

# Communications Manual

MC 5010

MC 5005

MC 5004

MC 5004 P STO

MCS

MC 3603

**EtherCAT**<sup>®</sup>  
Technology Group



## Imprint

---

Version:  
6th edition, 30.08.2021

Copyright  
by Dr. Fritz Faulhaber GmbH & Co. KG  
Daimlerstr. 23 / 25 · 71101 Schönaich

All rights reserved, including those to the translation.  
No part of this description may be duplicated, reproduced,  
stored in an information system or processed or  
transferred in any other form without prior express written  
permission of Dr. Fritz Faulhaber GmbH & Co. KG.

This document has been prepared with care.  
Dr. Fritz Faulhaber GmbH & Co. KG cannot accept any  
liability for any errors in this document or for the  
consequences of such errors. Equally, no liability can be  
accepted for direct or consequential damages resulting  
from improper use of the equipment.

The relevant regulations regarding safety engineering  
and interference suppression as well as the requirements  
specified in this document are to be noted and followed  
when using the software.

Subject to change without notice.

The respective current version of this technical manual is  
available on FAULHABER's internet site:  
[www.faulhaber.com](http://www.faulhaber.com)

# Content

<b>1</b>	<b>About this document</b>	<b>5</b>
1.1	Validity of this document	5
1.2	Associated documents	5
1.3	Using this document	5
1.4	List of abbreviations	6
1.5	Symbols and designations	7
<b>2</b>	<b>Overview</b>	<b>8</b>
2.1	Basic structure of an EtherCAT device	8
2.2	FAULHABER Motion Manager	9
2.3	Pre-conditions for communication (Physical Layer)	10
2.4	ESI file	11
2.5	Identification of a slave	11
<b>3</b>	<b>EtherCAT communication</b>	<b>12</b>
3.1	Introduction	12
3.2	Data Link Layer	12
3.2.1	EtherCAT frames and datagrams	13
3.2.2	SyncManager management	14
3.2.3	Addressing	15
3.2.4	Interfaces to the Application Layer	15
3.3	Application Layer	16
3.4	PDO (Process Data Object)	17
3.4.1	PDO configuration	17
3.4.2	PDO mapping in the standard configuration	17
3.5	SDO (Service Data Object)	18
3.5.1	SDO error description	19
3.6	Emergency object (error message)	20
3.7	Synchronization	22
3.7.1	Synchronization via distributed clocks (DC-Sync)	22
3.7.2	Synchronization via a SyncManager event (SM-Sync)	23
3.8	Layer management	24
3.8.1	Controlling the EtherCAT state machine	24
3.8.2	Slave Information Interface (SII)	25
3.9	Entries in the object dictionary	25
3.10	Error handling	26
3.10.1	Device faults	26
3.10.2	Communication error	27
3.10.2.1	Checking EtherCAT frame entries for errors	27
3.10.2.2	Error response	28
3.10.2.3	Analysis of the network traffic	29
3.10.2.4	EtherCAT AL status codes and troubleshooting	29
3.11	Saving and restoring parameters	31
3.11.1	Save parameters	31
3.11.2	Restoring settings	32
3.11.3	Changing the parameter set	32

## Content

---

<b>4</b>	<b>Trace recorder .....</b>	<b>35</b>
4.1	Trace settings .....	35
4.2	Reading the trace buffer .....	37
4.3	Typical execution of the trace function .....	38
<b>5</b>	<b>Parameter description .....</b>	<b>39</b>
5.1	Communication objects acc. to CiA 301 .....	39
5.2	Manufacturer-specific objects .....	47

## About this document

# 1 About this document

## 1.1 Validity of this document

This document describes:

- Communication with the drive via EtherCAT
- Basic services provided by the Communication structure
- Methods for accessing the parameters
- Drive from the viewpoint of the communication system

This document is intended for software developers with EtherCAT experience, and for EtherCAT project engineers.

All data in this document relate to the standard versions of the drives. Changes relating to customer-specific versions can be found in the corresponding data sheet.

All data in this document relate to the firmware revision J.

## 1.2 Associated documents

For certain actions during commissioning and operation of FAULHABER products additional information from the following manuals is useful:

Manual	Description
Motion Manager 6	Operating instructions for FAULHABER Motion Manager PC software
Quick start guide	Description of the first steps for commissioning and operation of FAULHABER Motion Controllers
Drive functions	Description of the operating modes and functions of the drive
Technical manual	Instructions for installation and use of the FAULHABER Motion Controller
CiA 301	CANopen application layer and communication profile
CiA 402	CANopen device profile for drives and motion control

These manuals can be downloaded in pdf format from the web page [www.faulhaber.com/manuals](http://www.faulhaber.com/manuals)

## 1.3 Using this document

- ▶ Read the document carefully before undertaking configuration.
- ▶ Retain the document throughout the entire working life of the product.
- ▶ Keep the document accessible to the operating personnel at all times.
- ▶ Pass the document on to any subsequent owner or user of the product.

## About this document

### 1.4 List of abbreviations

Abbreviation	Meaning
AL	Application Layer
Attr.	Attribute
CAN	Controller Area Network
CSP	Cyclic Synchronous Position
CSV	Comma-Separated Values
DC	Distributed Clocks
DL	Data Link Layer
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMCY	Emergency
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information
ETG	EtherCAT Technology Group
EtherCAT	Ethernet for Control Automation Technology
FCS	Frame Check Sequence
FMMU	Fieldbus Memory Management Unit
HB	High Byte
HHB	Higher High Byte
HLB	Higher Low Byte
LB	Low Byte
LHB	Lower High Byte
LLB	Lower Low Byte
LSB	Least Significant Byte
LSS	Layer Setting Service
MSB	Most Significant Byte
OD	Object dictionary
PDO	Process Data Object
PP	Profile Position
PV	Profile Velocity
ro	read only
RTR	Remote Request
rw	read-write
RxPDO	Receive Process Data Object (PDO received from the drive)
SDO	Service Data Object
SII	Slave Information Interface
PLC	Programmable Logic Controller
Sxx	Data type signed (negative and positive numbers) with bit size xx
TxPDO	Transmit Process Data Object (PDO sent from the drive)
Uxx	Data type unsigned (positive numbers) with bit size xx

## About this document

---

### 1.5 Symbols and designations



#### NOTICE!

Risk of damage.


- ▶ Measures for avoidance




Instructions for understanding or optimizing the operational procedures

- ✓ Pre-requirement for a requested action

1. First step for a requested action

-  Result of a step

2. Second step of a requested action

-  Result of an action

- ▶ Request for a single-step action

## 2 Overview

EtherCAT is a registered trade mark and patented technology licensed by Beckhoff Automation GmbH, Germany.

### 2.1 Basic structure of an EtherCAT device

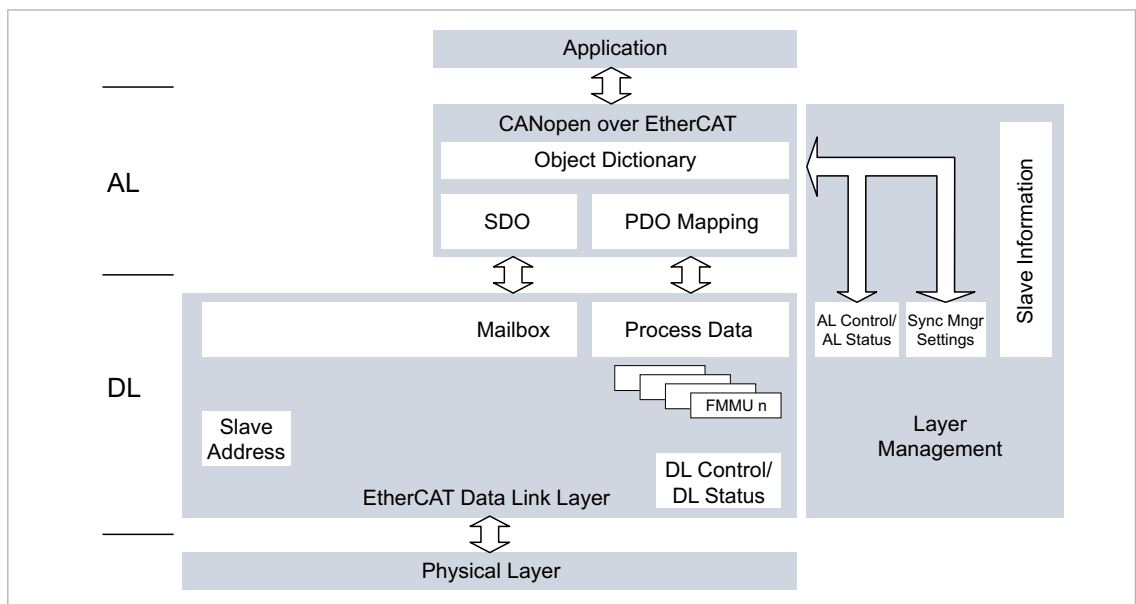


Fig. 1: Basic structure of an EtherCAT device

#### Physical Layer

The EtherCAT Physical Layer is structured according to IEEE 802.3, the specification for the Ethernet, with the standard 100Base-TX. It represents the link between the EtherCAT master and the EtherCAT slaves. The Physical Layer exchanges data packets with the Data Link Layer, and encodes / decodes these data packets by adding or removing the Framing Information.

#### Data Link Layer

As the EtherCAT frame data passes through, the Data Link Layer extracts data from it or inserts data into it. It also checks the EtherCAT frame for completeness. In so doing, the Data Link Layer complies with the rules that are saved in the Data Link Layer parameters. The data is made available in the respective memory sections of the EtherCAT slave, either as mailbox data or as process data (see chap. 3.2, p. 12).

#### Application Layer

The Application Layer contains all the services and objects necessary for communication between the Data Layer and the drive. The services are configured based on CANopen (see chap. 3.2, p. 12).

#### Application

The application part contains drive functions corresponding to CiA 402. The drive functions read parameters from the object dictionary, obtain the set-points from the object dictionary and return actual values. The parameters from the object dictionary determine the behavior of the drive.



**i** No further details of the application part are given in this document. The communication with the drive and the associated operating modes are described in the separate “Drives Functions” manual.

## 2.2 FAULHABER Motion Manager

We recommend that the first commissioning of a FAULHABER drive is performed using the “FAULHABER Motion Manager” software via the USB port or the serial COM port of the Motion Controller (depending on which port is available).

**i** If multiple interfaces are used simultaneously, impermissible transitional states may arise.

Before starting configuration of the FAULHABER drive via the USB port or RS232 port, disconnect the Motion Controller from the EtherCAT network.

The FAULHABER Motion Manager enables simple access to the settings and parameters of the connected motor controllers. The graphical user interface allows configurations to be read, changed and reloaded. Individual commands or complete parameter sets and program sequences can be input and loaded to the controller.

Wizard functions support the user when commissioning the drive controllers. The wizard functions are arranged on the user interface in the sequence they are normally used:

- Connection wizard: Supports the user in setting up the connection to the connected controller
- Motor wizard: Supports the user in adapting an external controller to the connected motor by selecting the respective FAULHABER motor
- Controller setting wizard: Supports the user in optimizing the controller parameters.

The software can be downloaded free of charge from the FAULHABER website.

**i** We recommend always using the latest version of the FAULHABER Motion Manager.

The FAULHABER Motion Manager is described in the separate “Motion Manager 6” manual. The contents of the manual are also available as context-sensitive online help within the FAULHABER Motion Manager.

## 2.3 Pre-conditions for communication (Physical Layer)

**i** Ethernet patch cables or crossover cables of category 5e (Cat5e to EN 50288) or higher up to a maximum length of 100 m can be used as network connection cables.

Never use EtherCAT and standard Ethernet alongside each other in a physical network. Such use can impair communications.

**i** If multiple interfaces are used simultaneously, impermissible transitional states may arise.

Before connecting the Motion Controller into the EtherCAT network, make sure that no other interfaces (such as USB, RS232) are connected.

1. Connect the controller to a power supply (supply at least for the electronics).
2. Connect the EtherCAT IN port to the master side port (see Fig. 2).
3. If multiple controllers are in use, connect each EtherCAT OUT port to the EtherCAT IN port of the next controller.
  - ↗ The EtherCAT OUT port of the last controller (slave) in the chain remains free. A telegram coming from the master passes through all the slaves and is then sent back to the master using the same cable.
4. Switch on the power.
5. Establish a connection via the configuration application (see chap. 2.2, p. 9).
6. Provide EtherCAT slave information (see chap. 2.4, p. 11).

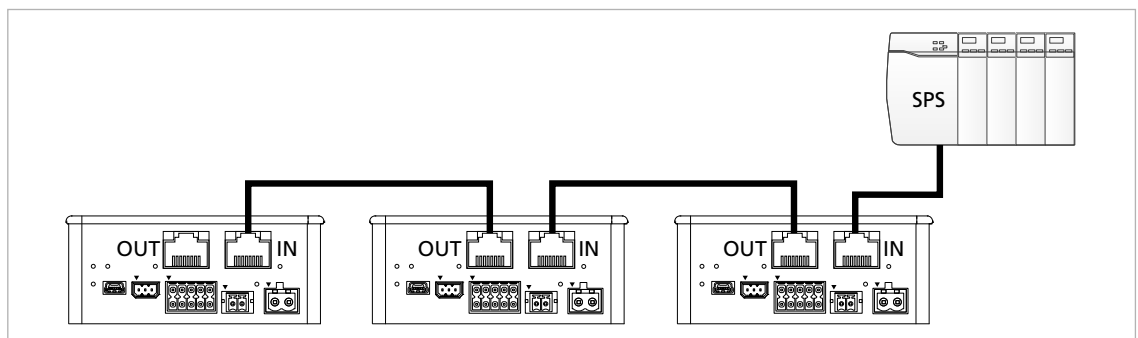


Fig. 2: Connection to the EtherCAT network

After switching on and initializing, the Motion Controller is at first in the *Init.* state. In order to be able to perform drive functions, the Motion Controller must be brought into the *Operational* state.

### 2.4 ESI file

The ESI file (EtherCAT Slave Information) contains information about the connected drive and its behavior. This information is required to enable the EtherCAT master to communicate with the slave.

The ESI file for the FAULHABER Motion Controller is held at the following places:

- As an XML file, which is stored in a subdirectory of the Motion Manager installation directory
- In a slightly simplified form on the EtherCAT EEPROM of the Motion Controller (Slave Information Interface, see chap. 3.8.2, p. 25)

The appropriately configured EtherCAT master can read the information from the ESI file. The master compares the drives found in the network with the ESI files available to it. If the manufacturer's number (0147), the product code and where applicable the revision number match, the ESI file for the drive has been found, and the master can configure the drive with the settings saved in the ESI file. Multiple revision numbers can be entered into an ESI file, to cater for multiple versions of the firmware.

#### Cycle time


The ESI file also contains the *Cycle Time* entry with the *AdaptAutomatically* attribute, which operates according to ETG2000 so that the EtherCAT master enters the cycle time at this point.

If despite this setting the EtherCAT master fails to enter the cycle time automatically, the entry must be modified manually. To do this, the default value **0** must be replaced with the cycle time of the master in nanoseconds. For a cycle time of 4 ms the value **4000000** must therefore be entered.

### 2.5 Identification of a slave

The EtherCAT master on a network has the following capabilities for identification of a slave:

- Identification via the position number:  
Due to its position within the logical Ethernet segment, each slave has a number by which it can be identified. The numbering is in ascending order starting with the EtherCAT master (the 1st slave after the EtherCAT master has the number 1, the slave that follows has the number 2, and so on).
- Identification via the Explicit Device ID:  
During the configuration phase, the user sets the content of object 2400.08 (Explicit Device ID) to any value and saves it with the SAVE command in the application EEPROM (see chap. 3.11, p. 31). During operation, the EtherCAT master reads this ID by means of EtherCAT mechanisms and compares it with a previously saved version. This ensures that swapped-out devices or incorrectly connected cables are detected.

 The FAULHABER Motion Manager offers a user-friendly facility for inputting and saving the Explicit Device ID.

# EtherCAT communication

## 3 EtherCAT communication

### 3.1 Introduction

#### EtherCAT

EtherCAT is an Ethernet-based communication technology. An EtherCAT master is required for communication using EtherCAT. The EtherCAT master controls the network and the communication with the connected EtherCAT slaves. More than 65,000 devices in a segment can be addressed within an EtherCAT network. Since EtherCAT uses the full-duplex process, transmission speeds of up to 100 MBit/s can be achieved.

#### EtherCAT specifications

The ETG specifications that are important for the FAULHABER drives define the following aspects:

- ETG1000 series: EtherCAT technology and communications structure
- ETG2000 series: Specification of the EtherCAT Slave Information (ESI)
- ETG6010: Implementation of the CiA 402 drive profiles

CANopen device profiles have been defined for a wide range of device classes, such as:

- CiA 402 for drives
- CiA 401 for input and output devices

### 3.2 Data Link Layer

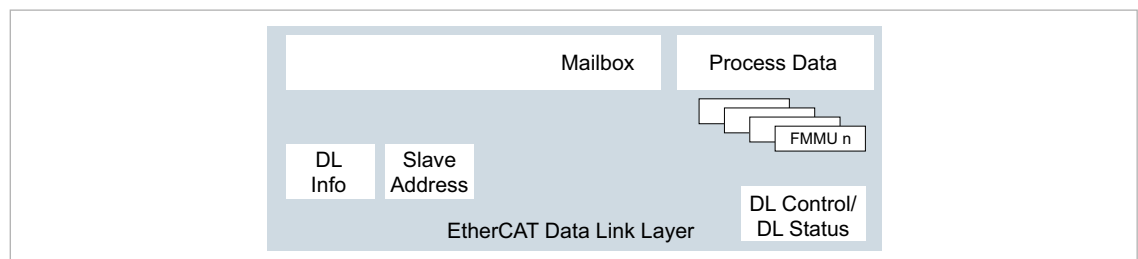


Fig. 3: Data Link Layer

The Data Link Layer connects the Physical Layer to the Application Layer. The Data Link Layer contains essentially the following control and communication services:

- Interface management to the Physical Layer (see chap. 3.2.1, p. 13)
- Interface management to the Application Layer (see chap. 3.2.4, p. 15)
- Access to the EtherCAT EEPROM
- ESC configuration
- Distributed clock (see chap. 3.7.1, p. 22)
- Addressing the EtherCAT slave (see chap. 3.2.3, p. 15)
- SyncManager management (see chap. 3.2.2, p. 14)

## EtherCAT communication

### 3.2.1 EtherCAT frames and datagrams

#### Frame structure

An EtherCAT frame consists of up to 1 518 bytes and can contain up to 1 450 bytes of user data.

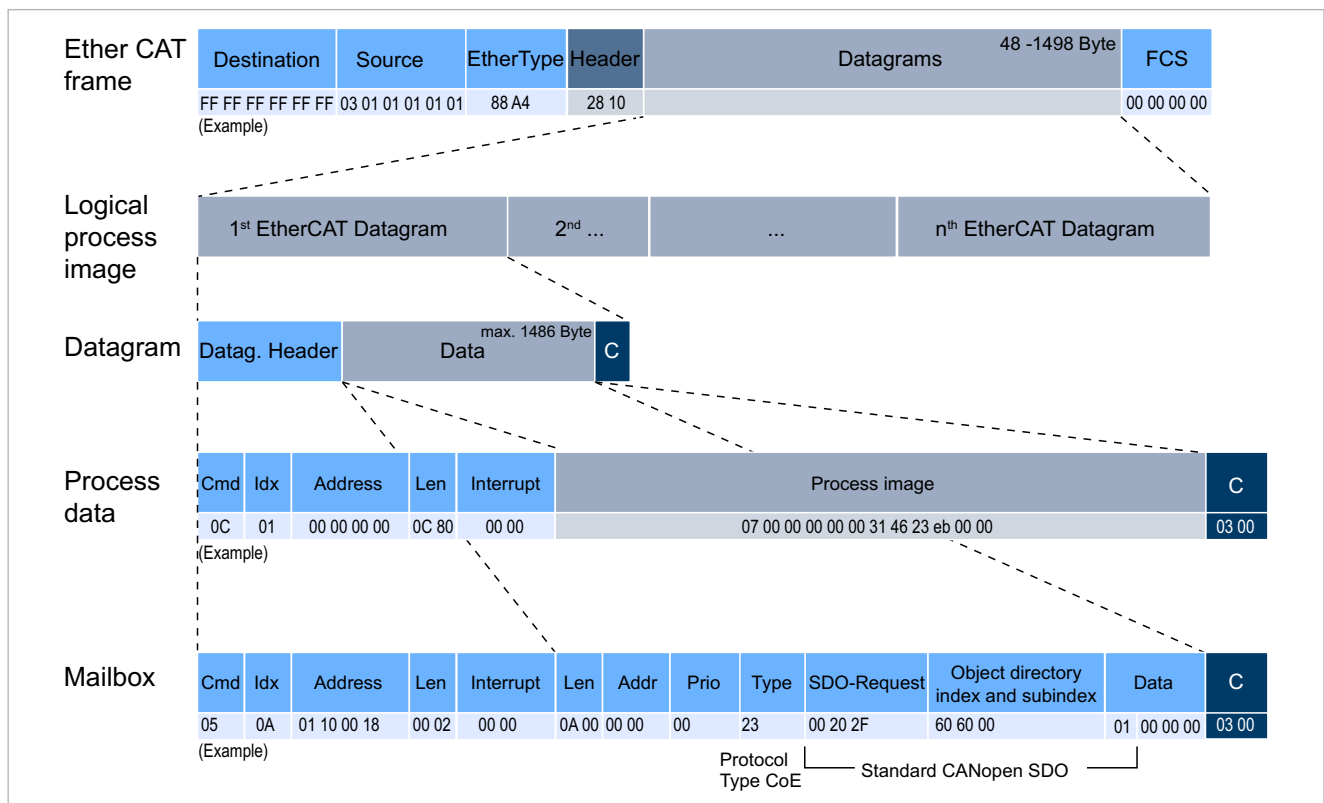


Fig. 4: Frame structure

An EtherCAT frame consists of the following data areas:

- Ethernet header: The Ethernet header contains the source and destination address of the frame, and the type of protocol used.
- Datagram(s): one or more datagrams (see below).
- Frame Check Sequence: This data is used to check the freedom from errors (see chap. 3.10.2, p. 27).

Each datagram consists of the following data areas:

- Datagram header: The datagram header contains information about the type of communication, memory access rights, addresses and length of the user data.
- User data: The user data are structured differently for mailbox and process data communication. They contain the service data objects (SDOs) or process data objects (PDOs) used for CANopen.
- Working Counter: The working counter is used to detect data exchange errors (see chap. 3.10.2, p. 27).

The process data size per EtherCAT slave can be almost any size and if necessary can be segmented into several datagrams. Setup of the process data can be different for every cycle.

## EtherCAT communication

### The path of an EtherCAT frame

On one pair of wires, the EtherCAT master sends the EtherCAT frame to the first EtherCAT slave. The slave processes the frame and forwards it to the next EtherCAT slave. In this way, the message is sent through the entire network and passes through every EtherCAT slave. The EtherCAT slave controllers (ESC) take data from the EtherCAT frame as it passes through, and add this data to their data. The last EtherCAT slave in the network uses the second of the pair of wires to send the EtherCAT frame back to the EtherCAT master.

### 3.2.2 SyncManager management

#### Data transmission through the SyncManager

The PDOs and SDOs are read out from the EtherCAT frame by the SyncManager (Receive Parameter) or are incorporated in the EtherCAT frame (Transmit Parameter). Four Sync channels are available for data transmission:

SyncManager channel	Function
0	Transmission of the service data from the EtherCAT frame into the mailbox (Receive SDO)
1	Transmission of the service data from the mailbox into the EtherCAT frame (Transmit SDO)
2	Transmission of the process data from the EtherCAT frame (Receive PDO 1/2/3/4)
3	Transmission of the process data into the EtherCAT frame (Transmit PDO 1/2/3/4)

The SyncManager objects 0x1C12 and 0x1C13 are available for process data transmission (see chap. 5.1, p. 39).

#### Monitoring the read/write access

The SyncManager protects the data exchange memory against simultaneous access by the EtherCAT master and EtherCAT slave. This prevents another memory area from being overwritten while a memory area is being read, thereby ensuring that the data being read out are consistent.

2 types of memory are available for data exchange:

Memory type	Description
Mailbox memory	<p>The mailbox memory consists of a single memory area.</p> <p>The SyncManager performs the following functions:</p> <ul style="list-style-type: none"> <li>Reading of the memory is prevented while the memory is being written to.</li> <li>Writing to the memory is prevented while the memory is being read.</li> <li>The memory is protected against overflow.</li> </ul> <p>This type of memory is unsuitable for real-time data and is therefore used only for service data.</p>
Buffer memory for process data	<p>The buffer memory is split into in 3 buffer areas.</p> <p>The SyncManager performs the following functions:</p> <ul style="list-style-type: none"> <li>A buffer area that is not currently being read is selected for writing. While writing is being performed, read access to the memory is blocked.</li> <li>Once a buffer area has been written to, it is released for reading. While reading is being performed, write access is blocked.</li> </ul> <p>The following buffer areas are thus available at all times:</p> <ul style="list-style-type: none"> <li>Buffer area 1, which is currently being written to</li> <li>Buffer area 2, which is currently being read</li> <li>Buffer area 3, which has been written to and is ready for reading (in the event that the read operation in buffer area 2 is completed before the write operation to buffer area 1 is complete)</li> </ul> <p>If the write operation to buffer area 1 is completed before the read operation in buffer area 2 is completed, the data in buffer area 1 is released for reading and the data in buffer area 3 is discarded.</p>

## EtherCAT communication

### 3.2.3 Addressing

The EtherCAT protocol permits the following addressing procedures:

- Position addressing: The physical positions of the EtherCAT slaves in the network serve as addresses. In each EtherCAT slave, a specific memory area is reserved for the address.
- Node addressing: Configured node addresses which the EtherCAT master assigned to the EtherCAT slaves during commissioning, serve as addresses. In each EtherCAT slave, a specific memory area is reserved for the address.
- Logical addressing: The entire memory area of the network, i.e. the memory areas of the EtherCAT master and all EtherCAT slaves, is reproduced in a logical memory which can be addressed using a parameter. The assignment of the physical addresses of the EtherCAT slaves to the logical addresses is stored in the EtherCAT master. During the start phase, it is transferred to the Field Bus Memory Management Units (FMMU) of the Data Link Layer. The FMMU converts the logical addresses into physical addresses.

### 3.2.4 Interfaces to the Application Layer

*Tab. 1: Data Link interfaces to the Application Layer*

Interface	Description
Mailbox	The mailbox is used exclusively for data that are not time-critical. This includes service data. Service data are transmitted using service data object frames (SDO frames) based on CANopen (CiA 301) (see chap. 3.5, p. 18). Transmission of service data is performed acyclically.
Process data	Process data are real-time data. This means that the latest saved data can always be accessed. Data that is not processed (such as cycle times, for which the data cannot be processed sufficiently quickly by the EtherCAT slave) is discarded. Process data are transmitted using process data object frames (PDO frames) based on CANopen (CiA 301) (see chap. 3.4, p. 17). Transmission of process data is performed cyclically.

# EtherCAT communication

## 3.3 Application Layer

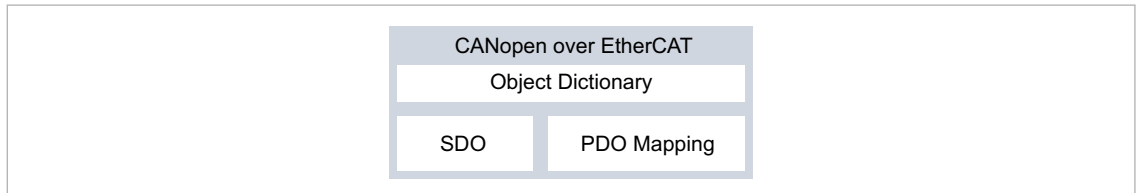


Fig. 5: Application Layer

### CANopen over EtherCAT

FAULHABER Motion Controllers support the CANopen over EtherCAT (CoE) protocol with the CANopen communication profile acc. to CiA 301:

- 4 transmit PDOs (TxPDOs)
- 4 receipt PDOs (RxPDOs)
- 2 SDOs

CANopen telegrams can be up to 250 bytes long and thus have more capacity than the original CAN telegrams with only 8 bytes.

The CANopen drive profiles acc. to CiA 402 can be used unchanged for EtherCAT (see the Functional Manual).

### Object dictionary

The object dictionary contains parameters, set-points and actual values of a drive. The object dictionary is the link between the application (drive functions) and the communication services. All objects in the object dictionary can be addressed by a 16-bit index number (0x1000 to 0x6FFF) and an 8-bit subindex (0x00 to 0xFF).

Index	Assignment of the objects
0x1000 to 0x1FFF	Communication objects
0x2000 to 0x5FFF	Manufacturer-specific objects
0x6000 to 0x6FFF	Objects of the drive profile acc. to CiA 402

The values of the parameters can be changed by the communication side or by the drive side.

The communication part contains communication services as specified in CiA 301.

The data assignment of the PDOs is pre-set to the "PDO set for servo drive" as specified in CiA 402.



## EtherCAT communication

### 3.4 PDO (Process Data Object)

PDOs contain process data for controlling and monitoring the behavior of the device. The drive makes the distinction between receipt PDOs and transmission PDOs.

- Receipt PDOs (RxPDO): are received by a drive and typically contain control data
- Transmission PDOs (TxPDO): are sent by a drive and typically contain monitoring data

PDOs are evaluated or transmitted only when the device is in the NMT *Operational* state (see chap. 3.8.1, p. 24).

#### 3.4.1 PDO configuration

- A maximum of 4 parameters can be mapped in one PDO.
- The data assignment of PDOs can be changed via the objects 0x1600 to 0x1603 and 0x1A00 to 0x1A03. The mapping procedure necessary for this is described in CiA 301. A suitable tool (such as FAULHABER Motion Manager or System Manager for the PLC controller used) is necessary for the mapping procedure.

#### 3.4.2 PDO mapping in the standard configuration

##### RxPDO1: Controlword

2 bytes user data

LB	HB
----	----

The RxPDO1 contains the 16-bit Controlword to CiA DSP402. The Controlword controls the state machine of the drive unit and points to the object index 0x6040 in the object dictionary. The bit distribution is described in the documentation for the drive functions.

##### TxPDO1: Statusword

2 bytes user data

LB	HB
----	----

The TxPDO1 contains the 16-bit Statusword to CiA 402. The Statusword indicates the status of the drive unit and points to the object index 0x6041 in the object dictionary. The bit distribution is described in the documentation for the drive functions.

##### RxPDO2: Controlword, Target Position (PP and CSP)

6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The RxPDO2 contains the 16-bit Controlword and the 32-bit value of the target position (object 0x607A) for the Profile Position mode (PP).

##### TxPDO2: Statusword, Position Actual Value

6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The TxPDO2 contains the 16-bit Statusword and the 32-bit value of the actual position (object 0x6064).

## EtherCAT communication

### RxPDO3: Controlword, Target Velocity (PV and CSV)

#### 6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The RxPDO3 contains the 16-bit controlword and the 32-bit value of the target speed (object 0x60FF) for the Profile Velocity mode (PV).

### TxPDO3: Statusword, Velocity Actual Value

#### 6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The TxPDO3 contains the 16-bit statusword and the 32-bit value of the actual speed (object 0x606C).

### RxPDO4: Controlword, Target Torque (PV and CSV)

#### 6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The RxPDO4 contains the 16-bit controlword and the 16-bit value of the target torque (object 0x6071) for Cyclic Torque mode (CST).

### TxPDO4: Statusword, Torque Actual Value

#### 6 bytes user data

LB	HB	LLB	LHB	HLB	HHB
----	----	-----	-----	-----	-----

The RxPDO4 contains the 16-bit statusword and the 16-bit value of the actual torque (object 0x6077) for Cyclic Torque mode (CST).

## 3.5 SDO (Service Data Object)

The SDO reads and writes parameters in the OD (object dictionary). The SDO accesses the object dictionary via the 16-bit index and the 8-bit subindex. At the request of the master (PC, PLC) the Motion Controller makes data available (upload) or receives data from the master (download).

Tab. 2: General structuring of the SDO user data

Byte 0	Byte 1 to 2	Byte 3	Byte 4 to 7
Command specifier	16-bit index	8-bit subindex	4-byte parameter data

Tab. 3: Distribution of the SDO transfer types

Transfer type	Number of bytes	Purpose
Expedited transfer	Maximum 250 bytes	–
Segmented Transfer	Any size	Transmission of data blocks (such as the trace buffer)

The transfer types are described in CiA 301.

## EtherCAT communication

### 3.5.1 SDO error description

If the SDO protocol cannot be processed, an SDO abort telegram is sent. The error types are coded as follows (see Tab. 2):

- Byte 4 + 5: Additional error code LB + HB
- Byte 6: Error code
- Byte 7: Error class

Error class	Error code	Additional code	Description
0x05	0x03	0x0000	The toggle bit is not changed
0x05	0x04	0x0001	SDO command specifier invalid or unknown
0x06	0x01	0x0000	Access to this object is not supported
0x06	0x01	0x0001	Attempt to read a write-only parameter
0x06	0x01	0x0002	Attempt to write to a read-only parameter
0x06	0x02	0x0000	Object not present in the object dictionary
0x06	0x04	0x0041	Object cannot be mapped in a PDO
0x06	0x04	0x0042	Number and/or length of the mapped objects exceed the PDO length
0x06	0x04	0x0043	General parameter incompatibility
0x06	0x04	0x0047	General internal incompatibility error in the device
0x06	0x07	0x0010	Data type or parameter length do not match or are unknown
0x06	0x07	0x0012	Data types do not match, parameter length too long
0x06	0x07	0x0013	Data types do not match, parameter length too short
0x06	0x09	0x0011	Subindex not present
0x06	0x09	0x0030	General value range error
0x06	0x09	0x0031	Value range error: Parameter value too high
0x06	0x09	0x0032	Value range error: Parameter value too low
0x06	0x09	0x0036	Value range error: Maximum value smaller than minimum value
0x08	0x00	0x0000	General SDO error
0x08	0x00	0x0020	Cannot be accessed
0x08	0x00	0x0022	Cannot be accessed at current device status

## EtherCAT communication

### 3.6 Emergency object (error message)

Emergency messages are not sent out by the slave at its own initiative as they are under CANopen, instead the EtherCAT master must request them via the mailbox protocol. Since this is an extremely slow procedure, we advise against the use of emergency. A better procedure is to map the error register 1001h or the Faulhaber error register 2320h to a PDO. This ensures that the master receives error information in the shortest possible time.

The emergency object is always 8 bytes in size:

8 bytes user data							
Error0(LB)	Error1(HB)	Error-Reg	FE0 (LB)	FE1 (HB)	0	0	0

Assignment of user data:

- Error0(LB)/Error1(HB): 16-bit error code
- Error-Reg: Error register (contents of object 0x1001, see chap. 5.2, p. 47)
- FE0(LB)/FE1(HB): 16-bit FAULHABER error register (contents of object 0x2320, see Tab. 7)
- Bytes 5 to 7: unused (0)

The error register identifies the error type. The individual error types are bit-coded and are assigned to the respective error codes. The object 0x1001 contains the last value of the error register.

A maximum of 3 emergencies can be saved. If the EtherCAT master does not request any emergencies, the 3 oldest are saved and those that are registered later are discarded. This allows errors to be detected that led to subsequent errors.

Tab. 4 lists all the errors that have been reported by emergency messages, provided that the respective error is included in the emergency mask for the FAULHABER error register (Tab. 8).

Tab. 4: Emergency error codes

Emergency message		FAULHABER error register 0x2320			Error register 0x1001	
Error Code	Designation	Error mask 0x2321	Bit	Designation	Bit	Designation
0x0000	No error (is sent out when an error is no longer present or has been acknowledged)	–	–	–	–	–
–	–	–	–	–	0	Generic error (is set if one of the error bits 1 to 7 is set)
0x3210	Overvoltage	0x0004	2	OverVoltageError	2	Voltage error
0x3220	Undervoltage	0x0008	3	UnderVoltageError	2	Voltage error
0x43F0	Temperature Warning	0x0010	4	TempWarning	1	Current error <sup>a)</sup>
0x4310	Temperature Error	0x0020	5	TempError	3	Temperature error
0x5410	Output stages	0x0080	7	IntHWEError	7	Manufacturer-specific error

## EtherCAT communication

Emergency message		FAULHABER error register 0x2320		Error register 0x1001		
Error Code	Designation	Error mask 0x2321	Bit	Designation	Bit	Designation
0x5530	EEPROM fault	0x0400	10	MemError	–	–
0x6100	Software error	0x1000	12	CalcError	7	Manufacturer-specific error
0x7200	Measurement Circuit: Current Measurement	0x0200	9	CurrentMeasError	7	Manufacturer-specific error
0x7300	Sensor Fault (Encoder)	0x0040	6	EncoderError	7	Manufacturer-specific error
0x7400	Computation circuit: module fault	0x0100	8	ModuleError	7	Manufacturer-specific error
0x8110	CAN overrun	0x0800	11	ComError	4	Communication error
0x8130	CAN guarding failed					
0x8140	CAN recovered from bus					
0x8310	off RS232 overrun					
0x84F0	Deviation error (velocity controller)	0x0001	0	SpeedDeviationError	5	Drive-specific error
0x84FF	Max speed error	0x2000	13	DynamicError	7	Manufacturer-specific error
0x8611	Following error (position controller)	0x0002	1	FollowingError	5	Drive-specific error

- a) *The current controller keeps the motor current below the specified limit at all times. The overcurrent error bit is set if the warning temperature is exceeded. The permissible motor current is then reduced from the peak current value to the continuous current value.*

### Example

An emergency message with the user data assignment in Tab. 5 is sent in the following event:

- In the Error Mask 0x2321, bit 1 (following error) is set under subindex 1 (emergency mask) (see Tab. 9).
- The control deviation corridor set in object 0x6065.00 for the position controller has been exceeded for an extended period as defined by the value set for the error delay time in object 0x6066.00 (see the documentation of the drive functions).

Tab. 5: Example of user data assignment to an emergency message

8 bytes user data							
0x11	0x86	0x20	0x02	0x00	0x00	0x00	0x00

## EtherCAT communication

### 3.7 Synchronization

FAULHABER Motion Controller supports synchronization by means of distributed clocks and via a SyncManager event. The type of synchronization is selected using the object 0x1C32.01 for receipt PDOs and the object 0x1C33.01 for transmit PDOs. The values are as follows:

- 0: No synchronization (FreeRun), the EtherCAT slave operates independently according to its own clock, which is set by the cycle time 0x1C32.02
- 1 or 34: Synchronization via a SyncManager event (see chap. 3.7.2, p. 23)
- 2: Synchronization via Distributed Clocks (see chap. 3.7.1, p. 22)

Only the following combinations are permitted for this:

0x1C32.01	SM2	0x1C33.01	SM3	Process data receipt
0x00	FreeRun	0x00	FreeRun	No checking of the cycle time
0x01	SM synchro-nized	0x22	SM synchronized with SM2	The cycle time is monitored if it is entered with a value > 0
0x02	DC Sync0	0x02	DC Sync0	The cycle time is monitored

The cycle time generated by the master must always be a multiple of 500  $\mu$ s. A minimum cycle time of 1 ms is specified in SM-synchronous mode and in *FreeRun* mode. In DC-synchronous mode the minimum cycle time is 500  $\mu$ s.

To simplify configuration of the SyncManager, the ESI file contains two *Slots*. This informs the master that the Motion Controller contains both the operating modes DC-synchronous and SM-synchronous as options. Only one of the options can be active at a time. If the master supports the *Slots* concept, the choice of the desired *Slots* allows the right SM-configuration to be generated easily and without errors.

#### 3.7.1 Synchronization via distributed clocks (DC-Sync)

Each EtherCAT slave has its own clock which is managed by the ESC. The time at the first EtherCAT slave (reference slave) serves as the reference time for the entire network. The clocks of all other EtherCAT slaves and of the EtherCAT master take their time from this reference time.

For synchronization of the clocks, at frequent intervals the EtherCAT master sends a special datagram into which the EtherCAT slave with the reference clock enters its current time. All other EtherCAT slaves and the EtherCAT master read this time from the datagram. Since the EtherCAT participants in a network are arranged in a logical ring structure, the first EtherCAT slave after the EtherCAT master is the reference slave.

Each reference time read in by the EtherCAT participants is corrected by the time taken for the datagram to travel from the reference clock to the corresponding EtherCAT participant. In order to determine these travel times, the EtherCAT master sends a special datagram to the EtherCAT slaves. When the ESCs receive the datagram they write the receipt time into a datagram. The EtherCAT master reads these receipt times and performs the appropriate calculation.

The ESC of the drive has an internal master clock that is synchronized to the master clock of the reference slave. The synchronization allows for the telegram travel time. The internal master clock generates a Sync0 signal which starts the local cycle of the drive.

The local cycle requires process data from an EtherCAT telegram which was received earlier and temporarily saved. If the local cycle is started by the Sync0 signal, it reads the saved

## EtherCAT communication

---

data and executes the control loop. Finally it writes the input data back to the process data so that the input data is available to the master.

The master should send telegrams with the same cycle as the cycle time of the slave so that the slave always processes the latest data. If due to a jitter in the cycle of the master a packet is sent out too late, it can no longer be processed in the current control cycle, instead it must be held back for processing in the next control cycle. In this case, the current control cycle uses the data from the previous telegram.

The DC cycle time is not set by the object 0x1C32.02, instead the master sets it directly in the ESC registers. The DC cycle time must be at least 500  $\mu$ s or a multiple thereof.

### 3.7.2 Synchronization via a SyncManager event (SM-Sync)

The local cycle of the EtherCAT slave is started when a process data telegram is received (SyncManager event). If the RxPDOs are transmitted cyclically, the EtherCAT slave is synchronized at the SyncManager2 event (SM2 event). If only TxPDOs are transmitted, the EtherCAT slave is synchronized at the SyncManager3 event (SM3 event).

The parameter for synchronization via a SyncManager event are set via the objects 0x1C32 (SM2) and 0x1C33 (SM3) in the *Pre-Operational* state (see chap. 5.1, p. 39).

#### Monitoring of the process data entry

The purpose of the entry in 0x1C32.02 (cycle time) is to monitor the telegrams sent by the master. The process data must arrive in the slave within the specified timescale. If a fault (such as a broken wire) occurs and no data arrives at the slave and the slave is appropriately configured, it will output an emergency message and switch into an error state. If the cycle time is set to zero, this monitoring mechanism is deactivated.

## EtherCAT communication

### 3.8 Layer management

The layer management provides the following services:

- Controlling the EtherCAT state machine (chap. 3.8.1, p. 24)
- Reading and writing at the Slave Information Interface (see chap. 3.8.2, p. 25)

The EtherCAT master communicates directly with the ESC in order to perform these services.

#### 3.8.1 Controlling the EtherCAT state machine

After switching on and initializing, the Motion Controller is automatically set to the *Pre-Operational* state. In the *Pre-Operational* state the Motion Controller can communicate with the device only using mailbox communication.

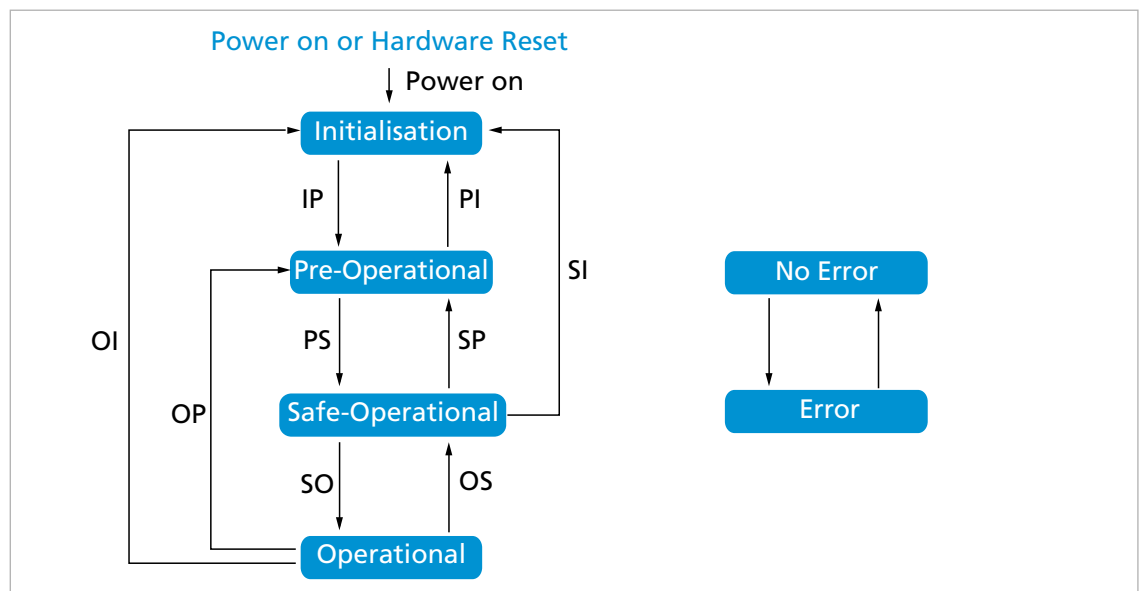


Fig. 6: EtherCAT state machine

Tab. 6: Changes of state

Transfer	Actions
Power on	<ul style="list-style-type: none"> <li>■ The initialization state is achieved automatically on switching on.</li> <li>■ Neither mailbox communication nor process data communication are available.</li> <li>■ The EtherCAT master initializes the SyncManager channels for mailbox communication.</li> </ul>
IP	<ul style="list-style-type: none"> <li>■ The EtherCAT master synchronizes the EtherCAT field bus.</li> <li>■ The EtherCAT master initializes the SyncManager channels for process data communication, the FMMU channels and the SyncManager-PDO assignment.</li> <li>■ Mailbox communication is established between the EtherCAT master and EtherCAT slaves.</li> <li>■ Settings for process data transmission are transmitted.</li> </ul>
PS	<ul style="list-style-type: none"> <li>■ The EtherCAT slave checks that the SyncManager channels for process data communication and the settings for the Distributed Clocks are correct.</li> <li>■ The EtherCAT slave copies the current input data in the memory areas of the ESC.</li> <li>■ Mailbox and process data communication are now available. The outputs of the EtherCAT slave remain in a safe state and are not output. The input data are updated cyclically.</li> </ul>
SO	<ul style="list-style-type: none"> <li>■ The EtherCAT master transmits valid output data to the EtherCAT slave.</li> <li>■ The EtherCAT master switches the EtherCAT slave into the <i>Operational</i> state.</li> <li>■ In the <i>Operational</i> state, the EtherCAT slave copies the input data to its outputs.</li> <li>■ Mailbox and process data communication are now available.</li> </ul>



## EtherCAT communication

The ESI file for the FAULHABER Motion Controller contains the default configuration for all objects (see chap. 2.4, p. 11). In most cases no further parametrization is necessary at system start.

Any necessary parameter settings can be performed by the FAULHABER Motion Manager using the USB interface and saved permanently in the EEPROM (see chap. 3.11, p. 31). Settings in the EEPROM are immediately available at system start.

**i** In the *Init* state, all values of the drive are reset to the switch-on values. Values previously set by the user in another state are overwritten if they have not been saved by a "Save" command 1010h. If this behavior is not intended, the drive should not be switched into the *Init* state, instead it should at least remain in the *Pre-Operational* state.

**i** The drive is controlled by objects of the drive profile (controlword, statusword). The communication with the drive and the associated operating modes are described in the separate "Functions Manual".

Switching into the *Pre-Operational* state takes just a few milliseconds. The master must enquire on the AL register (130h) and wait until the state has been successfully switched. No SDO communication is possible beforehand.

### 3.8.2 Slave Information Interface (SII)

The Slave Information Interface contains data specific to the EtherCAT slave and the connected drive (e.g. values of object 0x1018) as well as the mailbox SyncManager configuration.

This data is saved in the EtherCAT EEPROM, which is read out when the network is commissioned (see chap. 2.4, p. 11).

## 3.9 Entries in the object dictionary

The object dictionary manages the configuration parameters. The object dictionary is divided into three areas. Each object can be referenced by its index and subindex (SDO protocol).

- Communication parameters (index 0x1000 to 0x1FFF) contains communications objects to CiA 301, see chap. 5.1, p. 39)
- Manufacturer-specific area (index 0x2000 to 0x5FFF) contains manufacturer-specific objects, see chap. 5.2, p. 47)
- The standardized device profiles area (0x6000 to 0x9FFF) contains objects supported by the Motion Controller (see the documentation of the drive functions)

## EtherCAT communication

### 3.10 Error handling

#### 3.10.1 Device faults

Tab. 7: FAULHABER error register (0x2320)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2320	0x00	Fault Register	U16	ro	–	FAULHABER error register

The FAULHABER error register contains the most recent errors in bit-coded form. The errors can be masked by selection of the desired types of error via the Error Mask (0x2321) object.

Tab. 8: Error coding

Error bit	Error message	Description
0x0001	SpeedDeviationError	Speed deviation too big
0x0002	FollowingError	Following error
0x0004	OverVoltageError	Overvoltage detected
0x0008	UnderVoltageError	Undervoltage detected
0x0010	TempWarning	Temperature exceeds that at which a warning is output
0x0020	TempError	Temperature exceeds that at which an error message is output
0x0040	EncoderError	Error detected at the encoder
0x0080	IntHWError	Internal hardware error
0x0100	ModuleError	Error at the external module
0x0200	CurrentMeasError	Current measurement error
0x0400	MemError	Memory error (EEPROM)
0x0800	ComError	Communication error
0x1000	CalcError	Internal software error
0x2000	DynamicError	The current velocity is higher than the maximum speed set for the motor.
0x4000	–	Not used, value = 0
0x8000	–	Not used, value = 0

All of these errors correspond to an Emergency Error Code. (see chap. 3.6, p. 20).

## EtherCAT communication

The error mask describes the handling of internal errors depending on the error coding (see Tab. 8).

Tab. 9: Error Mask (0x2321)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2321	0x00	Number of Entries	U8	ro	6	Number of object entries
	0x01	Emergency Mask	U16	rw	0xFFFF	Errors for which an error message is sent
	0x02	Fault Mask	U16	rw	0x0000	Errors for which the state machine of the drive switches into <i>Fault Reaction Active</i> state
	0x03	Error Out Mask	U16	rw	0x0000	Errors for which the error output pin is set
	0x04	Disable Voltage Mask	U16	ro	0x4024	Errors which switch off the drive (not configurable)
	0x05	Disable Voltage User Mask	U16	rw	0x0000	Errors which switch off the drive (configurable)
	0x06	Quick Stop Mask	U16	rw	0x0000	Errors for which the state machine of the drive switches into <i>Quick Stop Active</i> state

### Examples:

- When the fault mask (subindex 2) of object 0x2321 is set to 0x0001 the drive is switched off due to overcurrent and its state machine is set to a *Fault Reaction Active* state.
- When the subindex 3 of object 0x2321 is set to 0, the error output (fault pin) indicates no error. When the subindex 3 of object 0x2321 is set to 0xFFFF, the error output (fault pin) indicates all errors.

### 3.10.2 Communication error

The network is monitored for communications data errors and also for missing data. If an error occurs, this procedure allows the drives to be brought into a safe state and error messages displayed. Network traffic analysis must then be performed in order to localize and remedy the error.

#### 3.10.2.1 Checking EtherCAT frame entries for errors

Since the EtherCAT slave cannot communicate directly with the EtherCAT master, the monitoring for defective data is performed via entries in the EtherCAT frame.

- Frame Check Sequence (FCS): The ESC uses a check sum to check the EtherCAT frame for errors as it passes through. The information from the EtherCAT frame is used only if the result of the check is positive. If the result of the check is negative, the EtherCAT frame is flagged as defective by incrementing the count value for the subsequent EtherCAT slaves and the EtherCAT master.
- Working counter: The working counter is part of the datagram. After a successful data exchange, the EtherCAT slave increments the count value by 1. The EtherCAT master compares the count value of the returned EtherCAT datagram with the expected count value, and thereby can detect any errors in the data exchange.

## EtherCAT communication

### 3.10.2.2 Error response

The drive must receive the output data from the EtherCAT master at the right time and must be able to send its state regularly to the EtherCAT master. For this purpose, the EtherCAT slave must receive process data at regular intervals. Variations during data reception must remain within certain limits.

The exchange of the process image is monitored by two mechanisms in the drive which operate on different principles:

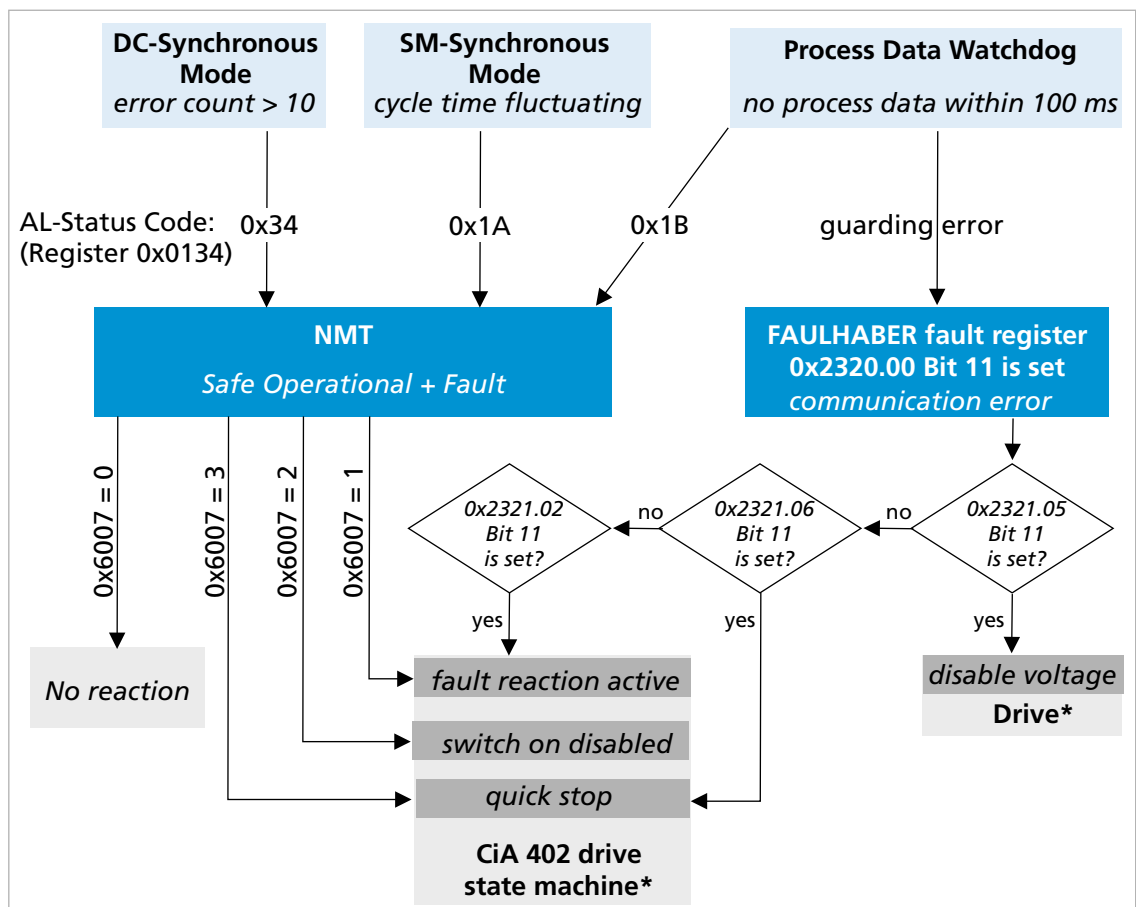


Fig. 7: Mechanisms of error monitoring

\* The objects of the drive profile to CiA402 are described in detail in the documentation of the drive functions

#### Monitoring the arrival time of the process data

When the process data arrive at the drive a check is made whether the arrival time matches the expected time. If on multiple occasions the deviation is too large, the EtherCAT state machine switches into the states *Safe Operational* and *Error*. Depending on the error response set in object 0x6007, the drive brakes to a standstill (see documentation of the drive functions).

This type of monitoring has different names depending on the type of synchronization:

- DC synchronization: SYNC0 monitoring
- SM synchronization: SyncManager monitoring

## EtherCAT communication

### Monitoring receipt of the process data

If no process data is received by the drive (e. g. because of a break in the cable), the monitoring of the arrival time is not activated. If for a period longer than 100 ms no process data has been received, the process data watchdog triggers an error response. The error response depends on the state of bit 11 in the FAULHABER error register 0x2320.00:

- Bit 11 not set:

The EtherCAT state machine changes to the states *Safe Operational* and *Error*. Depending on the error response set in object 0x6007, the drive brakes to a standstill (see documentation of the drive functions).

- Bit 11 set:

The drive stops as defined in the objects 0x2321.02, 0x2321.04, 0x2321.05 and 0x2321.06 (see chap. 5.2, p. 47).

### 3.10.2.3 Analysis of the network traffic

The network traffic can be analyzed using software tools (such as *Wireshark*). The software tool can be installed either on a separate PC connected to the network or directly on the EtherCAT master. The analysis of the network traffic consists of reading and comparing the frame sent by the EtherCAT master and the frame received by the EtherCAT master. Particularly distinctive points for the error analysis are the EtherCAT frame entries (FCS, Working Counter) mentioned above.

### 3.10.2.4 EtherCAT AL status codes and troubleshooting

If a communications error occurs, an error code is loaded to the AL status code register (0x0134). The table below describes the available codes and lists the actions for troubleshooting.

AL status code	Description	Troubleshooting
0x0001	EtherCAT system (hardware or software) did not initialize.	Inform FAULHABER support.
0x0002	No memory is available for an internal buffer of a SyncManager (mapping or buffering).	Inform FAULHABER support.
0x0011	Requested target state is not reachable / requested transition is not allowed.	Only switch the AL state machine in valid steps. <b>Invalid</b> are, for example: <ul style="list-style-type: none"> <li>▪ Init → Safe-Operational</li> <li>▪ Init → Operational</li> <li>▪ Pre-Operational → Operational</li> </ul>
0x0012	Requested target state does not exist.	Use valid AL state codes. Valid codes are: <ul style="list-style-type: none"> <li>▪ Init: 1</li> <li>▪ Pre-Operational: 2</li> <li>▪ Safe-Operational: 4</li> <li>▪ Operational: 8</li> </ul>
0x0013	The <i>Boot State</i> is not implemented in this product.	Do not change to boot state.
0x0015	When changing to boot state, an error was detected in the mailbox configuration (SDO communication).	See 0x0016.

## EtherCAT communication

AL status code	Description	Troubleshooting
0x0016	When the system was switched to <i>Pre-Operational</i> state an error in the configuration of the mailbox (SDO communication) was detected.	Check configuration: <ul style="list-style-type: none"> <li>SM0 and SM1 must be switched on</li> <li>The control bytes of the SM must be correct</li> <li>The physical and configured addresses of the SM must be correct</li> <li>The length of the SM must be within the permitted limits.</li> </ul>
0x0017	An error was detected while mapping an SM.	Check configuration: <ul style="list-style-type: none"> <li>Length of all mapped objects must not exceed SM's length</li> <li>A maximum of 8 SM can be used</li> <li>SMs may not overlap</li> </ul>
0x001A	Synchronization error: The fault threshold is > 0 and the internal error counter has exceeded this threshold because the process data was sent too fast. Only in SM synchronous mode.	Send process data slower. or: Check the cycle time that is set in 0x1C32.02. It must match the actual cycle time.
0x001B	Synchronization error: SM watchdog reported a communication fault, since there was no process data at all for more than 100 ms.	Send process data.
0x001D	Faulty configuration of an output SM.	Correctly configure SM: <ul style="list-style-type: none"> <li>SM with zero length must be disabled, all others must be enabled</li> <li>The control byte of the SM must be correct</li> <li>Physical and configured addresses of the SM must match</li> <li>The length of the SM must be within the permitted limits</li> <li>SMs may not overlap</li> </ul>
0x001E	Faulty configuration of an input SM.	Correctly configure SM: <ul style="list-style-type: none"> <li>SM with zero length must be disabled, all others must be enabled</li> <li>The control byte of the SM must be correct</li> <li>Physical and configured addresses of the SM must match</li> <li>The length of the SM must be within the permitted limits</li> <li>SMs may not overlap</li> </ul>
0x0026	Inconsistent Settings for SyncManager.	Check settings. Only the following combinations are valid: <ul style="list-style-type: none"> <li>SM0 = 0x00 &amp; SM1 = 0x00 (Freerun)</li> <li>SM0 = 0x01 &amp; SM1 = 0x22 &amp; ADO 0981 (SyncOut Unit) Bits [3,1,0] = 000 (SYNCO deactivated)</li> <li>SM0 = 0x02 &amp; SM1 = 0x02 &amp; ADO 0981 (SyncOut Unit) Bits [3,1,0] = 011 (SYNCO activated)</li> </ul>
0x002C	SYNCO signal no longer received in DC.	Check configuration of SYNCO: <ul style="list-style-type: none"> <li>SM: cycle time at least 1 ms</li> <li>DC: cycle time at least 500 µs</li> </ul>
0x002E	Cycle time too small.	In Freerun and SM synchronous minimum cycle time is 1 ms.
0x0030	Faulty configuration of DC.	Check configuration of DC: <ul style="list-style-type: none"> <li>SM0 = 0x02</li> <li>SM1 = 0x02</li> <li>ADO 0981 (SyncOut Unit) Bits [3,1,0] = 011 (SYNCO activated)</li> </ul>

## EtherCAT communication

AL status code	Description	Troubleshooting
0x0034	Before process data can be processed, the slave must receive it. The master had failed to send this data in time too often. Only in DC mode.	Send a process image at the right time, so that it can be used in response to the SYNC0 pulse.
0x0036	DC time is too small.	Minimum DC time is 500 µs.
0xB001	No PDOs were found for the SyncManager channel.	Inform FAULHABER support.
0xB003	PDO mapping does not exist.	Inform FAULHABER support.
0xB004	PDO index not within the necessary range.	Inform FAULHABER support.
0xB007	PDO entry does not exist in the object dictionary.	Inform FAULHABER support.

### 3.11 Saving and restoring parameters

So that changed parameters in the OD remain active in the controller when it is switched on again, the "Save" command must be executed to save them permanently in the non-volatile memory (EEPROM application) (see chap. 5.1, p. 39). When the motor is switched on, the parameters are loaded automatically from the non-volatile memory into RAM.

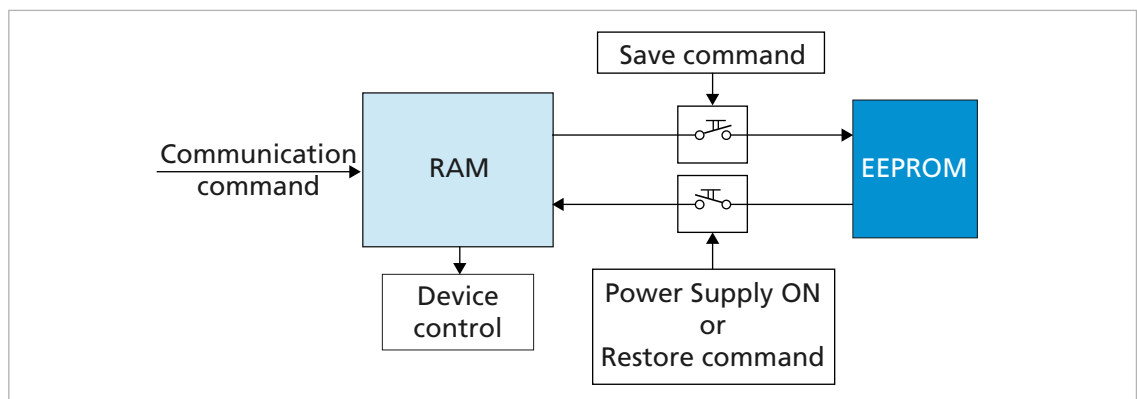


Fig. 8: Saving and restoring parameters

The following parameters can be loaded using the "Restore" command (see chap. 5.1, p. 39):

- Factory settings
- Parameters saved using the "Save" command


#### 3.11.1 Save parameters

The current parameter settings can be saved in the internal EEPROM (SAVE) (see Tab. 14), either completely or for individual ranges.



- ▶ Write the "save" signature to the subindex 01 to 05 of the object 0x1010 (see Tab. 15).


## EtherCAT communication

### 3.11.2 Restoring settings

 When the drive is switched on, the saved parameters are loaded automatically.

Factory settings or last saved parameter settings can be loaded from the internal EEPROM at any time, completely or for specific ranges (RESTORE) (see Tab. 16).

1. Write the "Load" signature to the subindex 01 to 06 of the object 0x1011 (see Tab. 17).
  -  After Restore Factory (01), Restore Communication (02) and Restore Application (03), the drive must be reset. Only then are the parameters updated.
2. Application parameters (04), together with record 1 and record 2 of the special application parameters (05/06) can be updated with the "Reload" command.
  -  The "Reload" command overwrites the values last saved as application parameters.

 If it is desired that the values currently loaded remain available after a "Restore", these must be saved to the PC using a suitable program (such as FAULHABER Motion Manager).

### 3.11.3 Changing the parameter set

The repository for the application parameters (motor data, I/O configuration, controller parameters, operating mode, etc.) includes a common basic set of parameters (App) as well as a storage area for parameters which often need to be adapted to variations in the load situation (App1/App2):

#### Speed controller and filter

Index	Subindex	Name	Type	Attr.	Meaning
0x2344	0x01	Gain $K_p$	U32	rw	Controller gain [ $As\ 1e^{-6}$ ]
	0x02	Integral time $T_N$	U16	rw	Controller reset time [100 $\mu s$ ]
0x2346	0x01	Set Point Velocity Filter Time $T_F$	U16	rw	Filter time $T_F$ [100 $\mu s$ ]
	0x02	Setpoint Filter Enable	U8	rw	0: inactive 1: Active
0x2347	0x01	Gain Factor	U8	rw	Gain factor (used by the speed control in PP mode on the $K_p$ ) 0: The gain factor of the speed controller is reduced to 0 at the target 128: no variable gain 255: The gain factor of the speed controller is doubled at the target

#### Position controller

Index	Subindex	Name	Type	Attr.	Meaning
0x2348	0x00	Number of entries	U8	ro	Number of object entries
	0x01	$K_v$ [1/s]	U8	rw	Range: 1-250



## EtherCAT communication

### Pre-controls

Index	Subindex	Name	Type	Attr.	Meaning
0x2349	0x01	Torque/force feed forward factor	U8	rw	Factor for the torque or force control 0: 0% activation of the feedforward value 128: 100% feedforward control
	0x02	Torque/Force feed forward delay	U8	rw	Set-point delay: 0: undelayed activation 1: Activation delayed by one sampling
0x234A	0x01	Velocity feed forward factor	U8	rw	Factor for the torque or force control 0: 0% feedforward control 128: 100% feedforward control
	0x02	Velocity feed forward delay	U8	rw	Set-point delay: 0: undelayed activation 1: Activation delayed by one sampling

### General settings

Index	Subindex	Name	Type	Attr.	Meaning
0x6060	0x00	Modes of Operation	S8	rw	Select the operating mode -4: ATC -3: AVC -2: APC -1: Voltage mode 0: Controller not activated 1: PP 3: PV 6: Homing 8: CSP 9: CSV 10: CST
0x6081	0x00	Profile Velocity	U32	rw	Profile velocity in user-defined units
0x6083	0x00	Profile acceleration	U32	rw	Profile acceleration [1/s <sup>2</sup> ]
0x6084	0x00	Profile deceleration	U32	rw	Profile deceleration [1/s <sup>2</sup> ]
0x6086	0x00	Motion Profile Type	S16	rw	Speed profile type: 0: Linear profile 1: Sin <sup>2</sup> speed
0x60E0	0x00	Positive torque limit value	U16	rw	Value of the upper limit value [in relative scaling]
0x60E1	0x00	Negative torque limit value	U16	rw	Value of the lower limit value [in relative scaling]

These parameters are stored twice. During operation, the system can switch quickly between these different presets.

## EtherCAT communication

---

### **Create an application set**

- ▶ Save application parameters 1: Write the "save" signature to subindex 04 of object 0x1010.
  - ↻ The current data is saved as the application parameter set 1.
- ▶ Save application parameters 2: Write the "save" signature to subindex 05 of object 0x1010.
  - ↻ The current data is saved as the application parameter set 2.

### **Activate an application set**

- ▶ Reload application parameters 1: Write the "load" signature to subindex 05 of object 0x1011.
  - ↻ Current data from the application parameter set 1 is activated directly.
- ▶ Reload application parameters 2: Write the "load" signature to subindex 06 of object 0x1011.
  - ↻ Current data from the application parameter set 2 is activated directly.

## Trace recorder

### 4 Trace recorder

The trace recorder allows up to 4 parameters of the controller to be recorded. For this purpose, one trigger source and a maximum of 4 signal sources are selected in the object dictionary. The parameter values are written to an internal buffer and can then be read out (see chap. 4.2, p. 37). The advantage compared to the cyclical transmission of process data is the higher speed. The trace recorder can record values at a controller sampling time of 100  $\mu$ s. By comparison, process data cannot be transmitted faster than every 500  $\mu$ s.

**i** The FAULHABER Motion Manager provides a user-friendly means of setting and evaluating the trace functions.

The configuration and reading of data with the trace recorder is performed via SDO.

The trace recorder is configured using the object 0x2370 in the OD.

The recorded data are read using the segmented SDO upload protocol. The object 0x2371 is available in the OD for this purpose (see chap. 4.2, p. 37).

#### 4.1 Trace settings

Object 0x2370 is available for configuration of the trace recorder. The data sources to be recorded, the buffer size, the resolution and the trigger conditions can be set here.

Tab. 10: Trace Configuration (0x2370)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2370	0x00	Number of Entries	U8	ro	10	Number of object entries
	0x01	Trigger Source	U32	wo	0	Trigger source
	0x02	Trigger Threshold	S32	rw	0	Trigger threshold
	0x03	Trigger Delay Offset	S16	rw	0	Trigger delay
	0x04	Trigger Mode	U16	rw	0	Trigger mode
	0x05	Buffer Length	U16	rw	100	Buffer length
	0x06	Sample Time	U8	rw	1	Recording sampling rate 1: in every sampling step
	0x07	Trace Source of Channel 1	U32	wo	0	Trace source of channel 1
	0x08	Trace Source of Channel 2	U32	wo	0	Trace source of channel 2
	0x09	Trace Source of Channel 3	U32	wo	0	Trace source of channel 3
	0x0A	Trace Source of Channel 4	U32	wo	0	Trace source of channel 4

#### Trigger Source (0x2370.01), trace source 1 to 4 (0x2370.07 to 0A)

The parameters to be recorded, trace source 1 to trace source 4, must be entered in objects 0x2370.07 to 0x2370.0A as pointers to a corresponding object entry (index and subindex of the desired parameter). The trigger source must be entered in object 0x2370.01 as a pointer to a corresponding object entry (index and subindex of the desired parameter).

## Trace recorder

### Example

Object 0x6064.00 (position actual value) must be recorded as the first data source: The value 0x606400 must be entered in object 0x2370.07.

### Trigger Threshold (0x2370.02)

The trigger threshold is entered in object 0x2370.02.

Depending on the settings of bits 1 to 3 in trigger type object 0x2370.04, recording is started when the threshold set here is exceeded or undershot.

### Trigger Delay Offset (0x2370.03)

The trigger delay is stated in object 0x2370.03 as a multiple of the sample time set in object 0x2370.06.

- Delay > 0: After the trigger, the start of recording is delayed by the set multiples of the sample time.
- Delay < 0: Negative delays are possible up to the length of the buffer. Recording ends at the point in the ring buffer where the recording would have had to start before the actual trigger. This ensures that the values recorded before the trigger are retained.

### Trigger Mode (0x2370.04)

The trigger type and the type of data sources are set using object 0x2370.04. Bit 0 activates the trigger and thus, providing the trigger conditions are satisfied, starts the recording.

Tab. 11: Trigger Mode (0x2370.04)

Bit	Entry	Description
0 (LSB)	EN	<ul style="list-style-type: none"> <li>■ 0: No trigger active</li> <li>■ 1: Trigger active. Is automatically reset in trigger modes 1 and 3</li> </ul>
1	Edge 0	<ul style="list-style-type: none"> <li>■ 0: rising flank or trigger &gt; threshold</li> </ul>
2	Edge 1	<ul style="list-style-type: none"> <li>■ 1: falling flank or trigger &lt; threshold</li> </ul>
3	Edge 2	
4 to 5	Reserved	–
6	Mode 0	<ul style="list-style-type: none"> <li>■ 0: No trigger</li> </ul>
7	Mode 1	<ul style="list-style-type: none"> <li>■ 1: Single Shot</li> <li>■ 2: Repeating</li> </ul>
8 to 10	Reserved	–
11	Source Type 1	<ul style="list-style-type: none"> <li>■ 0: An object dictionary entry is used as the source</li> </ul>
12	Source Type 2	<ul style="list-style-type: none"> <li>■ 1: Not currently supported</li> </ul>
13	Source Type 3	
14	Source Type 4	
15 (MSB)	Trigger Type	

### Buffer Length (0x2370.05)

The length of the buffer (number of values) available for recording is set in object 0x2370.05. The permissible length depends on the data type and the number of parameters to be recorded. In total, a maximum of 8 kBytes (2 kBytes per channel) of buffer are available.

### Sample Time (0x2370.06)

The sampling rate is stated in object 0x2370.06 as a multiple of the controller sampling time.

## Trace recorder


### 4.2 Reading the trace buffer

The recorded data buffer can be read using the object 0x2371.

Tab. 12: Trace Buffer (0x2371)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2371	0x00	Number of Entries	U8	ro	5	Number of object entries
	0x01	Trace State	U16	ro	0	Trigger status
	0x02	Trace Value of Channel 1	Vis string	ro	–	Signal buffer, channel 1
	0x03	Trace Value of Channel 2	Vis string	ro	–	Signal buffer, channel 2
	0x04	Trace Value of Channel 3	Vis string	ro	–	Signal buffer, channel 3
	0x05	Trace Value of Channel 4	Vis string	ro	–	Signal buffer, channel 4

The user data length of the individual data sources is dependent on the data length of the parameter to be transmitted (according to the OD entry) and the set buffer size. A memory area the size of the data length times the buffer size must therefore be provided for each data source, for reading the recorded values.

 The individual data points can be recorded to the highest resolution of the recorder.

#### Trace State (0x2371.01)

Tab. 13: Trace State (0x2371.01)

Bit	Entry	Description
0 (LSB)	Status 0	<ul style="list-style-type: none"> <li>▪ 0: No trigger active</li> <li>▪ 1: Trigger not yet reached</li> <li>▪ 2: Recording not yet completed</li> <li>▪ 3: Recording completed, data are available</li> </ul>
1	Status 1	
2 to 7	Not used	–
8 to 15 (MSB)	Start index	First value in the buffer after triggering

Before the recorded data are read, the Triggerstatus 0x2371.01 must be checked. If bit 0 and bit 1 are set (status = 3), recording is completed and the contents of the buffer can be read using the objects 0x2371.02 to 0x2371.05 via Segmented SDO-Upload Protokoll.

### 4.3 Typical execution of the trace function

1. Set the trigger type and the type of the data sources (2370.04).
2. Set the trigger source and the signals to be recorded (2370.01, 07 to 0A).
3. Set the recording length (2370.05).
4. If necessary, set the sampling rate (2370.06).
5. Set the threshold value (2370.02) for the trigger.
6. Set the flank for the trigger and activate recording (2370.04).
  - ↳ The settings for the trace recorder are complete.
7. Check the trigger status (2371.01) for the value 3.
8. Read the recorded buffer content (2371.02 to 05).

## Parameter description

### 5 Parameter description

#### 5.1 Communication objects acc. to CiA 301

##### Device Type

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1000	0x00	Device Type	U32	ro	0x00420192	Indication of the device type

Contains information on the device type, coded in two 16-bit fields:

- Byte MSB (Most Significant Byte): Additional Information = 0x42 (Servo drive, type specific PDO mapping)
- Byte LSB (Least Significant Byte): Device Profile Number = 0x192 (402d)

##### Error Register

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1001	0x00	Error Register	U8	ro	yes	Error register

The error register contains the last error types that occurred in bit-coded form.

This parameter can be mapped in a PDO.

##### Predefined Error Field (error log)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1003	0x00	Number of Errors	U8	rw	–	Number of saved errors
	0x01– 0x08	Standard Error Field	U32	ro	–	Error codes that have occurred most recently

The error log contains the coding of the last errors that occurred.

- Byte MSB: Error Register
- Byte LSB: Error Code

The meaning of the error codes is described in chap. 3.6, p. 20.

Writing a 0 to the subindex 0 clears down the error log.

##### Manufacturer Device Name

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1008	0x00	Manufacturer Device Name	Vis string	const	–	Device name

##### Manufacturer Hardware Version

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1009	0x00	Manufacturer Hardware Version	Vis string	const	–	Hardware version

##### Manufacturer Software Version

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x100A	0x00	Manufacturer Software Version	Vis string	const	–	Software version

## Parameter description

### Store Parameters

Tab. 14: Save parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1010	0x00	Number of Entries	U8	ro	9	Number of object entries
	0x01	Save All Parameters	U32	rw	1	Saves all parameters
	0x02	Save Comm Parameters	U32	rw	1	Save communication parameters (object dictionary entries 0x0000 to 0x1FFF)
	0x03	Save App Parameters	U32	rw	1	Save application parameters (object dictionary entries 0x2000 to 0x6FFF)
	0x04	Save App Parameters 1	U32	rw	1	Save application parameters for immediate changes (set 1)
	0x05	Save App Parameters 2	U32	rw	1	Save application parameters for immediate changes (set 2)

The Store Parameters object saves the configuration parameters into the flash memory. Read access supplies information about the save options. Writing the "Save" signature to the respective subindex initiates the save procedure.

Tab. 15: Signature "save"

Signature	ISO 8 859 ("ASCII")	hex
MSB	e	65h
	v	76h
	a	61h
LSB	s	73h



### NOTICE!

The flash memory is designed to accommodate 10,000 write cycles. If this command is executed more than 10,000 times, the correct operation of the flash memory can no longer be guaranteed.

- ▶ Avoid performing frequent saves.
- ▶ After 10,000 save cycles, replace the device.

### Restore Default Parameters

Tab. 16: Restoring parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1011	0x00	Number of Entries	U8	ro	6	Number of object entries
	0x01	Restore all Default Parameters	U32	rw	1	Restore all factory settings
	0x02	Restore Comm Default Parameters	U32	rw	1	Restore all factory settings for communication parameters (0x0000 to 0x1FFF)
	0x03	Restore App Default Parameters	U32	rw	1	Restore all factory settings for application parameters (from 0x2000)
	0x04	Reload User Parameters	U32	rw	1	Restore the user's last saved application parameters (from 0x2000)
	0x05	Reload Application Parameters 1	U32	rw	1	Application parameter set 1 for immediate changes
	0x06	Reload Application Parameters 2	U32	rw	1	Application parameter set 2 for immediate changes





## Parameter description

The Restore Default Parameters object loads the standard configuration parameters. The standard configuration parameters are either as delivered or as saved last. Read access supplies information about the restore options. Writing the “Load” signature to the respective subindex initiates the restore procedure:

Tab. 17: “Load” signature

Signature	ISO 8859 (“ASCII”)	hex
MSB	d	64h
	a	61h
	o	6Fh
LSB	l	6Ch

 The delivery state may be loaded only when the output stage is switched off.

 To activate the parameters restored by **Restore Factory Settings**, the drive must be switched off and on again.

### Identity Object

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1018	0x00	Number of Entries	U8	ro	4	Number of object entries
	0x01	Vendor ID	U32	ro	327	Manufacturer’s code number (FAULHABER: 327)
	0x02	Product Code	U32	ro	48	Product code number
	0x03	Revision Number	U32	ro	–	Version number
	0x04	Serial Number	U32	ro	–	Serial number

### Receive PDO1 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1600	0x00	Number of Mapped Objects	U8	ro	1	Number of mapped objects
	0x01	RxPDO1 Mapping Entry 1	U32	rw	0x60400010	Pointer to the 16-bit Controlword (0x6040)
	0x02	RxPDO1 Mapping Entry 2	U32	rw	0	
	0x03	RxPDO1 Mapping Entry 3	U32	rw	0	
	0x04	RxPDO1 Mapping Entry 4	U32	rw	0	

## Parameter description

### Receive PDO2 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1601	0x00	Number of Mapped Objects	U8	ro	2	Number of mapped objects
	0x01	RxPDO2 Mapping Entry 1	U32	rw	0x60400010	Pointer to the 16-bit Controlword (0x6040)
	0x02	RxPDO2 Mapping Entry 2	U32	rw	0x607A0020	Pointer to the 32-bit Target Position (0x607A)
	0x03	RxPDO2 Mapping Entry 3	U32	rw	0	
	0x04	RxPDO2 Mapping Entry 4	U32	rw	0	

### Receive PDO3 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1602	0x00	Number of Mapped Objects	U8	ro	2	Number of mapped objects
	0x01	RxPDO3 Mapping Entry 1	U32	rw	0x60400010	Pointer to the 16-bit Controlword (0x6040)
	0x02	RxPDO3 Mapping Entry 2	U32	rw	0x60FF0020	Pointer to the 32-bit Target Velocity (0x60FF)
	0x03	RxPDO3 Mapping Entry 3	U32	rw	0	
	0x04	RxPDO3 Mapping Entry 4	U32	rw	0	

### Receive PDO4 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1603	0x00	Number of Mapped Objects	U8	ro	2	Number of mapped objects
	0x01	RxPDO4 Mapping Entry 1	U32	rw	0x60400010	Pointer to the 16-bit Controlword (0x6040)
	0x02	RxPDO4 Mapping Entry 2	U32	rw	0x60710010	Pointer to the 16-bit Target Torque (0x6071)
	0x03	RxPDO4 Mapping Entry 3	U32	rw	0	
	0x04	RxPDO4 Mapping Entry 4	U32	rw	0	

## Parameter description

### Transmit PDO1 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A00	0x00	Number of Mapped Objects	U8	rw	1	Number of mapped objects
	0x01	TxPDO1 Mapping Entry 1	U32	rw	0x60410010	Pointer to the 16-bit Statusword (0x6041)
	0x02	TxPDO1 Mapping Entry 2	U32	rw	0	
	0x03	TxPDO1 Mapping Entry 3	U32	rw	0	
	0x04	TxPDO1 Mapping Entry 4	U32	rw	0	

### Transmit PDO2 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A01	0x00	Number of Mapped Objects	U8	rw	2	Number of mapped objects
	0x01	TxPDO2 Mapping Entry 1	U32	rw	0x60410010	Pointer to the 16-bit Statusword (0x6041)
	0x02	TxPDO2 Mapping Entry 2	U32	rw	0x60640020	Pointer to the 32-bit Position Actual Value (0x6064)
	0x03	TxPDO2 Mapping Entry 3	U32	rw	0	
	0x04	TxPDO2 Mapping Entry 4	U32	rw	0	

### Transmit PDO3 Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A02	0x00	Number of Mapped Objects	U8	rw	2	Number of mapped objects
	0x01	TxPDO3 Mapping Entry 1	U32	rw	0x60410010	Pointer to the 16-bit Statusword (0x6041)
	0x02	TxPDO3 Mapping Entry 2	U32	rw	0x606C0020	Pointer to the 32-bit Velocity Actual Value (0x606C)
	0x03	TxPDO3 Mapping Entry 3	U32	rw	0	
	0x04	TxPDO3 Mapping Entry 4	U32	rw	0	

## Parameter description

### Transmit PDO Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A03	0x00	Number of Mapped Objects	U8	rw	2	Number of mapped objects
	0x01	TxPDO4 Mapping Entry 1	U32	rw	0x60410010	Pointer to the 32-bit Position Actual Value (0x6064)
	0x02	TxPDO4 Mapping Entry 2	U32	rw	0x60770010	Pointer to the 16-bit Torque Actual Value (0x6077)
	0x03	TxPDO4 Mapping Entry 3	U32	rw	0	
	0x04	TxPDO4 Mapping Entry 4	U32	rw	0	

### SyncManager Communication Type

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C00	0x00	Number of Objects	U8	ro	4	Number of objects
	0x01	SM0 Communication Type	U8	ro	0	0: SyncManager not in use 1: mailbox receive (master to slave)
	0x02	SM1 Communication Type	U8	ro	0	2: mailbox send (slave to master) 3: process data output (master to slave)
	0x03	SM2 Communication Type	U8	ro	0	4: process data input (slave to master)
	0x04	SM3 Communication Type	U8	ro	0	

### SyncManager 2 (RxPDO, master to the drive): PDO Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C12	0x00	Number of Objects	U8	rw	4	Number of objects
	0x01	SM2: 1st RxPDO Assignment	U16	rw	0x1600	Assignment of the SyncManager channel 2 to the receipt PDO 1 Possible values: 0x1600...0x1603
	0x02	SM2: 2nd RxPDO Assignment	U16	rw	0x1601	Assignment of the SyncManager channel 2 to the receipt PDO 2 Possible values: 0x1600...0x1603
	0x03	SM2: 3rd RxPDO Assignment	U16	rw	0x1602	Assignment of the SyncManager channel 2 to the receipt PDO 3 Possible values: 0x1600...0x1603
	0x04	SM2: 4th RxPDO Assignment	U16	rw	0x1603	Assignment of the SyncManager channel 2 to the transmit PDO 4 Possible values: 0x1600...0x1603

## Parameter description

### SyncManager 3 (TxPDO, master to the drive): PDO Mapping

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C13	0x00	Number of Objects	U8	rw	4	Number of objects
	0x01	SM3: 1st TxPDO Assignment	U16	rw	0x1A00	Assignment of the SyncManager channel 3 to the transmit PDO 1 Possible values: 0x1A00...0x1A03
	0x02	SM3: 2nd TxPDO Assignment	U16	rw	0x1A01	Assignment of the SyncManager channel 3 to the transmit PDO 2 Possible values: 0x1A00...0x1A03
	0x03	SM3: 3rd TxPDO Assignment	U16	rw	0x1A02	Assignment of the SyncManager channel 3 to the transmit PDO 3 Possible values: 0x1A00...0x1A03
	0x04	SM3: 4th TxPDO Assignment	U16	rw	0x1A03	Assignment of the SyncManager channel 3 to the transmit PDO 4 Possible values: 0x1A00...0x1A03

### SyncManager 2 (RxPDO, master to the drive): Parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C32	0x00	Number of Objects	U8	ro	12	SyncManager parameters for input PDOs
	0x01	SM2: Synchronization Type	U16	rw	1	Synchronization type: <ul style="list-style-type: none"> <li>▪ 0: FreeRun</li> <li>▪ 1: SM-Sync</li> <li>▪ 2: DC-Sync</li> </ul>
	0x02	SM2: Cycle Time	U32	rw	500000	Cycle time (value must be a multiple of 500000 ns)
	0x04	SM2: Synchronization Types Supported	U16	ro	0	Synchronization types supported
	0x05	SM2: Minimum Cycle Time	U32	ro	0	Minimum cycle time (only in DC-Sync mode)
	0x06	SM2: Calc and Copy Time	U32	ro	0	The earliest time in ns after which the next SyncManager event can arrive (only in DC-Sync mode)
	0x09	SM2: Delay Time	U32	ro	0	Hardware delay time until the outputs are output (only in DC-Sync mode)
	0x0B	SM2: SM-Event Missed Counter	U16	ro	0	Number of SyncManager events missed (only in DC-Sync mode)
	0x0C	SM2: Cycle Time Too Short Counter	U16	ro	0	Error counter that is incremented by 1 if process input data is not updated before the next SM2 event occurs (not in FreeRun mode)

## Parameter description

### SyncManager 3 (TxPDO, drive to the master): Parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C33	0x00	Sync Manager 3(TxPDO): Parameter	U8	ro	12	SyncManager parameters for transmit PDOs
	0x01	SM3: Synchronization Type	U16	rw	34	Synchronization type: <ul style="list-style-type: none"> <li>0: FreeRun</li> <li>2: SM-Sync</li> <li>34: DC-Sync to SM2</li> </ul>
	0x02	SM3: Cycle Time	U32	ro	0	Copy of the value of 0x1C32.02 1C33.02 must be set if no outputs are defined but only inputs. In this case 1C32.02 cannot be set. Normally (i.e. both inputs and outputs are defined), 1C32.02 and 1C33.02 are defined, but both point internally to the same variable. This ensures that only identical times can be used.
	0x04	SM3: Synchronization Types Supported	U16	ro	0	Synchronization types supported
	0x05	SM3: Minimum Cycle Time	U32	ro	0	Minimum cycle time (only in DC-Sync mode)
	0x06	SM3: Calc and Copy Time	U32	ro	0	Time in ns between reading the inputs and availability of the inputs to the master (only in DC-Sync mode)
	0x0B	SM3: SM-Event Missed Counter	U16	ro	0	Number of SyncManager events missed (only in DC-Sync mode)
	0x0C	SM3: Cycle Time Too Short Counter	U16	ro	0	Error counter that is incremented by 1 if process input data is not updated before the next SM2 event occurs (not in FreeRun mode)

## Parameter description

### 5.2 Manufacturer-specific objects

#### FAULHABER error register (0x2320)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2320	0x00	Fault Register	U16	ro	–	FAULHABER error register

#### Error Mask (0x2321)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2321	0x00	Number of Entries	U8	ro	6	Number of object entries
	0x01	Emergency Mask	U16	rw	0xFFFF	Errors for which an error message is sent
	0x02	Fault Mask	U16	rw	0x0000	Errors for which the state machine of the drive switches into <i>Fault Reaction Active</i> state
	0x03	Error Out Mask	U16	rw	0x0000	Errors for which the error output pin is set
	0x04	Disable Voltage Mask	U16	ro	0x4024	Errors which switch off the drive (not configurable)
	0x05	Disable Voltage User Mask	U16	rw	0x0000	Errors which switch off the drive (configurable)
	0x06	Quick Stop Mask	U16	rw	0x0000	Errors for which the state machine of the drive switches into <i>Quick Stop Active</i> state

The states of the drive state machine are described in the documentation for the drive functions.

#### Trace Configuration

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2370	0x00	Number of Entries	U8	ro	10	Number of object entries
	0x01	Trigger Source	U32	wo	0	Trigger source
	0x02	Trigger Threshold	S32	rw	0	Trigger threshold
	0x03	Trigger Delay Offset	S16	rw	0	Trigger delay
	0x04	Trigger Mode	U16	rw	0	Trigger mode
	0x05	Buffer Length	U16	rw	100	Buffer length
	0x06	Sample Time	U8	rw	1	Recording sampling rate 1: in every sampling step
	0x07	Trace Source of Channel 1	U32	wo	0	Trace source of channel 1
	0x08	Trace Source of Channel 2	U32	wo	0	Trace source of channel 2
	0x09	Trace Source of Channel 3	U32	wo	0	Trace source of channel 3
	0x0A	Trace Source of Channel 4	U32	wo	0	Trace source of channel 4

## Parameter description

### Trace Buffer

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2371	0x00	Number of Entries	U8	ro	5	Number of object entries
	0x01	Trace State	U16	ro	0	Trigger status
	0x02	Trace Value of Channel 1	Vis string	ro	–	Signal buffer, channel 1
	0x03	Trace Value of Channel 2	Vis string	ro	–	Signal buffer, channel 2
	0x04	Trace Value of Channel 3	Vis string	ro	–	Signal buffer, channel 3
	0x05	Trace Value of Channel 4	Vis string	ro	–	Signal buffer, channel 4

### RS232 baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	0x00	Number of Entries	U8	ro	8	Number of object entries
	0x02	RS232 Rate	U8	rw	3	Baud rate index
	0x03	Node ID	U8	rw	1	Node number
	0x08	Explicit Device ID	U16	rw	0	Identification of the drive



