EtherCAT Specification – Part 3

Data Link Layer service definition

Document: ETG.1000.3 S (R) V1.0.4

Nomenclature:

ETG-Number	ETG.1000.3
Type	S (Standard)
State	R (Release)
Version	V1.0.4

Created by: ETG
Contact: info@ethercat.org
Date 15.09.2017

EtherCAT Technology Group

Trademarks and Patents

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Disclaimer

The documentation has been prepared with care. The technology described is, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

DOCUMENT HISTORY

Version	Comment
1.0	Adopted from IEC-Standard 61158-3 Type 12 for ETG use only!
1.0.1	Correctionsand clarifications according to ETG1000_ES_D_V0i6_EcatSpecErrata.pdf no changes in this document
1.0.2	Correctionsand clarifications according to ETG1000_V1i0i2_ES_R_V0i7_EcatSpecErrata.doc and IEC SC65C MT 9 editorial comments editorial changes in this document only
1.0.3	Correctionsand clarifications according to ETG1000_V1i0i2_ES_R_V0i8_EcatSpecErrata.doc
1.0.4	Correctionsand clarifications according to ETG1000_V1i0i3_ES_R_V0i9_EcatSpecErrata.doc

CONTENTS

Figures

Tables

1 Scope

1.1 Scope of this standard and accordance to IEC Standards

The ETG.1000 series specifies the EtherCAT Technology within the EtherCAT Technology group. It is devided into the following parts:

- ETG.1000.2: Physical Layer service definition and protocol specification
- ETG.1000.3: Data Link Layer service definition
- ETG.1000.4: Data Link Layer protocol specification
- ETG.1000.5: Application Layer service definition
- ETG.1000.6: Application Layer protocol specification

These parts are based on the corresponding parts of the IEC 61158 series Type 12. EtherCAT is named Type 12 in IEC 61158 to avoid the usage of brand names.

1.2 Overview

This part of the ETG.1000 series provides common elements for basic time-critical messaging communications between devices in an automation environment. 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 service provided by the EtherCAT fieldbus data-link layer in terms of

a) the primitive actions and events of the service;

b) the parameters associated with each primitive action and event, and the form which they take;

c) the interrelationship between these actions and events, and their valid sequences.

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

- the EtherCAT fieldbus application layer at the boundary between the application and data-link layers of the fieldbus reference model;
- systems management at the boundary between the data-link layer and systems management of the fieldbus reference model.

1.3 Specifications

The principal objective of this standard is to specify the characteristics of conceptual data-link layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of data-link protocols for time-critical communications. A secondary objective is to provide migration paths from previously-existing industrial communications protocols.

This specification may be used as the basis for formal DL-Programming-Interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this specification, including

a) the sizes and octet ordering of various multi-octet service parameters, and

b) the correlation of paired request and confirm, or indication and response, primitives.

1.4 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of data-link entities within industrial automation systems.

There is no conformance of equipment to this data-link layer service definition standard. Instead, conformance is achieved through implementation of the corresponding data-link protocol that fulfills the EtherCAT data-link layer services defined in this standard.

2 Normative references

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

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: Standard for Ethernet*

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

IEEE 802.1D, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges;* available at <http://www.ieee.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 DL-address [7498-3]

3.1.2 DL-connectionless-mode transmission [7498-1]

3.1.3 correspondent (N)-entities [7498-1]
correspondent DL-entities (N=2)
correspondent Ph-entities (N=1)

3.1.4 DL-duplex-transmission [7498-1]

3.1.5 (N)-entity [7498-1]
DL-entity (N=2)
Ph-entity (N=1)

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

3.1.7 layer-management [7498-1]

3.1.8 peer-entities [7498-1]

3.1.9 primitive name [7498-3]

3.1.10 DL-protocol [7498-1]

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

3.1.12 DL-relay [7498-1]

3.1.13 reset [7498-1]

3.1.14 responding-DL-address [7498-3]

3.1.15 routing [7498-1]

3.1.16 segmenting [7498-1]

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

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

3.1.19 DL-service-data-unit [7498-1]

3.1.20 DL-simplex-transmission [7498-1]

3.1.21 DL-subsystem [7498-1]

3.1.22 systems-management [7498-1]

3.1.23 DLS-user [7498-1]

3.1.24 DLS-user-data [7498-1]

3.2 Service convention terms and definitions

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

3.2.1 acceptor

3.2.2 asymmetrical service

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

3.2.4 deliver (primitive)

3.2.5 DL-service-primitive;
primitive

3.2.6 DL-service-provider

3.2.7 DL-service-user

3.2.8 DL-user-optional-facility

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

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

3.2.11 requestor

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

3.2.13 submit (primitive)

3.2.14 symmetrical service

3.3 Data-link service terms and definitions

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
basic slave
slave device that supports only physical addressing of data

3.3.4
bit
unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted

3.3.5
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

3.3.6
connection
logical binding between two application objects within the same or different devices

3.3.7
cyclic
events which repeat in a regular and repetitive manner

3.3.8
cyclic redundancy check
CRC
residual value computed from an array of data and used as a representative signature for the array

3.3.9
data
generic term used to refer to any information carried over a fieldbus

3.3.10
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.3.11
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.)

3.3.12
distributed clocks
method to synchronize slaves and maintain a global time base

3.3.13
DL-segment, link, local link
single DL-subnetwork in which any of the connected DLEs may communicate directly, without any intervening DL-relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication

3.3.14
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.3.15
event
instance of a change of conditions

3.3.16
fieldbus memory management unit
function that establishes one or several correspondences between logical addresses and physical memory

3.3.17
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

3.3.18
frame
denigrated synonym for DLPDU

3.3.19
full slave
slave device that supports both physical and logical addressing of data

3.3.20
interface
shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

3.3.21
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

3.3.22
mapping
correspondence between two objects in the way that one object is part of the other object

3.3.23
medium
cable, optical fibre, or other means by which communication signals are transmitted between two or more points

NOTE "media" is used as the plural of medium.

3.3.24
message
ordered series of octets intended to convey information

NOTE Normally used to convey information between peers at the application layer.

3.3.25
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.3.26
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 [derived from IEC 61158-2]

3.3.27
object
abstract representation of a particular component within a device

NOTE An object can be

a) an abstract representation of the capabilities of a device, 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); or

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

3.3.28
process data
data object containing application objects designated to be transferred cyclically or acyclically for the purpose of processing

3.3.29
receiving DLS-user
DL-service user that acts as a recipient of DL-user-data

NOTE A DL-service user can be concurrently both a sending and receiving DLS-user.

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

3.3.31
server
object which provides services to another (client) object

3.3.32
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.3.33
slave
DL-entity accessing the medium only after being initiated by the preceding slave or the master

3.3.34
Sync manager
collection of control elements to coordinate access to concurrently used objects.

3.3.35
Sync manager channel
single control elements to coordinate access to concurrently used objects.

3.3.36
switch
MAC bridge as defined in IEEE 802.1D

3.4 Symbols and abbreviations

APRD	Auto-increment physical read
APRW	Auto-increment physical read/write
APWR	Auto-increment physical write
ARMW	Auto-increment physical read / multiple write
BRD	Broadcast read
BRW	Broadcast read/write
BWR	Broadcast write
CAN	Controller area network
CoE	CAN application protocol over EtherCAT services
CSMA/CD	Carrier sense multiple access with collision detection
DC	Distributed clocks

DL-	Data-link layer (as a prefix)
DLC	DL-connection
DLCEP	DL-connection-end-point
DLE	DL-entity (the local active instance of the data-link layer)
DLL	DL-layer
DLPCI	DL-protocol-control-information
DLPDU	DL-protocol-data-unit
DLM	DL-management
DLME	DL-management entity (the local active instance of DL-management)
DLMS	DL-management service
DLS	DL-service
DLSAP	DL-service-access-point
DLSDU	DL-service-data-unit
E²PROM	Electrically erasable programmable read only memory
EoE	Ethernet tunneled over EtherCAT services
ESC	EtherCAT slave controller
FCS	Frame check sequence
FIFO	First-in first-out (queuing method)
FMMU	Fieldbus memory management unit
FoE	File access with EtherCAT services
FPRD	Configured address physical read
FPRW	Configured address physical read/write
FPWR	Configured address physical write
FRMW	Configured address physical read/multiple write
HDR	Header
ID	Identifier
IP	Internet protocol
LAN	Local area network
LRD	Logical memory read
LRW	Logical memory read/write
LWR	Logical memory write
MAC	Medium access control
MDI	Media-dependent interface (specified in ISO/IEC 8802-3)
MDX	Mailbox data exchange
MII	Media-independent interface (specified in ISO/IEC 8802-3)
PDI	Physical device interface (a set of elements that allows access to DL-services from the DL-user)
PDO	Process data object
Ph-	Physical layer (as a prefix)
PhE	Ph-entity (the local active instance of the physical layer)
PhL	Ph-layer
PHY	Physical layer device (specified in ISO/IEC 8802-3)
PNV	Publish network variable
OSI	Open systems interconnection
QoS	Quality of service
RAM	Random access memory
Rx	Receive
SDO	Service data object

SII	Slave information interface
SyncM	Synchronization manager
TCP	Transmission control protocol
Tx	Transmit
UDP	User datagram protocol
WKC	Working counter

3.5 Common conventions

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

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

Service primitives, used to represent service user/service provider interactions (see ISO/IEC 10731), convey parameters that indicate information available in the user/provider interaction.

This standard uses a tabular format to describe the component parameters of the DLS primitives. The parameters that apply to each group of DLS primitives are set out in tables throughout the remainder of this standard. Each table consists of up to six columns, containing the name of the service parameter and a column each for those primitives and parameter-transfer directions used by the DLS:

— the request primitive's input parameters;

— the indication primitive's output parameters;

— the response primitive's input parameters; and

— the confirm primitive's output parameters.

NOTE The request, indication, response and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

One parameter (or part of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive and parameter direction specified in the column:

M parameter is mandatory for the primitive.

U parameter is a user option, and may or may not be provided depending on the dynamic usage of the DLS-user. When not provided, a default value for the parameter is assumed.

C parameter is conditional upon other parameters or upon the environment of the DLS-user.

(blank) parameter is never present.

Some entries are further qualified by items in brackets. These may be a parameter-specific constraint:

(=) indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

In any particular interface, not all parameters need be explicitly stated. Some may be implicitly associated with the primitive.

In the diagrams which illustrate these interfaces, dashed lines indicate cause-and-effect or time-sequence relationships, and wavy lines indicate that events are roughly contemporaneous.

4 Data-link layer services and concepts

4.1 Operating principle

This standard describes 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, an EtherCAT 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 downstream microprocessor, but may consist of a large number of EtherCAT slave devices. These process the incoming Ethernet frames while they are in transit within the device, reading data from the Ethernet frame and/or inserting their own data into the frame before transferring the frame to the next slave device. The last slave device within the segment sends the fully processed Ethernet frame back in the reverse direction through the chain of devices, returning the collected information through the first slave device to the master, which receives it as an Ethernet response frame.

This procedure utilizes the full-duplex capability of Ethernet: both communication directions are operated independently with reading and writing by the slaves on the outbound path and only transmission-to-reception timing measurements on the inbound path as the Ethernet frame retraverses each intermediate slave device.

Full-duplex communication between a master device and an EtherCAT segment consisting of one or several slave devices may be established without using a switch.

4.2 Topology

The topology of a communication system is one of the crucial factors for the successful application in automation. The topology has significant influence on the cabling effort, diagnostic features, redundancy options and hot-plug-and-play features.

The star topology commonly used for Ethernet can lead to increased cabling effort and infrastructure cost. Particularly for automation applications, a line or tree topology is often preferable.

The slave node arrangement represents an open-loop bus. At the open end, the master device sends frames, either directly or via Ethernet switches; it receives them at the other end after they have been processed by each intervening device. Each Ethernet frame is relayed from the first node to the next one, and successively to each other node in series. The last node returns the Ethernet frame back to the master using the full-duplex capabilities of Ethernet. The resulting topology is a physical line.

Branches, which in principle are possible anywhere, can be used to enhance the line structure into a tree structure form. A tree structure supports very simple wiring; individual branches, for example, can branch into control cabinets or machine modules, while the main line runs from one module to the next. Branches are possible if a device has more than two ports. This standard allows up to two branching links in addition to the basic set of two series interfaces.

An Ethernet frame received on port *n* (*n* not zero) is forwarded to port *n*+1. If there is no port *n*+1, the Ethernet frame is forwarded to port 0. If no device is connected or the port is closed by the master, a request to send to that port will be processed as if the same data are received by this port (i.e. loop is closed).

4.3 Data-link layer overview

A single Ethernet frame can carry several EtherCAT DLPDUs which are packed into the Ethernet frame without gaps. Several nodes can be addressed individually by these DLPDUs. The Ethernet frame is terminated with the last EtherCAT DLPDU, except when the frame size is less than 64 octets, in which case the Ethernet frame is padded to 64 octets.

The EtherCAT DLPDU packing leads to better utilization of the Ethernet bandwidth than would separate Ethernet frames to and from each slave node. However, for e.g. a 2-channel digital

input node with just two bits of user data, the overhead of a single EtherCAT DLPDU can still be excessive.

Therefore slave nodes may also support logical address mapping. The process data can be inserted anywhere within a logical address space. If an EtherCAT DLPDU is sent that contains read or write services for a certain process image area located at the corresponding logical address, instead of addressing a particular node, the nodes insert the data at or extract the data from their appropriate place(s) within the process data, as noted in Figure 1.

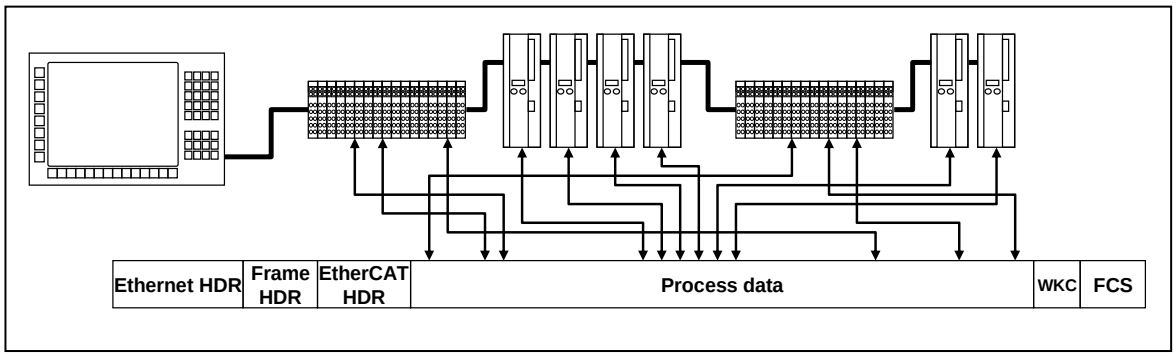

Figure 1 – Mapping of logical data in an Ethernet frame consisting of a single EtherCAT DLPDU

Each node that detects an address match with the process image inserts its data, so that many nodes can be addressed simultaneously with a single EtherCAT DLPDU. The master can assemble a completely sorted logical process image via a single EtherCAT DLPDU, independent of the physical wiring order of the slave devices.

Additional mapping is no longer required in the master, so that the process data can be transferred directly to one or more different control tasks. Each task can create its own process image and exchange it within its own timeframe. The physical order of the nodes is completely arbitrary and is only relevant during the first initialization phase.

The logical address space is 2^{32} octets (= 4 GB). Thus, an EtherCAT fieldbus can be considered to be a serial backplane for automation systems that enables connection to distributed process data for both large and very small automation devices. Using a standard Ethernet controller and standard Ethernet cables, a very large number of I/O channels can be connected to automation devices so that they can be accessed with high bandwidth, minimum delay and a near-optimum effective usable data rate. At the same time, devices such as fieldbus scanners can be connected as well, thus preserving existing technologies and standards.

4.4 Error detection overview

EtherCAT master and slave nodes (DLEs) check the Ethernet frame check sequence (FCS) to determine whether a frame is received correctly. Since one or several slaves may modify the frame during the transfer, the FCS is checked by each node on reception and recalculated during retransmission. If a slave detects a checksum error, the slave does not repair the FCS but flags the master by incrementing an error counter, so that the source of a single fault can be located precisely within the open-loop topology.

When reading data from or writing data to an EtherCAT DLPDU, the addressed slave increments a working counter (WKC) positioned at the end of the DLPDU. Slaves which are merely forwarding the DLPDU, but not extracting information from it or inserting information into it, do not modify the counter. By comparing the working counter with the expected number of accessing slave nodes, a master can check whether the expected number of nodes have processed the corresponding DLPDU.

4.5 Parameter and process data handling introduction

Industrial communication systems need to meet different requirements in terms of their data transmission characteristics. Parameter data can be transferred acyclically and in large quantities, usually in situations where the timing requirements are relatively non-critical and the transmission is triggered by the control system. Diagnostic data is also transferred acyclically

in an event-driven mode, 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. This international standard supports a variety of services and protocols to meet these differing requirements.

4.6 Node reference model

4.6.1 Mapping onto OSI Basic Reference Model

EtherCAT services are described using the principles, methodology and model of ISO/IEC 7498-1 (OSI). The OSI model provides a layered approach to communications standards, whereby the layers can be developed and modified independently. The EtherCAT specification defines functionality from top to bottom of a full OSI communications stack. Functions of the intermediate OSI layers, layers 3–6, are consolidated into either the EtherCAT data-link layer or the DL-user of the EtherCAT data-link layer. The EtherCAT data-link reference model is shown in Figure 2.

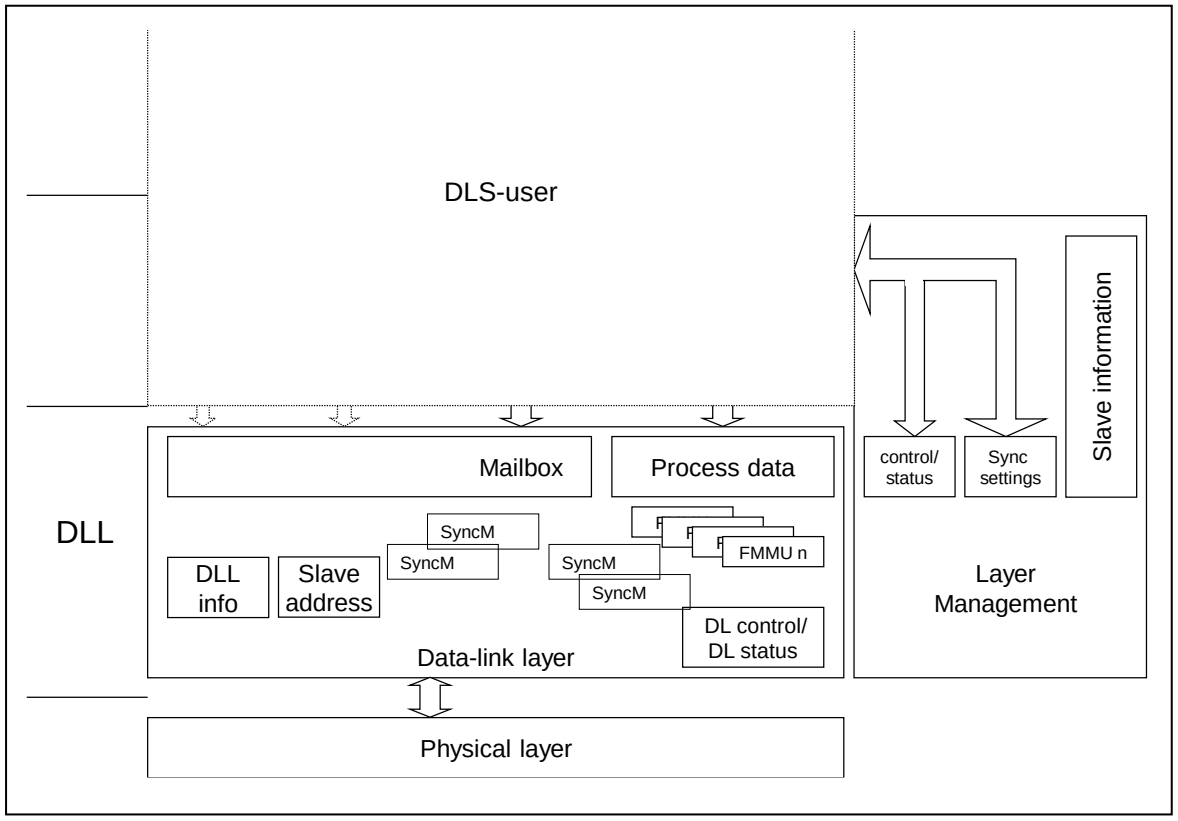

Figure 2 – EtherCAT data-link reference model

4.6.2 Data-link layer features

The data-link layer provides basic time-critical support for data communications among devices connected. 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 data is made available to the DL-user in physical memory, either in a mailbox configuration or within the process data section.

4.7 Operation overview

4.7.1 Relation to ISO/IEC 8802-3

This part specifies data-link layer services in addition to those specified in ISO/IEC 8802-3.

4.7.2 EtherCAT modes

4.7.2.1 Open mode

In the open mode, one or several EtherCAT segments may be connected to a standard switching device as shown in Figure 3. The first slave device within an EtherCAT segment then has an ISO/IEC 8802-3 MAC address representing the entire segment. This segment address slave device replaces the destination address field with the source address field and the source address field with its own MAC address within the Ethernet frame if the frame follows the coding rules of EtherCAT. If this type of frame is transported via UDP, this device will handle the source and destination IP addresses and the UDP source and destination port numbers in the same way as the MAC addresses in order to ensure that the response frame fully satisfies UDP/IP protocol standards. Additionally, this device protects the slaves within the segment against unauthorized access by master devices or generic Ethernet devices.

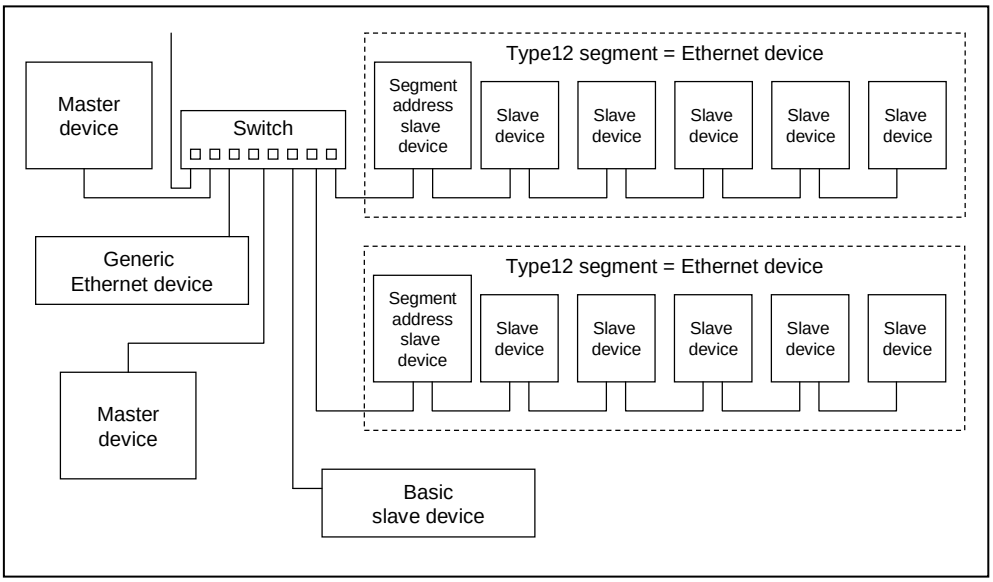

Figure 3 – EtherCAT segments in open mode

4.7.2.2 Direct mode

In the direct mode, one EtherCAT segment is connected to the standard Ethernet port of the controlling or master device as shown in Figure 4. The MAC address fields of the Ethernet frames are not checked.

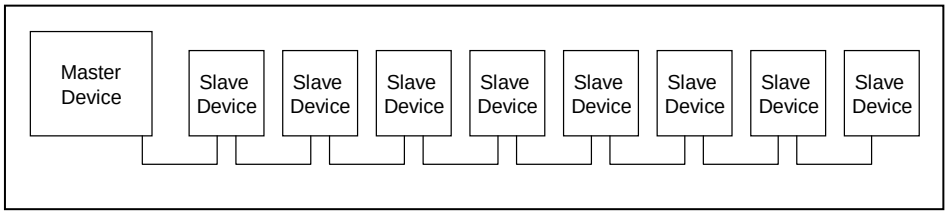

Figure 4 – EtherCAT segment in direct mode

4.7.3 Logical topology

In logical terms, the slave arrangement within an EtherCAT segment represents a bus connected as an open full-duplex loop. At the input end of the open loop, the master device inserts Ethernet frames, either directly or via standard Ethernet switches, and receives them at the output end of the open loop after they have been processed by all slave devices. All frames are relayed from the first slave device to the next one. The last slave device returns the frame

back through all the other slave devices to the master. The result is an open logical loop realized by consecutive segments of full-duplex physical lines.

Received Ethernet frames are processed octet by octet "on the fly" by the slave devices according to their physical sequence within the open loop structure. In this case, each slave device recognizes relevant commands and executes them accordingly while the frames (delayed by a constant time, typically below 1 µs) are forwarded to the next device in the open loop. Data extraction and insertion are performed by the data-link layer as the Ethernet frame transits the slave device, in a manner that is independent of the response times of any microprocessors within (or connected to) the slave device.

Full-duplex physical branches are possible in the EtherCAT segment at any location, since a branch does not break the logical loop. Branches can be used to build a flexible tree structure, thus permitting very simple wiring.

4.8 Addressing

4.8.1 Addressing overview

Different addressing modes are supported for slaves, as noted in Figure 5. The header within the EtherCAT DLPDU contains a 32-bit address, which is used for physical node addressing or logical addressing.

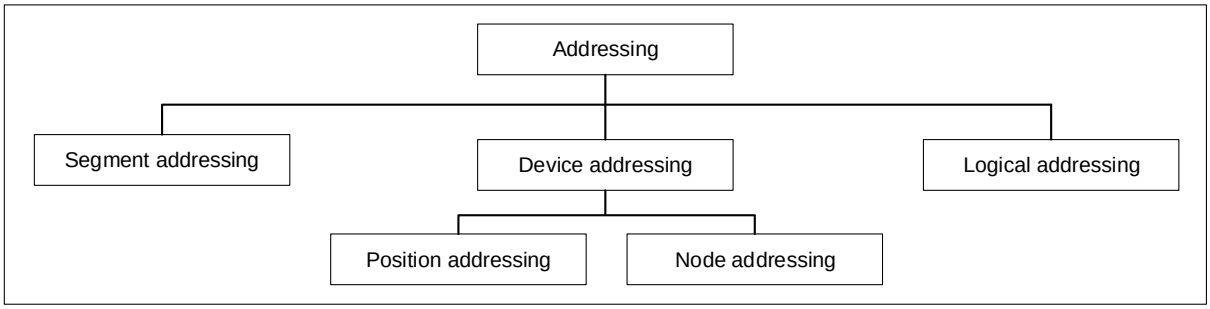

Figure 5 – Addressing mode overview

4.8.2 Segment addressing

MAC addresses according to ISO/IEC 8802-3 are used for segment addressing.

4.8.3 Device addressing

4.8.3.1 Structure of device addresses

With this address mode, a 32-bit address within each EtherCAT DLPDU is split into a 16-bit slave device address and a 16-bit physical address within the slave device, thus leading to 2^{16} slave device addresses, each with an associated 16-bit local address space. With device addressing, each EtherCAT DLPDU uniquely addresses one single slave device.

This mode is most suitable for transferring parameter data. There are two different device addressing mechanisms:

- position addressing;
- node addressing.

4.8.3.2 Position addressing

Position addressing is used to address each slave device via its physical position within the segment. Each slave device increments the 16-bit address field as the DLPDU transits the slave device; that device which receives a DLPDU with an address field of value 0 is the one being addressed. Due to the mechanism employed to update this address while transiting the node, the slave device address in position addressing is referred to as an *auto-increment address*.

EXAMPLE If the 10th slave device in the segment is to be addressed, the master device sends a DLPDU with position addressing with a start device address value of -9, which is incremented by one by each device which the DLPDU transits.

In practice, position addressing is used during a start-up phase, during which the master assigns configured node addresses to the slaves, after which they can be addressed irrespective of their physical position in the segment via use of those node addresses.

This topology-based addressing mechanism has the advantage that no slave node addresses need to be set manually at the slaves.

4.8.3.3 Node addressing

With node addressing, the slaves are addressed via configured node addresses assigned by the master during the data-link start-up phase. This ensures that, even if the segment topology is changed or devices are added or removed, the slave devices can still be addressed via the same configured address.

The slave device address for node addressing is referred to as *configured station address*.

4.8.4 Logical addressing

For logical addressing within a segment the entire 32-bit address field of each EtherCAT DLPDU is used as a single unstructured address. With logical addressing, slaves are not addressed individually, but instead a section of the segment-wide 4 GB logical address space is addressed. Any number of slaves may use the same or overlapping sections.

The data region address used in this mode is referred to as a *logical address*.

The logical addressing mode is particularly suitable for transferring and/or exchanging cyclic process data.

4.8.5 FMMU introduction

Fieldbus memory management units (FMMU) handle the local assignment of physical slave memory addresses to logical segment-wide addresses, as shown in Figure 6.

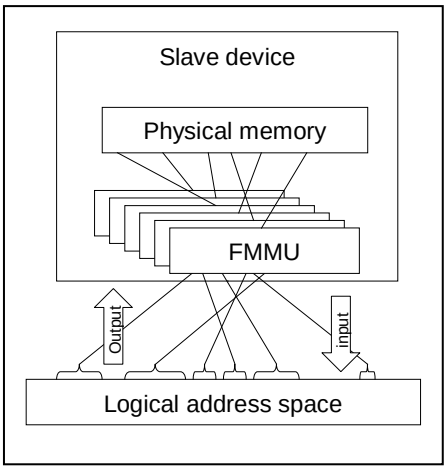

Figure 6 – Fieldbus memory management unit overview

Configuration of the FMMU entities is performed by the master device and transferred to the slave devices during the data-link start-up phase. For each FMMU entity, the following items are configured: a logical, bit-oriented start address, a physical memory start address, a bit length, and a type that specifies the direction of the mapping (input or output). Any data within the memory of a slave device can thus be mapped bit-wise to any logical address.

When an EtherCAT DLPDU with logical addressing is received, the slave device checks whether one of its FMMU entities shows an address match. If appropriate, it inserts data at the associated position of the data field into the DLPDU (input type) or extracts data from the associated position of the DLPDU (output type). DLPDUs can therefore be assembled flexibly and optimized to the requirements of the control application.

4.8.6 Sync manager introduction

The sync manager (SyncM) controls the access to the DLS-user memory. Each SyncM channel defines a consistent area of the DLS-user memory.

4.9 Slave classification

4.9.1 Full slave

There is a difference between full slaves, which support all addressing modes, and basic slaves, which support only a subset of the addressing modes. Master devices may support the basic slave functionality to allow for direct communication with another master device. Slave devices should support the full slave functionality.

A full slave supports

- logical addressing;
- position addressing; and
- node addressing.

Thus full slave devices need both an FMMU and address auto-increment functionality.

Full slaves may support segment addressing. Full slaves that support segment addressing are called segment address slave devices.

Only full slaves can be connected within an EtherCAT segment.

4.9.2 Basic slave

Basic slave devices support node addressing and segment addressing.

4.10 Structure of the communication layer in the slave

The attributes are related to the physical memory of a slave, which can be read or written from the master. The physical memory consists of registers and DL-user memory. The register area contains information for configuration, management and device identification in the DLL. The use of the DL-user memory is defined by the DL-user. Figure 7 shows the outline of the interactions between DL-user and DLL and between DLL and PhL.

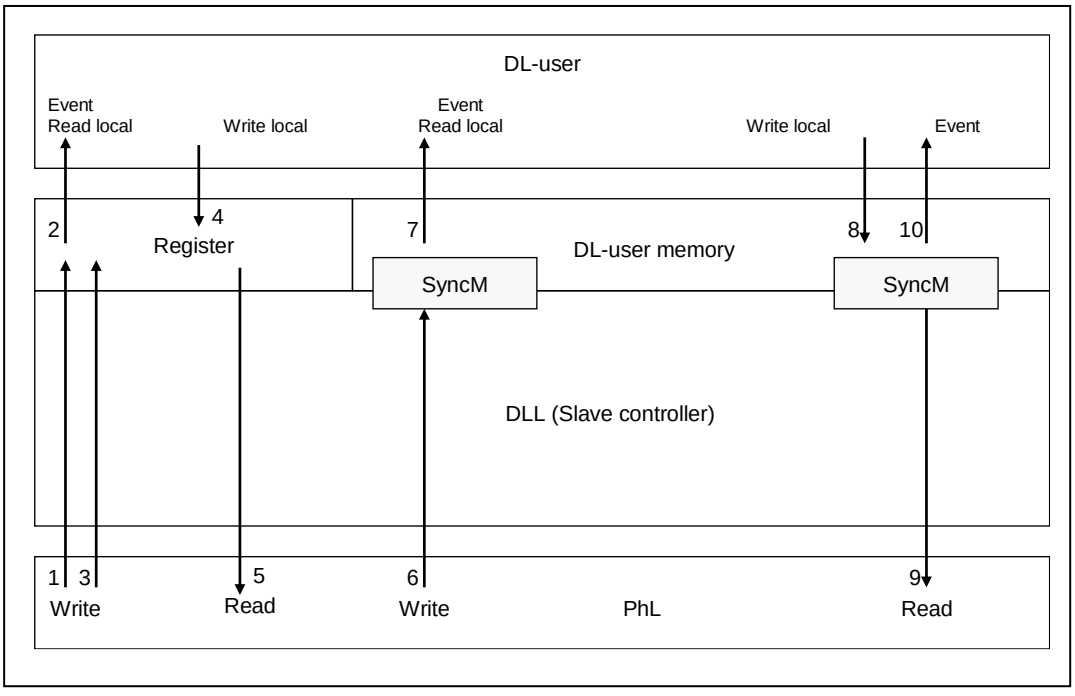

Figure 7 – Layering of communication

A DL write service to the register area (1) may (depending on the written register) result in an event indication primitive to the DL-User, followed by a read local request primitive from the DL-user to get the written value (2). Otherwise, the DL write service will only access the register area without informing the DL-user (3). The DL-user can read the register area with a read local primitive at any time.

The DL-user will set up the register with write register local and update it if needed and possible (4). A DL read service to the register area will only access the register area without informing the DL-user (5).

The DL-user memory area access is coordinated by the sync manager. Access without the sync manager can be done in a similar way as the register access, but consistency constraints and the lack of events indicating changes caused by the master may limit this method of use. The description of DL-user memory access assumes the use of the sync manager.

A DL write service to the DL-user memory area (6) will result in an event indication primitive to the DL-user, followed by a read local request primitive from the DL-user to get the written values (7) which can issue an event indication primitive to the PhL.

The DL-user will write the DL-user memory area with a write local request primitive (8) which can issue an event indication primitive to the PhL. The DL-user memory area will be read by the master with a DL read service (9) which will issue an event indication primitive to the DL-user (10) to indicate that the DL-user memory area has been read and can be written by the DL-user again.

A slave responds to all read and write requests, and may respond to combined read/write requests.

5 Communication services

5.1 Overview

Services are described from the point of view of the master. The execution of the service within the slave is described in Clause 6.

The data-link layer specifies services for reading, writing and exchanging (reading followed immediately by overwriting) data to and from physical memory within the slaves.

NOTE For simplification, the expression "reads memory" is used instead of "reads data from physical memory". Equivalently, the expression "writes memory" is used instead of "writes data to physical memory".

With the read service a master reads registers or DL-user memory from one or many slaves.

The basic service procedure of all variants of read service is the same except for broadcast read. The addressed unit will copy the data in the data parameter. With broadcast service, the slave will execute a bitwise-OR operation of the parameter data with the memory or register data.

If there is only one slave connected to a master, the service procedure is executed according to the client server model. If several slaves are connected (always in series), the invocation of the service procedure is handled in such a way that the output of one slave serves as the input to the next slave.

The service procedures are similar to the procedures defined in IEEE 802.1D, but forwarding and processing are combined. As EtherCAT uses confirmed services instead of unconfirmed services as specified in IEEE 802.1D, the flow of information between service primitives is adapted to this situation. The master initiates a request service and receives a corresponding confirmation. Each slave receives an indication of the data it receives, while forwarding that data to the next slave after a possible update. Figure 8 shows this control flow.

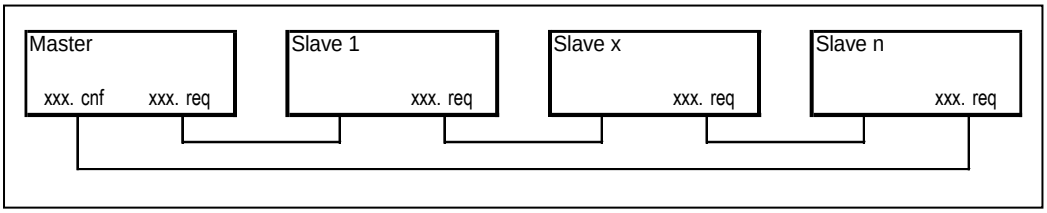

Figure 8 – Flow of EtherCAT service primitives

5.2 Read services

5.2.1 Overview

With the read services, a master reads data from the memory of one or many slaves.

5.2.2 Positional physical read (APRD)

With the APRD service, a master reads data from memory or register of one slave selected by the physical ordering of the slave in the segment. Table 1 shows the service primitives and parameters of the APRD service.

Table 1 – Auto-increment physical read (APRD)

DL-AUTOINCREMENT-PHYSICALREAD	Request	Confirm
Parameter name	input	output
Ordinal device number	M	M
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Ordinal device number

This parameter specifies the ordinal index of the addressed device in the wired communication chain. In the confirmation to the master, the number of transited slave devices is given.

Device data area

This parameter specifies the location in the physical memory of the slave where the data to be read is stored.

DLS-user data

On confirm this parameter, specifies the data read from the device if the access was valid at the addressed slave. Otherwise the value specified by the request is returned unchanged.

Working counter

This parameter is incremented if the data was successfully read.

5.2.3 Configured-address physical read (FPRD)

With the FPRD service, a master reads data from memory or register of one slave selected by the slave's configured station address. Table 2 shows the service primitives and parameters of the FPRD service.

Table 2 – Configured-addresse physical read (FPRD)

DL-CONFIGURED-PHYSICALREAD Parameter name	Request input	Confirm output
Configured device number	M	
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Configured device number

This parameter specifies the configured station address of the addressed device.

Device data area

This parameter specifies the location in the physical memory of the slave where the data to be read is stored.

DLS-user data

On confirm this parameter, specifies the data read from the device if the access was valid at the addressed slave. Otherwise the value specified by the request is returned unchanged.

Working counter

This parameter is incremented if the data was successfully read.

5.2.4 Broadcast read (BRD)

With the BRD service, a master reads data from a physical memory area or register, which will be a bitwise-OR between the incoming data and the selected object at all slaves. Table 3 shows the service primitives and parameters of the BRD service.

Table 3 – Broadcast read (BRD)

DL-BROADCAST-READ Parameter name	Request input	Confirm output
Broadcast address	M	M
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Broadcast address

This parameter is incremented at each slave.

Device data area

This parameter specifies the location in the physical memory where the data to be read is stored.

DLS-user data

On confirm this parameter, specifies the result of the bitwise-OR operation between the parameter data of the request and the selected object at the response.

Working counter

This parameter is incremented by all slaves which made the bitwise-OR of the requested data.

5.2.5 Logical read (LRD)

With the LRD service, a master reads data from the memory or a register of one or many slaves selected by a logical address. Table 4 shows the service primitives and parameters of the LRD service.

Table 4 – Logical read (LRD)

DL-LOGICAL-READ	Request	Confirm
Parameter name	input	output
Logical memory address	M	
DLS-user data		U
Working counter	M	M
NOTE 1 The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Logical memory address

This parameter specifies the start address in the logical memory where the data to be read is located.

DLS-user data

This parameter specifies the data that was read.

Working Counter

This parameter is incremented by all slaves which detect an address match of the requested logical memory area.

5.3 Write services

5.3.1 Overview

With the write services, a master writes data to a register or the memory of one or many slaves.

5.3.2 Positional physical write (APWR)

With the APWR service, a master writes to memory or register of one slave selected by the physical ordering of the slave in the segment. Table 5 shows the service primitives and parameters of the APWR service.

Table 5 – Auto-increment physical write (APWR)

DL-AUTOINCREMENT-PHYSICALWRITE	Request	Confirm
Parameter name	input	output
Ordinal device number	M	M
Device data area	M	
DLS-user data	U	
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Ordinal device number

This parameter specifies the ordinal index of the address device in the wired communications chain. In the confirmation to the master, the number of transited slave devices is given.

Device data area

This parameter specifies the location in the physical memory of the slave where the data to be written is stored.

DLS-user data

This parameter specifies the data to be written.

Working counter

This parameter is incremented if the data was successfully written.

5.3.3 Configured-address physical write (FPWR)

With the FPWR service, a master writes to memory or register of one slave selected by the slave's configured station address. Table 6 shows the service primitives and parameters of the FPWR service.

Table 6 – Configured-address physical write (FPWR)

DL-CONFIGURED-PHYSICALWRITE	Request	Confirm
Parameter name	input	output
Configured device number	M	
Device data area	M	
DLS-user data	U	
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Configured device number

This parameter specifies the configured station address of the addressed device.

Device data area

This parameter specifies the location in the physical memory of the slave where the data to be written is stored.

DLS-user data

This parameter specifies the data to be written.

Working counter

This parameter is incremented if the data was successfully written.

5.3.4 Broadcast write (BWR)

With the BWR service, a master writes to a physical memory area of all slaves. Table 7 shows the service primitives and parameters of the BWR service.

Table 7 – Broadcast write (BWR)

DL-BROADCAST-WRITE	Request	Confirm
Parameter name	input	output
Broadcast address	M	M
Device data area	M	
DLS-user data	U	
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Broadcast address

This parameter is incremented at each slave.

Device data area

This parameter specifies the location in the physical memory where the data to be written is stored.

DLS-user data

This parameter specifies the data to be written.

Working counter

This parameter is incremented by each slave that writes data to its physical memory.

5.3.5 Logical write (LWR)

With the LWR service, a master writes to memory or register of one or many slaves selected by a logical address. Table 8 shows the service primitives and parameters of the LWR service.

Table 8 – Logical write (LWR)

DL-LOGICAL-WRITE	Request	Confirm
Parameter name	input	output
Logical memory address	M	
DLS-user data	U	
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Logical memory address

This parameter specifies the start address in the logical memory where the data to be written is located.

DLS-user data

This parameter specifies the data to be written.

Working counter

This parameter is incremented by all slaves that detect an address match of the requested logical memory area.

5.4 Combined read/write services

5.4.1 Overview

For the combined read/write services, the rules for read and/or write of the addressed slave apply.

5.4.2 Positional physical read/write (APRW)

With the APRW service, a master reads out memory or register of one slave selected by the physical ordering of the slave in the segment and writes data to memory or register of the same slave. Table 9 shows the service primitives and parameters of the APRW service.

Table 9 – Auto-increment physical read/write (APRW)

DL-AUTOINCREMENT-PHYSICALREADWRITE	Request	Confirm
Parameter name	input	output
Ordinal device number	M	M
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Ordinal device number

This parameter specifies the ordinal index of the addressed device in the wired communication chain. In the confirmation to the master, the number of transited slave devices is given.

Device data area

This parameter specifies the location in the physical memory of the slave where data to be read and written is stored.

DLS-user data

This parameter specifies the data to be written, or the data that was read.

Working counter

This parameter is incremented if the data was successfully written or read.

5.4.3 Configured-address physical read/write (FPRW)

With the FPRW service, a master reads from memory or register of one slave selected by the slave's configured station address and writes data to the same object. Table 10 shows the service primitives and parameters of the FPRW service.

Table 10 – Configured-address physical read/write (FPRW)

DL-CONFIGURED-PHYSICALREADWRITE	Request	Confirm
Parameter name	input	output
Configured device number	M	
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Configured device number

This parameter specifies the configured station address of the addressed device.

Device data area

This parameter specifies the location in the physical memory of the slave where the data to be read and written is stored.

DLS-user data

This parameter specifies the data to be written, or the data that was read.

Working counter

This parameter is incremented if the data was successfully written or read.

5.4.4 Broadcast read/write (BRW)

With the BRW service, a master reads a physical memory area or register, which will be bitwise-OR by all slaves and writes data to a physical memory area or register collected from all previous slaves. Table 11 shows the service primitives and parameters of the BRW service.

Table 11 – Broadcast read/write (BRW)

DL-BROADCAST-READWRITE	Request	Confirm
Parameter name	input	Output
Broadcast address	M	M
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Broadcast address

This parameter is incremented at each slave.

Device data area

This parameter specifies the location in the physical memory where the data to be read and written is stored.

DLS-user data

This parameter specifies the data to be written to the device, or the result of the bitwise-OR operation of the data that was read from each device.

Working counter

This parameter is incremented by all slaves which made the bitwise-OR of the requested data or wrote data into their physical memory.

5.4.5 Logical read/write (LRW)

With the LRW service, a master writes and reads memory to one or many slaves selected by a logical address. Table 12 shows the service primitives and parameters of the LRW service.

Table 12 – Logical read/write (LRW)

DL-LOGICAL-READWRITE	Request	Confirm
Parameter name	input	output
Logical memory address	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Logical memory address

This parameter specifies the start address in the logical memory where the data to be read or written is located.

DLS-user data

This parameter specifies the data to be written, or the data that was read.

Working counter

This parameter is incremented if data was successfully written or read.

5.4.6 Positional physical read / multiple write (ARMW)

With the ARMW service, a master reads data out of memory or register of one slave selected by the physical ordering of the slave in the segment and writes the value of the parameter data to the same memory or register of all other slaves following. Table 13 shows the service primitives and parameters of the ARMW service.

Table 13 – Auto-increment physical read / multiple write (ARMW)

DL-AUTOINCREMENT-READMULTIPLEWRITE	Request	Confirm
Parameter name	input	output
Ordinal device number	M	M
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Ordinal device number

This parameter specifies the ordinal index of the device in the wired communications chain that executes the read action. In the confirmation to the master, the number of transited slave devices is given.

Device data area

This parameter specifies the location in the physical memory of the slave where data to be read and written is stored.

DLS-user data

This parameter specifies the data to be written, or the data that was read.

Working counter

This parameter is incremented if the data was successfully written or read.

5.4.7 Configured-address physical read / multiple write (FRMW)

With the FRMW service, a master reads from memory or register of one slave selected by the slave's configured station address and writes data to the same object of all other slaves. Table 14 shows the service primitives and parameters of the FRMW service.

Table 14 – Configured-address physical read / multiple write (FRMW)

DL-CONFIGURED-READMULTIPLEWRITE	Request	Confirm
Parameter name	input	output
Configured device number	M	
Device data area	M	
DLS-user data	U	U
Working counter	M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Configured device number

This parameter specifies the configured station address of the device that is to perform the read operation.

Device data area

This parameter specifies the location in the physical memory of the slave where data to be read and written is stored.

DLS-user data

This parameter specifies the data to be written, or the data that was read.

Working counter

This parameter is incremented if the data was successfully written or read.

5.5 Network services

5.5.1 Overview

Network variable services are described from the point of publisher. The data-link layer specifies services for publishing. This service is dedicated for the communication between masters or between master and standard Ethernet devices.

5.5.2 Publisher network variables (PNV)

With the PNV service, a master provides data to one or many other stations (master or slaves). The primary addressing is done by the destination MAC address (group address/ individual address). The stations receiving an indication will pass the data to the DL-user. Table 15 shows the service primitives and parameters of the PNV service.

Table 15 – Publisher network variable (PNV)

DL-PROVIDE-NETWORKVARIABLE	Request	Indication
Parameter name	input	output
Publisher ID	M	M (=)
Cycle	M	M (=)
List of network variables	M	M (=)

Parameter description

Publisher ID

This parameter specifies the values of the identification octet string.

Cycle

This parameter represents a numeric identifier of the devices' cycle and may be used to detect new values.

List of network variables

This parameter specifies a list of network variables. Each element within the list specifies:

Index

This parameter specifies the unique identifier within the provider of the network variable.

Hash value

This parameter specifies a hash value of the variable structure description of the network variables. The hash algorithm is provider-specific.

Data

This parameter specifies the values of the publisher data.

5.6 Mailbox

5.6.1 Overview

The mailbox works in both directions – from the master to a slave and from a slave to the master. It supports full-duplex, independent communication in both directions and multiple DL-user protocols. Slave to slave communication is managed by the master, operating as router. The mailbox header contains an address field that allows the master to redirect services.

The mailbox uses the two sync manager channels, one for each direction (e.g. sync manager channel 0 from the master to the slave and sync manager channel 1 from the slave to the master). The sync manager channels configured as mailbox prevent the other side from an overrun. Normally the mailbox communication is non-cyclic and addresses a single slave. Therefore, physical addressing without the need for an FMMU is used instead of logical addressing.

5.6.1.1 Communication from master to slave

After the master issues a mailbox command to a slave, the master has to check the working counter of the reply. If the working counter did not increment (normally because the slave has not completely read the last command) or there is no response within the time limit, the master has to retransmit the mailbox command. Further error recovery is the responsibility of higher-layer protocols.

5.6.1.2 Communication from slave to master

The master has to determine that a slave has filled the sync manager with a mailbox command and has to send an appropriate read command as quickly as possible.

There are different ways to determine that a slave has filled its sync manager. A clever solution is to configure the “written bit” of the configuration header of sync manager 1 to a logical address and to read this bit cyclically. Using a logical address enables the possibility to read the bits from several slaves collectively and to configure each slave on an individual bit address. The drawback of this solution is that one FMMU per slave is needed.

Another solution is to simply poll the sync manager data area. The working counter of that read command will only be incremented once if the slave has filled the area with a new command.

The master has to check the working counter of the reply to the mailbox command issued to a slave. If the working counter did not increment (normally because the slave has not completely read the last command) or there is no response within the time limit, the master has to toggle the retry parameter in the sync manager area. With a toggled retry parameter, the slave has to put the last read data in the mailbox. Further error recovery is the responsibility of higher-layer protocols.

The primitives of the mailbox services are mapped to the DL-user memory primitives at the slave:

Mailbox write	event, read local
Mailbox read update	write local
Mailbox read	event

Figure 9 shows the primitives between master and slave in case of a successful mailbox write sequence.

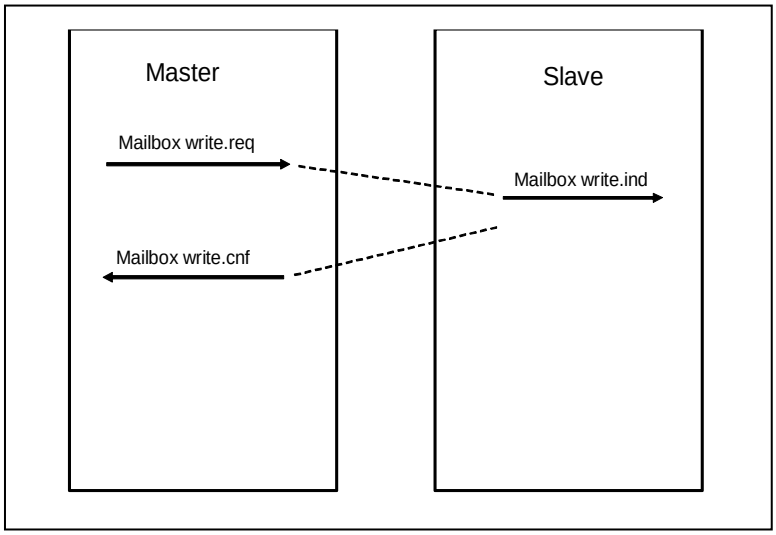

Figure 9 – Successful mailbox write sequence

Figure 10 shows the primitives between master and slave in case of a successful mailbox read sequence.

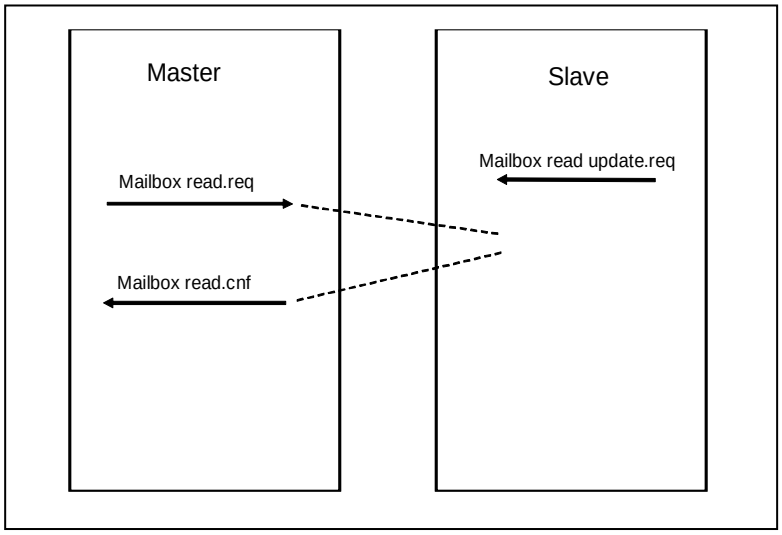

Figure 10 – Successful mailbox read sequence

5.6.2 Mailbox data transmission services

5.6.2.1 Mailbox write

The mailbox Write service as specified in Table 16 is based on writing (transmission from master to slave) memory to get an acknowledged transmission of data.

Table 16 – Mailbox write

DL-MAILBOX-WRITE Parameter name	Request input	Indication output	Confirm output
D_address	M		
MBX	M		
S_address	M	M (=)	
Channel	M	M (=)	
Priority	M	M (=)	
Cnt	M	M (=)	
Type	M	M (=)	
DLS-user data	U	U (=)	
DL-status			M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.			

Parameter description

D_address

This parameter specifies the node address of the destination node to allow slave-to-slave communication or communication beyond network boundaries using a virtual address.

MBX

This parameter specifies the mailbox.

S_address

This parameter specifies the station address of the source station to allow slave-to-slave communication or communication beyond network boundaries using a virtual address.

Channel

This parameter specifies the communication channel.

Priority

This parameter specifies a communication priority.

Type

This parameter specifies the protocol type of the used mailbox service.

Cnt

This parameter specifies a service counter

DLS-user data

This parameter specifies the data to be written or that was written.

DL-status

This parameter specifies the result of the operation.

5.6.2.2 Mailbox read update

The mailbox read update service as specified in Table 17 is based on a local write to memory. The update buffer has to be retained as long as a repeated operation is possible.

Table 17 – Mailbox read update

DL-MAILBOX-READUPD Parameter name	Request input	Confirm output
D_address	M	
Channel	M	
Priority	M	
Cnt	M	
Type	M	
DLS-user data	U	
DL-status		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

D_address

This parameter specifies the station address of the destination station to allow slave-to-slave communication or communication beyond network boundaries using a virtual address.

Channel

This parameter specifies the communication channel.

Priority

This parameter specifies a communication priority.

Type

This parameter specifies the protocol type of the used mailbox service.

Cnt

This parameter specifies a service counter

DLS-user data

This parameter specifies the data that was read.

DL-status

This parameter specifies the result of the operation.

5.6.2.3 Mailbox read

The mailbox read service as specified in Table 18 is based on reading (transmission from slave to master) memory to get an acknowledged transmission of data.

Table 18 – Mailbox read

DL-MAILBOX-READ Parameter name	Request input	Indication output	Confirm output
S_address	M		
MBX	M		
D_address			C
Channel			C
Priority			C
Type			C
Cnt			C
DLS-user data			C
DL-status		M	M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.			

Parameter description

S_address

This parameter specifies the station address of the source station.

MBX

This parameter specifies the mailbox.

D_address

This parameter specifies the station address of the destination station to allow slave-to-slave communication or communication beyond network boundaries using a virtual address.

Channel

This parameter specifies the communication channel.

Priority

This parameter specifies a communication priority.

Type

This parameter specifies the protocol type of the used mailbox service.

Cnt

This parameter specifies a service counter

DLS-user data

This parameter specifies the data that was read.

DL-status

This parameter specifies the result of the operation.

6 Local interactions

6.1 Read local

With this function, the DL-user reads data from a memory area. Table 19 shows the primitives and parameters of this function.

Table 19 – Read local

DL-READ LOCAL	Request	Confirm
Parameter name	input	output
Memory area	M	
DLS-user data		U
DL-status		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Memory area

This parameter specifies the memory area to be read.

DLS-user data

This parameter specifies the data that was read from the specified memory area.

DL-status

This parameter specifies the result of the operation.

6.2 Write local

With this function, the DL-user writes data to a memory area. Table 20 shows the primitives and parameters of this function.

Table 20 – Write local

DL-WRITE LOCAL	Request	Confirm
Parameter name	input	output
Memory area	M	
DLS-user data	U	
DL-status		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.		

Parameter description

Memory area

This parameter specifies the memory area to be written.

DLS-user data

This parameter specifies the data to be written to the specified memory area.

DL-status

This parameter specifies the result of the operation.

6.3 Event local

With this function, the DL-user gets an indication of an event. Table 21 shows the primitives and parameters of this function.

Table 21 – Event local

DL-EVENT LOCAL	Indication
Parameter name	output
Sync manager	C
Type	C

Parameter description

Sync manager

This parameter indicates the sync manager channel

Type

This parameter indicates the type of the event (read or write).

Bibliography

IEC 61158-1:2013, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-2, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-12, *Industrial communication networks – Fieldbus specifications – Part 3-12: Data-link layer service definition – Type 12 elements*

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

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

IEC 61158-6-12, *Industrial communication networks – Fieldbus specifications – Part 6-12: Application layer protocol specification – Type 12 elements*

IEC 61588, *Precision clock synchronization protocol for networked measurement and control system*

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

ISO/IEC/TR 8802-1, *Information technology —Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements —Part 1: Overview of Local Area Network Standards*

IEEE 802.1Q, *IEEE Standard for Local and metropolitan area networks – Virtual Bridged Local Area Networks;* available at <http://www.ieee.org>

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

IETF RFC 791, *Internet Protocol*; available at http://www.ietf.org

ETG.1000.2 *EtherCAT Specification Part 2: Physical layer specification and service definition*

ETG.1000.3 *EtherCAT Specification Part 3: Data Link Layer service definition*

ETG.1000.4 *EtherCAT Specification Part 4: Data-link layer protocol specification*

ETG.1000.5 *EtherCAT Specification Part 5: Application layer service definition*

ETG.1000.6 *EtherCAT Specification Part 6: Application layer protocol specification*

ETG.1100 *EtherCAT Specification Communication profiles*

———————