

**DIN EN 61158-6-12**

ICS 35.100.70; 35.160

Ersatz für  
DIN EN 61158-6-12:2013-03  
Siehe Anwendungsbeginn

**Industrielle Kommunikationsnetze –  
Feldbusse –  
Teil 6-12: Protokollspezifikation des Application Layer  
(Anwendungsschicht) – Typ 12-Elemente  
(IEC 61158-6-12:2014);  
Englische Fassung EN 61158-6-12:2014, nur auf CD-ROM**

Industrial communication networks –  
Fieldbus specifications –  
Part 6-12: Application layer protocol specification – Type 12 elements  
(IEC 61158-6-12:2014);  
English version EN 61158-6-12:2014, only CD-ROM

Réseaux de communication industriels –  
Spécifications des bus de terrain –  
Partie 6-12: Spécification du protocole de la couche application – Éléments de type 12  
(CEI 61158-6-12:2014);  
Version anglaise EN 61158-6-12:2014, seulement en CD-ROM

Gesamtumfang 154 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

## Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2014-09-23 angenommene Europäische Norm als DIN-Norm ist 2015-09-01.

Für DIN EN 61158-6-12:2013-03 besteht eine Übergangsfrist bis 2017-09-23.

## Nationales Vorwort

*Vorausgegangener Norm-Entwurf: E DIN EN 61158-600:2013-08.*

Für dieses Dokument ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE ([www.dke.de](http://www.dke.de)) zuständig.

Die enthaltene IEC-Publikation wurde vom SC 65C „Industrial networks“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (stability date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe des Dokuments.

Der Zusammenhang der zitierten Dokumente mit den entsprechenden Deutschen Dokumenten ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für die vorliegende Norm erfüllt.

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61158-6-12 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als Nationaler Anhang NA der vorliegenden Norm hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

## Einführung

Ein Feldbus ist ein digitaler serieller Datenbus für die Kommunikation zwischen Geräten der industriellen Automatisierungstechnik wie beispielsweise Messeinrichtungen, Stellgliedern, Reglern und speicherprogrammierbaren Steuerungen. Die Normen der Reihe DIN EN 61158 legen eine Anzahl von Feldbus-Protokolltypen fest. Die vorliegende Norm legt das Protokoll des Application Layer für Typ 12 fest.

## **Änderungen**

Gegenüber DIN EN 61158-6-12:2013-03 wurden folgende Änderungen vorgenommen:

- a) redaktionelle Überarbeitung;
- b) Abschnitt 6 (ESM) explizite Geräteidentifikation eingeführt.

## **Frühere Ausgaben**

DIN V 19245-1: 1988-01

DIN EN 61158-6: 2005-05

DIN EN 61158-6-12: 2008-09, 2013-03

## Nationaler Anhang NA (informativ)

### 1 Anwendungsbereich<sup>N1)</sup>

#### 1.1 Allgemeines

Der Feldbus-Application Layer (FAL) liefert Benutzerprogramme mit Hilfsmitteln für den Zugriff auf die Feldbuskommunikationsumgebung. In dieser Hinsicht, kann der FAL als ein Fenster zwischen zugehörigen Anwendungsprogrammen betrachtet werden.

Diese Norm stellt für Typ 12 Feldbusse maßgebliche allgemeine Elemente für zeitkritischen und nicht zeitkritischen Datentransfer zwischen Anwendungsprogrammen in einer Automatisierungsumgebung bereit. Der Begriff „zeitkritisch“ (en: time-critical) wird verwendet um die Präsenz eines Zeitfensters darzustellen, in dem eine oder mehrere festgelegte Aktivitäten mit einem vorherbestimmten Grad an Gewissheit erledigt werden. Das Scheitern der Erledigung der festgelegten Aktivität innerhalb des Zeitfensters, trägt das Risiko des Betriebsausfalls der Anwendung mit, und somit das aufgeforderte Eingreifen mit dazugehörigem Risiko für Geräte, Anlage und möglicherweise Menschenleben.

Diese Norm definiert auf eine abstrakte Weise das von außen sichtbare Verhalten, das von den verschiedenen Typen des Feldbus-Application Layer bereitgestellt wird, in Form von:

- einer abstrakten Syntax zur Festlegung der Application Layer Protokoll-Dateneinheiten die zwischen kommunizierenden Anwendungseinheiten übermittelt werden,
- der übertragenen Syntax zur Festlegung der Application Layer Protokoll-Dateneinheiten die zwischen kommunizierenden Anwendungseinheiten übermittelt werden,
- des Anwendungskontext der Zustandsmaschine zur Festlegung des sichtbaren Dienstverhalten zwischen kommunizierenden Anwendungseinheiten,
- der Anwendungsbeziehung der Zustandsmaschine zur Festlegung des sichtbaren Kommunikationsverhalten zwischen kommunizierenden Anwendungseinheiten.

Der Zweck dieser Norm ist, das Protokoll zu definieren, welches bereitgestellt wird zur

- Festlegung der Leitungsdarstellung der Dienstprimitiven, welche in IEC 61158-5-12 beschrieben sind und
- zur Festlegung des von außen sichtbaren Verhaltens verbunden mit ihrer Übertragung.

Diese Norm beschreibt das Protokoll des Typ 12-Feldbus-Application Layer in Übereinstimmung mit dem OSI-Basis-Referenzmodell (ISO/IEC 7498-1) und der OSI-Application Layer-Struktur (ISO/IEC 9545).

FAL Dienste und Protokolle sind von FAL Applikations-Einheiten (AE) bereitgestellt, welche innerhalb von Anwendungsprozessen enthalten sind. Die FAL AE besteht aus einer Menge von objekt-orientierten Anwendungs-Dienst-Elementen (ASEs) und Layer-Management-Einheiten (LME) welche die AE leiten. Die ASEs liefern Kommunikationsdienste, die auf einer Menge von zugehörigen Applikationsprozessobjekt (APO) - Klassen arbeitet. Eine der FAL ASEs ist eine Management ASE, welche eine allgemeine Menge von Diensten für das Management der Instanzen der FAL-Klassen bereitstellt.

Obwohl diese Dienste, aus dem Blickwinkel der Anwendung, beschreiben wie Anfrage und Antwort ausgegeben und abgegeben werden, beinhalten sie keine Festlegung was die anfragende und antwortende Anwendungen mit diesen vorzunehmen haben. Die Verhaltensaspekte der Anwendungen sind nicht festgelegt, nur eine Definition von welchen Anfragen und Antworten diese senden/empfangen können, ist beschrieben. Dies erlaubt den FAL-Benutzern eine größere Flexibilität, um solch ein Objektverhalten zu standardisieren. Zusätzlich zu diesen Diensten, sind in dieser Norm ebenfalls einige Dienste zur Unterstützung festgelegt, um den Zugriff zur FAL bereitzustellen und somit bestimmte Aspekte des Betriebs zu steuern.

<sup>N1)</sup> Die Nummerierung aus IEC 61158-6-12 wurde beibehalten.

## 1.2 Spezifikationen

Das hauptsächliche Ziel dieser Norm, ist das Festlegen der Syntax und des Verhaltens des Application Layer-Protokolls, welcher die in IEC 61158-5-12 beschriebenen Application Layer-Dienste, vermittelt.

Ein zweites Ziel, ist das Bereitstellen von Migrationspfade von bisherigen bestehenden industriellen Kommunikationsprotokollen. Es ist dieses letztere Ziel, welches zur Steigerung von unterschiedlichen Protokollen, die in Teilen der IEC 61158-6 genormt werden, führt.

## 1.3 Konformität

Diese Norm legt keine individuellen Implementierungen oder Produkte fest, noch beschränkt sie die Implementierungen von Application Layer-Einheiten innerhalb eines industriellen Automatisierungssystems.

Es gibt keine Konformität von Geräten zu dieser Application Layer-Dienstdefinitionsnorm. Stattdessen wird Konformität durch das Implementieren dieser Application Layer-Protokoll-Spezifikation erreicht.

## 3.3 Begriffe zur Anwendungsschicht

Für die Anwendung dieses Dokuments gelten die folgenden Begriffe.

### 3.3.1

#### Anwendung

(en: application)

Funktion oder Datenstruktur, für die Daten konsumiert oder produziert werden

### 3.3.2

#### Anwendungsobjekte

(en: application objects)

mehrere Objektklassen, die zur Laufzeit für einen Nachrichtenaustausch im Netzwerk und innerhalb des Netzwerkgerätes sorgen und diesen verwalten

### 3.3.3

#### Anwendungsprozess

(en: application process)

Teil einer verteilten Anwendung in einem Netzwerk, der sich auf einem Gerät befindet und eindeutig adressiert wird

### 3.3.4

#### Anwendungsbeziehung

(en: application relationship)

kooperative Verbindung zwischen zwei oder mehr Aufrufen von Anwendungsentitäten zum Zweck des Informationsaustausches und der Koordination des gemeinsamen Betriebs

Anmerkung 1 zum Begriff: Diese Beziehung wird entweder durch den Austausch von Protokolldateneinheiten der Anwendung oder als Ergebnis von Vorkonfigurationsaktivitäten aktiviert.

### 3.3.5

#### Attribut

(en: attribute)

Beschreibung eines von außen sichtbaren Merkmals oder einer von außen sichtbaren Funktion eines Objekts

Anmerkung 1 zum Begriff: Die Attribute eines Objekts enthalten Informationen zu veränderlichen Teilen eines Objekts. Attribute liefern üblicherweise Statusinformationen oder steuern den Betrieb eines Objekts. Attribute können auch das Verhalten eines Objekts beeinflussen. Attribute werden in Klassenattribute und Instanzattribute unterteilt.

### 3.3.6

#### Basis-Slave

(en: basic slave)

Slave-Gerät, das nur die physische Adressierung von Daten unterstützt

### 3.3.7

#### **Verhalten**

(en: behavior)

Anzeige dafür, wie das Objekt auf bestimmte Ereignisse reagiert

### 3.3.8

#### **Bit**

(en: bit)

Informationseinheit, die aus einer 0 oder 1 besteht

Anmerkung 1 zum Begriff: Ein Bit ist die kleinste Dateneinheit, die übertragen werden kann.

### 3.3.9

#### **Kanal**

(en: channel)

Darstellung eines einzelnen physischen oder logischen Verwaltungsobjekts eines Slaves zur Steuerung der Datenübermittlung

### 3.3.10

#### **Client**

(en: client)

- 1) Objekt, das die Dienste eines anderen (Server-)Objekts zur Ausführung einer Aufgabe nutzt;
- 2) Initiator einer Nachricht, auf die ein Server reagiert

### 3.3.11

#### **Uhrensynchronisation**

(en: clock synchroization)

Darstellung einer Abfolge von Interaktionen zur Synchronisation der Uhren aller Zeitempfänger mit einem Zeit-Master

### 3.3.12

#### **Kommunikationsobjekt**

(en: communication object)

Komponente, die zur Laufzeit für einen Nachrichtenaustausch im Netzwerk sorgt und diesen verwaltet

### 3.3.13

#### **Verbindung**

(en: connection)

logische Verknüpfung zwischen zwei Anwendungsobjekten innerhalb eines Gerätes oder in unterschiedlichen Geräten

### 3.3.14

#### **konsumieren**

(en: consume)

Vorgang des Empfangs von Daten von einem Provider

### 3.3.15

#### **Konsument**

(en: consumer)

Knoten oder Senke, der/die Daten von einem Provider empfängt

### 3.3.16

#### **Übertragungspfad**

(en: conveyance path)

unidirektionaler Strom von APDUs in einer Anwendungsbeziehung

### 3.3.17

#### **zyklisch**

(en: cyclic)

Ereignisse, die sich regelmäßig wiederholen

**3.3.18****Daten**

(en: data)

allgemeine Bezeichnung für jegliche über einen Feldbus übertragene Informationen

**3.3.19****Datenkonsistenz**

(en: data consistency)

Mittel für die kohärente Übertragung des Eingabe- oder Ausgabedatenobjekts zwischen Client und Server und für den kohärenten Zugriff auf das Eingabe- oder Ausgabedatenobjekt innerhalb von Client und Server

**3.3.20****Datentyp**

(en: data type)

Beziehung zwischen den Werten und der Codierung für Daten dieses Typs entsprechend den Definitionen in IEC 61131-3

**3.3.21****Datentypobjekt**

(en: data type object)

Eintrag im Objektverzeichnis, der einen Datentyp angibt

**3.3.22****Standard-Gateway**

(en: default gateway)

Gerät mit mindestens zwei Schnittstellen in zwei verschiedenen IP-Subnetzen, das als Router für ein Subnetz agiert

**3.3.23****Gerät**

(en: device)

an den Feldbus angeschlossene physische Entität, die aus mindestens einem Datenübermittlungselement (dem Netzelement) besteht und die ein Steuerungselement und/oder ein Abschlusselement (Wandler, Stellgerät usw.) enthalten darf

**3.3.24****Geräteprofil**

(en: device profile)

Zusammenstellung geräteabhängiger Informationen und Funktionalitäten, um Konsistenz zwischen ähnlichen Geräten desselben Gerätetyps herzustellen

**3.3.25****Diagnose-Informationen**

(en: diagnosis information)

alle Daten, die serverseitig für Wartungszwecke verfügbar sind

**3.3.26****verteilte Uhren**

(en: distributed clocks)

Methode zur Synchronisierung von Slaves und zur Aufrechterhaltung einer globalen Zeitbasis

**3.3.27****Fehler**

(en: error)

Nichtübereinstimmung zwischen einem berechneten, beobachteten oder gemessenen Wert oder Zustand und dem festgelegten oder theoretisch richtigen Wert oder Zustand

**3.3.28****Fehlerklasse**

(en: error class)

allgemeine Gruppierung für zugehörige Fehlerdefinitionen und entsprechende Fehlercodes

**3.3.29**

**Fehlercode**

(en: error code)

Kennzeichnung eines bestimmten Fehlertyps innerhalb einer Fehlerklasse

**3.3.30**

**Ereignis**

(en: event)

Instanz einer Änderung von Zuständen

**3.3.31**

**Speicherverwaltungseinheit des Feldbusses**

(en: fieldbus memory management unit)

Funktion, die eine oder mehrere Korrespondenzen zwischen logischen Adressen und dem physischen Speicher aufbaut

**3.3.32**

**Entität der Speicherverwaltungseinheit des Feldbusses**

(en: fieldbus memory management unit entity)

einzelnes Element der Speicherverwaltungseinheit des Feldbusses: eine Korrespondenz zwischen einem kohärenten logischen Adressraum und einem kohärenten physischen Speicherort

**3.3.33**

**Rahmen**

(en: frame)

umgangssprachliches Synonym für DLPDU

**3.3.34**

**Full-Slave**

(en: full slave)

Slave-Gerät, das sowohl die physische als auch die logische Adressierung von Daten unterstützt

**3.3.35**

**Index**

(en: index)

Adresse eines Objekts innerhalb eines Anwendungsprozesses

**3.3.36**

**Schnittstelle**

(en: interface)

gemeinsam genutzte Grenze zwischen zwei funktionalen Einheiten, die durch funktionale Merkmale, Signalmerkmale oder entsprechend andere Merkmale festgelegt ist

**3.3.37**

**Little-Endian**

(en: little endian)

Datendarstellung von Multi-Oktett-Feldern, wobei das am wenigsten signifikante Oktett zuerst übertragen wird

**3.3.38**

**Master**

(en: master)

Gerät, das den Datentransfer im Netzwerk steuert und den Medienzugriff der Slaves durch Senden von Nachrichten initiiert und das die Schnittstelle zum Steuerungssystem bildet

**3.3.39**

**Zuordnung**

(en: mapping)

Verbindung zwischen zwei Objekten in einer Weise, dass ein Objekt ein Teil des anderen Objekts ist

**3.3.40****Zuordnungsparameter**

(en: mapping parameters)

Satz von Werten, der die Verbindung zwischen Anwendungsobjekten und Prozessdatenobjekten festlegt

**3.3.41****Medium**

(en: medium)

Kabel, Lichtwellenleiter oder anderes Mittel, über das Kommunikationssignale zwischen zwei oder mehr Punkten übertragen werden

Anmerkung 1 zum Begriff: „Medien“ ist der Plural von „Medium“.

**3.3.42****Nachricht**

(en: message)

geordnete Folge von Oktetten für die Übermittlung von Informationen

Anmerkung 1 zum Begriff: Nachrichten werden normalerweise für die Übermittlung von Informationen zwischen Peers auf der Anwendungsschicht verwendet.

**3.3.43****Netzwerk**

(en: network)

Menge von Knoten, die über ein bestimmtes Kommunikationsmedium miteinander verbunden sind, einschließlich zwischengeschalteter Repeater, Bridges, Router und Gateways unterer Schichten

**3.3.44****Knoten**

(en: node)

- a) einzelne DL-Entität, wie sie auf einem lokalen Verbindungsweig existiert;
- b) Endpunkt eines Verbindungszugs in einem Netzwerk oder Punkt, an dem zwei oder mehrere Verbindungswege aufeinandertreffen

[abgeleitet von IEC 61158-2]

**3.3.45****Objekt**

(en: object)

abstrakte Darstellung einer bestimmten Komponente innerhalb eines Gerätes

Anmerkung 1 zum Begriff: Ein Objekt kann sein:

- a) eine abstrakte Darstellung der Fähigkeiten eines Gerätes. Objekte können aus einigen der folgenden Komponenten oder aus allen folgenden Komponenten bestehen:
  - 1) Daten (Informationen, die sich im Laufe der Zeit ändern);
  - 2) Konfiguration (Parameter für das Verhalten);
  - 3) Methoden (Dinge, die mithilfe der Daten und der Konfiguration durchgeführt werden können);
- b) eine Zusammenstellung zugehöriger Daten (in Form von Variablen) und Methoden (Prozeduren) für Operationen mit diesen Daten, die eine klar definierte Schnittstelle und ein klar definiertes Verhalten aufweisen.

**3.3.46****Objektverzeichnis**

(en: object dictionary)

Datenstruktur, die von Index und Subindex adressiert wird und Beschreibungen von Datentypobjekten, Kommunikationsobjekten und Anwendungsobjekten enthält

**3.3.47****Prozessdaten**

(en: process data)

Zusammenstellung von Anwendungsobjekten, die zum Zweck der Messung und Steuerung zyklisch oder azyklisch übertragen werden

**3.3.48**

**Prozessdatenobjekt**

(en: process data object)

Struktur, die durch Zuordnungsparameter beschrieben wird und eine oder mehrere Prozessdatenentitäten enthält

**3.3.49**

**Produzent**

(en: producer)

Knoten oder Quelle, der/die Daten an einen oder viele Konsumenten sendet

**3.3.50**

**Ressource**

(en: resource)

Verarbeitungs- oder Informationsfähigkeit

**3.3.51**

**Segment**

(en: segment)

Zusammenstellung von einem realen Master und einem oder mehreren Slaves

**3.3.52**

**Server**

(en: server)

Objekt, das einem anderen (Client-)Objekt Dienste bereitstellt

**3.3.53**

**Dienst**

(en: service)

Operation oder Funktion, die ein Objekt und/oder eine Objektklasse auf Anforderung eines anderen Objekts und/oder einer anderen Objektklasse hin ausführt

**3.3.54**

**Slave**

(en: slave)

DL-Entität, die nur nach Initiierung durch den vorausgehenden Slave oder den Master Zugang zum Medium hat

**3.3.55**

**Subindex**

(en: subindex)

untergeordnete Adresse eines Objekts innerhalb des Objektverzeichnisses

**3.3.56**

**Sync-Manager**

(en: sync manager)

Zusammenstellung von Steuerelementen zur Koordinierung des Zugriffs auf gleichzeitig verwendete Objekte

**3.3.57**

**Sync-Manager-Kanal**

(en: sync manager channel)

einzelne Steuerelemente zur Koordinierung des Zugriffs auf gleichzeitig verwendete Objekte

**3.3.58**

**Switch**

(en: switch)

MAC-Bridge, wie in IEEE 802.1D definiert

English Version

**Industrial communication networks - Fieldbus specifications -  
Part 6-12: Application layer protocol specification - Type 12  
elements  
(IEC 61158-6-12:2014)**

Réseaux de communication industriels - Spécifications des  
bus de terrain - Partie 6-12: Spécification du protocole de la  
couche application - Éléments de type 12  
(CEI 61158-6-12:2014)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-12:  
Protokollspezifikation des Application Layer  
(Anwendungsschicht) - Typ 12-Elemente  
(IEC 61158-6-12:2014)

This European Standard was approved by CENELEC on 2014-09-23. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



European Committee for Electrotechnical Standardization  
Comité Européen de Normalisation Electrotechnique  
Europäisches Komitee für Elektrotechnische Normung

**CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels**

## **Foreword**

The text of document 65C/764/FDIS, future edition 3 of IEC 61158-6-12, prepared by SC 65C "Industrial networks" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-6-12:2014.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2015-06-23
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2017-09-23

This document supersedes EN 61158-6-12:2012.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

## **Endorsement notice**

The text of the International Standard IEC 61158-6-12:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

|                  |      |                                               |
|------------------|------|-----------------------------------------------|
| IEC 61131-3      | NOTE | Harmonized as EN 61131-3.                     |
| IEC 61158-1:2014 | NOTE | Harmonized as EN 61158-1:2014 (not modified). |
| IEC 61158-4-12   | NOTE | Harmonized as EN 61158-4-12.                  |
| IEC 61784-1      | NOTE | Harmonized as EN 61784-1.                     |
| IEC 61784-2      | NOTE | Harmonized as EN 61784-2.                     |

## CONTENTS

|                                                                                                                                |     |
|--------------------------------------------------------------------------------------------------------------------------------|-----|
| INTRODUCTION.....                                                                                                              | 8   |
| 1 Scope .....                                                                                                                  | 9   |
| 1.1 General.....                                                                                                               | 9   |
| 1.2 Specifications .....                                                                                                       | 10  |
| 1.3 Conformance.....                                                                                                           | 10  |
| 2 Normative references .....                                                                                                   | 10  |
| 3 Terms, definitions, symbols, abbreviations and conventions .....                                                             | 11  |
| 3.1 Reference model terms and definitions.....                                                                                 | 11  |
| 3.2 Service convention terms and definitions.....                                                                              | 12  |
| 3.3 Application layer definitions .....                                                                                        | 13  |
| 3.4 Common symbols and abbreviations .....                                                                                     | 18  |
| 3.5 Additional symbols and abbreviations.....                                                                                  | 18  |
| 3.6 Conventions .....                                                                                                          | 20  |
| 4 Application layer protocol specification.....                                                                                | 24  |
| 4.1 Operating principle .....                                                                                                  | 24  |
| 4.2 Node reference model .....                                                                                                 | 25  |
| 5 FAL syntax description .....                                                                                                 | 26  |
| 5.1 Coding principles.....                                                                                                     | 26  |
| 5.2 Data types and encoding rules .....                                                                                        | 26  |
| 5.3 AR coding .....                                                                                                            | 30  |
| 5.4 SII coding.....                                                                                                            | 35  |
| 5.5 Isochronous PDI coding.....                                                                                                | 40  |
| 5.6 CoE coding.....                                                                                                            | 43  |
| 5.7 EoE coding.....                                                                                                            | 82  |
| 5.8 FoE Coding .....                                                                                                           | 90  |
| 6 FAL protocol state machines.....                                                                                             | 96  |
| 6.1 Overall structure.....                                                                                                     | 96  |
| 6.2 AP-Context state machine.....                                                                                              | 98  |
| 6.3 FAL service protocol machine (FSPM).....                                                                                   | 98  |
| 6.4 Application Relationship Protocol Machines (ARPMs).....                                                                    | 98  |
| 6.5 DLL mapping protocol machine (DMPM).....                                                                                   | 141 |
| Bibliography .....                                                                                                             | 142 |
| Annex ZA (normative) Normative references to international publications with their<br>corresponding European publications..... | 143 |
| Figure 1 – Common structure of specific fields .....                                                                           | 20  |
| Figure 2 – Type description example .....                                                                                      | 22  |
| Figure 3 – Slave Node Reference Model .....                                                                                    | 25  |
| Figure 4 – Encoding of Time of Day value .....                                                                                 | 27  |
| Figure 5 – Encoding of Time Difference value.....                                                                              | 27  |
| Figure 6 – AL Control Request structure.....                                                                                   | 30  |
| Figure 7 – AL Control Response structure .....                                                                                 | 31  |
| Figure 8 – AL State Changed structure.....                                                                                     | 33  |

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| Figure 9 – PDI Control type description .....                       | 34  |
| Figure 10 – Sync Configuration type description .....               | 34  |
| Figure 11 – Distributed Clock sync and latch type description ..... | 41  |
| Figure 12 – CoE general structure.....                              | 44  |
| Figure 13 – SDO Download Expedited Request structure .....          | 45  |
| Figure 14 – SDO Download Expedited Response structure .....         | 46  |
| Figure 15 – SDO Download Normal Request structure.....              | 47  |
| Figure 16 – Download SDO Segment Request structure .....            | 49  |
| Figure 17 – Download SDO Segment Response structure.....            | 50  |
| Figure 18 – SDO Upload Expedited Request structure.....             | 50  |
| Figure 19 – SDO Upload Expedited Response structure .....           | 51  |
| Figure 20 – SDO Upload Normal Response structure .....              | 53  |
| Figure 21 – Upload SDO Segment Request structure .....              | 54  |
| Figure 22 – Upload SDO Segment Response structure.....              | 54  |
| Figure 23 – Abort SDO Transfer Request structure.....               | 55  |
| Figure 24 – SDO Information Service structure.....                  | 58  |
| Figure 25 – Get OD List Request structure .....                     | 59  |
| Figure 26 – Get OD List Response structure .....                    | 60  |
| Figure 27 – Get Object Description Request structure .....          | 61  |
| Figure 28 – Get Object Description Response structure.....          | 62  |
| Figure 29 – Get Entry Description Request structure .....           | 63  |
| Figure 30 – Get Entry Description Response structure.....           | 64  |
| Figure 31 – SDO Info Error Request structure .....                  | 65  |
| Figure 32 – EoE general structure.....                              | 82  |
| Figure 33 – EoE Timestamp structure .....                           | 84  |
| Figure 34 – EoE Fragment Data structure .....                       | 85  |
| Figure 35 – Set IP Parameter Request structure.....                 | 86  |
| Figure 36 – Set IP Parameter Response structure .....               | 87  |
| Figure 37 – Set MAC Filter Request structure.....                   | 88  |
| Figure 38 – Set MAC Filter Response structure .....                 | 90  |
| Figure 39 – Read Request structure.....                             | 91  |
| Figure 40 – Write Request structure .....                           | 91  |
| Figure 41 – Data Request structure.....                             | 92  |
| Figure 42 – Ack Request structure .....                             | 93  |
| Figure 43 – Error Request structure .....                           | 94  |
| Figure 44 – Busy Request structure .....                            | 96  |
| Figure 45 – Relationship among Protocol Machines.....               | 97  |
| Figure 46 – AR Protocol machines .....                              | 98  |
| Figure 47 – ESM Diagramm .....                                      | 100 |
| Table 1 – PDU element description example .....                     | 21  |
| Table 2 – Example attribute description.....                        | 22  |

|                                                                  |    |
|------------------------------------------------------------------|----|
| Table 3 – State machine description elements .....               | 23 |
| Table 4 – Description of state machine elements .....            | 23 |
| Table 5 – Conventions used in state machines .....               | 24 |
| Table 6 – Transfer Syntax for bit sequences .....                | 28 |
| Table 7 – Transfer syntax for data type Unsignedn .....          | 28 |
| Table 8 – Transfer syntax for data type Integern .....           | 29 |
| Table 9 – AL Control Description .....                           | 30 |
| Table 10 – AL Control Response .....                             | 31 |
| Table 11 – AL Status Codes .....                                 | 31 |
| Table 12 – AL State Changed .....                                | 33 |
| Table 13 – PDI Control .....                                     | 34 |
| Table 14 – PDI Configuration .....                               | 34 |
| Table 15 – Sync Configuration .....                              | 35 |
| Table 16 – Slave Information Interface Area .....                | 35 |
| Table 17 – Slave Information Interface Categories .....          | 36 |
| Table 18 – Mailbox Protocols Supported Types .....               | 36 |
| Table 19 – Categories Types .....                                | 37 |
| Table 20 – Structure Category String .....                       | 37 |
| Table 21 – Structure Category General .....                      | 38 |
| Table 22 – Structure Category FMMU .....                         | 39 |
| Table 23 – Structure Category SyncM for each Element .....       | 39 |
| Table 24 – Structure Category TXPDO and RXPDO for each PDO ..... | 40 |
| Table 25 – Structure PDO Entry .....                             | 40 |
| Table 26 – Distributed Clock sync parameter .....                | 42 |
| Table 27 – Distributed Clock latch data .....                    | 43 |
| Table 28 – CoE elements .....                                    | 44 |
| Table 29 – SDO Download Expedited Request .....                  | 46 |
| Table 30 – SDO Download Expedited Response .....                 | 47 |
| Table 31 – SDO Download Normal Request .....                     | 48 |
| Table 32 – Download SDO Segment Request .....                    | 49 |
| Table 33 – Download SDO Segment Response .....                   | 50 |
| Table 34 – SDO Upload Expedited Request .....                    | 51 |
| Table 35 – SDO Upload Expedited Response .....                   | 52 |
| Table 36 – SDO Upload Normal Response .....                      | 53 |
| Table 37 – Upload SDO Segment Request .....                      | 54 |
| Table 38 – Upload SDO Segment Response .....                     | 55 |
| Table 39 – Abort SDO Transfer Request .....                      | 56 |
| Table 40 – SDO Abort Codes .....                                 | 57 |
| Table 41 – SDO Information Service .....                         | 58 |
| Table 42 – Get OD List Request .....                             | 59 |
| Table 43 – Get OD List Response .....                            | 60 |
| Table 44 – Get Object Description Request .....                  | 61 |

|                                                    |    |
|----------------------------------------------------|----|
| Table 45 – Get Object Description Response .....   | 62 |
| Table 46 – Get Entry Description Request.....      | 63 |
| Table 47 – Get Entry Description Response .....    | 64 |
| Table 48 – SDO Info Error Request.....             | 66 |
| Table 49 – Emergency Request .....                 | 67 |
| Table 50 – Emergency Error Codes.....              | 68 |
| Table 51 – Error Code .....                        | 68 |
| Table 52 – Diagnostic Data.....                    | 69 |
| Table 53 – Sync Manager Length Error .....         | 69 |
| Table 54 – Sync Manager Address Error .....        | 69 |
| Table 55 – Sync Manager Settings Error .....       | 69 |
| Table 56 – RxPDO Transmission via mailbox .....    | 70 |
| Table 57 – TxPDO Transmission via mailbox.....     | 70 |
| Table 58 – RxPDO Remote Transmission Request ..... | 71 |
| Table 59 – TxPDO Remote Transmission Request ..... | 71 |
| Table 60 – Command object structure .....          | 72 |
| Table 61 – Object Dictionary Structure .....       | 72 |
| Table 62 – Object Code Definitions .....           | 72 |
| Table 63 – Basic Data Type Area.....               | 73 |
| Table 64 – Extended Data Type Area.....            | 74 |
| Table 65 – Enumeration Definition .....            | 75 |
| Table 66 – CoE Communication Area.....             | 75 |
| Table 67 – Device Type .....                       | 76 |
| Table 68 – Error Register.....                     | 77 |
| Table 69 – Manufacturer Device Name.....           | 77 |
| Table 70 – Manufacturer Hardware Version.....      | 77 |
| Table 71 – Manufacturer Software Version.....      | 78 |
| Table 72 – Identity Object.....                    | 78 |
| Table 73 – Receive PDO Mapping.....                | 79 |
| Table 74 – Transmit PDO Mapping.....               | 79 |
| Table 75 – Sync Manager Communication Type .....   | 80 |
| Table 76 – Sync Manager Channel 0-31 .....         | 81 |
| Table 77 – Sync Manager Synchronization .....      | 82 |
| Table 78 – Initiate EoE Request.....               | 83 |
| Table 79 – Initiate EoE Response .....             | 84 |
| Table 80 – EoE Fragment Data .....                 | 85 |
| Table 81 – EoE Data .....                          | 86 |
| Table 82 – Set IP Parameter Request .....          | 86 |
| Table 83 – Set IP Parameter Response.....          | 88 |
| Table 84 – EoE Result Parameter .....              | 88 |
| Table 85 – Set MAC Filter Request .....            | 89 |
| Table 86 – Set MAC Filter Response.....            | 90 |

|                                                                            |     |
|----------------------------------------------------------------------------|-----|
| Table 87 – Read Request .....                                              | 91  |
| Table 88 – Write Request.....                                              | 92  |
| Table 89 – Data Request .....                                              | 93  |
| Table 90 – Ack Request.....                                                | 94  |
| Table 91 – Error Request.....                                              | 95  |
| Table 92 – Error codes of FoE .....                                        | 95  |
| Table 93 – Busy Request .....                                              | 96  |
| Table 94 – State transitions and local management services.....            | 100 |
| Table 95 – Primitives issued by ESM to DL .....                            | 101 |
| Table 96 – Primitives issued by DL to ESM .....                            | 101 |
| Table 97 – Primitives issued by Application to ESM .....                   | 102 |
| Table 98 – Primitives issued by ESM to Application .....                   | 102 |
| Table 99 – ESM Variables.....                                              | 103 |
| Table 100 – ESM macros .....                                               | 104 |
| Table 101 – ESM functions .....                                            | 104 |
| Table 102 – ESM state table .....                                          | 105 |
| Table 103 – Primitives issued by Mailbox handler to DL.....                | 117 |
| Table 104 – Primitives issued by DL to Mailbox handler.....                | 118 |
| Table 105 – Primitives issued by Protocol handler to Mailbox handler ..... | 118 |
| Table 106 – Primitives issued by Mailbox handler to Protocol handler ..... | 118 |
| Table 107 – Primitives issued by Application to CoESM.....                 | 119 |
| Table 108 – Primitives issued by CoESM to Application.....                 | 120 |
| Table 109 – CoESM state table.....                                         | 120 |
| Table 110 – Primitives issued by Application to EoESM.....                 | 130 |
| Table 111 – Primitives issued by EoESM to Application .....                | 131 |
| Table 112 – EoESM state table .....                                        | 131 |
| Table 113 – Primitives issued by Application to FoESM .....                | 136 |
| Table 114 – Primitives issued by FoESM to Application .....                | 137 |
| Table 115 – FoESM state table .....                                        | 137 |

## 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<br>correspondent AL-entities (N=7) | [ISO/IEC 7498-1] |
| 3.1.2  | (N)-entity<br>AL-entity (N=7)                                 | [ISO/IEC 7498-1] |
| 3.1.3  | (N)-layer<br>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<br>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);<br>requestor.deliver (primitive)   |
| 3.2.4 | deliver (primitive)                                     |
| 3.2.5 | AL-service-primitive;<br>primitive                      |
| 3.2.6 | AL-service-provider                                     |
| 3.2.7 | AL-service-user                                         |
| 3.2.8 | indication (primitive);<br>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 <sup>2</sup> PROM | 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 task 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.

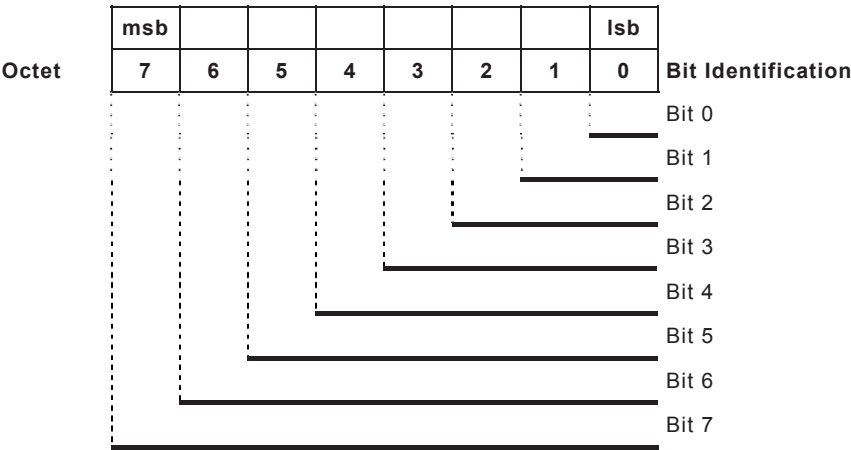


Figure 1 – Common structure of specific fields

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<br>0x01: size of Data in Data Set Size specified |
|            | Transfer Type     | Unsigned1 | 0x01: Expedited transfer                                                               |
|            | Data Set Size     | Unsigned2 | 0x00: 4 Octet Data<br>0x01: 3 Octet Data<br>0x02: 2 Octet Data<br>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<br>/Condition<br>=> Action | Next state |
|---|---------------|----------------------------------|------------|
|   |               |                                  |            |

**Table 4 – Description of state machine elements**

| Description element | Meaning                                                                                                |
|---------------------|--------------------------------------------------------------------------------------------------------|
| Current state       | name of the given states.                                                                              |
| Next state          |                                                                                                        |
| #                   | 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.<br>EXAMPLE<br>Identifier = reason<br>means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'             |
| "xxx"      | indicates fixed visible string.<br>EXAMPLE Identifier = "abc"<br>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.

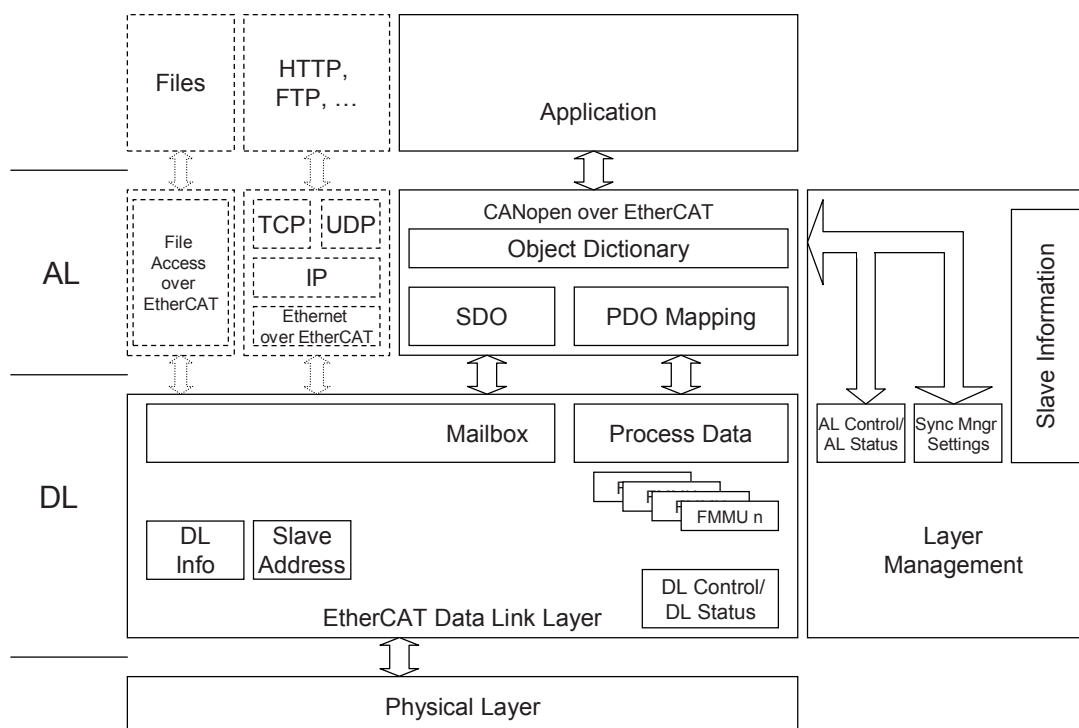


Figure 3 – Slave Node Reference Model

### 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<br>milliseconds<br>since<br>midnight                               |
| 2      | $2^{23}$ | $2^{22}$ | $2^{21}$ | $2^{20}$ | $2^{19}$ | $2^{18}$ | $2^{17}$ | $2^{16}$ |                                                                              |
| 3      | $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$    | $2^8$    |                                                                              |
| 4      | $2^7$    | $2^6$    | $2^5$    | $2^4$    | $2^3$    | $2^2$    | $2^1$    | $2^0$    |                                                                              |
| 5      | $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$    | $2^8$    | number of<br>days since<br>1984-01-01<br><br>only with<br>date<br>indication |
| 6      | $2^7$    | $2^6$    | $2^5$    | $2^4$    | $2^3$    | $2^2$    | $2^1$    | $2^0$    |                                                                              |
| 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      | $2^{31}$ | $2^{30}$ | $2^{29}$ | $2^{28}$ | $2^{27}$ | $2^{26}$ | $2^{25}$ | $2^{24}$ | milliseconds                 |
| 2      | $2^{23}$ | $2^{22}$ | $2^{21}$ | $2^{20}$ | $2^{19}$ | $2^{18}$ | $2^{17}$ | $2^{16}$ |                              |
| 3      | $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$    | $2^8$    |                              |
| 4      | $2^7$    | $2^6$    | $2^5$    | $2^4$    | $2^3$    | $2^2$    | $2^1$    | $2^0$    |                              |
| 5      | $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$    | $2^8$    | days                         |
| 6      | $2^7$    | $2^6$    | $2^5$    | $2^4$    | $2^3$    | $2^2$    | $2^1$    | $2^0$    | only with date<br>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  $b = b_0 \dots b_{n-1}$  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):

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

**Table 6 – Transfer Syntax for bit sequences**

| octet number | 1.              | 2.                 | k.                        |
|--------------|-----------------|--------------------|---------------------------|
|              | $b_7 \dots b_0$ | $b_{15} \dots b_8$ | $b_{8k-1} \dots b_{8k-8}$ |

EXAMPLE

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

The bit sequence  $b = b_0 \dots b_9 = 0011\ 1000\ 01_0$  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 Unsigned $n$  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

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

is assigned the value

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

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 Unsigned $n$  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 Unsigned $n$**

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

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

### 5.2.7 Encoding of a Signed Integer value

Data of basic data type Integer $n$  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

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

is assigned the value

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

and, performing two's complement arithmetic,

$$\text{Integer}_n(b) = -\text{Integer}_n(\text{^}b) - 1 \quad \text{if } b_{n-1} = 1$$

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

EXAMPLE The value  $-266 = 0x\text{FEF6}$  with data type Integer16 is transferred in two octets, first 0xF6 and then 0xFE.

The Integer $n$  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 Integer $n$**

| 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

- The encoding of a variable length VisibleString value shall be primitive.
- There is no Length field and no termination symbol; the length is encoded implicitly.
- 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

- The encoding of a variable length UnicodeString value shall be primitive.
- There is no Length field; the length is encoded implicitly.
- 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<br>2: Pre-Operational<br>3: Bootstrap<br>4: Safe-Operational<br>8: Operational                                         |
| Acknowledge          | R1               | Unsigned1 | 0: Parameter Change of the AL Status Register will be unchanged<br>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;
} TALSTATUS;
```

**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<br>2: Pre-Operational<br>3: Bootstrap<br>4: Safe-Operational<br>8: Operational |
| Change               | R3               | Unsigned1  | 0: State transition successful<br>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<br>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             |

| Code   | Description                    | Current state<br>(or state change) | Resulting state   |
|--------|--------------------------------|------------------------------------|-------------------|
| 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      |
| 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                 |

| Code                    | Description                         | Current state<br>(or state change) | Resulting state |
|-------------------------|-------------------------------------|------------------------------------|-----------------|
| 0x0061                  | Device Identification value updated | P                                  | P + E           |
| 0x0062<br>...0x00EF     | Reserved                            |                                    |                 |
| 0x00F0                  | Application controller available    | I                                  | I + E           |
| other codes<br>< 0x8000 | reserved                            |                                    |                 |
| 0x8000 –<br>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;
} TALSTATUS;
```

**Figure 8 – AL State Changed structure**

The AL State Changed coding is specified in Table 12.

**Table 12 – AL State Changed**

| Parameter            | DL-user<br>Register | Data Type  | Value                                                                                  |
|----------------------|---------------------|------------|----------------------------------------------------------------------------------------|
| State                | R3                  | Unsigned4  | 1: Init<br>2: Pre-Operational<br>3: Bootstrap<br>4: Safe-Operational<br>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    PDType:                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<br>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<br>0x01: enable |
| Enable Interrupt Sync0    | R8               | unsigned1 | R         | R            | 0x00: disable<br>0x01: enable |
| Signal Conditioning Sync1 | R8               | unsigned2 | R         | R            | controller specific           |
| Enable Signal Sync1       | R8               | unsigned1 | R         | R            | 0x00: disable<br>0x01: enable |
| Enable Interrupt Sync1    | R8               | unsigned1 | R         | R            | 0x00: disable<br>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]    | shall 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)<br><br>Standard Mailbox size and Bootstrap Mailbox can differ. A bigger Mailbox size in Bootstrap mode can be used for optimization |
| Bootstrap Send Mailbox Offset    | 0x0016  | Unsigned16 | send Mailbox Offset for Bootstrap state (slave to master)                                                                                                                                      |
| Bootstrap Send Mailbox           | 0x0017  | Unsigned16 | send Mailbox Size for Bootstrap state (slave to                                                                                                                                                |

| Parameter                       | Address | Data Type  | Value/Description                                                                                                                       |
|---------------------------------|---------|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Size                            |         |            | master)<br>Standard Mailbox size and Bootstrap Mailbox can differ. A bigger Mailbox size in Bootstrap mode can be used for optimization |
| 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<br>NOTE: KiBit means 1024 Bit.<br>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<br>shall not be overwritten by Master or configuration tool.<br>Might be used for calibration values) |
| STRINGS         | 10                | string repository for other Categories structure of this category data<br>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<br>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-<br>0x0FFF | Vendor specific categories                                                                                                       |
| Reserved        | 0x1000-<br>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<br>[str1_len] | String1 Data                      |
| str2_len  | 0x0002+str1_len | Unsigned8          | length String2                    |
| str_2     | 0x0003+str1_len | BYTE<br>[str2_len] | String2 Data                      |
| ....      |                 |                    |                                   |
| strn_len  | 0x000z          | Unsigned8          | length Stringn                    |
| str_n     | 0x000z+1        | BYTE<br>[strn_len] | Stringn Data                      |
| PAD_Byte  | 0x000y          | BYTE               | padding if Category length is odd |