: activation
Enable Interrupt Sync1	R8	unsigned1	R	R	0x00: désactivation 0x01: activation

5.4 Codage de l'interface SII

Le codage des zones de l'interface SII est spécifié dans le Tableau 16 et le Tableau 17. "Adresse" désigne une adresse de mot (par exemple, 0 est le premier mot, 1 est le deuxième).

Tableau 16 – Zones de l'interface SII

Paramètre	Adresse	Data type	Valeur/Description
PDI Control	0x0000	Unsigned16	valeur d'initialisation du registre de commande de l'interface PDI (0x140-0x141)
PDI Configuration	0x0001	Unsigned16	valeur d'initialisation du registre de configuration de l'interface PDI (0x150-0x151)
SyncImpulseLen	0x0002	Unsigned16	impulsion de synchronisation, en multiples de 10 ns
PDI Configuration2	0x0003	Unsigned16	valeur d'initialisation du mot de poids le plus fort du registre de configuration R8 de l'interface PDI (0x152-0x153)
Configured Station Alias	0x0004	Unsigned16	adresse d'alias
Réservé	0x0005	BYTE[4]	doit être égal à zéro
Checksum	0x0007	Unsigned16	l'octet bas contient le reste de la division du mot 0 au mot 6, exprimés sous la forme d'un nombre non signé, par le polynôme x^8+x^2+x+1 (valeur initiale 0xFF)
Vendor ID	0x0008	Unsigned32	objet CAN 0x1018, sous-indice 1
Product Code	0x000A	Unsigned32	objet CAN 0x1018, sous-indice 2
Revision Number	0x000C	Unsigned32	objet CAN 0x1018, sous-indice 3
Serial Number	0x000E	Unsigned32	objet CAN 0x1018, sous-indice 4
Réservé	0x0010	BYTE[8]	doit être égal à zéro
Bootstrap Receive Mailbox Offset	0x0014	Unsigned16	réception de l'adresse relative de la boîte aux lettres pour l'état d'amorçage (maître à esclave)
Bootstrap Receive Mailbox Size	0x0015	Unsigned16	réception de la taille de boîte aux lettres pour l'état d'amorçage (maître à esclave) La taille de boîte aux lettres normalisée et celle de la boîte aux lettres en mode Bootstrap peuvent différer. Une plus grande taille de boîtes aux lettres en mode Bootstrap peut être utilisée pour l'optimisation.

Paramètre	Adresse	Data type	Valeur/Description
Bootstrap Send Mailbox Offset	0x0016	Unsigned16	envoi de l'adresse relative de la boîte aux lettres pour l'état d'amorçage (esclave à maître)
Bootstrap Send Mailbox Size	0x0017	Unsigned16	envoi de la taille de boîte aux lettres pour l'état d'amorçage (esclave à maître) La taille de boîte aux lettres normalisée et celle de la boîte aux lettres en mode Bootstrap peuvent différer. Une plus grande taille de boîtes aux lettres en mode Bootstrap peut être utilisée pour l'optimisation.
Standard Receive Mailbox Offset	0x0018	Unsigned16	réception de l'adresse relative de la boîte aux lettres pour l'état standard (maître à esclave)
Standard Receive Mailbox Size	0x0019	Unsigned16	réception de la taille de boîte aux lettres pour l'état standard (maître à esclave)
Standard Send Mailbox Offset	0x001A	Unsigned16	envoi de l'adresse relative de la boîte aux lettres pour l'état standard (esclave à maître)
Standard Send Mailbox Size	0x001B	Unsigned16	envoi de la taille de boîte aux lettres pour l'état standard (esclave à maître)
Mailbox Protocol	0x001C	Unsigned16	protocoles de boîte aux lettres pris en charge, selon les définitions données dans le Tableau 18
Réservé	0x001D	BYTE[66]	doit être égal à zéro
Size	0x003E	Unsigned16	Taille de E2PROM en [KiBit] + 1 NOTE: KiBit signifie 1 024 Bits. NOTE: “size” = 0 (taille = 0) signifie une taille d’EEPROM de 1 KiBit
Version	0x003F	Unsigned16	cette version est la version 1

Tableau 17 – Catégories de l'interface SII

Paramètre	Adresse	Data type	Valeur/Description
First Category Header	0x0040	Unsigned15	type de catégorie, selon les définitions données dans le Tableau 19
	0x0040	Unsigned1	spécifique au fournisseur
	0x0041	Unsigned16	suivant la taille de mot de la catégorie x
First Category Data	0x0042	Dépend de la catégorie.	données de la catégorie
Second Category Header	0x0042 + x	Unsigned15	type de catégorie, selon les définitions données dans le Tableau 19
	0x0042 + x	Unsigned1	spécifique au fournisseur
	0x0043 + x	Unsigned16	suivant la taille de mot de la catégorie
Second Category Data	0x0044 + x	Dépend de la catégorie.	données de la catégorie
...			et ainsi de suite jusqu'à la dernière catégorie

Tableau 18 – Types de protocoles de boîte aux lettres pris en charge

Protocole	Valeur	Description
EoE	0x0002	Ethernet via Type 12 (tunnellisation des services de liaison de données)
CoE	0x0004	Protocole d'application CAN via Type 12 (accès à l'objet SDO)
FoE	0x0008	Accès aux fichiers via Type 12
SoE	0x0010	Profil de servocommande via Type 12
VoE	0x0020	Protocole spécifique au fournisseur via Type 12

Tableau 19 – Types de catégories

Protocole	Valeur	Description
NOP	00	absence d'informations
Spécifique à l'appareil	01 – 09	Catégories spécifiques à un appareil ne doit pas être écrasé par le Maître ou l'outil de configuration Pourrait être utilisé pour des valeurs d'étalonnage)
STRINGS	10	référentiel de chaînes destiné aux autres catégories. Pour connaître la structure des données de cette catégorie, voir le Tableau 20
DataTypes	20	data types pour utilisation future
Généralités	30	informations générales. Pour connaître la structure des données de cette catégorie, voir le Tableau 21
FMMU	40	unités FMMU pour utilisation future. Pour connaître la structure des données de cette catégorie, voir le Tableau 22
Gestionnaire de synchronisation	41	configuration du gestionnaire de synchronisation. Pour connaître la structure des données de cette catégorie, voir le Tableau 23
objet TxPDO	50	description d'objet TxPDO. Pour connaître la structure des données de cette catégorie, voir le Tableau 24
Objet RxPDO	51	description d'objet RxPDO. Pour connaître la structure des données de cette catégorie, voir le Tableau 24
DC	60	horloge distribuée pour utilisation future
Réservé	60-2047	Réservé
Spécifique au fournisseur	0x0800-0x0FFF	Catégories spécifiques à un fournisseur
Réservé	0x1000-0xFFFF	Réservé
END	0xffff	

Tableau 20 – Structure de la catégorie des données de chaînes

Paramètre	Adresse de l'octet	Data type	Valeur/Description
nStrings	0x0000	Unsigned8	nombre de chaînes
str1_len	0x0001	Unsigned8	longueur de la 1ère chaîne
str_1	0x0002	BYTE[str1_len]	données de la 1ère chaîne
str2_len	0x0002+str1_len	Unsigned8	longueur de la 2e chaîne
str_2	0x0003+str1_len	BYTE[str2_len]	données de la 2e chaîne
....			
strn_len	0x000z	Unsigned8	longueur de la n^{e} chaîne
str_n	0x000z+1	BYTE[strn_len]	données de la n^{e} chaîne
PAD_Byte	0x000y	BYTE	remplissage si le nombre d'octets de la catégorie est impair

Tableau 21 – Structure de la catégorie des informations générales

Paramètre	Adresse de l'octet	Data type	Valeur/Description
GroupIdx	0x0000	Unsigned8	Informations sur le groupe (spécifiques au fournisseur) – Indice des chaînes
ImgIdx	0x0001	Unsigned8	Nom d'image (spécifique au fournisseur) – Indice des chaînes
OrderIdx	0x0002	Unsigned8	Numéro de commande de l'appareil (spécifique au fournisseur) – Indice des chaînes
NameIdx	0x0003	Unsigned8	Désignation de l'appareil (spécifique au fournisseur) – Indice des chaînes
Réservé	0x0004	Unsigned8	réservé
CoE details	0x0005	Unsigned8	Bit 0: activation d'objets SDO Bit 1: activation des informations d'objet SDO Bit 2: activation de l'attribution d'objets PDO Bit 3: activation de la configuration d'objets PDO Bit 4: activation du chargement au démarrage Bit 5: activation de l'accès total aux objets SDO
FoE details	0x0006	Unsigned8	Bit 0: activation de l'accès FoE
EoE Details	0x0007	Unsigned8	Bit 0: activation du protocole EoE
SoEChannels	0x0008	Unsigned8	réservé
DS402Channels	0x0009	Unsigned8	réservé
SysmanClass	0x000a	Unsigned8	réservé
Flags	0x000b	Unsigned8	Bit 0: activation de l'état d'exploitation sûre Bit 1: activation de non-LRW
CurrentOnEBus	0x000c	Signed16	Consommation de courant de l'E-bus (en mA) Les valeurs négatives indiquent que l'E-bus fournit du courant.
PAD_Byte1	0x000b	BYTE[2]	réservé
PhysicalPort	0x0010	Unsigned16	Description des ports physiques: 0x00: non utilisé 0x01: MII 0x02: réservé 0x03: E-BUS Chaque port est décrit par 4 bits: 3:0: port 0

Paramètre	Adresse de l'octet	Data type	Valeur/Description
			7:4: port 1 11:8: port 2 15:9: port 3
PAD_Byte2	0x0012	BYTE[14]	réservé

Tableau 22 – Structure de la catégorie FMMU

Paramètre	Adresse de l'octet	Data type	Valeur/Description
FMMU0	0x0000	Unsigned8	0x00: unité FMMU0 non utilisée 0x01: unité FMMU0 utilisée pour les sorties 0x02: unité FMMU0 utilisée pour les entrées 0x03: unité FMMU0 utilisée pour l'état du gestionnaire de synchronisation (lecture de la boîte aux lettres) 0xFF: unité FMMU0 non utilisée
FMMU1	0x0001	Unsigned8	0x00: unité FMMU1 non utilisée 0x01: unité FMMU1 utilisée pour les sorties 0x02: unité FMMU1 utilisée pour les entrées 0x03: unité FMMU1 utilisée pour l'état du gestionnaire de synchronisation (lecture de la boîte aux lettres) 0xFF: unité FMMU1 non utilisée
	...		et ainsi de suite si l'on utilise plus de deux unités FMMU

Tableau 23 – Structure de la catégorie du gestionnaire de synchronisation pour chaque élément

Paramètre	Adresse de l'octet	Data type	Valeur/Description
PhysicalStartAddress	0x0000	WORD	origine des données (voir l'adresse de début physique du gestionnaire de synchronisation)
Length (Longueur)	0x0002	WORD	
Control Register	0x0004	Unsigned8	définit le mode d'exploitation (voir le registre de commande du gestionnaire de synchronisation)
Status Register	0x0005	BYTE	indifférent
Enable Synch Manager	0x0006	Unsigned8	activation du gestionnaire de synchronisation Bit 0: activation Bit 1: contenu fixe (informations destinées à l'outil de configuration – le gestionnaire de synchronisation possède du contenu fixe) Bit 2: gestionnaire de synchronisation virtuel (gestionnaire de synchronisation virtuel – pas d'utilisation de ressources matérielles) Bit 3: opOnly (il convient de n'activer le gestionnaire de synchronisation qu'à l'état d'exploitation) Bit 7 à bit 4: réservé
Sync Manager Type	0x0007	BYTE	Type de gestionnaire de synchronisation 0x00 = non utilisé ou inconnu 0x01 = utilisé pour les sorties de la boîte aux

Paramètre	Adresse de l'octet	Data type	Valeur/Description
			lettres 0x02 = utilisé pour les entrées de la boîte aux lettres 0x03 = utilisé pour les sorties de données de processus 0x04 = utilisé pour les entrées de données de processus

Tableau 24 – Structure de la catégorie TXPDO et RXPDO pour chaque objet PDO

Paramètre	Adresse	Data type	Valeur/Description
PDO index	0x0000	Unsigned16	RxPDO en lecture seule: 0x1600 à 17FF, TxPDO en lecture seule: 0x1A00 à 1BFF
nEntry	0x0002	Unsigned8	nombre d'entrées
SyncM	0x0003	Unsigned8	gestionnaire de synchronisation associé
Synchronization	0x0004	Unsigned8	renvoi à la synchronisation d'horloges DC
NameIdx	0x0005	Unsigned8	nom de l'objet – indice des chaînes
Flags	0x0006	WORD	pour utilisation future
Entry 1	0x0008	8 BYTE	répété pour chaque entrée, selon la définition dans le Tableau 25
...			
Entry nEntry	nEntry*8	8 BYTE	répété pour chaque entrée, selon la définition dans le Tableau 25

Tableau 25 – Structure des entrées d'objet PDO

Paramètre	Position relative dans la structure d'entrée	Data type	Valeur/Description
Entry Index	0x0000	Unsigned16	indice de l'entrée
Subindex	0x0002	Unsigned8	sous-indice
Entry Name Idx	0x0003	Unsigned8	nom de l'entrée – indice des chaînes
Data type	0x0004	Unsigned8	data type de l'entrée (indice dans le dictionnaire d'objets CoE)
Bitlen	0x0005	Unsigned8	longueur des données de l'entrée
Flags	0x0006	WORD	pour utilisation future

5.5 Codage de l'interface PDI isochrone

Les types d'attributs de verrouillage et de synchronisation des horloges distribuées sont décrits à la Figure 11.

```
typedef struct
{
  BYTE          Reserved1;
  unsigned      CyclicOperationEnable:   1;
  unsigned      SYNC0Activate:           1;
  unsigned      SYNC1Activate:           1;
  unsigned      Reserved2:               5;
  WORD          SYNCPulse;
  BYTE          Reserved5[10];
  unsigned      Interrupt1Status:        1;
  unsigned      Reserved2:               7;
  unsigned      Interrupt2Status:        1;
  unsigned      Reserved3:               7;
  DWORD         CyclicOperationStartTime;
  BYTE          Reserved4[12];
  DWORD         SYNC0CycleTime;
  DWORD         SYNC1CycleTime;
  unsigned      Latch0PosEdge:           1;
  unsigned      Latch0NegEdge:           1;
  unsigned      Reserved5:               14;
  unsigned      Latch1PosEdge:           1;
  unsigned      Latch1NegEdge:           1;
  unsigned      Reserved6:               14;
  BYTE          Reserved7[4];
  unsigned      Latch0PosEvnt:           1;
  unsigned      Latch0NegEvnt:           1;
  unsigned      Reserved8:               6;
  unsigned      Latch1PosEvnt:           1;
  unsigned      Latch1NegEvnt:           1;
  unsigned      Reserved9:               6;
  DWORD         Latch0PosEdgeValue;
  BYTE          Reserveda[4];
  DWORD         Latch0NegEdgeValue;
  BYTE          Reservedb[4];
  DWORD         Latch1PosEdgeValue;
  BYTE          Reservedc[4];
  DWORD         Latch1NegEdgeValue;
  BYTE          Reservedd[4];
} TDCISOCHRON;
```

Figure 11 – Description des types de verrouillage et de synchronisation des horloges distribuées

L'encodage des paramètres de synchronisation des horloges distribuées est décrit dans le Tableau 26. Les paramètres sont mappés avec les paramètres P1 à P6 de l'utilisateur d'horloges DC de couche DLL. Les événements SYNC0 et SYNC1 sont mappés avec les événements de couche DLL.

Tableau 26 – Paramètres de synchronisation des horloges distribuées

Paramètre	Paramètre de l'utilisateur d'horloges DC	Data type	Type d'accès DLL de Type 12	PDI du type d'accès	Valeur/Description
Cyclic operation enable	P1	Unsigned1	RW	R	0: désactivé 1: activé
SYNC0 activate	P1	Unsigned1	RW	R	0: désactivé 1: impulsion SCNC0 générée
SYNC1 activate	P1	Unsigned1	RW	R	0: désactivé 1: impulsion SCNC1 générée
SYNC Pulse	P2	Unsigned16	R	R	obtenu auprès de l'interface SII
Interrupt 0 Status	P3	Unsigned1	R	R	0: inactive 1: active
Réservé	P3	Unsigned7	R	R	
Interrupt 1 Status	P3	Unsigned1	R	R	0: inactive 1: active
Réservé	P3	Unsigned7	R	R	
Cyclic Operation Start Time	P4	Unsigned32	RW	R	la génération d'interruptions commencera lorsque les 32 bits bas de l'heure système atteindront cette valeur (en ns)
SYNC0 cycle time	P5	DWORD	RW	R	durée du cycle de l'événement SYNC0
SYNC1 cycle time	P6	DWORD	RW	R	durée du cycle de l'événement SYNC1

L'encodage des données de verrouillage des horloges distribuées est décrit dans le Tableau 27.

Tableau 27 – Données de verrouillage des horloges distribuées

Paramètre	Paramètre de l'utilisateur d'horloges DC	Data type	Type d'accès DLL de Type 12	PDI du type d'accès	Valeur/Description
Latch0 positive Edge	P7	Unsigned1	RW	R	0: continu 1: unique
Latch0 negative Edge	P7	Unsigned1	RW	R	0: continu 1: unique
Réservé	P7	Unsigned6	R	R	
Latch1 positive Edge	P7	Unsigned1	RW	R	0: continu 1: unique
Latch1 negative Edge	P7	Unsigned1	RW	R	0: continu 1: unique
Réservé	P7	Unsigned6	R	R	
Latch0 positive Event	P8	Unsigned1	RW	R	0: absence d'événements 1: événement mémorisé
Latch0 negative Event	P8	Unsigned1	RW	R	0: absence d'événements 1: événement mémorisé
Réservé	P8	Unsigned6	R	R	
Latch1 positive Event	P8	Unsigned1	RW	R	0: absence d'événements 1: événement mémorisé
Latch1 negative Event	P8	Unsigned1	RW	R	0: absence d'événements 1: événement mémorisé
Réservé	P8	Unsigned6	R	R	
Latch 0 positive Edge value	P9	DWORD	R	R	Latch0 Value positive Event (événement de valeur positive de Latch0)
Latch 0 negative Edge value	P10	DWORD	R	R	Latch0 Value negative Event (événement de valeur négative de Latch0)
Latch 1 positive Edge value	P11	DWORD	R	R	Latch1 Value positive Event (événement de valeur positive de Latch1)
Latch 1 negative Edge value	P12	DWORD	R	R	Latch1 Value negative Event (événement de valeur négative de Latch1)

5.6 Codage CoE

5.6.1 Structure des unités PDU

Les types d'attributs généraux du protocole CoE sont décrits à la Figure 12.

```
typedef struct
{
  unsigned          NumberLo:         8;
  unsigned          NumberHi:         1;
  unsigned          Reserved:         3;
  unsigned          Service:          4;
} TCOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
BYTE              Data[MBX_DATA_SIZE-2];
} TCOEMBX;
```

Figure 12 – Structure générale du protocole CoE

Le codage CoE est spécifié dans le Tableau 28.

Tableau 28 –Éléments CoE

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number (valeur)	Unsigned9	suivant le service CoE
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x01: Urgence 0x02: demande d'objet SDO 0x03: réponse de l'objet SDO 0x04: objet TxPDO 0x05: Objet RxPDO 0x06: demande distante d'objet TxPDO 0x07: demande distante d'objet RxPDO 0x08: Informations sur l'objet SDO

5.6.2 Objet SDO

5.6.2.1 SDO Download Expedited (Téléchargement express de l'objet SDO

5.6.2.1.1 SDO DownloadExpedited Request

Les types d'attributs de SDO DownloadExpedited Request (demande de téléchargement express de l'objet SDO) sont décrits à la Figure 13.

```
typedef struct
{
  unsigned          SizeIndicator:    1;
  unsigned          TransferType:     1;
  unsigned          DataSetSize:      2;
  unsigned          CompleteAccess:   1;
  unsigned          Command:          3;
  BYTE              IndexLo;
  BYTE              IndexHi;
  BYTE              SubIndex;
} TINITSDOHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  BYTE              Data[4];
} TINITSDODOWNLOADEXPREQMBX;
```

Figure 13 – Structure de la demande de téléchargement express de l'objet SDO

Le codage de SDO DownloadExpedited Request (demande de téléchargement express de l'objet SDO) est spécifié dans le Tableau 29.

Tableau 29 – Demande de téléchargement express de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x01: taille des données dans la taille de l'ensemble de données spécifiée
	Type de transfert	Unsigned1	0x01: transfert express
	Taille de l'ensemble de données	Unsigned2	0x00: 4 octets de données 0x01: 3 octets de données 0x02: 2 octets de données 0x03: 1 octet de données
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera téléchargée 0x01: l'objet entier sera téléchargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)
	Spécificateur de	Unsigned3	0x01: demande de téléchargement

Partie de la trame	Champ de données	Data type	Valeur/Description
	commande		
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	Données	BYTE[4]	données de l'objet

5.6.2.1.2 SDO DownloadExpedited Response

Les types d'attributs de SDO Download Expedited Response (réponse de téléchargement express de l'objet SDO) sont décrits à la Figure 14.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
} TINITSDODOWNLOADEXPRESMBX;
```

Figure 14 – Structure de la réponse de téléchargement express de l'objet SDO

Le codage de SDO DownloadExpedited Response (réponse de téléchargement express de l'objet SDO) est spécifié dans le Tableau 30.

Tableau 30 – Réponse de téléchargement express de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x03: réponse de l'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x00
	Type de transfert	Unsigned1	0x00
	Taille de l'ensemble de données	Unsigned2	0x00
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera téléchargée 0x01: l'objet entier sera chargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)

Partie de la trame	Champ de données	Data type	Valeur/Description
	Spécificateur de commande	Unsigned3	0x03: réponse de téléchargement
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	réservé	DWORD	

5.6.2.2 SDO DownloadNormal

5.6.2.2.1 SDO DownloadNormal Request

Les types d'attributs de SDO Download Normal Expedited (demande de téléchargement normal de l'objet SDO) sont décrits à la Figure 15.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             CompleteSize;
  BYTE              Data[MBX_DATA_SIZE-10];
} TINITSDODOWNLOADNORMREQMBX;
```

Figure 15 – Structure de la demande de téléchargement normal de l'objet SDO

Le codage de SDO DownloadNormal Request (demande de téléchargement normal de l'objet SDO) est spécifié dans le Tableau 31.

Tableau 31 – Demande de téléchargement normal de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x01
	Type de transfert	Unsigned1	0x00: transfert normal
	Taille de l'ensemble de données	Unsigned2	0x00

Partie de la trame	Champ de données	Data type	Valeur/Description
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera téléchargée 0x01: l'objet entier sera chargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)
	Spécificateur de commande	Unsigned3	0x01: demande de téléchargement
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	Complete size	DWORD	taille totale des données de l'objet
	Données	BYTE[n-10]	si ((Length-10) >= Complete Size): données de l'objet si ((Length-10) < Complete Size): première partie de données de l'objet. Le téléchargement de segment SDO suit

5.6.2.2.2 SDO DownloadNormal Response

Les types d'attributs et le codage de SDO Download Normal Response (réponse de téléchargement normal de l'objet SDO) sont les mêmes que ceux de SDO Download Expedited Response (réponse de téléchargement express de l'objet SDO) (voir 5.6.2.1.2).

5.6.2.3 Download SDO Segment (Téléchargement de segment SDO)

5.6.2.3.1 Download SDO Segment Request (Demande de téléchargement de segment SDO)

Les types d'attributs de Download SDO Segment Request (demande de téléchargement de segment SDO) sont décrits à la Figure 16.

```
typedef struct
{
  unsigned          MoreFollows:      1;
  unsigned          SegDataSize:      3;
  unsigned          Toggle:           1;
  unsigned          Command:          3;
} TSDOSEGHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
  BYTE              Data[MBX_DATA_SIZE-3];
} TDOWNLOADSDOSEGREQMBX;
```

Figure 16 – Structure de la demande de téléchargement de segment SDO

Le codage de Download SDO Segment Request (demande de téléchargement de segment SDO) est spécifié dans le Tableau 32.

Tableau 32 – Demande de téléchargement de segment SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x0A: longueur des données du service de boîte aux lettres
	Address (Adresse)	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (Voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (Priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt (Compteur)	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number (Numéro)	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	More follows (À suivre)	Unsigned1	0x00: le téléchargement du segment SDO suit 0x01: dernier téléchargement de segment SDO
	SegData Size	Unsigned3	définit le nombre d'octets de données, parmi les 7 derniers (qui doivent toujours être envoyés), qui contiennent des données (s'applique uniquement si la longueur est 0x0A; sinon, doit être égal à 0x00): 0x00: 7 octets de données 0x01: 6 octets de données 0x02: 5 octets de données 0x03: 4 octets de données 0x04: 3 octets de données 0x05: 2 octets de données 0x06: 1 octet de données 0x07: 0 octet de données
	Toggle (Bascule)	Unsigned1	doit basculer à chaque demande de téléchargement de segment SDO, en partant de 0x00
	Command specifier	Unsigned3	0x00: demande de téléchargement de segment
	Data (Données)	BYTE[n-3]	partie de données de l'objet

5.6.2.3.2 Réponse de téléchargement de segment SDO

Les types d'attributs de Download SDO Segment Response (réponse de téléchargement de segment SDO) sont décrits à la Figure 17.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
} TDOWNLOADSDOSEGRESMBX;
```

Figure 17 – Structure de la réponse de téléchargement de segment SDO

Le codage de Download SDO Segment Response (réponse de téléchargement de segment SDO) est spécifié dans le Tableau 33.

Tableau 33 – Réponse de téléchargement de segment SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Address	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x03: réponse de l'objet SDO
Objet SDO	Réservé	Unsigned4	0x00
	Toggle	Unsigned1	doit être le même que pour la demande de téléchargement de segment SDO correspondante
	Command Specifier	Unsigned3	0x01: réponse de téléchargement de segment
	Réservé	BYTE[7]	

5.6.2.4 SDO UploadExpedited (Chargement express de l'objet SDO)

5.6.2.4.1 SDO UploadExpedited Request (Réponse de chargement express de l'objet SDO)

Les types d'attributs de SDO UploadExpedited Request (demande de chargement express de l'objet SDO) sont décrits à la Figure 18.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
} TINITSDOUPLOADEXPREQMBX;
```

Figure 18 – Structure de la demande de chargement express de l'objet SDO

Le codage de SDO UploadNormal Request (demande de chargement normal de l'objet SDO) est spécifié dans le Tableau 34.

Tableau 34 – Demande de chargement express de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Réservé	Unsigned4	0x00
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera chargée 0x01: l'objet entier sera chargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)
	Spécificateur de commande	Unsigned3	0x02: demande de chargement
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	Réservé	DWORD	

5.6.2.4.2 SDO UploadExpedited Response

Les types d'attributs de SDO UploadExpedited Response (réponse de chargement express de l'objet SDO) sont décrits à la Figure 19.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  BYTE              Data[4];
} TINITSDOUPLOADEXPREQMBX;
```

Figure 19 – Structure de la réponse de chargement express de l'objet SDO

Le codage de SDO UploadExpedited Response (réponse de chargement express de l'objet SDO) est spécifié dans le Tableau 35.

Tableau 35 – Réponse de chargement express de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Address (Adresse)	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x03: réponse de l'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x01: taille des données dans la taille de l'ensemble de données spécifiée
	Type de transfert	Unsigned1	0x01: transfert express
	Taille de l'ensemble de données	Unsigned2	0x00: 4 octets de données 0x01: 3 octets de données 0x02: 2 octets de données 0x03: 1 octet de données
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera chargée 0x01: l'objet entier sera chargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)
	Spécificateur de commande	Unsigned3	0x02: réponse de chargement
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	Data (Données)	BYTE[4]	données de l'objet

5.6.2.5 SDO UploadNormal

5.6.2.5.1 SDO UploadNormal Request

Les types d'attributs et le codage de SDO Upload Normal Request (demande de chargement normal de l'objet SDO) sont les mêmes que ceux de SDO Upload Expedited Request (demande de chargement express de l'objet SDO (voir 5.6.2.4.1).

5.6.2.5.2 SDO UploadNormal Response

Les types d'attributs de SDO UploadNormal Response (réponse de chargement normal de l'objet SDO) sont décrits à la Figure 20.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             CompleteSize;
  BYTE              Data[MBX_DATA_SIZE-10];
} TINITSDOUPLOADNORMRESMBX;
```

Figure 20 – Structure de la réponse de chargement normal de l'objet SDO

Le codage de SDO UploadNormal Response (réponse de chargement normal de l'objet SDO) est spécifié dans le Tableau 36. Si le nombre d'octets du paramètre Data est inférieur ou égal à 4, la réponse spécifiée en 5.6.2.4.2 peut être utilisée.

Tableau 36 – Réponse de chargement normal de l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x0A: longueur des données du service de boîte aux lettres
	Address	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x03: réponse de l'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x01
	Type de transfert	Unsigned1	0x00: transfert normal
	Taille de l'ensemble de données	Unsigned2	0x00
	Complete access	Unsigned1	0x00: l'entrée dont l'adresse correspond à l'indice et au sous-indice sera chargée 0x01: l'objet entier sera chargé, le sous-indice doit être égal à zéro (sous-indice 0 inclus) ou un (sous-indice 0 exclu)
	Command Specifier	Unsigned3	0x02: réponse de chargement
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet. Il doit être égal à zéro ou un, en cas d'accès total = 0x01
	Complete size	DWORD	taille totale des données de l'objet
	Data	BYTE[n-10]	si ((Length-10) >= Complete Size): données de l'objet

Partie de la trame	Champ de données	Data type	Valeur/Description
			si ((Length-10) < Complete Size): première partie de données de l'objet. Le chargement du segment SDO suit

5.6.2.6 Chargement de segment SDO

5.6.2.6.1 Demande de chargement de segment SDO

Les types d'attributs de Upload SDO Segment Request (demande de chargement de segment SDO) sont décrits à la Figure 21.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
} TUPLOADSDOSEGREQMBX;
```

Figure 21 – Structure de la demande de chargement de segment SDO

Le codage de Upload SDO Segment Request (demande de chargement de segment SDO) est spécifié dans le Tableau 37.

Tableau 37 – Demande de chargement de segment SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Réservé	Unsigned4	0x00
	Toggle	Unsigned1	doit basculer à chaque demande de chargement de segment SDO, en partant de 0x00
	Command Specifier	Unsigned3	0x03: demande de chargement de segment
	Réservé	BYTE[7]	

5.6.2.6.2 Réponse de chargement de segment SDO

Les types d'attributs de Upload SDO Segment Response (réponse de chargement de segment SDO) sont décrits à la Figure 22.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOSEGHEADER     SdoHeader;
  BYTE              Data[MBX_DATA_SIZE-3];
} TUPLOADSDOSEGRESMBX;
```

Figure 22 – Structure de la réponse de chargement de segment SDO

Le codage de Upload SDO Segment Response (réponse de chargement de segment SDO) est spécifié dans le Tableau 38.

Tableau 38 – Réponse de chargement de segment SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x03: réponse de l'objet SDO
Objet SDO	More follows	Unsigned1	0x00: le chargement du segment SDO suit 0x01: dernier chargement de segment SDO
	SegData Size	Unsigned3	définit le nombre d'octets de données, parmi les 7 derniers (qui doivent toujours être envoyés), qui contiennent des données (s'applique uniquement si la longueur est 0x0A; sinon, doit être égal à 0x00): 0x00: 7 octets de données 0x01: 6 octets de données 0x02: 5 octets de données 0x03: 4 octets de données 0x04: 3 octets de données 0x05: 2 octets de données 0x06: 1 octet de données 0x07: 0 octet de données
	Toggle	Unsigned1	doit être le même que pour la demande de chargement de segment SDO correspondante
	Command specifier	Unsigned3	0x00: réponse de chargement de segment
	Data	BYTE[n-3]	partie de données de l'objet

5.6.2.7 Abandon du transfert d'objet SDO

5.6.2.7.1 Demande d'abandon du transfert d'objet SDO

Les types d'attributs de Abort SDO Transfer Request (demande d'abandon du transfert d'objet SDO) sont décrits à la Figure 23.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TINITSDOHEADER    SdoHeader;
  DWORD             AbortCode;
} TABORTSDOTRANSFERREQMBX;
```

Figure 23 – Structure de la demande d'abandon du transfert d'objet SDO

Le codage de Abort SDO Transfer Request (demande d'abandon du transfert d'objet SDO) est spécifié dans le Tableau 39.

Tableau 39 – Demande d'abandon du transfert d'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x02: demande d'objet SDO
Objet SDO	Indicateur de taille	Unsigned1	0x00
	Type de transfert	Unsigned1	0x00
	Taille de l'ensemble de données	Unsigned2	0x00
	Réservé	Unsigned1	0x00
	Command Specifier	Unsigned3	0x04: demande d'abandon du transfert
	Index (Indice)	WORD	indice de l'objet
	Sous-indice	BYTE	sous-indice de l'objet
	Abort Code	DWORD	code d'abandon selon les spécifications du Tableau 40

5.6.2.7.2 Codes d'abandon SDO

Les codes d'abandon SDO sont spécifiés dans le Tableau 40.

Tableau 40 – Codes d'abandon SDO

Valeur	Signification
0x05 03 00 00	bit de basculement inchangé
0x05 04 00 00	expiration de protocole SDO
0x05 04 00 01	spécificateur de commande client/serveur non valide ou inconnu
0x05 04 00 05	mémoire saturée
0x06 01 00 00	accès non pris en charge à un objet
0x06 01 00 01	tentative de lecture d'un objet en écriture seule
0x06 01 00 02	tentative d'écriture dans un objet en lecture seule
0x06 01 00 03	Le sous-indice ne peut pas être écrit, SI0 doit être nul pour un accès en écriture
0x06 01 00 04	Un accès complet à SDO n'est pas pris en charge pour les objets de longueur variable tels que les types d'objets ENUM
0x06 01 00 05	La longueur de l'objet dépasse la taille de la boîte aux lettres
0x06 01 00 06	Objet mappé avec RxPDO, Téléchargement SDO bloqué
0x06 02 00 00	objet absent du dictionnaire d'objets
0x06 04 00 41	l'objet ne peut pas être mappé avec l'objet PDO
0x06 04 00 42	le nombre et la longueur des objets à mapper dépasseraient la longueur de l'objet PDO
0x06 04 00 43	raison d'incompatibilité de paramètre général
0x06 04 00 47	incompatibilité interne générale dans l'appareil
0x06 06 00 00	échec de l'accès en raison d'une erreur matérielle
0x06 07 00 10	non-concordance du data type, non-concordance de la longueur du paramètre de service
0x06 07 00 12	non-concordance du data type, longueur du paramètre de service trop élevée
0x06 07 00 13	non-concordance du data type, longueur du paramètre de service trop basse
0x06 09 00 11	sous-indice inexistant
0x06 09 00 30	dépassement de la plage de valeurs du paramètre (uniquement pour l'accès en écriture)
0x06 09 00 31	valeur du paramètre écrit trop haute
0x06 09 00 32	valeur du paramètre écrit trop basse
0x06 09 00 36	valeur maximum inférieure à la valeur minimum
0x08 00 00 00	erreur générale
0x08 00 00 20	les données ne peuvent pas être transférées ou mémorisées dans l'application
0x08 00 00 21	les données ne peuvent pas être transférées ou mémorisées dans l'application en raison d'une commande locale
0x08 00 00 22	les données ne peuvent pas être transférées ou mémorisées dans l'application en raison de l'état actuel de l'appareil
0x08 00 00 23	échec de la génération dynamique du dictionnaire d'objets ou absence de dictionnaire d'objets

5.6.3 Informations sur l'objet SDO

5.6.3.1 Généralités

L'ensemble de la partie "données de service d'information sur l'objet SDO" des services suivants doit être traité comme un bloc et n'être envoyé qu'une fois.

En cas de nécessité de fragmenter le service, le champ de longueur peut être <= 0x0A pour le dernier fragment.

5.6.3.2 Service d'information sur l'objet SDO

Les types d'attributs de SDO Information Service (service d'information sur l'objet SDO) sont décrits à la Figure 24.

```
typedef struct
{
  unsigned          OpCode:           7;
  unsigned          InComplete:       1;
  unsigned          Reserved:         8;
  WORD              FragmentsLeft;
} TSDOINFOHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
} TSDOINFOSERVICE;
```

Figure 24 – Structure du service d'information sur l'objet SDO

Le codage de SDO Information Service (service d'information sur l'objet SDO) est spécifié dans le Tableau 41.

Tableau 41 – Service d'information sur l'objet SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n > 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x01: Demande d'obtention de la liste de dictionnaires OD 0x02: Réponse d'obtention de la liste de dictionnaires OD 0x03: Demande d'obtention de la description d'objet 0x04: Réponse d'obtention de la description d'objet 0x05: Demande d'obtention de la description d'entrée 0x06: Réponse d'obtention de la description d'entrée

Partie de la trame	Champ de données	Data type	Valeur/Description
			0x07: demande d'informations d'erreur SDO
	Incomplete	Unsigned1	0x00: dernier fragment d'informations sur l'objet SDO 0x01: autres fragments d'informations sur l'objet SDO à suivre
	Réservé	Unsigned8	0x00
	Fragments left	WORD	nombre de fragments à suivre
Données sur le service d'information sur l'objet SDO	Données	BYTE[n-6]	données sur le service d'information sur l'objet SDO

5.6.3.3 Obtention de la liste de dictionnaires OD

5.6.3.3.1 Demande d'obtention de la liste de dictionnaires OD

Les types d'attributs de Get OD List Request (demande d'obtention de la liste de dictionnaires OD) sont décrits à la Figure 25.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              ListType;
} TGETODLISTREQ;
```

Figure 25 – Structure de la demande d'obtention de la liste de dictionnaires OD

Le codage de Get OD List Request (emande d'obtention de la liste de dictionnaires ODà est spécifié dans le Tableau 42.

Tableau 42 – Demande d'obtention de la liste de dictionnaires OD

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x01: Demande d'obtention de la liste de dictionnaires OD
	Incomplete	Unsigned1	doit être égal à zéro
	Réservé	Unsigned8	0x00
	Fragments left	WORD	doit être égal à zéro
Données sur le service d'information sur l'objet SDO	List type	WORD	0x00: obtention du nombre d'objets dans les 5 listes différentes 0x01: tous les objets du dictionnaire d'objets doivent être remis dans la réponse 0x02: seuls les objets pouvant être mappés avec un objet RxPDO doivent être remis dans la réponse 0x03: seuls les objets pouvant être mappés avec un objet TxPDO doivent être remis dans la réponse 0x04: seuls les objets qui doivent être mémorisés en vue d'un remplacement d'appareil doivent être remis dans la réponse 0x05: seuls les objets pouvant être utilisés comme paramètre de démarrage doivent être remis dans la réponse

5.6.3.3.2 Réponse d'obtention de la liste de dictionnaires OD

Les types d'attributs de la réponse d'obtention de la liste de dictionnaires OD sont décrits à la Figure 26.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              ListType;
} TGETODLISTRES;
```

Figure 26 – Structure de la réponse d'obtention de la liste de dictionnaires OD

Le codage de la réponse d'obtention de la liste de dictionnaires OD est spécifié dans le Tableau 43.

Tableau 43 – Get Réponse d'obtention de la liste de dictionnaires OD

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x02: Réponse d'obtention de la liste de dictionnaires OD
	Incomplete	Unsigned1	0x00: dernier fragment d'informations sur l'objet SDO 0x01: autres fragments d'informations sur l'objet SDO à suivre
	Réservé	Unsigned8	0x00
	Fragments left	WORD	nombre de fragments à suivre L'en-tête d'informations SDO sera émis avec chaque fragment, la donnée d'informations SDO sera émise fragmentée
Données sur le service d'information sur l'objet SDO	List type	WORD	0x00: la liste des longueurs doit être remise dans la réponse 0x01: tous les objets du dictionnaire d'objets doivent être remis dans la réponse 0x02: seuls les objets pouvant être mappés avec un objet RxPDO doivent être remis dans la réponse 0x03: seuls les objets pouvant être mappés avec un objet TxPDO doivent être remis dans la réponse 0x04: seuls les objets qui doivent être mémorisés en vue d'un remplacement d'appareil doivent être remis dans la réponse 0x05: seuls les objets pouvant être utilisés comme paramètre de démarrage doivent être remis dans la réponse
	Index list	WORD [(n-8)/2]	liste des indices d'objet ou 5 mots avec la longueur des types de liste si le type de liste est 0 L'ordre des indices d'objets est libre. Il convient de les trier dans l'ordre croissant

5.6.3.4 OD list segment

S'il se révèle nécessaire de segmenter les données de service d'information sur l'objet SDO figurant dans la réponse d'obtention de la liste de dictionnaires OD, les données de service

d'information sur l'objet SDO sont fragmentées et le service de segment de liste de dictionnaires OD doit transférer les segments.

5.6.3.5 Obtention de la description d'objet

5.6.3.5.1 Demande d'obtention de la description d'objet

Les types d'attributs de la demande d'obtention de la description d'objet sont décrits à la Figure 27.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
} TGETOBJDESCREQ;
```

Figure 27 – Structure de la demande d'obtention de la description d'objet

Le codage de la demande d'obtention de la description d'objet est spécifié dans le Tableau 44.

Tableau 44 – Demande d'obtention de la description d'objet

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x03: Demande d'obtention de la description d'objet
	Incomplete	Unsigned1	doit être égal à zéro
	Réservé	Unsigned8	0x00
	Fragments left	WORD	doit être égal à zéro
Données sur le service d'information sur l'objet SDO	Index (Indice)	WORD	indice de la description d'objet demandée

5.6.3.5.2 Réponse d'obtention de la description d'objet

Les types d'attributs de la réponse d'obtention de la description d'objet sont décrits à la Figure 28.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  WORD              DataType;
  BYTE              MaxSubindex;
  BYTE              ObjCode;
} TGETOBJDESCRES;
```

Figure 28 – Get Structure de la réponse d'obtention de la description d'objet

Le codage de la réponse d'obtention de la description d'objet est spécifié dans le Tableau 45.

Tableau 45 – Réponse d'obtention de la description d'objet

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n > 0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x04: Réponse d'obtention de la description d'objet
	Incomplete	Unsigned1	0x00: dernier fragment d'informations sur l'objet SDO
	Réservé	Unsigned8	0x00
	Fragments left	WORD	nombre de fragments à suivre
Données sur le service d'information sur l'objet SDO	Index (Indice)	WORD	indice de la description d'objet
	Data type	WORD	renvoi à la liste des data types
	Max subindex	BYTE	nombre maximum de sous-indice de l'objet
	Code objet	BYTE	Code objet 7: variable 8: tableau 9: enregistrement
	Nom	char[n-12]	nom de l'objet

5.6.3.6 Obtention de la description d'entrée

5.6.3.6.1 Demande d'obtention de la description d'entrée

Les types d'attributs de la demande d'obtention de la description d'entrée sont décrits à la Figure 29.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  BYTE              Subindex;
  BYTE              ValueInfo;
} TGETENTRYDESCREQ;
```

Figure 29 – Structure de la demande d'obtention de la description d'entrée

Le codage de la demande d'obtention de la description d'entrée est spécifié dans le Tableau 46.

Tableau 46 – Demande d'obtention de la description d'entrée

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x05: Demande d'obtention de la description d'entrée
	Incomplete	Unsigned1	doit être égal à zéro
	Réservé	Unsigned8	0x00
	Fragments left	WORD	doit être égal à zéro
Données sur le service d'information sur l'objet SDO	Index (Indice)	WORD	indice de la description d'objet demandée
	Sous-indice	BYTE	sous-indice de la description d'objet demandée
	ValueInfo	BYTE	les informations sur la valeur comprennent les éléments qui doivent figurer dans la réponse: Bit 0: réservé Bit 1: réservé Bit 2: réservé

Partie de la trame	Champ de données	Data type	Valeur/Description
			Bit 3: type d'unité Bit 4: valeur par défaut Bit 5: valeur minimum Bit 6: valeur maximum

5.6.3.6.2 Réponse d'obtention de la description d'entrée

Les types d'attributs de la réponse d'obtention de la description d'entrée sont décrits à la Figure 30.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  WORD              Index;
  BYTE              Subindex;
  BYTE              ValueInfo;
  WORD              DataType;
  WORD              BitLength;
  WORD              ObjAccess;
} TGETENTRYDESCRES;
```

Figure 30 – Structure de la réponse d'obtention de la description d'entrée

Le codage de la réponse d'obtention de la description d'objet est spécifié dans le Tableau 47.

Tableau 47 – Réponse d'obtention de la description d'entrée

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n >= 0x10: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x06: Réponse d'obtention de la description d'entrée
	Incomplete	Unsigned1	0x00: dernier fragment d'informations sur l'objet SDO 0x01: autres fragments d'informations sur l'objet SDO à suivre
	Réservé	Unsigned8	0x00
	Fragments left	WORD	nombre de fragments à suivre
Données sur le service d'information sur l'objet SDO	Index (Indice)	WORD	indice de la description d'objet demandée

Partie de la trame	Champ de données	Data type	Valeur/Description
	Sous-indice	BYTE	sous-indice de la description d'objet demandée
	ValueInfo	BYTE	les informations sur la valeur comprennent les éléments qui figurent dans la réponse: Bit 0: réservé Bit 1: réservé Bit 2: réservé Bit 3: type d'unité Bit 4: valeur par défaut Bit 5: valeur minimum Bit 6: valeur maximum
	Data type	WORD	indice du data type de l'objet
	Bit length	WORD	longueur de l'objet (en bits) si longueur = 0xFFFF: la longueur de l'objet est supérieure à 64 kBit ou pour les objets de longueur variable. Pour l'obtention de sa longueur, cet objet doit être chargé
	Object Access	WORD	Bit 0: accès en lecture à l'état de pré-exploitation Bit 1: accès en lecture à l'état d'exploitation sûre Bit 2: accès en lecture à l'état d'exploitation Bit 3: accès en écriture à l'état de pré-exploitation Bit 4: accès en écriture à l'état d'exploitation sûre Bit 5: accès en écriture à l'état d'exploitation Bit 6: objet pouvant être mappé avec un objet RxPDO Bit 7: objet pouvant être mappé avec un objet TxPDO Bit 8: objet pouvant être utilisé à des fins de sauvegarde Bit 9: objet pouvant être utilisé à des fins de paramétrage Bit 10 à bit 15: réservé
	Données	BYTE[n-16]	si la réponse contient le type d'unité, le type d'unité de l'objet suit (DWORD) si la réponse contient la valeur par défaut, la valeur par défaut de l'entrée d'objet suit (même data type que la valeur d'objet) si la réponse contient la valeur minimum, la valeur minimum de l'entrée d'objet suit (même data type que la valeur d'objet) si la réponse contient la valeur maximum, la valeur maximum de l'entrée d'objet suit (même data type que la valeur d'objet) si la longueur est supérieure au paramètre de réponse décrit, la description suit (tableau de caractères)

5.6.3.7 Segment de description d'entrée

Les types d'attributs et le codage du segment de description d'entrée sont les mêmes que pour la réponse d'obtention de la description d'entrée.

5.6.3.8 Informations d'erreur SDO

Les types d'attributs de la demande d'informations d'erreur SDO sont décrits à la Figure 31.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TSDOINFOHEADER    SdoInfoHeader;
  DWORD             AbortCode;
} TABORTSDOTRANSFERREQMBX;
```

Figure 31 – Structure de la demande d'informations d'erreur SDO

Le codage de la demande d'informations d'erreur SDO est spécifié dans le Tableau 48.

Tableau 48 – Demande d'informations d'erreur SDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x08: Informations sur l'objet SDO
En-tête d'informations SDO	Opcode	Unsigned7	0x07: demande d'informations d'erreur SDO
	Incomplete	Unsigned1	doit être égal à zéro
	Réservé	Unsigned8	0x00
	Fragments left	WORD	doit être égal à zéro
	Abort Code	DWORD	code d'abandon selon les spécifications du Tableau 40

5.6.4 Urgence

5.6.4.1 Demande d'urgence

Le codage de la demande d'urgence est spécifié dans le Tableau 49.

Tableau 49 – Demande d'urgence

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n = 0x0A: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	0x00
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x01: Urgence
Urgence	Code d'erreur	WORD	Code d'erreur
	Registre d'erreurs	BYTE	Registre d'erreurs
	Données	BYTE[5]	code d'erreur 0000-9FFF: champ d'erreur spécifique au fabricant code d'erreur A000-EFFF: Données de diagnostic code d'erreur F000-FFFF: champ d'erreur spécifique au fabricant
	Réservé	BYTE[n-10]	

5.6.4.2 Codes d'erreur d'urgence

Les codes d'erreur d'urgence sont spécifiés dans le Tableau 50.

Tableau 50 – Codes d'erreur d'urgence

Code d'erreur (hexadécimal)	Signification
00xx	Réinitialisation après erreur ou absence d'erreurs
10xx	Erreur générique
20xx	Courant
21xx	Courant, côté entrée de l'appareil
22xx	Courant, intérieur de l'appareil
23xx	Courant, côté sortie de l'appareil
30xx	Tension
31xx	Tension réseau
32xx	Tension, intérieur de l'appareil
33xx	Tension de sortie
40xx	Température
41xx	Température ambiante
42xx	Température de l'appareil
50xx	Matériel de l'appareil
60xx	Logiciel de l'appareil
61xx	Logiciel interne
62xx	Logiciel de l'utilisateur
63xx	Ensemble de données
70xx	Modules supplémentaires
80xx	Surveillance
81xx	Communication
82xx	Erreur de protocole
8210	Objet PDO non traité en raison d'une erreur de longueur
8220	Dépassement de la longueur de l'objet PDO
90xx	Erreur externe
A0xx	Erreur de changement d'état du diagramme d'états externe (External State Machine, ESM)
F0xx	Fonctions supplémentaires
FFxx	Spécifique à l'appareil

5.6.4.3 Erreur de changement d'état du diagramme d'états externe (External State Machine, ESM)

5.6.4.3.1 Code d'erreur

Les codes d'erreur de changement d'état du diagramme d'états ESM sont spécifiés dans le Tableau 51.

Tableau 51 – Code d'erreur

Code d'erreur (hexadécimal)	Signification
A000	échec du passage de l'état de pré-exploitation à l'état d'exploitation sûre
A001	échec du passage de l'état d'exploitation sûre à l'état d'exploitation

5.6.4.3.2 Données de diagnostic

La structure des données de diagnostic des changements d'états du diagramme d'états ESM est spécifiée dans le Tableau 52.

Tableau 52 – Données de diagnostic

Donnée[0]	Données[1..4]	Signification
0x00 + channel*4	Erreur de longueur du gestionnaire de synchronisation	non-concordance de la longueur du canal de gestionnaire de synchronisation
0x01 + channel*4	Erreur d'adresse du gestionnaire de synchronisation	non-concordance de l'adresse physique de début du canal de gestionnaire de synchronisation
0x02 + channel*4	Erreur de paramètre du gestionnaire de synchronisation	non-concordance des paramètres du canal de gestionnaire de synchronisation

5.6.4.3.3 Erreur de longueur du gestionnaire de synchronisation

Le codage des erreurs de longueur du gestionnaire de synchronisation est spécifié dans le Tableau 53.

Tableau 53 – Erreur de longueur du gestionnaire de synchronisation

Données[1..4]	Data type	Valeur/Description
Longueur minimum	WORD	valeur minimum du paramètre de longueur du canal de gestionnaire de synchronisation
Longueur maximum	WORD	valeur maximum du paramètre de longueur du canal de gestionnaire de synchronisation

5.6.4.3.4 Erreur d'adresse du gestionnaire de synchronisation

Le codage des erreurs d'adresse du gestionnaire de synchronisation est spécifié dans le Tableau 54.

Tableau 54 – Erreur d'adresse du gestionnaire de synchronisation

Données[1..4]	Data type	Valeur/Description
Adresse minimum	WORD	valeur minimum du paramètre d'adresse physique de début du canal de gestionnaire de synchronisation
Adresse maximum	WORD	valeur maximum du paramètre d'adresse physique de début du canal de gestionnaire de synchronisation

5.6.4.3.5 Erreur de paramètre du gestionnaire de synchronisation

Le codage des erreurs de paramètre du gestionnaire de synchronisation est spécifié dans le Tableau 55.

Tableau 55 – Erreur de paramètre du gestionnaire de synchronisation

Données[1..4]	Data type	Valeur/Description
Type de tampon attendu	Unsigned2	valeur attendue pour le paramètre de type de tampon du canal de gestionnaire de synchronisation
Direction attendue	Unsigned2	valeur attendue pour le paramètre de direction du canal de gestionnaire de synchronisation
Réservé	Unsigned1	0x00 (réservé pour utilisation future)
AL Event Enable attendu	Unsigned1	valeur attendue pour le paramètre AL Event Enable du canal de gestionnaire de synchronisation
Réservé	Unsigned10	0x00 (réservé pour utilisation future)
Channel Enable attendu	Unsigned1	valeur attendue pour le paramètre Channel Enable du canal de gestionnaire de synchronisation
Réservé	Unsigned15	0x00 (réservé pour utilisation future)

5.6.5 donnée de processus

5.6.5.1 Objet RxPDO

Le protocole d'émission RxPDO via la boîte aux lettres est spécifié dans le Tableau 56.

Tableau 56 – Émission RxPDO via la boîte aux lettres

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n > 0x02: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	numéro RxPDO associé
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x05: Objet RxPDO
PDO	Données	BYTE[n-2]	données de sortie du processus

5.6.5.2 Objet TxPDO

Le codage d'émission TxPDO via la boîte aux lettres est spécifié dans le Tableau 57.

Tableau 57 – Émission TxPDO via la boîte à lettres

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	n > 0x02: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	numéro TxPDO associé
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x04: objet TxPDO
PDO	Données	BYTE[n-2]	données d'entrée du processus

5.6.5.3 Demande d'émission distante de l'objet RxPDO

Le codage de la demande d'émission distante de l'objet RxPDO est spécifié dans le Tableau 58.

Tableau 58 – Demande d'émission distante de l'objet RxPDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x02: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	numéro RxPDO associé
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x07: Demande d'émission distante de l'objet RxPDO

5.6.5.4 Demande d'émission distante de l'objet TxPDO

Le codage de la demande d'émission distante de l'objet TxPDO est spécifié dans le Tableau 59.

Tableau 59 – Demande d'émission distante de l'objet TxPDO

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x02: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête CoE	Number	Unsigned9	numéro TxPDO associé
	Réservé	Unsigned3	0x00
	Service	Unsigned4	0x06: Demande d'émission distante de l'objet TxPDO

5.6.6 Commande

La structure des objets de commande est spécifiée dans le Tableau 60. Chaque objet de commande doit avoir le data type 0x0025. La structure peut être utilisée par tout objet déclaré comme objet de commande.

Tableau 60 – Structure des objets de commande

Subindex	Description	Data type	Valeur
0	Nombre d'entrées	UNSIGNED8	3
1	Commande	OCTET_STRING	Octets 0-n: données de demande un accès en écriture aux données de commande exécutera la commande
2	État	UNSIGNED8	0: dernière commande exécutée, absence d'erreurs, absence de réponse 1: dernière commande exécutée, absence d'erreurs, réponse dans l'objet 2: dernière commande exécutée, erreur, absence de réponse 3: dernière commande exécutée, erreur, réponse dans l'objet 4 – 99: réservée pour un usage futur 100 – 200: indique le pourcentage d'exécution de la commande (100 = 0 %, 200 = 100 %) 201 – 254: réservée pour un usage futur 255: commande en cours d'exécution (si l'affichage du pourcentage n'est pas pris en charge)
3	Réponse	OCTET-STRING	Octets 0-n: données de réponse

5.6.7 Object dictionary (Dictionnaire d'objets)

5.6.7.1 Structure du dictionnaire d'objets

La structure du dictionnaire d'objets est indiquée dans le Tableau 61.

Tableau 61 – Structure du dictionnaire d'objets

Index (hexadécimal)	Zone du dictionnaire d'objets
0x0000-0x0FFF	Zone de data type
0x1000-0x1FFF	Zone de communication CoE
0x2000-0x5FFF	Zone spécifique au fabricant
0x6000-0xFFFF	Zone de profil

5.6.7.2 Définitions de codes objet

La structure des entrées de définition de code objet est indiquée dans le Tableau 62.

Tableau 62 – Définitions de codes objet

Code d'objet	Nom d'objet
0002	DOMAIN
0005	DEFTYPE
0006	DEFSTRUCT
0007	VAR
0008	ARRAY
0009	RECORD

5.6.7.3 Zone Data type

La zone de data type de base est spécifiée dans le Tableau 63.

Tableau 63 – Zone de data type de base

Index (hexadécimal)	Type d'objet	Name
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	Integer8
0003	DEFTYPE	Integer16
0004	DEFTYPE	Integer32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000A	DEFTYPE	OCTET_STRING
000B	DEFTYPE	UNICODE_STRING
000C	DEFTYPE	TIME_OF_DAY
000D	DEFTYPE	TIME_DIFFERENCE
000E		Réservé
000F	DEFTYPE	DOMAIN
0010	DEFTYPE	Integer24
0011	DEFTYPE	REAL64
0012	DEFTYPE	Integer40
0013	DEFTYPE	Integer48
0014	DEFTYPE	Integer56
0015	DEFTYPE	Integer64
0016	DEFTYPE	UNSIGNED24
0017		Réservé
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001A	DEFTYPE	UNSIGNED56
001B	DEFTYPE	UNSIGNED64
001C		Réservé
001D	DEFTYPE	GUID
001E	DEFTYPE	BYTE
001F-002C		Réservé
002D	DEFTYPE	BitARR8
002E	DEFTYPE	BITARR16
002F	DEFTYPE	BITARR32

La zone de data type étendue est spécifiée dans le Tableau 64.

Tableau 64 – Zone de data type étendue

Index (hexadécimal)	Objet	Name
0020		réservé
0021	DEFSTRUCT	PDO_MAPPING
0022		réservé
0023	DEFSTRUCT	Identity
0024		réservé
0025	DEFSTRUCT	COMMAND_PAR
0026-0028		réservé
0029	DEFSTRUCT	SYNC_PAR
002A-002F		réservé
0030	DEFTYPE	BIT1
0031	DEFTYPE	BIT2
0032	DEFTYPE	BIT3
0033	DEFTYPE	BIT4
0034	DEFTYPE	BIT5
0035	DEFTYPE	BIT6
0036	DEFTYPE	BIT7
0037	DEFTYPE	BIT8
0038-003F		réservé
0040-005F	DEFSTRUCT	Types "Manufacturer Specific Complex Data"
0060-007F	DEFTYPE	Device Profile 0 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 0)
0080-009F	DEFSTRUCT	Device Profile 0 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 0)
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 1)
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 1)
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 2)
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 2)
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 3)
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 3)
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 4)
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 4)
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 5)
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 5)
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types (Type des données standard spécifiques au Profil d'appareil 6)
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 6)
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types (Type des données

Index (hexadécimal)	Objet	Name
		standard spécifiques au Profil d'appareil 7)
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types (Type des données complexes spécifiques au Profil d'appareil 7)
0260-07FF		réservé

La zone de data type énuméré occupe les indices 0x800 à 0xFFF. Chaque élément possède un data type, qui spécifie le nombre de bits occupé (par exemple, BIT3 ou UNSIGNED16), une liste d'entrées qui spécifie la valeur entière (données de type unsigned32) et l'énumération au format VisibleString, comme indiqué dans le Tableau 65.

Tableau 65 – Définition d'énumération

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre d'entrées	UNSIGNED8	O	R	Non	nombre de valeurs d'énumération n
	Remplissage	UNSIGNED8				0 remplissage visant à maintenir la parité à la frontière d'octets pour les objets suivants
1	Enum1	Chaîne d'octets	O	R	Non	UNSIGNED32 comme valeur entière VISIBLE STRING comme chaîne énumération
	…					
1	Enumn	Chaîne d'octets	O	R	Non	UNSIGNED32 comme valeur entière VISIBLE STRING comme chaîne énumération

5.6.7.4 Zone "CoE Communication"

La zone de dictionnaire d'objets de communication CoE se compose des éléments décrits dans le Tableau 66.

Tableau 66 – Zone de communication CoE

Index (hexadécimal)	Type d'objet	Name	Type	M/O/C
1000	VAR	Type d'appareil	UNSIGNED32	M
1001		Registre d'erreurs	UNSIGNED8	O
1002		Réservé		
::::	::::	::::	::::	
1007		Réservé		
1008	VAR	Nom d'appareil attribué par le fabricant	VisibleString	O
1009	VAR	Version matérielle attribuée par le fabricant	VisibleString	O
100A	VAR	Version logicielle attribuée par le fabricant	VisibleString	O
100B		Réservé		
::::	::::	::::	::::	
1017		Réservé		
1018	RECORD	Objet d'identité	Identity (0x23)	M
1019		Réservé		
::::	::::	::::	::::	
15FF		Réservé		
1600	RECORD	1er mapping PDO en réception	PDO Mapping (0x21)	C
1601	RECORD	2^{e} mapping PDO en réception	Mapping PDO	C
::::	::::	::::	::::	
17FF	RECORD	512^{e} mapping PDO en réception	Mapping PDO	C
1800		Réservé		
::::	::::	::::	::::	
19FF		Réservé		
1A00	RECORD	mapping du 1er PDO en émission	PDO Mapping (0x21)	C
1A01	RECORD	mapping du 2^{e} PDO en émission	Mapping PDO	C
::::	::::	::::	::::	
1BFF	RECORD	mapping du 512^{e} PDO en émission	Mapping PDO	C
1C00	ARRAY	Type de communication du gestionnaire de synchronisation	UNSIGNED8	M
1C01		Réservé		
::::	::::	::::	::::	

Index (hexadécimal)	Type d'objet	Name	Type	M/O/C
1C0F		Réservé		
1C10	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 0	UNSIGNED16	C
1C11	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 1	UNSIGNED16	C
1C12	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 2	UNSIGNED16	C
1C13	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 3	UNSIGNED16	C
1C14	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 4	UNSIGNED16	C
::::	::::	::::	::::	
1C2F	ARRAY	Attribution d'objet PDO du gestionnaire de synchronisation 31	UNSIGNED16	C
1C30	RECORD	Synchronisation du gestionnaire de synchronisation 0		O
::::	::::	::::	::::	
1C4F	RECORD	Synchronisation du gestionnaire de synchronisation 31		O
1C50		Réservé		
::::	::::	::::	::::	
1FFF		Réservé		

5.6.7.4.1 Device type (Type de l'appareil)

L'entrée de type d'appareil du dictionnaire d'objets (indice 0x1000) est spécifiée dans le Tableau 67.

Tableau 67 – Type d'appareil

Attribut	Valeur
Nom	Type d'appareil
Code objet	VAR
Data type	UNSIGNED32
Catégorie	Obligatoire
Accès	Lecture seule
Mapping PDO	Non
Valeur	Bit 0 à bit 15: profil d'appareil utilisé, 0x0000 si aucun profil d'appareil normalisé n'est utilisé Bit 16 à bit 31: informations supplémentaires dépendant du profil d'appareil utilisé

5.6.7.4.2 Registre d'erreurs

L'entrée de registre d'erreurs du dictionnaire d'objets (indice 0x1001) est spécifiée dans le Tableau 68.

Tableau 68 – Registre d'erreurs

Attribut	Valeur
Nom	Registre d'erreurs
Code objet	VAR
Data type	UNSIGNED8
Catégorie	Facultative
Accès	Lecture seule
Mapping PDO	
Valeur	Bit 0: erreur générique Bit 1: erreur de courant Bit 2: erreur de tension Bit 3: erreur de température Bit 4: erreur de communication Bit 5: erreur spécifique au profil d'appareil Bit 6: réservé Bit 7: erreur spécifique au fabricant

5.6.7.4.3 Nom d'appareil attribué par le fabricant

L'entrée du dictionnaire d'objets correspondant au nom d'appareil attribué par le fabricant (indice 0x1008) est spécifiée dans le Tableau 69.

Tableau 69 – Nom d'appareil attribué par le fabricant

Attribut	Valeur
Nom	Nom d'appareil attribué par le fabricant
Code objet	VAR
Data type	VISIBLE_STRING
Catégorie	Facultative
Accès	Lecture seule
Mapping PDO	Non
Valeur	Nom de l'appareil, sous forme de chaîne ne se terminant pas par zéro

5.6.7.4.4 Version matérielle attribuée par le fabricant

L'entrée du dictionnaire d'objets correspondant à la version matérielle attribuée par le fabricant (indice 0x1009) est spécifiée dans le Tableau 70.

Tableau 70 – Version matérielle attribuée par le fabricant

Attribut	Valeur
Nom	Version matérielle attribuée par le fabricant
Code objet	VAR
Data type	VISIBLE_STRING
Catégorie	Facultative
Accès	Lecture seule
Mapping PDO	Non
Valeur	Version matérielle de l'appareil, sous forme de chaîne ne se terminant pas par zéro

5.6.7.4.5 Version logicielle attribuée par le fabricant

L'entrée du dictionnaire d'objets correspondant à la version logicielle attribuée par le fabricant (indice 0x100A) est spécifiée dans le Tableau 71.

Tableau 71 – Version logicielle attribuée par le fabricant

Attribut	Valeur
Nom	Version logicielle attribuée par le fabricant
Code objet	VAR
Data type	VISIBLE_STRING
Catégorie	Facultative
Accès	R
Mapping PDO	Non
Valeur	Version logicielle de l'appareil, sous forme de chaîne ne se terminant pas par zéro

5.6.7.4.6 Objet d'identité

L'entrée d'objet d'identité du dictionnaire d'objets (indice 0x1018) est spécifiée dans le Tableau 72.

Tableau 72 – Objet d'identité

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre d'entrées	UNSIGNED8	M	R	Non	4
1	ID du fournisseur	UNSIGNED32	M	R	Non	attribué de manière non ambiguë par l'ETG
2	Code produit	UNSIGNED32	M	R	Non	attribué de manière non ambiguë par le fournisseur
3	Numéro de révision	UNSIGNED32	M	R	Non	attribué de manière non ambiguë par le fournisseur
4	Numéro de série	UNSIGNED32	M	R	Non	attribué de manière unique à cet appareil par le fournisseur 0 si aucun numéro de série n'a été attribué

5.6.7.4.7 Mapping PDO en réception

L'entrée de mapping PDO en réception du dictionnaire d'objets (indice 0x1600 à 0x17FF) est spécifiée dans le Tableau 73.

Tableau 73 – Mapping PDO en réception

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre d'objets dans l'objet PDO	UNSIGNED8	M	R/RW [a]	Non	0 – 254 écriture possible si le mapping de variable est pris en charge
1	Premier objet de sortie à mapper	UNSIGNED32	C	R/RW [a]	Non	Bit 0 à bit 7: longueur des objets mappés, exprimée en bits (pour un espace dans l'objet PDO: il doit s'agir de la longueur en bits de l'espace) Bit 8 à bit 15: sous-indice de l'objet mappé (0 pour un espace dans l'objet PDO) Bit 16 à bit 31: indice de l'objet mappé (pour un espace dans l'objet PDO: doit être égal à zéro)
..						
n	Dernier objet de sortie à mapper	UNSIGNED32	C	R/RW [a]	Non	
[a] L'écriture est possible si l'attribution d'objet PDO variable est prise en charge.						

5.6.7.4.8 Mapping de PDO en émission

L'entrée de mapping de PDO en émission du dictionnaire d'objets (indice 0x1A00 à 0x1BFF) est spécifiée dans le Tableau 74.

Tableau 74 – Mapping de PDO en émission

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre d'objets dans l'objet PDO	UNSIGNED8	M	R/RW [a]	Non	0 – 254 écriture possible si le mapping de variable est pris en charge
1	Premier objet de sortie à mapper	UNSIGNED32	C	R/RW [a]	Non	Bit 0 à bit 7: longueur des objets mappés, exprimée en bits (pour un espace dans l'objet PDO: il doit s'agir de la longueur en bits de l'espace) Bit 8 à bit 15: sous-indice de l'objet mappé (0 pour un espace dans l'objet PDO) Bit 16 à bit 31: indice de l'objet mappé (pour un espace dans l'objet PDO: doit être égal à zéro)
..						
n	Dernier objet de sortie à mapper	UNSIGNED32	C	R/RW [a]	Non	
[a] L'écriture est possible si l'attribution d'objet PDO variable est prise en charge.						

5.6.7.4.9 Type de communication du gestionnaire de synchronisation

L'entrée du type de communication du gestionnaire de synchronisation dans le dictionnaire d'objets (indice 0x1C00) est spécifiée dans le Tableau 75.

Tableau 75 – Type de communication du gestionnaire de synchronisation

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre de canaux de gestionnaire de synchronisation utilisés	UNSIGNED8	M	R	Non	4– 32
1	Type de communication du gestionnaire de synchronisation 0	UNSIGNED8	C	R	Non	configurable[a] 0: inutilisé 1: réception dans la boîte aux lettres (maître à esclave) 2: envoi dans la boîte aux lettres (esclave à maître) 3: sortie de données de processus 4: entrée de données de processus (esclave à maître)
2	Type de communication du gestionnaire de synchronisation 1	UNSIGNED8	C	R	Non	configurable[a] 0: inutilisé 1: réception dans la boîte aux lettres (maître à esclave) 2: envoi dans la boîte aux lettres (esclave à maître) 3: sortie de données de processus 4: entrée de données de processus (esclave à maître)
3	Type de communication du gestionnaire de synchronisation 2	UNSIGNED8	C	R	Non	configurable[a] 0: inutilisé 1: réception dans la boîte aux lettres (maître à esclave) 2: envoi dans la boîte aux lettres (esclave à maître) 3: sortie de données de processus 4: entrée de données de processus (esclave à maître)
4	Type de communication du gestionnaire de synchronisation 3	UNSIGNED8	C	R	Non	configurable[a] 0: inutilisé 1: réception dans la boîte aux lettres (maître à esclave) 2: envoi dans la boîte aux lettres (esclave à maître) 3: sortie de données de processus 4: entrée de données de processus (esclave à maître)
5 – n	Type de communication du gestionnaire de synchronisation 4 – (n-1)	UNSIGNED8	C	R	Non	configurable[a] 0: inutilisé 1: réception dans la boîte aux lettres (maître à esclave) 2: envoi dans la boîte aux lettres (esclave à maître) 3: sortie de données de processus

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
						4: entrée de données de processus (esclave à maître)
a Il convient d'utiliser le type de communication du gestionnaire de synchronisation comme suit: Type de communication du gestionnaire de synchronisation 0: 1 réception dans la boîte aux lettres Type de communication du gestionnaire de synchronisation 1: 2 envoi dans la boîte aux lettres Type de communication du gestionnaire de synchronisation 2: 3 sortie de données de processus Type de communication du gestionnaire de synchronisation 3: 4 entrée de données de processus Si la boîte aux lettres n'est pas prise en charge, il convient d'utiliser ce type comme suit: Type de communication du gestionnaire de synchronisation 0: 3 sortie de données de processus Type de communication du gestionnaire de synchronisation 1: 4 entrée de données de processus						

5.6.7.4.10 Attribution d'objet PDO du gestionnaire de synchronisation

5.6.7.4.10.1 Voie de Sync Manager

L'entrée du dictionnaire d'objets correspondant aux canaux 0 à 31 du gestionnaire de synchronisation (indice 0x1C10 à 0x1C2F) est spécifiée dans le Tableau 76. Si un canal de gestionnaire de synchronisation est utilisé comme gestionnaire de synchronisation de boîte aux lettres, l'objet correspondant doit être indisponible ou le sous-indice 0 doit être égal à zéro.

Tableau 76 – Canaux 0 à 3 1 du gestionnaire de synchronisation

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre d'objets TxPDO attribués	UNSIGNED8	C	R/RW[a]	Non	0– 254
1 – n	Indice d'objet de mapping PDO de l'objet PDO attribué	UNSIGNED16	C	R/RW[a]	Non	Soit 0x1600:RxPDO 1 0x1601:RxPDO 2 ... 0x17FF:RxPDO 512 Soit 0x1A00: TxPDO 1 0x1A01: TxPDO 2 ... 0x1BFF: TxPDO 512
a L'écriture est possible si l'attribution d'objet PDO variable est prise en charge.						

5.6.7.4.11 Synchronisation du gestionnaire de synchronisation

L'entrée du dictionnaire d'objets correspondant à la synchronisation du gestionnaire de synchronisation (indice 0x1C30 à 0x1C4F) est spécifiée dans le Tableau 77.

Tableau 77 – Synchronisation du gestionnaire de synchronisation

Sous-indice	Description	Data type	M/O/C	Access	Mapping PDO	Valeur
0	Nombre de paramètres de synchronisation	UNSIGNED8	O	R	Non	1 – 3
1	Type de synchronisation	UNSIGNED16	O	RW	Non	• 0: non synchronisé • 1: Synchrone – synchronisé avec l'événement de couche AL sur ce gestionnaire de synchronisation • 2: DC Sync0 – synchronisé avec l'événement de couche AL de Sync0 • 3: DC Sync1 – synchronisé avec l'événement de couche AL de Sync1 • 32: SyncSm0 – synchronisé avec l'événement de couche AL de SM0 • 33: SyncSm1 – synchronisé avec l'événement de couche AL de SM1 • … • 63: SyncSm31 – synchronisé avec l'événement de couche AL de SM31
2	Durée de cycle	UNSIGNED32	O	RW	Non	laps de temps séparant deux événements (en ns)
3	Délai de changement	UNSIGNED32	O	RW	Non	laps de temps séparant l'événement de couche AL lié et l'action associée (en ns)

5.7 Codage du protocole EoE

5.7.1 Initiate EoE

5.7.1.1 Demande de déclenchement du protocole EoE

Les types d'attributs de la demande de déclenchement du protocole EoE sont décrits à la Figure 32.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          FragmentNumber:     6;
  unsigned          CompleteSize:       6;
  unsigned          FrameNumber:        4;
} TEOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TCOEHEADER        CoeHeader;
  TEOEHEADER        EoeHeader;
BYTE              Data[MAX_EOE_DATA_SIZE];
} TINIEOEREQ;
```

Figure 32 – Structure générale du protocole EoE

Le codage de la demande de déclenchement du protocole EoE est spécifié dans le Tableau 78.

Tableau 78 – Demande de déclenchement EoE

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N = 0x24 + y*0x20: longueur des données du service de boîte aux lettres (y = 0 à 0x2F)
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x00
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x00: au moins un service de fragment EoE suit 0x01: la trame Ethernet complète figure dans la partie de données
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée après les données EoE dans le dernier fragment 0x01: une valeur d'horodatage sera ajoutée après les données EoE dans le dernier fragment
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée 0x01: valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	
	Fragment Number (Numéro de	Unsigned6	0x00

Partie de la trame	Champ de données	Data type	Valeur/Description
	fragment)		
	Complete size	Unsigned6	en blocs de 32 octets Tronc ((taille totale de la trame Ethernet en octets + 31)/32)
	Frame number	Unsigned4	numéro de la trame Ethernet
	EoE Data (Données EoE)	BYTE[N-4]	(N-4) octets de la première portion de la trame Ethernet (sans préambule, SFD ni FCS; 4 octets de moins si un horodatage est compris)
(facultatif)	Timestamp	Unsigned32	heure de réception de la trame, exprimée en ns, au début de l'adresse de destination (Destination Address, DA)

5.7.1.2 Réponse de déclenchement du protocole EoE

Les types d'attributs de la réponse de déclenchement du protocole EoE sont décrits à la Figure 33.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
  TimeStamp         Unsigned32;
} TINIEOERES;
```

Figure 33 – Structure de l'horodatage EoE

Le codage de la réponse de déclenchement du protocole EoE est spécifié dans le Tableau 79.

Tableau 79 – Réponse de déclenchement du protocole EoE

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N = 0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x03
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x00
	Time appended	Unsigned1	0x01: une valeur d'horodatage sera ajoutée
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	

Partie de la trame	Champ de données	Data type	Valeur/Description
	Fragment Number (Numéro de fragment)	Unsigned6	0x00
	Complete size	Unsigned6	0x00
	Frame number	Unsigned4	numéro de la trame Ethernet
	Timestamp	Unsigned32	heure d'envoi de la trame, exprimée en ns, au début de l'adresse DA

5.7.2 Données de fragmentation EoE

Les types d'attributs de EoE Fragment Data (données de fragmentation EoE) sont décrits à la Figure 34.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 34 – Structure de données de fragmentation EoE

Le codage des EoE Fragment Data (données de fragmentation EoE) est spécifié dans le Tableau 80.

Tableau 80 – Données de fragmentation EoE

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x04: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x00
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x00: au moins un service de fragment EoE suit 0x01: dernière partie de données de la trame Ethernet concernée (horodatage compris)
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée après les données EoE dans le dernier fragment 0x01: une valeur d'horodatage sera ajoutée après les données EoE dans le dernier fragment
	Time Request	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi

Partie de la trame	Champ de données	Data type	Valeur/Description
	(Demande de temps)		demandée 0x01: valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	
	Fragment Number (Numéro de fragment)	Unsigned6	0x01-0x2F: numéro du fragment de trame Ethernet
	Offset	Unsigned6	position relative en blocs de 32 octets du fragment de trame Ethernet
	Frame number	Unsigned4	numéro de la trame Ethernet
	EoE Data (Données EoE)	BYTE[N-4]	(N-4) octets de la première portion de la trame Ethernet (sans préambule, SFD ni FCS; 4 octets de moins si un horodatage est compris)
(facultatif)	Timestamp	Unsigned32	heure de réception de la trame, exprimée en ns, au début de l'adresse de destination (Destination Address, DA)

5.7.3 Élément de données EoE

Le codage des EoE Data (données EoE) est spécifié dans le Tableau 81.

Tableau 81 – Données EoE

Partie de la trame	Champ de données	Data type	Valeur/Description
Ethernet	Dest MAC	BYTE[6]	adresse MAC de destination, selon les spécifications de l ISO/CEI 8802-3
	Voir	BYTE[6]	adresse MAC source, selon les spécifications de l'ISO/CEI 8802-3
(facultatif)	Balise VLAN	BYTE[4]	0x81, 0x00 et deux octets d'informations de contrôle de la balise selon la norme IEEE 802.1Q
	Type Ether	BYTE[2]	attribué par l'IEEE
Trame de l'utilisateur	Données		données de l'utilisateur (chaîne d'octets) ou paramètre EoE
	Remplissage	BYTE[n]	doit être inséré si l'unité PDU de couche DLL compte moins de 64 octets, conformément à l' ISO/CEI 8802-3

5.7.4 Réglage de paramètre IP

5.7.4.1 Demande de réglage de paramètre IP

Les types d'attributs de Set IP Parameter Request (demande de réglage de paramètre IP) sont décrits à la Figure 35.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 35 – Structure de la demande de réglage de paramètre IP

Le codage de Set IP Parameter Request (demande de réglage de paramètre IP) est décrit dans le Tableau 82.

Tableau 82 – Demande de réglage de paramètre IP

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x02
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x01: dernière partie de données
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	
	Fragment Number (Numéro de fragment)	Unsigned6	0x00
	Offset	Unsigned6	0x00
	Frame number	Unsigned4	0x00
Paramètre EoE	MAC included (MAC incluse)	Unsigned1	1, adresse MAC selon l'ISO/CEI 8802-3
	IP address included (adresse IP incluse)	Unsigned1	1, adresse IP selon l'IETF RFC 791
	Subnet Mask included (masque de sous-réseau inclus)	Unsigned1	1, masque de sous-réseau l'IETF RFC 791
	Default Gateway included (passerelle par défaut incluse)	Unsigned1	1, adresse de passerelle par défaut selon l'IETF RFC 791
	DNS Server IP Address included (adresse IP de serveur DNS incluse)	Unsigned1	1, adresse IP du serveur DNS selon l'IETF RFC 791
	DNS Name included (nom de serveur DNS inlcus)	Unsigned1	1, nom du serveur DNS selon l'IETF RFC 791
	réservé	Unsigned26	
	MAC	BYTE[6]	adresse MAC selon ISO/CEI 8802-3
	IP address	BYTE[4]	adresse IP selon IETF RFC 791
	Subnet mask	BYTE[4]	masque de sous-réseau selon IETF RFC 791 et IETF RFC 826
	passerelle par défaut	BYTE[4]	adresse de passerelle par défaut selon IETF RFC 791

Partie de la trame	Champ de données	Data type	Valeur/Description
	DNS Server IP Address (adresse IP de serveur DNS)	BYTE[4]	adresse IP du serveur DNS selon IETF RFC 791
	DNS Name	char[32]	nom du serveur DNS selon IETF RFC 791

5.7.4.2 Réponse de réglage de paramètre IP

Les types d'attributs de Set IP Parameter Response (réponse de réglage de paramètre IP) sont décrits à la Figure 36.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          Result:            16;
} TEOEPARAHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEPARAHEADER    EoeHeader;
} TEOEFRAGREQ;
```

Figure 36 – Structure de la réponse de réglage de paramètre IP

Le codage de Set IP Parameter Response (réponse de réglage de paramètre IP) est décrit dans le Tableau 83.

Tableau 83 – Réponse de réglage de paramètre IP

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N = 0x04: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x03
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x01: dernière partie de données
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée

Partie de la trame	Champ de données	Data type	Valeur/Description
	Réservé	Unsigned5	
	Result	Unsigned16	voir le Tableau 84

Tableau 84 – Paramètre de résultat EoE

Code de résultat	Signification
0x0000	réussite
0x0001	erreur non spécifiée
0x0002	type de trame non pris en charge
0x0201	absence de prise en charge IP
0x0202	Protocole DHCP (Dynamic Host Configuration Protocol (protocole de configuration dynamique de l'hôte)) non pris en charge Si le Maître envoie une Demande de réglage de paramètre IP avec l'adresse IP “0.0.0.0”, il convient que l'esclave envoie un DCHP_Discover pour obtenir une adresse IP. Si l'esclave ne prend pas en charge le protocole DHCP, il convient qu'il envoie une réponse avec le champ Result mis à “DHCP not supported” («protocole DHCP non pris en charge»)
0x0401	absence de prise en charge du filtre

5.7.5 Réglage du filtre d'adresse

5.7.5.1 Demande de réglage du filtre MAC

Les types d'attributs de Set MAC Filter Request (demande de réglage du filtre MAC) sont décrits à la Figure 37.

```
typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEHEADER        EoeHeader;
BYTE              Data[MAX_EOE_DATA_SIZE];
} TEOEFRAGREQ;
```

Figure 37 – Structure de la demande de réglage du filtre MAC

Le codage de Set MAC Filter Request (demande de réglage de filtre MAC) est décrit dans le Tableau 85.

Tableau 85 – Demande de réglage de filtre MAC

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x04
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x01: dernière partie de données
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	
	Fragment Number (Numéro de fragment)	Unsigned6	0x00
	Offset	Unsigned6	0x00
	Frame number	Unsigned4	0x00
Paramètre EoE	MAC filter count (nombre total de filtres MAC)	Unsigned4	nombre d'adresses MAC selon l'ISO/CEI 8802-3 acceptées par les ports Ethernet de l'esclave concerné
	MAC filter mask (masque de filtrage MAC)	Unsigned2	nombre de masques d'adresses MAC selon ISO/CEI 8802-3 combinés avec un filtre par les ports Ethernet de l'esclave concerné
	Réservé	Unsigned1	masque de sous-réseau selon IETF RFC 791
	Inhibit Broadcast (Interdiction de diffusion)	Unsigned1	filtrage des messages de diffusion
	Réservé	Unsigned8	
(conditionnel)	List of MAC Address (Liste d'adresse MAC)	List of BYTE[6]	adresse MAC selon ISO/CEI 8802-3
(conditionnel)	List of MAC address filter	List of BYTE[6]	adresse MAC selon ISO/CEI 8802-3 la présence d'un bit défini indique une comparaison entre ce bit de l'adresse MAC de destination et l'entrée correspondante de l'adresse MAC affichée

5.7.5.2 Set MAC Filter Response (Réponse de réglage du filtre MAC)

Les types d'attributs de Set MAC Filter Response (réponse de réglage du filtre MAC) sont décrits à la Figure 38.

```
typedef struct
{
  unsigned          FrameType:          4;
  unsigned          Port:               4;
  unsigned          LastFragment:       1;
  unsigned          TimeAppended:       1;
  unsigned          TimeRequested:      1;
  unsigned          Reserved:           5;
  unsigned          Result:            16;
} TEOEPARAHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TEOEPARAHEADER    EoeHeader;
} TEOEFRAGREQ;
```

Figure 38 – Structure de la réponse de réglage du filtre MAC

Le codage de Set MAC Filter Response (réponse de réglage de filtre MAC) est décrit dans le Tableau 86.

Tableau 86 – Réponse de réglage de filtre MAC

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N = 0x08: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x02: EoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête EoE	FrameType	Unsigned4	0x05
	Port	Unsigned4	0x00: envoi sans port de destination spécifique 0x01-0x0F: ports sélectionnés
	Last fragment	Unsigned1	0x01: dernière partie de données
	Time appended	Unsigned1	0x00: aucune valeur d'horodatage ne sera ajoutée
	Time Request (Demande de temps)	Unsigned1	0x00: aucune valeur d'horodatage de l'envoi demandée
	Réservé	Unsigned5	
	Result	Unsigned16	voir le Tableau 84

5.8 Codage FoE

5.8.1 Demande de lecture

Les types d'attributs de la demande de lecture sont décrits à la Figure 39.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             Password;
  char              FileName[MAX_FILE_NAME_SIZE};
} TFOEREADREQ;
```

Figure 39 – Structure de la demande de lecture

Le codage de FoE Read Request (demande de lecture FoE) est spécifié dans le Tableau 87.

Tableau 87 – Demande de lecture

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x01: Demande de lecture
	Réservé	BYTE	doit être égal à zéro
En-tête de lecture	Mot de passe	DWORD	0: mot de passe inutilisé 1-0xFFFFFFFF: mot de passe
	File name (nom de fichier)	char[n-6]	nom du fichier à lire

5.8.2 Demande d'écriture

Les types d'attributs de Write Request (demande d'écriture) sont décrits à la Figure 40.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             Password;
  char              FileName[MAX_FILE_NAME_SIZE};
} TFOEWRITEREQ;
```

Figure 40 – Structure de la demande d'écriture

Le codage de FoE Write Request (demande d'écriture FoE) est spécifié dans le Tableau 88.

Tableau 88 – Demande d'écriture

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x02: Demande d'écriture
	Réservé	BYTE	doit être égal à zéro
En-tête d'écriture	mot de passe	DWORD	0: mot de passe inutilisé 1-0xFFFFFFFF: mot de passe
	File name	char[n-6]	nom du fichier dans lequel les données seront écrites

5.8.3 Demande de données

Les types d'attributs de Data Request (demande de données) sont décrits à la Figure 41.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             PacketNo;
BYTE              Data[MAX_DATA_SIZE};
} TFOEDATAREQ;
```

Figure 41 – Structure de la demande de données

Le codage de FoE Data Request (demande de données FoE) est spécifié dans le Tableau 89.

Tableau 89 – Demande de données

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N > 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x03: Demande de données
	Réservé	BYTE	doit être égal à zéro
En-tête de données	Packet number	DWORD	1-0xFFFFFFFF
	Données	BYTE[n-6]	données de fichier

5.8.4 Demande d'acquittement (Acknowledgement, Ack)

Les types d'attributs de Ack Request (demande d'acquittement) sont décrits à la Figure 42.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             PacketNo;
} TFOEACKREQ;
```

Figure 42 – Structure de la demande d'acquittement

Le codage de FoE Ack Request (demande d'acquittement FoE) est spécifié dans le Tableau 90.

Tableau 90 – Demande d'acquittement

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x04: Demande d'acquittement (Acknowledgement, Ack)
	Réservé	BYTE	doit être égal à zéro
En-tête Ack	Packet number	DWORD	0: acquittement d'une demande d'écriture 1-0xFFFFFFFF: acquittement d'une demande de données

5.8.5 Demande d'informations d'erreur

Les types d'attributs de Error Request (demande d'informations d'erreur) sont décrits à la Figure 43.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  DWORD             ErrorCode;
  char              ErrorText[MAX_ERROR_TEXT_SIZE};
} TFOEERRORREQ;
```

Figure 43 – Structure de la demande d'informations d'erreur

Le codage de FoE Error Request (demande d'informations d'erreur FoE) est spécifié dans le Tableau 91.

Tableau 91 – Demande d'informations d'erreur

Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N >= 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse … 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x05: Demande d'informations d'erreur
	Réservé	BYTE	doit être égal à zéro
En-tête d'erreur	Code d'erreur	DWORD	1-0xFFFFFFFF
	Error text	char[n-6]	description de l'erreur (facultative)

Les codes d'erreur de FoE sont spécifiés dans le Tableau 92.

Tableau 92 – Codes d'erreur de FoE

code d'erreur	signification
0x8000	Non défini
0x8001	Introuvable
0x8002	Accès refusé
0x8003	Disque saturé
0x8004	Illégalité
0x8005	Erreur de numéro de paquet
0x8006	Existe déjà
0x8007	Absence d'utilisateur
0x8008	Amorçage uniquement
0x8009	Incompatible avec l'amorçage
0x800A	Absence de droits
0x800B	Erreur de programme

5.8.6 Demande d'état occupé (Busy)

Les types d'attributs de Busy Request (demande Busy) sont décrits à la Figure 44.

```
typedef struct
{
  BYTE              OpCode;
  BYTE              Reserved;
} TFOEHEADER;

typedef struct
{
  TMBXHEADER        MbxHeader;
  TFOEHEADER        FoeHeader;
  WORD              Done;
  WORD              Entire;
  char              BusyText[MAX_BUSY_TEXT_SIZE};
} TFOEBUSYREQ;
```

Figure 44 – Structure de la demande Busy

Le codage de FoE Busy Request (demande Busy FoE) est spécifié dans le Tableau 93.

Tableau 93 – Demande Busy

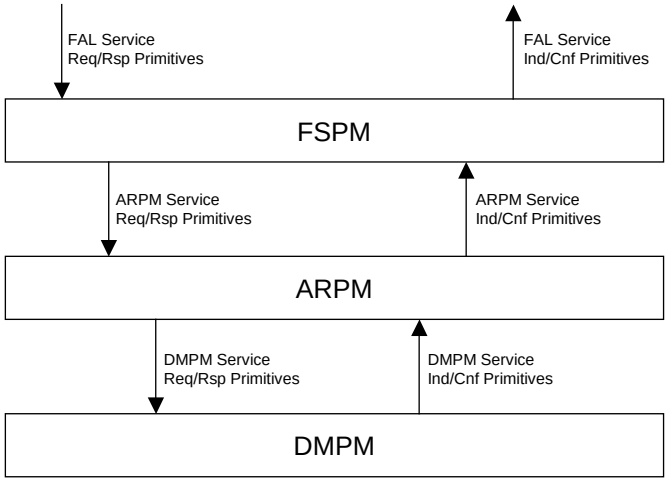
Partie de la trame	Champ de données	Data type	Valeur/Description
En-tête de boîte aux lettres	Length (Longueur)	WORD	N >= 0x06: longueur des données du service de boîte aux lettres
	Adresse	WORD	adresse de station de la source, si un maître est client; adresse de station de la destination, si un esclave est client
	Channel (voie)	Unsigned6	0x00 (réservé pour utilisation future)
	Priority (priorité)	Unsigned2	0x00: priorité la plus basse ... 0x03: priorité la plus haute
	Type	Unsigned4	0x04: FoE
	Cnt	Unsigned3	compteur des services de boîte aux lettres (0: réservé; 1: valeur de début; la valeur qui suit 7 est 1)
	Réservé	Unsigned1	0x00
En-tête FoE	OpCode	BYTE	0x06: Demande d'état occupé (Busy)
	Réservé	BYTE	doit être égal à zéro
En-tête Busy	Done	WORD	Si le champ Entire («entier») est “0”, le champ Done indique l'état d'avancement en pourcentage 0 à 100 (done en %) Si la valeur du champ Entire est différente de 0, la valeur indique la taille des données déjà transférées, exprimée dans la même unité que l'élément Entire
	Entire	WORD	Si la valeur de l'élément est égale à 0, le champ Done indique l'avancement du transfert de fichier, de 0 % à 100 %. Si la valeur de l'élément est différente de 0, le champ Entire indique la taille du fichier à transférer, dans une unité spécifiée. Dans ce cas, le champ Done indique la taille des données déjà transférées, dans la même unité. Un pourcentage peut être calculé à l'aide de ces deux valeurs: 100*Done/Entire
	Busy Text (texte de Busy)	char[n-6]	description de l'état occupé (facultative)

6 Diagrammes d'états de protocole de la FAL

6.1 Structure globale

6.1.1 Vue d'ensemble

La structure des diagrammes d'états de protocole de la couche FAL est définie à la Figure 45. Leur structure générale est conforme au modèle de machine de protocole de la sous-série CEI 61158-6.



Légende

Anglais	Français
FAL Service Req/Rsp Primitives	Primitives de demande/réponse du service FAL
FAL Service Ind/Cnf Primitives	Primitives d'indication/de confirmation du service FAL
ARPM Service Req/Rsp Primitives	Primitives de demande/réponse du service ARPM
ARPM Service Ind/Cnf Primitives	Primitives d'indication/de confirmation du service ARPM
DMPM Service Req/Rsp Primitives	Primitives de demande/réponse du service DMPM
DMPM Service Ind/Cnf Primitives	Primitives d'indication/de confirmation du service DMPM

Figure 45 – Relations entre les machines de protocole

Le comportement de la couche FAL est spécifié par trois machines de protocole intégrées. La machine de protocole de service FAL (Fieldbus Application Layer Service Protocol Machine (FSPM)) constitue l'interface de service entre les services FAL faisant partie de la spécification de classe FAL et le point AREP concerné.

La couche FAL de Type 12 fournit un ensemble de machines de protocole pour l'esclave. Les maîtres peuvent anticiper le comportement des esclaves.

La machine de protocole FSPM est chargée des activités suivantes.

– Accepter les primitives de service émises par l'utilisateur de service FAL et les convertir en primitives internes FAL.
– Sélectionner le diagramme d'états de la machine de protocole de relations AR (Application Relationship Protocol Machine (ARPM)) en fonction du mécanisme

d'adressage implicite et envoyer des primitives internes FAL avec les paramètres de service au diagramme d'états ARPM.

– Accepter les primitives FAL internes provenant du diagramme d'états ARPM et les convertir en primitives de service destinées à l'utilisateur de service FAL.

– Remettre les primitives de service FAL à l'utilisateur FAL.

La machine de protocole ARPM spécifie le type de transport associé à la relation d'application.

La machine de protocole de mapping de couche DLL (DL Mapping Protocol Machine (DMPM)) spécifie le mapping avec la couche DLL. Elle définit donc deux machines de protocole: la machine de protocole LMPM et la machine de protocole MAC.

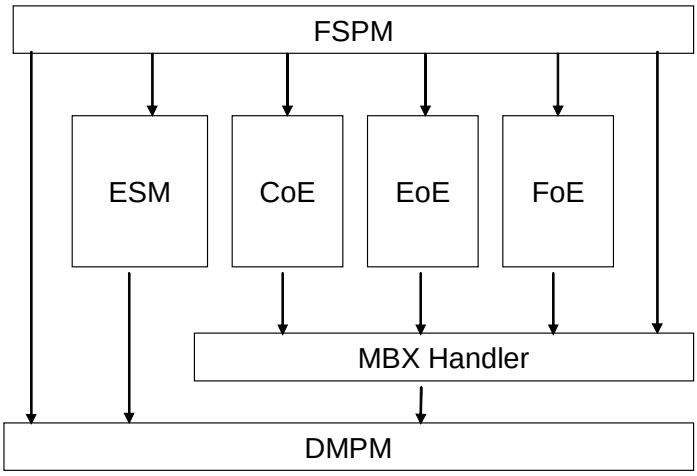
6.1.2 Machines de protocole FSPM

Les machines de protocole FSPM coordonnent les diagrammes d'états sous-jacents utilisés pour le traitement des divers services et relations entre applications.

La machine de protocole FSPM est essentiellement une machine de mapping de protocoles. Sa tâche principale consiste à transmettre le service à la machine de protocole qui en est responsable, ainsi qu'à faire suivre les confirmations et les réponses à l'utilisateur. Cette machine comprend également un schéma élémentaire de contrôle de redondance permettant de faire collaborer deux relations AR dans une même entité avec une disponibilité supérieure.

6.1.3 Machines de protocole ARPM

Les machines de protocole ARPM sont chargées de l'exécution des différentes procédures de service. Leur structure globale est décrite à la Figure 46. L'interaction entre les données de processus est traitée directement par la DLL (couche D)L et contrôlée par le diagramme d'états ESM. L'application dispose de diverses manières d'exploiter les protocoles de boîte aux lettres.



Légende

Anglais	Français
FSPM	FSPM
ESM	ESM
CoE	CoE
EoE	EoE
FoE	FoE

Anglais	Français
MBX Handler	Gestionnaire de boîte aux lettres
DMPM	DMPM

Figure 46 – Machines de protocole de relation AR

6.1.4 Machines de protocole DMPM

La machine de protocole DMPM connecte les autres diagrammes d'états à la couche 2. Elle assure la coordination de tous les diagrammes d'états, pour ce qui concerne la configuration et le traitement des erreurs associées à l'utilisation de la couche DLL. Elle mappe les fonctions avec les services DLL de couche 2. La machine de protocole DMPM génère les paramètres de couche 2 nécessaires au service, puis reçoit les confirmations et indications provenant de la couche 2 et les transmet à l'utilisateur DMPM concerné.

6.2 Diagramme d'états de l'entité ACE (AP Context Entity)

Aucun diagramme d'états de l'entité ACE n'est défini pour ce protocole.

NOTE Le diagramme d'états de l'entité ACE ((AP Context Entity) fait partie du modèle CEI 61158-6.

6.3 Machine de protocole FSPM

Les services spécifiés dans la CEI 61158-5-12 sont mappés directement avec les services des machines de protocole ARPM.

6.4 Machines de protocole ARPM

6.4.1 Diagramme d'états de couche AL

6.4.1.1 Description

Le diagramme d'états ESM est responsable de la coordination du maître et de l'esclave au démarrage et pendant l'exploitation. Les changements d'état résultent essentiellement des interactions entre maître et esclave. Ils sont principalement associés aux opérations d'écriture dans le mot AL Control.

Une fois la couche DL et la couche AL initialisées, la machine passe à l'état d'initialisation (INIT). L'état d'initialisation définit la racine de la relation de communication entre le maître et l'esclave au niveau de la couche Application. Aucune communication directe entre le maître et l'esclave dans la couche AL n'est possible. Le maître utilise l'état d'initialisation pour initialiser un ensemble de registres de configuration. Si l'esclave prend en charge une boîte aux lettres, les configurations de gestionnaire de synchronisation correspondantes sont également réalisées à l'état d'initialisation.

Si l'esclave prend en charge la boîte aux lettres facultative et que celle-ci a été paramétrée, la machine peut passer à l'état de pré-exploitation. Le maître comme l'esclave peuvent utiliser la boîte aux lettres et les protocoles appropriés pour échanger des paramètres et des initialisations spécifiques à l'application. Aucune communication de données de processus n'est possible dans cet état.

La machine peut passer à l'état d'exploitation sûre si le tampon d'entrée a été paramétré, que l'esclave prend les entrées en charge et que le maître demande des entrées. L'application de l'esclave doit remettre les données d'entrée réelles sans traiter les données de sortie. Les sorties réelles de l'esclave doivent être mises dans leur "état de sécurité".

La machine peut passer à l'état d'exploitation si le tampon de sortie a été paramétré et que les sorties réelles ont été remises à l'esclave (à condition que les sorties de l'esclave soient utilisées). L'application de l'esclave doit remettre les données d'entrée réelles et l'application du maître doit fournir les données de sortie.

Dans l'état facultatif d'amorçage (Bootstrap), l'application de l'esclave doit être en mesure d'accepter les paramètres persistants téléchargés avec le protocole FoE.

Le diagramme d'états ESM définit quatre états, qui doivent être pris en charge:

- Init,
- Pre-Operational,
- Safe-Operational, et
- Operational.

Tous les changements d'états sont possibles, à l'exception du cas de l'état ‘Init’ où seule la transition vers l'état ‘Pre-Operational’ est possible et celle du cas de l'état ‘Pre-Operational’ où il n'existe aucun changement direct d'état vers ‘Operational’.

Les changements d'états sont normalement demandés par le maître. Le maître demande une écriture dans le registre de commandes d'AL qui donne lieu à une indication d'événement de registre ‘AL Control’ dans l'esclave. L'esclave doit répondre au changement ayant affecté le paramètre AL Control par le biais d'un service d'écriture d'état AL local, après la réussite ou l'échec du changement d'état. En cas d'échec du changement d'état demandé, l'esclave doit répondre en définissant l'indicateur d'erreur.

Les appareils EtherCAT peuvent être des appareils dits simples ou complexes. Ces catégories d'appareils diffèrent par leur comportement, concernant la demande et la réponse AL Control. À la réception d'un indicateur d'accusé de réception AL, les esclaves complexes réinitialisent si possible l'indicateur d'erreur AL (émulation d'appareil inactive) alors que les esclaves simples copient l'indicateur d'accusé de réception dans l'indicateur d'erreur AL (émulation d'appareil active).

Malgré cette différence de comportement, il convient qu'un maître ait la possibilité d'initialiser le réseau au moyen d'une commande de diffusion. Il doit donc être autorisé de réinitialiser tous les appareils au moyen d'une demande d'initialisation de diffusion avec l'indicateur Ack réglé sur False (registre AL Control = 0x0001). Les appareils complexes doivent alors réinitialiser l'indicateur d'erreur.

En cas d'erreur, le code d'état AL doit être réglé en premier, l'indicateur d'erreur devant être réglé ensuite. Une fois l'indicateur d'erreur effacé, il convient que le code d'état AL le soit également. Le maître doit ignorer le code d'état AL si l'indicateur d'erreur est acquitté.

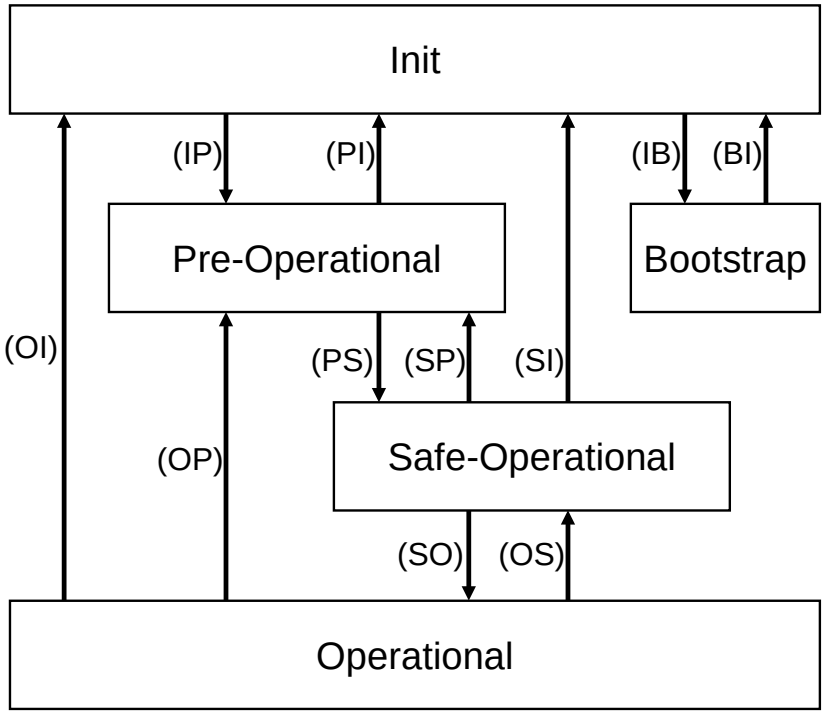
En cas de passage de l'état d'exploitation de la couche AL à l'état d'exploitation sûre avec erreur, le gestionnaire de synchronisation de sortie doit être désactivé. Le gestionnaire de synchronisation d'entrée ne doit être désactivé qu'en cas d'erreur se traduisant par des entrées non valides (erreur d'entrée ou de synchronisation).

Le gestionnaire de synchronisation de sortie doit être réactivé si l'erreur a été acquittée et qu'il ne reste aucune erreur de sortie.

Le gestionnaire de synchronisation d'entrée doit être réactivé si l'erreur a été acquittée et qu'il ne reste aucune erreur d'entrée.

L'état Bootstrap (amorce) est facultatif et il y a seulement une transition depuis l'état Init ou vers l'état Init. Le seul but de cet état est de télécharger vers l'aval le firmware (micro logiciel) de l’appareil. Dans l'état d'amorçage, la boîte aux lettres est active, mais limitée au protocole FoE.

Le diagramme d'états ESM est spécifié à la Figure 47.



Légende

Anglais	Français
Init	Initialisation
Pre-Operational	Pré-Exploitation
Bootstrap	Amorçage
Safe-Operational	Exploitation sûre
Operational	Exploitation

Figure 47 – Diagramme de l'ESM

Les services de gestion locaux ont, avec les changements dans le diagramme d'états ESM, les relations spécifiées dans le Tableau 94. Si plusieurs services sont associés à un même changement, l'application de l'esclave traitera tous les services liés.

Tableau 94 – Changements d'état et services de gestion locaux

Changement d'état	Service de gestion local
IP	Lancement de la communication de boîte aux lettres
PI	Arrêt de la communication de boîte aux lettres
PS	Lancement de la mise à jour des entrées
SP	Arrêt de la mise à jour des entrées
SO	Lancement de la mise à jour des sorties
OS	Arrêt de la mise à jour des sorties
OP	Arrêt de la mise à jour des sorties, arrêt de la mise à jour des entrées
SI	Arrêt de la mise à jour des entrées, arrêt de la communication de boîte aux lettres
OI	Arrêt de la mise à jour des sorties, arrêt de la mise à jour des entrées, arrêt de la communication de boîte aux lettres
IB	Lancement du mode amorçage
BI	Redémarrage de l'appareil

6.4.1.2 États du diagramme d'états ESM

6.4.1.2.1 Initialisation (Initialization, INIT)

L'état d'initialisation définit la racine de la relation de communication entre le maître et l'esclave au niveau de la couche Application. Aucune communication directe entre le maître et l'esclave dans la couche AL n'est possible. Le maître utilise l'état d'initialisation pour initialiser un ensemble de registres de configuration du contrôleur ESC. Si l'esclave prend en charge des services de boîte aux lettres, les configurations de gestionnaire de synchronisation correspondantes sont également réalisées dans l'état d'initialisation.

6.4.1.2.2 Pré-exploitation (Pre-Operational, PreOP)

Dans l'état de pré-exploitation, la boîte aux lettres est active si l'esclave prend en charge la boîte aux lettres facultative. Le maître comme l'esclave peuvent utiliser la boîte aux lettres et les protocoles appropriés pour échanger des paramètres et des initialisations spécifiques à l'application. Aucune communication de données de processus n'est possible dans cet état.

6.4.1.2.3 Exploitation sûre (Safe-Operational, SafeOP)

Dans l'état d'exploitation sûre, l'application de l'esclave doit remettre les données d'entrée réelles sans manipuler les données de sortie. Les sorties doivent être mises dans leur "état de sécurité".

6.4.1.2.4 Exploitation (Operational, OP)

Dans l'état d'exploitation, l'application de l'esclave doit remettre les données d'entrée réelles et l'application du maître doit remettre les données de sortie réelles.

6.4.1.2.5 Amorçage (Bootstrap, Boot)

Dans l'état facultatif d'amorçage (Bootstrap), l'application de l'esclave doit être en mesure d'accepter un nouveau microprogramme ("firmware") téléchargé avec le protocole FoE.

6.4.1.3 Définitions de primitives

6.4.1.3.1 Primitives échangées entre la couche DL et le diagramme d'états ESM

Le Tableau 95 montre les primitives de service émises par le diagramme d'états ESM et reçues par la couche DL, avec les paramètres associés.

Tableau 95 – Primitives adressées par le diagramme d'états ESM à la couche DL

Nom de la primitive	Paramètres associés	Fonctions
AL State Change.req	AL Status Application Specific AL Status Code	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-5-12
AL Control.rsp(+)	AL State AL Status Code	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-5-12
AL Control.rsp(-)	AL State AL Status Code	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-5-12

Le Tableau 96 montre les primitives de service émises par la couche DLL et reçues par le diagramme d'états ESM, avec les paramètres associés.

Tableau 96 – Primitives adressées par la couche DL au diagramme d'états ESM

Nom de la primitive	Paramètres associés	Fonctions
AL Control.ind	AL Control State Ack Flag ID Request	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-5-12

6.4.1.3.2 Primitives échangées entre la couche AL et le diagramme d'états ESM

Le Tableau 97 montre les primitives de service émises par la couche AL et reçues par le diagramme d'états ESM, avec les paramètres associés.

Tableau 97 – Primitives adressées par la couche Application au diagramme d'états ESM

Nom de la primitive	Paramètres associés	Fonctions
Stop Input		arrêt de la mise à jour des données de processus par l'application
SM Change		changement de configuration de gestionnaire de synchronisation (la primitive peut être activée/désactivée en local ou émise par la communication)

Le Tableau 98 montre les primitives de service émises par le diagramme d'états ESM et reçues par la couche AL, avec les paramètres associés.

Tableau 98 – Primitives adressées par le diagramme d'états ESM à la couche Application

Nom de la primitive	Paramètres associés	Fonctions
START_MBX_HANDLER		Lancement de la communication de boîte aux lettres INIT->PREOP: Lancement de la communication de boîte aux lettres par le biais de l'activation du gestionnaire de synchronisation
STOP_MBX_HANDLER		Arrêt de la communication de boîte aux lettres PREOP->INIT: Arrêt de la communication de boîte aux lettres par le biais de la désactivation du gestionnaire de synchronisation
START_INPUT_HANDLER		Lancement de la mise à jour des entrées PREOP->SAFEOP: Lancement de la mise à jour des entrées dans le contrôleur ESC par le biais de l'activation du gestionnaire de synchronisation Lancement de la mise à jour des sorties du contrôleur ESC par le biais de l'activation du gestionnaire de synchronisation
STOP_INPUT_HANDLER		Arrêt de la mise à jour des entrées SAFEOP->PREOP: Arrêt de la mise à jour des entrées dans le contrôleur ESC par le biais de la désactivation du gestionnaire

Nom de la primitive	Paramètres associés	Fonctions
		de synchronisation Arrêt de la mise à jour des sorties de le contrôleur ESC par le biais de la désactivation du gestionnaire de synchronisation Arrêt du chien de garde
START_LOCAL_OUTPUT_HANDLER		Lancement de la mise à jour des sorties locales SAFEOP->OP: Réglage des sorties sur les valeurs de traitement
STOP_LOCAL_OUTPUT_HANDLER		Arrêt de la mise à jour des sorties locales OP->SAFEOP: Réglage des sorties sur les valeurs de sécurité
ID_INFO		If ID Request AND IdSupported then AL_STATUS_CODE= ID value ID_FLAG = 1 else ID_FLAG = 0 end_if

6.4.1.3.3 Variables du diagramme d'états ESM

Le Tableau 99 définit les variables utilisées dans le diagramme d'états ESM.

Tableau 99 – Variables du diagramme d'états ESM

Nom de variable	Description
bootSupported	le mode d'amorçage est pris en charge par l'esclave
dcNotAccepted	l'appareil ne prend pas en charge les horloges DC et n'accepte pas les paramètres d'horloge DC. Si des paramètres d'horloge DC sont définis, l'appareil passera à l'état d'erreur. Le registre 0x0981 de commande de synchronisation doit être égal à zéro.
dcRequired	l'appareil requiert des paramètres d'horloge DC (les bits 0x0981:00 à 0x0981:02 doivent être réglés en conséquence). Les autres modes tels que FreeRun et le mode synchrone avec les événements du gestionnaire de synchronisation ne sont pas pris en charge
dcRunning	les horloges DC fonctionnent et sont utilisées pour la synchronisation entre le maître et l'application esclave.
dcSupported	l'appareil prend en charge au moins un mode d'exploitation qui utilise les horloges distribuées
IdSupported	L'appareil prend en charge l'Identification Explicite d'Appareil en utilisant le Registre 0x0134
localErrorCode	cause initiale d'une erreur locale antérieure qui a entraîné la désactivation de canaux de gestionnaire de synchronisation
localErrorFlag	indicateur de l'application locale signalant une erreur locale qui a entraîné la désactivation des gestionnaires de synchronisation. Utilisé en cas d'erreur provoquant le passage de l'état d'exploitation à l'état d'exploitation sûre
pllRunning	l'application locale est synchronisée avec l'événement d'horloge DC et l'application maîtresse
safeOp2OpTimeoutTimer	temporisation pour changement d'état de SafeOp en Op
dcEventReceived	L'esclave a reçu l'événement Sync0/Sync1 au moins une fois
waitForPllRunning	attendre que la boucle PLL locale soit verrouillée selon le cycle maître (attendre que pllRunning = TRUE). Le délai nécessaire doit être inférieur à l'expiration du passage de l'état d'exploitation sûre à l'état d'exploitation
wdEnabled	Le comportement du chien de garde est désactivé si au moins le temps du chien de garde (R400, R420) est nul.De plus, le comportement du chien de garde peut être désactivé si le bit du chien de garde du gestionnaire de synchronisation est désactivé (Registre du SyncManager +0x04.06=0) (car lors de l'utilisation de la fonctionnalité du chien de garde du ESC, ce chien de garde est activé seulement si le bit du chien de garde du gestionnaire de synchronisation est mis): TRUE: le comportement du chien de garde est activé FALSE: le comportement du chien de garde est désactivé Si un esclave possède seulement des entrées, la variable wdEnabled peut rester toujours FALSE
readyForOP	Variable interne qui est mise lorsque l'appareil est prêt à commuter vers OP en mode Non-DC, il convient que cette situation se produise lorsque les sorties ont été reçues pendant le temps de chien de garde si celui-ci est activé ou pendant l'état SafeOp si le chien de garde est désactivé. Cependant, un autre comportement peut se produire pour des raisons d'héritage.
smEventReceived	Variable interne qui indique que l'évènement SM (SM-event) a été reçu

6.4.1.3.4 Macros du diagramme d'états ESM

Le Tableau 100 définit les macros utilisées dans le diagramme d'états ESM.

Tableau 100 – Macros du diagramme d'états ESM

Nom de la macro	Description
SM_SETTINGS_0_1_MATCH	fonction locale comparant les paramètres du gestionnaire de synchronisation de boîte aux lettres, figurant dans la demande, aux paramètres locaux
SM_SETTINGS_2_TO_n_MATCH	fonction locale comparant les paramètres du gestionnaire de synchronisation des données de processus, figurant dans la demande, aux paramètres locaux Pour les esclaves dépourvus de boîte aux lettres, il peut s'agir du gestionnaire de synchronisation 0 à n
DC_ACTIVATED	l'application locale est synchronisée NOTE AssignActivate doit être combiné à une opération AND logique avec 0x0701 et doit être soit 0x0300, soit 0x0700 (tous les autres bits peuvent être mis selon les besoins)
DISABLE_SM_CHANNEL	le gestionnaire de synchronisation 2 doit être désactivé. C'est seulement en l'absence de gestionnaire de synchronisation de sortie ou en cas d'erreur d'entrée que le gestionnaire de synchronisation d'entrée doit être désactivé tandis que le gestionnaire de synchronisation 2 reste activé
ENABLE_SM_CHANNEL	les gestionnaires de synchronisation doivent être activés
RESTART_WD	Si le chien de garde est activé: le chien de garde du contrôleur ESC est mis en marche. De plus, une horloge de surveillance locale sur le contrôleur hôte peut être mise en marche

6.4.1.3.5 Fonctions du diagramme d'états ESM

Le Tableau 101 définit les fonctions utilisées dans le diagramme d'états ESM.

Tableau 101 – Fonctions du diagramme d'états ESM

Nom de la fonction	Description
Output Event	primitive adressée par la couche DL au diagramme d'états ESM: événement indiquant l'arrivée de nouvelles données de sortie
Le chien de garde a expiré	le chien de garde (local) est arrivé à expiration
SM_Chg	primitive adressée par la couche DL au diagramme d'états ESM: service d'événement de couche DLL utilisé pour indiquer un changement des paramètres de gestionnaire de synchronisation
AL_State Change.req (AL Status, AL Status Code)	primitive adressée par la couche Application au diagramme d'états ESM: l'application esclave signale un changement d'état
PLL Running Event	événement local indiquant que le maître est maintenant synchronisé avec l'application locale (les sorties sont envoyées par le maître et reçues par l'esclave avant l'événement DC)
SafeOp2OpTimeoutTimer Expired	événement indiquant que la temporisation de changement d'état local est arrivée à expiration
Start SafeOp2OpTimeoutTimer	la temporisation locale de la temporisation du passage de l'état d'exploitation sûre à l'état d'exploitation a démarré

Nom de la fonction	Description
AckSM_Chg	l'esclave doit renvoyer un accusé de réception de l'événement de changement du gestionnaire de synchronisation
Stop Inp	fonction locale de la couche AL permettant de désactiver la mise à jour des entrées
SM_SETTINGS_OBJECT_SYNC_TYPE_MATCH	fonction locale comparant les paramètres de synchronisation figurant dans la demande aux propriétés locales

6.4.1.3.6 Paramètres des primitives

Les paramètres utilisés avec les primitives échangées entre la couche DLL, le diagramme d'états ESM et la couche AL sont décrits dans la CEI 61158-5-12.

6.4.1.4 Table d'états du diagramme d'états ESM

Le Tableau 102 contient la description complète du diagramme d'états ESM.

Tableau 102 – Table d'états du diagramme d'états ESM

N	État actuel	Événement /Condition => Action	Prochain état
1.1	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_CONTROL_STATE = STATE_INIT ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
1.2	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
2	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
3	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP and SM_SETTINGS_0_AND_1_MATCH => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_PREOP START_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
4	initialisation (Initializa	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and	initialisation (Initialization, INIT)

N	État actuel	Événement /Condition => Action	Prochain état
	tion, INIT)	AL_CONTROL_STATE = STATE_PREOP and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	
5	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and BOOT_SUPPORTED and SM_SETTINGS_0_AND_1_MATCH => AL_ERROR_FLAG = 0 AL_STATE = STATE_BOOT AL_STATUS_CODE = 0 START_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	BOOT
6	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and BOOT_SUPPORTED and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x15 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
7	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT and not BOOT_SUPPORTED => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x13 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
8	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = STATE_SAFEOP OR AL_CONTROL_STATE = STATE_OP) => ID_FLAG = 0 AL_STATE = STATE_INIT AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
9	initialisation (Initialization, INIT)	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 L_STATE = STATE_INIT AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
10	initialisation (Initialization, INIT)	**SM_Chg** => ignore	initialisation (Initialization, INIT)

N	État actuel	Événement /Condition => Action	Prochain état
11.1	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER ID_INFO AL Control.rsp(-+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
11.2	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_ERROR_FLAG = 1 and .ACK_FLAG = 0 and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
12	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
13	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
14.1	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and ((not DC_ACTIVATED and not dcRequired) or (DC_ACTIVATED and not dcNotAccepted and not dcSupported)) => dcRunning = FALSE AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_SAFEOP START_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
14.2	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and DC_ACTIVATED and not dcNotAccepted and dcSupported => dcRunning = TRUE AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_SAFEOP START_INPUT_HANDLER ID_INFO	SAFEOP

N	État actuel	Événement /Condition => Action	Prochain état
		AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
14.3	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and SM_SETTINGS_2_TO_n_MATCH and ((DC_ACTIVATED and dcNotAccepted) or (not DC_ACTIVATED and dcRequired)) => ID_FLAG = 0 AL_STATUS_CODE = 0x30 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
17	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_SAFEOP and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATE = STATE_PREOP AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
18	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = STATE_BOOT or AL_CONTROL_STATE = STATE_OP) => ID_FLAG = 0 AL_STATE = STATE_PREOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
19	PREOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 AL_STATE = STATE_PREOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
20.1	PREOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH => AckSM_Chg	PREOP
20.2	PREOP	**SM_Chg** / AL_ERROR_FLAG = 1 => ignore	PREOP
21	PREOP	**SM_Chg** /AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_MBX_HANDLER AckSM_Chg	initialisation (Initialization, INIT)

N	État actuel	Événement /Condition => Action	Prochain état
		AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	
22.1	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and .ACK_FLAG = 0) and AL_CONTROL = STATE_INIT => AL_ERROR_FLAG = 0 (AL_STATUS_CODE = 0) AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
22.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and .ACK_FLAG = 0) and AL_CONTROL_STATE <> STATE_INIT => AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
23	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
24	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
25.1	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 1 and (not localErrorFlag and .ACK_FLAG = 1)) and AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ENABLE_SM_CHANNEL ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
25.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /((AL_ERROR_FLAG = 0 or (localErrorFlag and .ACK_FLAG = 1)) and AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and readyForOP and	OP

N	État actuel	Événement /Condition => Action	Prochain état
		(not dcRunning or pllRunning) and not localErrorFlag => AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_OP ENABLE_SM_CHANNEL START_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	
26.4	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and dcRunningand not pllRunning and not localErrorFlag => ENABLE_SM_CHANNEL waitForPllRunning = TRUE Start SafeOpT2OpTimeoutTimer ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.5	SAFEOP	**PLL Running Event** /waitForPllRunning => AL_STATE = STATE_OP AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 START_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	OP
26.6	SAFEOP	**PLL Running Event** /not waitForPllRunning => ignore	SAFEOP
26.7	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /waitForPllRunning and not dcEventReceived => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x2D AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.8	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /waitForPllRunning and dcEventReceived => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x32 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
26.9	SAFEOP	**SafeOp2OpTimeoutTimer Expired** /not waitForPllRunning => ignore	SAFEOP
27.1	SAFEOP	AL_Control.ind (AL Control State, Ack Flag, ID Request) /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OP and not readyForOP and (not dcRunning or pllRunning) and	SAFEOP

N	État actuel	Événement /Condition => Action	Prochain état
		not localErrorFlag => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x1B AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	
27.2	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_OPand localErrorFlag => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = localErrorCode AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
29	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and AL_CONTROL_STATE = STATE_BOOT => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
30	SAFEOP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
31.1	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and SM_SETTINGS_2_TO_n_MATCH => AckSM_Chg	SAFEOP
31.2	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 1 => ignore	SAFEOP
32	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	PREOP
33	SAFEOP	**SM_Chg** / AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH =>	initialisation (Initialization, INIT)

N	État actuel	Événement /Condition => Action	Prochain état
		ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_INPUT_HANDLER STOP_MBX_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	
34.1	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 DISABLE_SM_CHANNEL AL_STATE = STATE_SAFEOP AL_ERROR_FLAG = TRUE localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	SAFEOP
34.2	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 0 and localErrorFlag => ENABLE_SM_CHANNEL localErrorFlag = FALSE localErrorCode = 0	SAFEOP
34.3	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_PREOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_INPUT_HANDLER AL_STATE = STATE_PREOP AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	PREOP
34.4	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_MBX_HANDLER STOP_INPUT_HANDLER AL_STATE = STATE_INIT AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
34.5	SAFEOP	**AL_State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_INIT or STATE_PREOP) and AL_ERROR_FLAG = 0) or AL_STATE = STATE_OP or STATE_BOOT or unknown => ignore	SAFEOP
35.1	SAFEOP	**Output Event** /wdEnabled => RESTART_WD	SAFEOP
37	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATUS_CODE =0 AL_STATE = STATE_INIT STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	initialisation (Initialization, INIT)

N	État actuel	Événement /Condition => Action	Prochain état
38	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_PREOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_PREOP STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	PREOP
39	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_SAFEOP => AL_ERROR_FLAG = 0 AL_STATUS_CODE = 0 AL_STATE = STATE_SAFEOP STOP_LCOAL_OUTPUT_HANDLER ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
40	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_OP => ID_INFO AL Control.rsp(+) (AL State, AL Status Code, Error Flag, ID Flag)	OP
42	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = STATE_BOOT => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 DISABLE_SM_CHANNEL STOP_LOCAL_OUTPUT_HANDLER AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
43	OP	**AL_Control.ind (AL Control State, Ack Flag, ID Request)** /AL_CONTROL_STATE = unknownState => ID_FLAG = 0 AL_STATE = STATE_SAFEOP AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 DISABLE_SM_CHANNEL STOP_LCOAL_OUTPUT_HANDLER AL Control.rsp(-) (AL State, AL Status Code, Error Flag, ID Flag)	SAFEOP
44	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and SM_SETTINGS_2_TO_n_MATCH => AckSM_Chg	OP
45	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and SM_SETTINGS_0_AND_1_MATCH and not SM_SETTINGS_2_TO_n_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x17 AL_ERROR_FLAG = 1 AL_STATE = STATE_PREOP STOP_LCOAL_OUTPUT_HANDLER STOP_INPUT_HANDLER	PREOP

N	État actuel	Événement /Condition => Action	Prochain état
		AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	
46	OP	**SM_Chg** / AL_ERROR_FLAG = 0 and not SM_SETTINGS_0_AND_1_MATCH => ID_FLAG = 0 AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_LOCAL_OUTPUT_HANDLER STOP_INPUT_HANDLER STOP_MBX_HANDLER AckSM_Chg AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
47.1	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_SAFEOP and AL_ERROR_FLAG = 1 => STOP_LOCAL_OUTPUT_HANDLER DISABLE_SM_CHANNEL AL_STATE = STATE_SAFEOP localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	SAFEOP
47.2	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_PREOP and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER AL_STATE = STATE_PREOP localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	PREOP
47.3	OP	**AL State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => ID_FLAG = 0 STOP_MBX_HANDLER STOP_INPUT_HANDLER STOP_LOCAL_OUTPUT_HANDLER AL_STATE = STATE_INIT localErrorFlag = TRUE localErrorCode = AL_STATUS_CODE AL Status Changed.req (AL state, AL status code, Error Flag, ID Flag)	initialisation (Initialization, INIT)
47.4	OP	**AL State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_SAFEOP or STATE_PREOP or STATE_INIT) and AL_ERROR_FLAG = 0) or AL_CONTROL_STATE = STATE_BOOT or unknown => ignore	OP
48	OP	**Output event** => RESTART_WD Update Outputs	OP
49	OP	**WD expired** => ID_FLAG = 0 AL_STATUS_CODE = 0x1B AL_ERROR_FLAG = 1 AL_STATE = STATE_SAFEOP STOP_LOCAL_OUTPUT_HANDLER	SAFEOP

N	État actuel	Événement /Condition => Action	Prochain état
		DISABLE_SM_CHANNEL AL Status Changed.req (AL state, AL staus code, Error Flag, ID Flag)	
51	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /AL_CONTROL_STATE = STATE_INIT => AL_ERROR_FLAG = 0 AL_STATE = STATE_INIT STOP_MBX_HANDLER AL Control.rsp(+) (AL State, AL Status Code)	initialisation (Initialization, INIT)
52	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /AL_CONTROL_STATE = STATE_BOOT => AL_ERROR_FLAG = 0 AL Control.rsp(+) (AL State, AL Status Code)	BOOT
53.1	BOOT	**AL_Control.ind (ALControlState, Ack Flag)** /(AL_CONTROL_STATE = STATE_PREOP or AL_CONTROL_STATE = STATE_SAFEOP or AL_CONTROL_STATE = STATE_OP) => AL_STATUS_CODE = 0x11 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code)	initialisation (Initialization, INIT)
53.2	BOOT	**AL_Control.ind (ALControlState, Ack Flag)** /(AL_CONTROL_STATE = STATE_PREOP or AL_CONTROL_STATE = STATE_SAFEOP or AL_CONTROL_STATE = STATE_OP) => ignore	BOOT
54.1	BOOT	**AL_Control.ind (AL Control State, Ack Flag)** /(AL_CONTROL_STATE = unknownState) => AL_STATUS_CODE = 0x12 AL_ERROR_FLAG = 1 AL Control.rsp(-) (AL State, AL Status Code)	initialisation (Initialization, INIT)
54.2	BOOT	**AL_Control.ind (ALControlState, Ack Flag)** /(AL_ERROR_FLAG = 0 or .ACK_FLAG = 1) and (AL_CONTROL_STATE = unknownState) => ignore	BOOT
55	BOOT	**SM_Chg** /SM_SETTINGS_0_AND_1_MATCH => ignore	BOOT
56.1	BOOT	**SM_Chg** /not SM_SETTINGS_0_AND_1_MATCH => AL_STATUS_CODE = 0x16 AL_ERROR_FLAG = 1 AL_STATE = STATE_INIT STOP_MBX_HANDLER AL Status Changed.req (AL state, AL staus code)	initialisation (Initialization, INIT)
56.2	BOOT	**SM_Chg** /not SM_SETTINGS_0_AND_1_MATCH => ignore	BOOT

N	État actuel	Événement /Condition => Action	Prochain état
57	BOOT	**AL_State Change.req (AL Status, AL Status Code)** /AL_STATE = STATE_INIT and AL_ERROR_FLAG = 1 => STOP_MBX_HANDLER AL_STATE = STATE_INIT AL_ERROR_FLAG = TRUE AL Status Changed.req (AL state, AL status code)	initialisation (Initialization, INIT)
58	BOOT	**AL_State Change.req (AL Status, AL Status Code)** /((AL_STATE = STATE_INIT and AL_ERROR_FLAG = 0) or AL_STATE = STATE_PREOP or STATE_SAFEOP or STATE_OP or STATE_BOOT or unknown => ignore	BOOT

6.4.2 Mailbox handler state machine

6.4.2.1 Description

Le gestionnaire de boîte aux lettres est responsable de la coordination du maître et de l'esclave, relativement à l'exploitation de la boîte aux lettres. Les écritures dans la boîte aux lettres sont transmises aux machines concernées et les réponses seront placées dans la boîte aux lettres de lecture.

Etant donné l'absence de service spécial orienté états, il n'y a pas de table d'états.

Les tâches du gestionnaire de boîte aux lettres sont les suivantes.

– Mapper les services de boîte aux lettres d'écriture avec le gestionnaire de protocole
– Mettre en file d'attente les demandes de service à destination de la boîte aux lettres de lecture
– Confirmer l'émission de la boîte aux lettres de lecture

Le gestionnaire de protocole peut être

– Diagramme d'états CoE
– Diagramme d'états EoE
– Diagramme d'états FoE
– un diagramme d'états spécifique à un profil.

6.4.2.2 Primitives échangées entre la couche DLL, le gestionnaire de boîte aux lettres et le gestionnaire de protocole

Le Tableau 103 montre les primitives de service émises par le gestionnaire de boîte aux lettres et reçues par la couche DLL, avec les paramètres associés.

Tableau 103 – Primitives adressées par le gestionnaire de boîte aux lettres à la couche DL

Nom de la primitive	Paramètres associés	Fonctions
Mailbox Read Upd.req	Length Address Channel Priority Type Cnt Service Data	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-3-12

Le Tableau 104 montre les primitives de service émises par la couche DLL et reçues par le gestionnaire de boîte aux lettres, avec leurs paramètres associés.

Tableau 104 – Primitives adressées par la couche DL au gestionnaire de boîte aux lettres

Nom de la primitive	Paramètres associés	Fonctions
Mailbox Write.ind	Length Address Channel Priority Type Cnt Service Data	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-3-12
Mailbox Read Upd.cnf	success	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-3-12

Le Tableau 105 montre les primitives de service émises par le gestionnaire de protocole et reçues par le gestionnaire de boîte aux lettres, avec les paramètres associés. Le préfixe TYPE provient du paramètre de type de la primitive de service de couche DLL correspondante.

Tableau 105 – Primitives adressées par le gestionnaire de protocole au gestionnaire de boîte aux lettres

Nom de la primitive	Paramètres associés	Fonctions
TYPE Mailbox Read Upd.req	Length Address Channel Priority Service Data	voir la définition du service de mise à jour de la boîte aux lettres de lecture pour les bus de terrain de Type 12 dans la CEI 61158-3-12

Le Tableau 106 montre les primitives de service émises par le gestionnaire de boîte aux lettres et reçues par le gestionnaire de protocole, avec les paramètres associés. Le préfixe TYPE provient du paramètre de type de la primitive de service de couche DLL correspondante.

Tableau 106 – Primitives adressées par le gestionnaire de boîte aux lettres au gestionnaire de protocole

Nom de la primitive	Paramètres associés	Fonctions
TYPE Mailbox Write.ind	Length Address Channel Priority Service Data	voir la définition des services pour les bus de terrain de Type 12 dans la CEI 61158-3-12
TYPE Mailbox Read Upd.cnf	success	voir la définition du service de mise à jour de la boîte aux lettres de lecture pour les bus de terrain de Type 12 dans la CEI 61158-3-12

6.4.3 Diagramme d'états CoE

6.4.3.1 Description

Le diagramme d'états CoE est chargé du traitement des services CoE.

Sa tâche principale consiste à exécuter les demandes de service et à fournir la réponse

– soit sous la forme d'une trame unique,
– soit sous la forme d'une séquence de trames (services d'information),
– soit sous la forme d'une trame unique avec une indication de type "à suivre",
– soit sous la forme d'un abandon, pour l'arrêt du service à la suite d'une erreur.

6.4.3.2 Définitions de primitives

6.4.3.2.1 Primitives échangées entre le gestionnaire de boîte aux lettres et le diagramme d'états CoESM

Les primitives échangées entre le diagramme d'états CoESM et le gestionnaire de boîte aux lettres sont décrites en 6.4.2.2.

6.4.3.2.2 Primitives échangées entre la couche Application et le diagramme d'états CoESM

Le Tableau 107 montre les primitives de service émises par la couche AL et reçues par le diagramme d'états CoESM, avec les paramètres associés. Les services d'information sur l'objet SDO représentent un groupe de services (obtention de la liste de dictionnaires OD, obtention de la description de l'objet, obtention de la description de l'entrée) différenciés par le paramètre opcode.

Tableau 107 – Primitives adressées par la couche Application au diagramme d'états CoESM

Nom de la primitive	Paramètres associés	Fonctions
SDO Download Expedited.rsp	Success Address Index Subindex	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Download Normal.rsp	Success Address Index Subindex	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Download SDO Segment.rsp	Success Address Toggle	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Upload Expedited.rsp	Success Address Index Subindex Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Upload Normal.rsp	Success Address Index Subindex Complete Size Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Upload SDO Segment.rsp	Success Address Toggle More Follows Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Abort SDO Transfer.req	Address Index Subindex Reason	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Info Service.rsp	Address Opcode Incomplete	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

Nom de la primitive	Paramètres associés	Fonctions
	Fragments Left Size Data	
SDO Info Seg Service.req	Address Opcode Incomplete Fragments Left Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Emergency Service.req	Address Error Code Error Register Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

Le Tableau 108 montre les primitives de service émises par le diagramme d'états CoESM et reçues par la couche AL, avec les paramètres associés.

Tableau 108 – Primitives adressées par le diagramme d'états CoESM à la couche Application

Nom de la primitive	Paramètres associés	Fonctions
SDO Download Expedited.ind	Address Index Subindex Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Download Normal.ind	Address Index Subindex Complete Size Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Download SDO Segment.ind	Address Toggle More Follows Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Upload Expedited.ind	Address Index Subindex	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Upload Normal.ind	Address Index Subindex	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Upload SDO Segment.ind	Address Toggle	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Abort SDO Transfer.req	Address Index Subindex Reason	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
SDO Info Service.ind	Address Opcode Incomplete Fragments Left Size Data	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

6.4.3.2.3 Paramètres des primitives

Les paramètres utilisés avec les primitives échangées entre le diagramme d'états CoESM et la couche AL sont décrits dans la CEI 61158-5-12.

6.4.3.3 Table d'états du diagramme d'états CoESM

Le Tableau 109 contient la description complète du diagramme d'états CoESM.

Tableau 109 – Table d'états du diagramme d'états CoESM

N	État actuel	Événement /Condition => Action	Prochain état
1	OFF	**START MAILBOX** => Seg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** =>	OFF
5	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service != 2 && Service_Data.Service != 8 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service == 2 && Service_Data.Command_Specifier > 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 1 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
8	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 1 => Size = 4 - Service_Data.Data_Set_Size Index = Service_Data.Index SubIndex = Service_Data.SubIndex Data = Service_Data.Data SDO Download Expedited.ind (Address, Index, Subindex , Size, Data)	DWLE
9	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
10	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 1 && Service_Data.Transfer_Type == 0 => Seg = (Service Data.Complete_Size - (Length - 10)) Toggle = FALSE Size = Length - 10 Complete Size = Service Data.Complete_Size Index = Service_Data.Index SubIndex = Service_Data.SubIndex	DWLN

N	État actuel	Événement /Condition => Action	Prochain état
		Data = Service_Data.Data SDO Download Normal.ind (Address, Index, Subindex, Complete Size, Size, Data)	
11	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length < 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
12	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >=10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
13	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
14	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Index = Service_Data.Index SubIndex = Service_Data.SubIndex SDO Upload Expedited.ind (Address, Index, Subindex)	UPLE
15	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
16	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 2 => Index = Service_Data.Index SubIndex = Service_Data.SubIndex SDO Upload Normal.ind (Address, Index, Subindex)	UPLN
17	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 3 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
18	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 3 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 =>	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
19	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 8 && Service_Data.Service == 8 && Service_Data.Opcode == 1,3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
20	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 8 && Service_Data.Service == 8 && Service_Data.Opcode == 1,3 => Size = Length -6 Opcode = Service_Data.Opcode Incomplete = Service_Data.Incomplete FragmentsLeft = Service_Data.FragmentsLeft SDO Info Service.ind (Address, Opcode, Incomplete, Fragments Left, Size, Data)	INF
21	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 10 && Service_Data.Service == 8 && Service_Data.Opcode == 5 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
22	IDLE	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 10&& Service_Data.Service == 8 && Service_Data.Opcode == 5 => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ SDO Info Service.ind (Address, Opcode, Incomplete, Fragments Left, Size, Data)	INF
23	IDLE	**CoE Read Upd.cnf (success)** => ignore	IDLE
24	IDLE	**Emergency Service.req (Address, Error Code, Error Register, Size, Data)** => Length = 10 Service_Data.Service = 1 Service_Data.Error Code = Error Code Service_Data.Error Register = Error Register Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
25	IDLE	**other application service primitives** => ignore	IDLE
26	DWLE	**START MAILBOX** => ignore	DWLE
27	DWLE	**STOP MAILBOX** => CANCEL SERVICE	OFF
28	DWLE	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
29	DWLE	**SDO Download Expedited.rsp (Success, Address, Index, Subindex)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 3 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
30	DWLE	**other application service primitives** => ignore	DWLE
31	DWLN	**START MAILBOX** => ignore	DWLN
32	DWLN	**STOP MAILBOX** => CANCEL SERVICE	OFF
33	DWLN	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
34	DWLN	**SDO Download Normal.rsp (Success, Address, Index, Subindex)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 3 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
35	DWLN	**other application service primitives** => ignore	DWLN
36	DWLS	**START MAILBOX** => ignore	DWLS
37	DWLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
38	DWLS	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
39	DWLS	**Download SDO Segment.rsp (Success, Address, Toggle)** => Length = 6 Service_Data.Service = 3 Service_Data.Command_Specifier = 1 Service_Data.Toggle, Toggle = !Toggle CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
40	DWLS	**other application service primitives** => ignore	DWLS
41	UPLE	**START MAILBOX** => ignore	UPLE
42	UPLE	**STOP MAILBOX** => CANCEL SERVICE	OFF
43	UPLE	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
44	UPLE	**SDO Upload Expedited.rsp (Success, Address, Index, Subindex, Size, Data)** => Length = 10 Service_Data.Service = 3 Service_Data.Command_Specifier = 2 Service_Data.Transfer Type = 1 Service_Data.Data Set Size = 4 - Size Service_Data.Index = Index Service_Data.Subindex = Subindex Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
45	UPLE	**other application service primitives** => ignore	UPLE
46	UPLN	**START MAILBOX** => ignore	UPLN
47	UPLN	**STOP MAILBOX** => CANCEL SERVICE	OFF
48	UPLN	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
49	UPLN	**SDO Upload Normal.rsp (Success, Address, Index, Subindex, Complete Size, Size, Data)** => Seg = Size - Complete Size Toggle = FALSE Length = 10 + Size Service_Data.Service = 3 Service_Data.Command_Specifier = 2 Service_Data.Transfer Type = 0 Service_Data.Index = Index Service_Data.Subindex = Subindex Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
50	UPLN	**other application service primitives** => ignore	UPLN
51	UPLS	**START MAILBOX** =>	UPLS

N	État actuel	Événement /Condition => Action	Prochain état
		ignore	
52	UPLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
53	UPLS	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
54	UPLS	**Upload SDO Segment.rsp (Success, Address, Toggle, More Follows, Size, Data)** => Seg = (Seg -Size) if (Size > 7) then Service_Data.SegDataSize = 0 else Service_Data.SegDataSize =7-Size Length = Size + 3 + Service_Data.SegDataSize Service_Data.Service = 3 Service_Data.Command_Specifier = 0 Service_Data.Toggle = Toggle Service_Data.Data = Data Service_Data.More Follows = More Follows CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
55	UPLS	**other application service primitives** => ignore	UPLS
56	INF	**START MAILBOX** => ignore	INF
57	INF	**STOP MAILBOX** => CANCEL SERVICE	OFF
58	INF	**Abort SDO Transfer.req (Address, Index, Subindex, Reason)** => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = Reason Seg = 0 CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
59	INF	**SDO Info Service.rsp (Address, Opcode, Incomplete, Fragments Left, Size, Data)** => if FragmentsLeft >0) then Seg = 0x80000000 else Seg = 0 Length = Size + 6 Service_Data.Service = 8 Service_Data.Opcode = Opcode Service_Data. Incomplete = Incomplete Service_Data.Data = Data Service_Data. Fragments Left = Fragments Left CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
60	INF	**SDO Info Seg Service.req (Address, Opcode, Incomplete, Fragments Left, Size, Data)** => if FragmentsLeft >0) then Seg = 0x80000000 else Seg = 0 Length = Size + 6 Service_Data.Service = 8 Service_Data.Opcode = Opcode Service_Data. Incomplete = Incomplete Service_Data.Data = Data	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		Service_Data. Fragments Left = Fragments Left CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
61	WUPD	**START MAILBOX** =>	WUPD
62	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
63	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0 =>	IDLE
64	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0x80000000 && (Fragments Left == 0) => Seg = 0	IDLE
65	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg == 0x80000000 && (Fragments Left > 0) =>	INF
66	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg >0 && Seg < 0x80000000 =>	WSEG
67	WUPD	**CoE Mailbox Read Upd.cnf (success)** /Seg < 0 && Seg > 0x80000000 =>	WSEG
69	ERR	**START MAILBOX** =>	ERR
70	ERR	**STOP MAILBOX** => RESET MAILBOX	OFF
71	ERR	**ERR Mailbox Read Upd.cnf (success)** =>	IDLE
73	WSEG	**START MAILBOX** =>	WSEG
74	WSEG	**STOP MAILBOX** =>	OFF
75	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service != 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
76	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.Service == 2 && Service_Data.Command_Specifier != 0,3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
77	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length < 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE Seg = 0 ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
78	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 &&	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		Service_Data.Command_Specifier == 0 && Seg < (Length - 3 - Service_Data.SegDataSize) => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
79	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 && Seg >= (Length - 3 - Service_Data.SegDataSize) && (Service_Data.More Follows != (0 < (Seg -(Length - 3 - Service_Data.SegDataSize))) || Toggle = Service_Data.Toggle) => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
80	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length >= 10 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 0 && Seg >= (Length - 3 - Service_Data.SegDataSize) && Service_Data.More Follows == (0 < (Seg -(Length - 3 - Service_Data.SegDataSize))) && !Toggle = Service_Data.Toggle => Seg = (Seg -(Length - 3 - Service_Data.SegDataSize)) Size = Length - 3 - Service_Data.SegDataSize Toggle = Service_Data.Toggle More Follows = Service_Data.More Follows Data = Service_Data.Data Download SDO Segment.ind (Address, Toggle, More Follows, Size, Data)	DWLS
81	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length != 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
82	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 && Toggle = Service_Data.Toggle => Length = 10 Service_Data.Service = 2 Service_Data.Command_Specifier = 4 Service_Data.Abort_Code = ABT_SEQ CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
83	WSEG	**CoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 6 && Service_Data.Service == 2 && Service_Data.Command_Specifier == 3 && !Toggle = Service_Data.Toggle => Toggle = Service_Data.Toggle Upload SDO Segment.ind (Address, Toggle)	UPLS
84	WSEG	**Emergency Service.req (Address, Error Code, Error Register, Size, Data)** => Length = 10 Service_Data.Service = 1	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		Service_Data.Error Code = Error Code Service_Data.Error Register = Error Register Service_Data.Data = Data CoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
85	WSEG	**other application service primitives** => ignore	IDLE

6.4.4 Diagramme d'états EoE

6.4.4.1 Description

Le diagramme d'états EoE est chargé du transport des trames Ethernet standard dans les éléments de Type 12. Il reçoit et transmet des unités PDU de Type 12, soit avec la trame Ethernet complète, soit avec un fragment de la trame Ethernet, ce fragment pouvant être combiné avec une trame complète. Après le réassemblage (qui n'est pas du ressort de cette machine), la trame Ethernet résultante peut être traitée comme si elle avait été émise via Ethernet sans services de Type 12.

Les règles d'entrée et de sortie d'un port Ethernet pouvant être associées aux services EoE sont spécifiées dans la norme IEEE 802.1D.

Définitions générales des horodatages:

- 32 bits, résolution de 1 ns
- L'heure système des horloges DC peut être utilisée
- Début de l'adresse DA correspondant au déclenchement de l'horodatage

Horodatage ajouté (TA = 1):

- De l'esclave au maître: l'horodatage contient l'heure exacte de réception.
- Du maître à l'esclave: l'horodatage contient l'heure d'envoi souhaitée.
- Il convient que l'esclave ajoute toujours un horodatage s'il possède la fonction correspondante.
- L'horodatage ajoute 32 bits aux données de la trame.
- Le bit TA n'est autorisé que dans le dernier fragment (LF = 1).
- Si l'horodatage ne tient pas dans le dernier fragment, ajouter un fragment.
  - Remplir le "dernier fragment" avec des parties de l'horodatage (LF = 0, TA = 0) et envoyer un tout dernier fragment avec le reste de l'horodatage (LF = 1, TA = 1).

Horodatage demandé (TR = 1):

- Réponse avec l'heure exacte d'envoi et le numéro de trame demandé.
- Il convient que la réponse soit envoyée le plus tôt possible.

6.4.4.2 Définitions de primitives

6.4.4.2.1 Primitives échangées entre le gestionnaire de boîte aux lettres et le diagramme d'états EoESM

Les primitives échangées entre le diagramme d'états EoESM et le gestionnaire de boîte aux lettres sont décrites en 6.4.2.2.

6.4.4.2.2 Primitives échangées entre la couche AL et le diagramme d'états EoESM

Le Tableau 110 montre les primitives de service émises par la couche Application et reçues par le diagramme d'états EoESM, avec les paramètres associés.

Tableau 110 – Primitives adressées par la couche Appliication au diagramme d'états EoESM

Nom de la primitive	Paramètres associés	Fonctions
Initiate_EoE.req	Address, Port Time Appended Time Requested Frame Number Complete Size Last Fragment Size Data Time Stamp	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Initiate_EoE.rsp	Address, Frame Number Time Stamp	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
EoE Fragment.req	Address, Port Time Appended Frame Number Offset Last Fragment Size Data Time Stamp	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Set IP Parameter.rsp	Address, Reason	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Set Address Filter.rsp	Address, Reason	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

Le Tableau 111 montre les primitives de service émises par le diagramme d'états EoESM et reçues par la couche AL, avec les paramètres associés.

Tableau 111 – Primitives adressées par le diagramme d'états EoESM à la couche Application

Nom de la primitive	Paramètres associés	Fonctions
Initiate_EoE.ind	Address, Port Time Appended Time Requested Frame Number Complete Size Last Fragment Size Data Time Stamp	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
EoE Fragment.ind	Address, Port Time Appended Frame Number Offset Last Fragment Size Data Time Stamp	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Set IP Parameter.ind	Address,	voir la définition des services CoE

Nom de la primitive	Paramètres associés	Fonctions
	MAC Address IP Address Subnet Mask Default Gateway DNS Server DNS Name	pour les bus de terrain de Type 12 dans la CEI 61158-5-12
Set Address Filter.ind	Address, Broadcast Fowarding MAC Address Filters MAC Filter Masks	voir la définition des services CoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

6.4.4.2.3 Paramètres des primitives

Les paramètres utilisés avec les primitives échangées entre le diagramme d'états EoESM et la couche AL sont décrits dans la CEI 61158-5-12.

6.4.4.3 Table d'états du diagramme d'états EoESM

Le Tableau 112 contient la description complète du diagramme d'états EoESM. Les services Set IP Parameter et Set Address Filter sont traités dans tous les états, de telle sorte que le paramètre demandé soit changé si possible.

Tableau 112 – Table d'états du diagramme d'états EoESM

N	État actuel	Événement /Condition => Action	Prochain état
1	OFF	**START MAILBOX** => SSeg, RSeg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** => Terminate Segmented Services	OFF
5	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length < 36 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && (Service_Data.Fragment > 0 || Service_Data.CompleteSize < 2) => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 1 => TimeAppended = Service_Data.TimeAppended TimeRequested= Service_Data.TimeRequested Size = Length-4-TimeAppended*4 if (TimeAppend) Timestamp = Service_Data[Length -4..Length-1] Frame Number = Service_Data.Frame Number Complete Size = Service_Data.CompleteSize Fragment Number = Service_Data.Fragment LastFragment = 1 Data = Service_Data.EoE Data	IDLE

N	État actuel	Événement /Condition => Action	Prochain état
		Initiate_EoE.ind (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)	
8	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 0 && ((Length -4) mod 32) == 0 => TimeAppended = Service_Data.TimeAppended TimeRequested= Service_Data.TimeRequested Size,SSeg = Length-4 Frame Number = Service_Data.Frame Number Fragment Number = Service_Data.Fragment Complete Size = Service_Data.CompleteSize LastFragment = 0 Data = Service_Data.EoE Data Initiate_EoE.req (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)	IDLE
9	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg == 0 && Length >= 36 && Service_Data.Fragment == 0 && Service_Data.CompleteSize >= 2 && Service_Data.LastFragment == 0 && ((Length -4) mod 32) != 0 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
10	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && (Service_Data.Fragment == 0 || Service_Data.Offset*32 != SSeg) => SSeg = 0 Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
11	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 1 => SSeg = 0 TimeAppended = Service_Data.TimeAppended Size = Length-4-TimeAppended*4 if (TimeAppend) Timestamp = Service_Data[Length -4..Length-1] Frame Number = Service_Data.Frame Number Offset = Service_Data.Offset Fragment Number = Service_Data.Fragment LastFragment = 1 Data = Service_Data.EoE Data EoE_Fragment.ind (Address, Port, Time Appended, Frame Number, Offset, Last Fragment, Size, Data, Timestamp)	IDLE
12	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** / Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 0 && ((Length -4) mod 32) == 0 => SSeg = SSEG + Length-4 Frame Number = Service_Data.Frame Number Offset = Service_Data.Offset Fragment Number = Service_Data.Fragment LastFragment = 0 Data = Service_Data.EoE Data EoE_Fragment.ind (Address, Port, Time Appended, Frame Number, Offset, Last Fragment, Size, Data, Timestamp)	IDLE
13	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)**	ERR

N	État actuel	Événement /Condition => Action	Prochain état
		/ Service_Data.FrameType == 0 && SSeg != 0 && Service_Data.Fragment != 0 && Service_Data.Offset*32 == SSeg && Service_Data.LastFragment == 0 && ((Length -2) mod 32) != 0 => SSeg = 0 Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
14	IDLE	**Initiate_EoE.req (Address, Port, Time Appended, Time Requested, Frame Number, Complete Size, Last Fragment, Size, Data,Timestamp)** => Service_Data.TimeAppended = TimeAppended Service_Data.TimeRequested = TimeRequested Length = Size+4+TimeAppended*4 if (TimeAppend) Service_Data[Length -4..Length-1] = Timestamp Service_Data.Frame Number = Frame Number Service_Data.Fragment Number = Fragment Service_Data.Complete Size = CompleteSize Service_Data.LastFragment = LastFragment Service_Data.Port = Port Service_Data.EoE Data = Data EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
15	IDLE	**EoE_Fragment.req(Address, Port, TimeAppended, Frame Number, Offset, Last Fragment, Size, Data,Timestamp)** => Service_Data.TimeAppended = TimeAppended Length = Size+4+TimeAppended*4 if (TimeAppend) Service_Data[Length -4..Length-1] = Timestamp Service_Data.Frame Number = Frame Number Service_Data.Fragment Number = Fragment Service_Data.Complete Size = CompleteSize Service_Data.LastFragment = LastFragment Service_Data.Port = Port Service_Data.EoE Data = Data EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
16	IDLE	**EOE Read Upd.cnf (success)** => ignore	IDLE
17	IDLE	**other application service primitives** => ignore	IDLE
18	IDLE	**Initiate_EoE.rsp(Address, Frame Number, Timestamp)** => Service_Data.FrameType = 1 Service_Data.TimeAppended = 1 Service_Data.TimeStamp = Timestamp EOE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
19	WUPD	**START MAILBOX** =>	WUPD
20	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
21	WUPD	**EOE Read Upd.cnf (success)** =>	WUPD
22	WUPD	**EOE Read.ind** =>	IDLE
23	WUPD	**Timeout** => RESET MAILBOX	OFF
24	ERR	**START MAILBOX**=>	ERR
25	ERR	**STOP MAILBOX**	OFF

N	État actuel	Événement /Condition => Action	Prochain état
		=> RESET MAILBOX	
26	ERR	**ERR Read Upd.cnf (success)** =>	ERR
27	ERR	**ERR Read.ind** =>	IDLE
28	ERR	**Timeout** => RESET MAILBOX	OFF
29	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 8 (fits request) && Service_Data.FrameType == 2 => Extract IP Parameter from Service Data. IP Parameter Set IP Parameter.ind(Address, Mac Address, IP Adress, Subnet Mask, Default Gateway, DNS Server, DNS Name)	IPPAR
30	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length not (fits request) && Service_Data.FrameType == 2 => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = 2 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
31	IPPAR	**START MAILBOX** => ignore	IPPAR
32	IPPAR	**STOP MAILBOX** => CANCEL SERVICE	OFF
33	IPPAR	**Set IP Parameter.rsp(+) (Adress)** => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = 0 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
34	IPPAR	**Set IP Parameter.rsp(-) (Adress, Reason)** => Length = 4 Service_Data.FrameType = 3 Service_Data.Result = Reason Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
35	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 (fits request) && Service_Data.FrameType == 4 => Extract Address Fileter from Service Data SetAddress Filter.ind(Address, Broadcast forwarding, MAC Address Filter 1..16, MAC Filter Mask 1..4)	MFLT
36	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length not (fits request) && Service_Data.FrameType == 4 => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = 2 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
37	MFLT	**START MAILBOX** => ignore	MFLT

N	État actuel	Événement /Condition => Action	Prochain état
38	MFLT	**STOP MAILBOX** => CANCEL SERVICE	OFF
39	MFLT	**SetAddress Filter.rsp(+) (Adress)** => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = 0 Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
40	MFLT	**SetAddress Filter.rsp(-) (Adress, Reason)** => Length = 4 Service_Data.FrameType = 5 Service_Data.Result = Reason Service_Data.Port = 0 EoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
41	IDLE	**EoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.FrameType != 0,2,4 => ignore	IDLE

6.4.5 Diagramme d'états FoE

6.4.5.1 Description

Le diagramme d'états FoE est responsable du transport des fichiers dans les éléments de Type 12. Il reçoit et transmet des unités PDU de Type 12 avec une commande de transfert ("écrire" signifie charger le fichier dans l'esclave, "lire" signifie charger le fichier dans le maître).

6.4.5.2 Définitions de primitives

6.4.5.2.1 Primitives échangées entre le gestionnaire de boîte aux lettres et le diagramme d'états FoESM

Les primitives échangées entre le diagramme d'états FoESM et le gestionnaire de boîte aux lettres sont décrites en 6.4.2.2.

6.4.5.2.2 Primitives échangées entre la couche AL et le diagramme d'états FoESM

Le Tableau 113 montre les primitives de service émises par la couche AL et reçues par le diagramme d'états FoESM, avec les paramètres associés.

Tableau 113 – Primitives adressées par la couche Application au diagramme d'états FoESM

Nom de la primitive	Paramètres associés	Fonctions
FoE Data.req	Address, Packet Number Size Data	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Ack.req	Address, Packet Number	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Error.req	Address, Error Code, ErrorText	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

Le Tableau 114 montre les primitives de service émises par le diagramme d'états FoESM et reçues par la couche AL, avec les paramètres associés.

Tableau 114 – Primitives adressées par le diagramme d'états FoESM à la couche Application

Nom de la primitive	Paramètres associés	Fonctions
FoE Write.ind	Address, Password File Name	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Read.ind	Address, Password File Name	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Data.ind	Address, Packet Number Size Data	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Ack.ind	Address, Packet Number	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12
FoE Error.ind	Address, Error Code, ErrorText	voir la définition des services FoE pour les bus de terrain de Type 12 dans la CEI 61158-5-12

6.4.5.2.3 Paramètres des primitives

Les paramètres utilisés avec les primitives échangées entre le diagramme d'états FoESM et la couche AL sont décrits dans la CEI 61158-5-12.

6.4.5.3 Table d'états du diagramme d'états FoESM

Le Tableau 115 contient la description du complète du diagramme d'états FoESM.

Tableau 115 – Table d'états du diagramme d'états FoESM

N	État actuel	Événement /Condition => Action	Prochain état
1	OFF	**START MAILBOX** => Seg = 0	IDLE
2	OFF	**STOP MAILBOX** =>	OFF
3	IDLE	**START MAILBOX** =>	IDLE
4	IDLE	**STOP MAILBOX** =>	OFF
5	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 0 || Service_Data.OpCode> 4 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
6	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 6 && Service_Data.OpCode == 2 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
7	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 && Service_Data.OpCode == 2	DWLS

N	État actuel	Événement /Condition => Action	Prochain état
		=> Password = Service_Data.Password Filename = Service_Data.FileName FoE Write.ind (Address, Password, Filename)	
8	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length =< 6 && Service_Data.OpCode == 1 => Length = 4 Service_Data == 1, MBXERR_INVALIDSIZE ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
9	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length > 6 && Service_Data.OpCode == 1 => Password = Service_Data.Password Filename = Service_Data.FileName FoE Read.ind (Address, Password, Filename)	UPLS
10	IDLE	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Length == 3 && (Service_Data.OpCode == 3 || Service_Data.OpCode == 4) => Length = 6 Service_Data.OpCode = 5 Service_Data.Error_Code = ABT_SEQ FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
11	IDLE	**FoE Read Upd.cnf (success)** => ignore	IDLE
12	IDLE	**other application service primitives** => ignore	IDLE
13	DWLS	**START MAILBOX** => ignore	DWLS
14	DWLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
15	DWLS	**FoE Error.req (Address, ErrorCode, ErrorText)** => Seg = 0 Length = 6 + size(Error_Text) Service_Data.OpCode = 5 Service_Data.Error_Code = Error_Code Service_Data.Error_Text= Error_Text FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
16	DWLS	**FoE Ack.req (Address, Packet Number)** => Length = 6 Service_Data.OpCode = 4 Service_Data.PacketNumber = Packet Number Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	DWUPD
17	DWLS	**other application service primitives** => ignore	DWLS
18	UPLS	**START MAILBOX** => ignore	UPLS
19	UPLS	**STOP MAILBOX** => CANCEL SERVICE	OFF
20	UPLS	**FoE Error.req (Address, ErrorCode, ErrorText)** => Seg = 0 Length = 6 + size(Error_Text) Service_Data.OpCode = 5	WUPD

N	État actuel	Événement /Condition => Action	Prochain état
		Service_Data.Error_Code = Error_Code Service_Data.Error_Text= Error_Text FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
11	IDLE	**FoE Read Upd.cnf (success)** => ignore	IDLE
21	UPLS	**FoE Data.req (Address, Packet Number, Size, Data)** /Size = FullSize => Length = 6 + SIze Service_Data.OpCode = 3 Service_Data.PacketNumber = Packet Number Service_Data.Data = Data Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	UWUPD
22	UPLS	**FoE Data.req (Address, Packet Number, Size, Data)** /Size< FullSize => Length = 6 + SIze Service_Data.OpCode = 3 Service_Data.PacketNumber = Packet Number Service_Data.Data = Data Seg = Packet Number + 1 FoE Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	WUPD
23	UPLS	**other application service primitives** => ignore	UPLS
24	UWUPD	**START MAILBOX** =>	UWUPD
25	UWUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
26	UWUPD	**FoE Read Upd.cnf (success)** =>	UWSEG
27	UWUPD	**Timeout** => RESET MAILBOX	OFF
28	WUPD	**START MAILBOX** =>	WUPD
29	WUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
30	WUPD	**FoE Read Upd.cnf (success)** =>	IDLE
31	WUPD	**Timeout** => RESET MAILBOX	OFF
32	DWUPD	**START MAILBOX** =>	DWUPD
33	DWUPD	**STOP MAILBOX** => RESET MAILBOX	OFF
34	DWUPD	**FoE Read Upd.cnf (success)** =>	DWSEG
35	DWUPD	**Timeout** => RESET MAILBOX	OFF
36	ERR	**START MAILBOX** =>	ERR
37	ERR	**STOP MAILBOX**	OFF

N	État actuel	Événement /Condition => Action	Prochain état
		=> RESET MAILBOX	
38	ERR	**ERR Read Upd.cnf (success)** =>	IDLE
39	ERR	**Timeout** => RESET MAILBOX	OFF
40	DWSEG	**START MAILBOX** =>	DWSEG
41	DWSEG	**STOP MAILBOX** =>	OFF
42	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode != 3 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
43	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
44	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq && Length = FullSize + 6 => Size = Length - 6 Packet Number = Service_Data.Packet Number Data = Service_Data.Data FoE Data.ind (Address, Packet Number, Size, Data)	DWLS
45	DWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 3 && Service_Data.PackeNumber != Seq && Length < FullSize + 6 => Size = Length - 6 Packet Number = Service_Data.Packet Number Data = Service_Data.Data FoE Data.ind (Address, Packet Number, Size, Data)	IDLE
46	DWSEG	**other application service primitives** => ignore	DWSEG
47	UWSEG	**START MAILBOX** =>	UWSEG
48	UWSEG	**STOP MAILBOX** =>	OFF
49	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode != 4 => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0 FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	ERR
50	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 4 && Service_Data.PackeNumber != Seq => Length = 4 Service_Data == 1, MBXERR_INVALIDHEADER Seg =0	ERR

N	État actuel	Événement /Condition => Action	Prochain état
		FoE Error.ind (Address, ErrorCode = ABT_SEQ, ErrorText) ERR Mailbox Read Upd.req (Length, Address, Channel, Priority, Service_Data)	
51	UWSEG	**FoE Mailbox Write.ind (Length, Address, Channel, Priority, Service Data)** /Service_Data.OpCode == 4 && Service_Data.PackeNumber == Seq => Packet Number = Service_Data.Packet Number FoE Ack.req (Address, Packet Number)	UPLS
52	UWSEG	**other application service primitives** => ignore	UWSEG

6.5 Machine de protocole DMPM

Les services spécifiés dans la CEI 61158-3-12 sont utilisés directement dans les machines de protocole ARPM.

Bibliographie

CEI 61131-3, *Automates programmables – Partie 3: Langages de programmation*

CEI 61158-1:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61158-4-12, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-12: Spécification du protocole de la couche liaison de données – Éléments type 12*

CEI 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

CEI 61784-2, *Réseaux de communication industriels – Profils – Partie 2: Profils de bus de terrain supplémentaires pour les réseaux en temps réel basés sur l'ISO/CEI 8802-3*

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange* (disponible en anglais seulement)

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1* (disponible en anglais seulement)

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)* (disponible en anglais seulement)

ISO 8601, *Éléments de données et formats d'échange – Échange d'information – Représentation de la date et de l'heure* (disponible en anglais seulement)

ISO 15745-1, *Systèmes d'automatisation industrielle et intégration – Cadres d'intégration d'application pour les systèmes ouverts – Partie 1: Description générale de référence*

IETF RFC 792, *Internet Control Message Protocol;* disponible à l'adresse <http://www.ietf.org>

_____

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