FAULHABER

Communications Manual

MC 5010

MC 5005

MC 5004

MCS

EtherCAT® Technology Group

WE CREATE MOTION

EN

Imprint

Version:
15-04-2016

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

Content

Content

1 About this document

1.1 Validity of this document

This document describes:

- Communication with the drive via EtherCAT
- Basic services provided by the communications structure
- Methods for accessing the parameters
- The drive as viewed by 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 attached sheet.

1.2 Associated documents

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

Manual	Description
Motion Manager 6	Instruction Manual for FAULHABER Motion Manager PC software
Quick start description	Description of the first steps for commissioning and operation of FAULHABER Motion Controllers
Functional manual	Description the operating modes and functions of the drive
Technical manual	Guide 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 Internet page 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.

1.4 List of abbreviations

Abbreviation	Meaning
AL	Application Layer
Attr.	Attribute
CAN	Controller Area Network
CSP	Cyclic Synchronous Position mode
CSV	Cyclic Synchronous Velocity mode
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 (receive a PDO from the drive)
SDO	Service Data Object
SII	Slave Information Interface
PLC	Programmable Logic Controller - PLC
Sxx	Data type signed (negative and positive numbers) with bit size xx
TxPDO	Transmit Process Data Object (send a PDO to the drive)
Uxx	Data type unsigned (positive numbers) with bit size xx

1.5 Symbols and markers

NOTICE!

Risk of damage to equipment.

- Measures for avoidance

Instructions for understanding or optimising the operations

- ✓ Pre-requirement for a requested action
- Request for a single-step action

1. First step of a requested action
   - Result of a step
2. Second step of a requested action

- Result of an 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

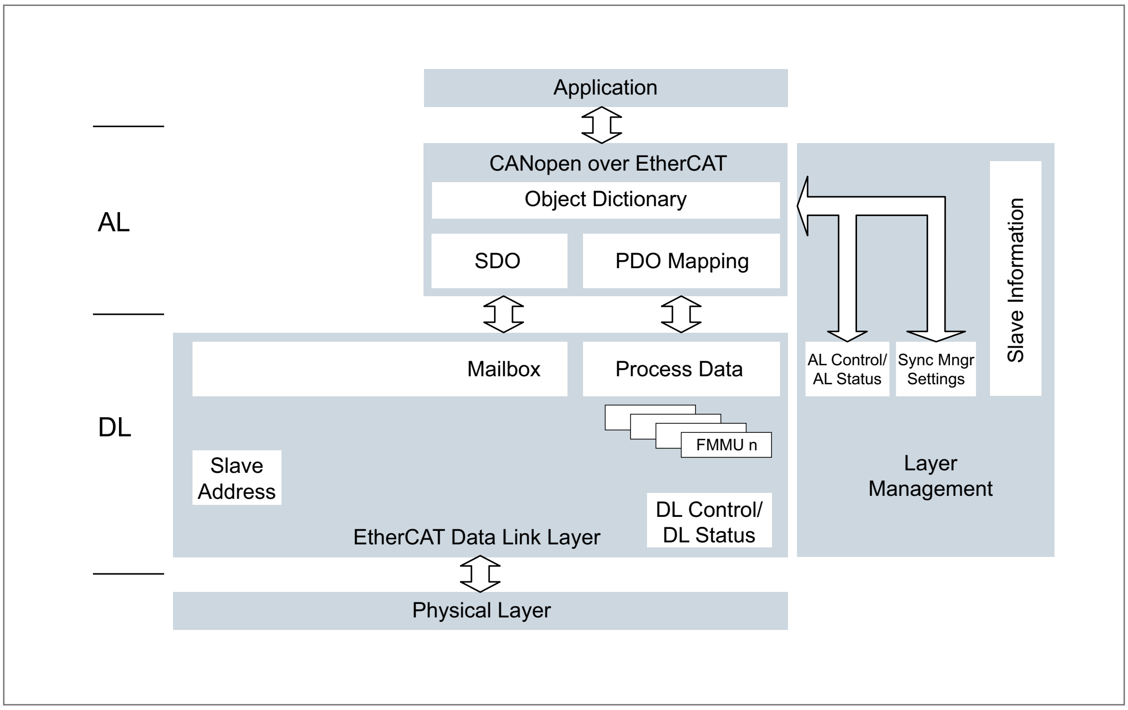

Fig. 1: Basic structure of an EtherCAT device

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 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 are 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 setpoints from the object dictionary and return actual values. The parameters from the object dictionary determine the behaviour of the drive.

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 “Functions Manual”.

2.2 FAULHABER Motion Manager

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

If multiple ports are used concurrently, 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 permits simple access to the settings and parameters of the connected motor controller. 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:

- Motor wizard: Supports the user when configuring an external controller to the connected motor, by selecting the respective FAULHABER motor
- Control setting wizard: Supports the user in optimising the control parameters.

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

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)

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.

If multiple ports are used concurrently, impermissible transitional states may arise.

- Before connecting the Motion Controller into the EtherCAT network, make sure that no other ports (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, p. 10).
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. Define the EtherCAT slave information (see chap. 2.4, p. 11).

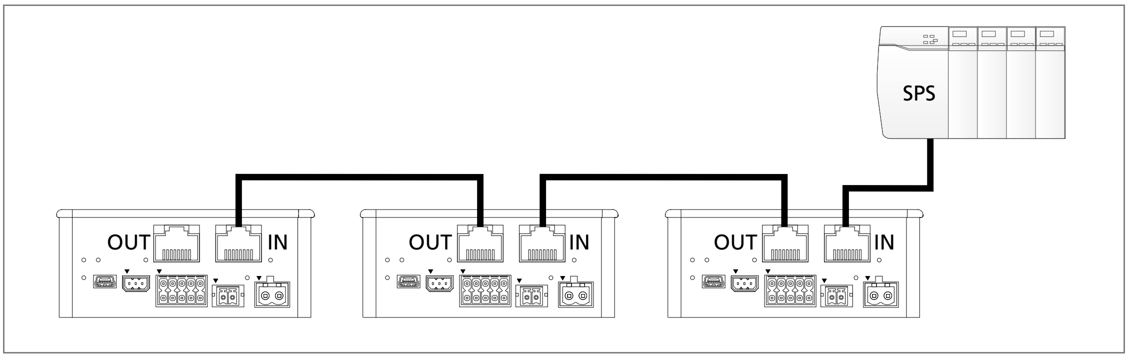

Fig. 2: Connection to the EtherCAT network

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

2.4 ESI file

Information about the connected drive and its behaviour necessary for communication with the EtherCAT slave is held in the ESI (EtherCAT Slave Information) file. (This includes the necessary settings, data types and object dictionary).

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

- In its comprehensive form as an XML file in the scope of supply of the Motion Controller, or available directly from FAULHABER
- In a slightly simplified form on the EtherCAT EEPROM of the Motion Controller (Slave Information Interface, see chap. 3.8.2, p. 27)

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 *Adapt Automatically* 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 altered 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.

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

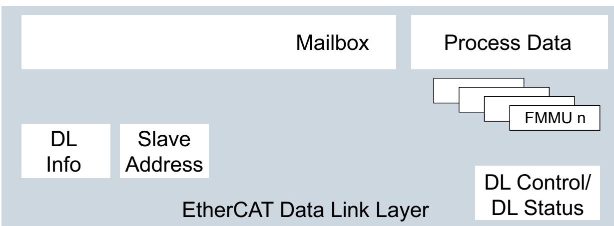

Fig. 3: Data Link Layer

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 Tab. 1, p. 16)
- Access to the EtherCAT EEPROM
- ESC configuration
- Distributed clock (see chap. 3.7.1, p. 24)

- Addressing the EtherCAT slave (see chap. 3.2.3, p. 15)
- SyncManager management (see chap. 3.2.2, p. 14)

3.2.1 EtherCAT frames and datagrams

Frame structure

An EtherCAT frame consists of up to 1518 bytes and can contain up to 1450 bytes of user data.

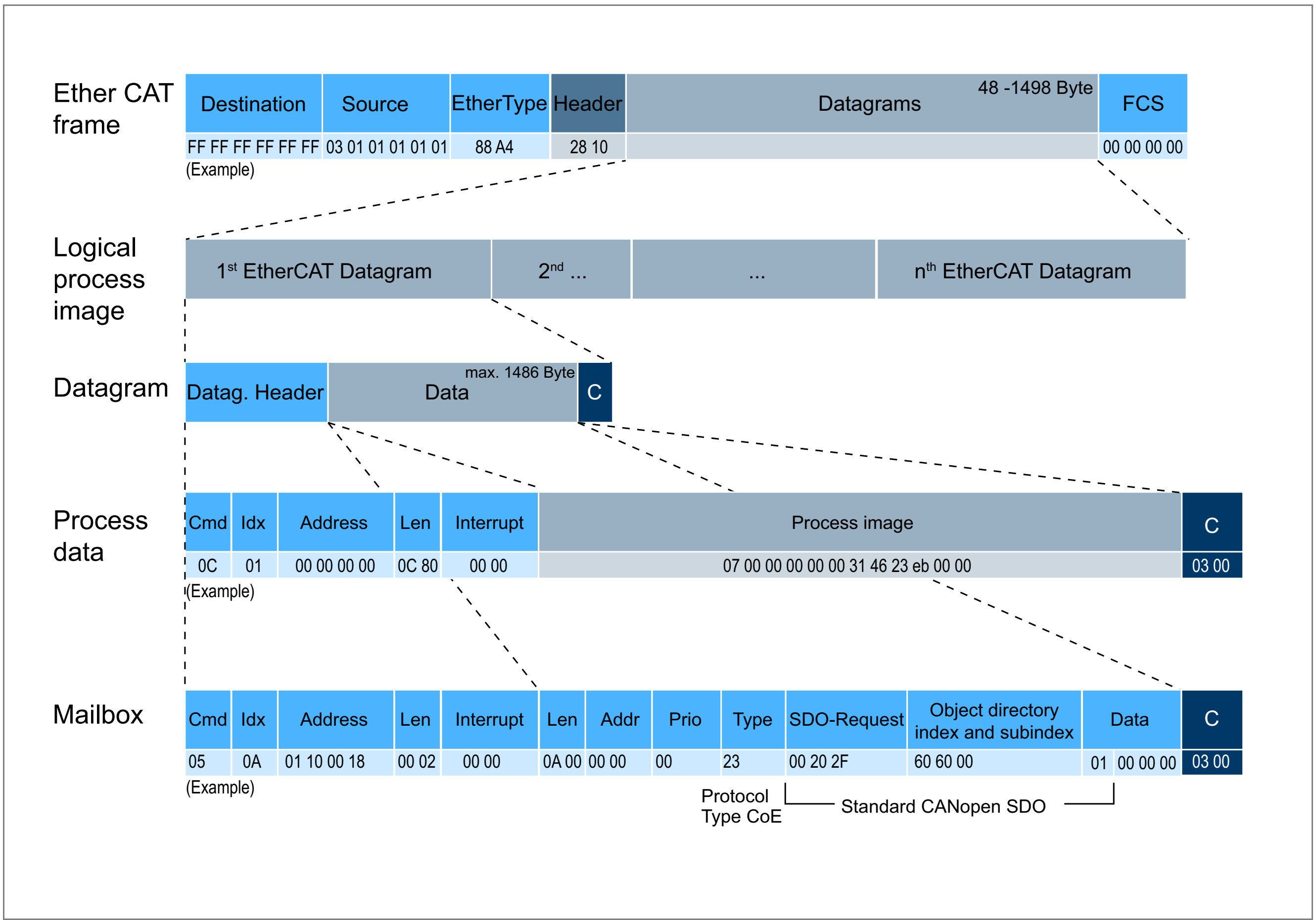

Fig. 4: Frame structure

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: These data are used to check the freedom from errors (see chap. 3.10.2, p. 30).

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. 30).

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

The path of an EtherCAT frame
The EtherCAT master uses the first of the pair of wires to send EtherCAT frames over a pair of wires. Each frame passes through every EtherCAT slave in the network. As the EtherCAT frame passes through, the EtherCAT slave controllers (ESC) can take data from it and insert data into it. Each ESC always has read/write rights only for a certain part of the EtherCAT frame. 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.

The sequence of the data is independent of the physical sequence of the EtherCAT slaves in the network. Broadcast, multicast and intercommunication between the EtherCAT slaves are available.

3.2.2 SyncManager management

Data transmission by the SyncManager
The PDOs and SDOs are read 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. 6.1, p. 38).

Monitoring the read/write access

The SyncManager protects the data exchange memory against simultaneous access by the EtherCAT master and EtherCAT slave. This prevents the risk that of another memory area being overwritten whilst a memory area is being read, so that the data being read are inconsistent.

2 types of memory are available for data exchange:

Memory type	Description
Mailbox memory	The mailbox memory consists of a single memory area. The SyncManager performs the following functions: ▪ Reading of the memory is prevented whilst the memory is being written to. ▪ Writing to the memory is prevented whilst the memory is being read. ▪ The memory is protected against overflow. This type of memory is unsuitable for real-time data and is therefore used only for service data.
Buffer memory for process data	The buffer memory is split into in 3 buffer areas. The SyncManager performs the following functions: ▪ A buffer area that is not currently being written to is selected for writing. Whilst writing is being performed, read access to the memory is blocked. ▪ Once a buffer area has been written to, it is read-enabled. Whilst reading is being performed, write access is blocked. The following buffer areas are available at all times: ▪ Buffer area 1, which is currently being written to ▪ Buffer area 2, which is currently being read ▪ Buffer area 3, which has been written to an is enabled for reading (in the event that the read access in buffer area 2 is completed before the write access to buffer area 1 is complete) If the write access to buffer area 1 is completed before the read access in buffer area 2 is completed, the data in buffer area 1 are read-enabled and the data in buffer area 3 are discarded.

3.2.3 Addressing

The EtherCAT protocol permits the following addressing procedure:

- Position addressing: The physical positions of the EtherCAT slaves in the network act as addresses. In each EtherCAT slave, a specific area of memory is reserved for the address.
- Node addressing: Configured node addresses which the EtherCAT master assigned to the EtherCAT slaves at commissioning, serve as addresses. In each EtherCAT slave, a specific area of memory is reserved for the address.
- Logical addressing: The entire memory area of the networks, i.e. the memory areas of the EtherCAT master and all the EtherCAT slaves, are 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. 19). Transmission of service data is performed acyclically.
Process data	Process data are real-time data. That means that the latest data can always be accessed. Data that are not called up (such as cycle times, for which the data cannot be processed sufficiently quickly by the EtherCAT slave) are 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 service data is performed cyclically.

3.3 Application Layer

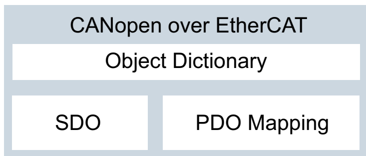

Fig. 5: Application Layer

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

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

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

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

Object Dictionary
The object dictionary contains parameters, set values 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 sub-index (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 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.

3.4 PDO (Process Data Object)

PDOs contain process data for controlling and monitoring the behaviour 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 "*Operational*" state see chap. 3.8.1, p. 26).

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 that is used) is helpful for performance of the mapping procedure.

3.4.2 PDO mapping in the standard configuration (status as delivered)

RxPDO1: Controlword

2 bytes user data	
LB	HB

The RxPDO1 contains the 16-bit control word to CiA 402. 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 an and points to the object index 0x6041 in the object dictionary. The bit distribution is described in the documentation for the drive functions.

RxPDO2: Control word, Target Position (PP and CSP)

6 bytes user data					
LB	HB	LLB	LHB	HLB	HHB

The RxPDO2 contains the 16-bit control word and the 32-bit value of the destination 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).

RxPDO3: Control word, Target Velocity (PV and CSV)

6 bytes user data					
LB	HB	LLB	LHB	HLB	HHB

The RxPDO3 contains the 16-bit control word and the 32-bit value of the set 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: Control word, 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)

3.4.3 Dealing with mapping errors

If the mapping procedure specified in CiA 301 is not complied with, one of the following SDO errors will be returned:

Tab. 2: SDO errors in response the incorrect mapping procedure

SDO error	Meaning	Cause
0x06090030	General value range error	The mapping parameter lies outside that specified in the mapping procedure.
0x06020000	Object not present in the object dictionary	The value for the number of mapped objects is greater than the number of valid entries in the respective sub-indexes for the mapping parameter objects.

Other mapping errors are described in the SDO error table (see chap. 3.5.1, p. 20).

3.5 SDO (Service Data Object)

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

Tab. 3: 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. 4: Distribution of the SDO types of transmission

Type of transmission	Number of bytes	Purpose
Expedited transfer	maximum 250 bytes	–
Segmented transfer	any size	Transmission of data blocks (such as the trace buffer)

The types of transfer are described in CiA 301.

3.5.1 SDO error description

If the SDO protocol cannot be processed further, an SDO-Abort telegram is sent. The error types are coded as follows:

- Error0: Additional error code HB
- Error1: Additional error code LB
- Error2: Error code
- Error3: 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	Sub-index not present
0x06	0x09	0x0030	General value range error
0x06	0x09	0x0031	Value range error: Parameter value too large
0x06	0x09	0x0032	Value range error: Parameter value too small
0x06	0x09	0x0036	Value range error: Maximum value greater 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

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 a slower procedure, we advise against the use of emergency. A better procedure is to map the error register 1001h or the Faulhaber error register 2320h in a PDO. This ensures that the Master receives error information in the shortest possible time.

The emergency object is always size 8 bytes:

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

Assignment of user data:

- Error0(LB)/Error1(HB): 16-bit error code
- Error register: Error register (contents of object 0x1001, see chap. 6.1, p. 38)
- FE0(LB)/FE1(HB): 16-bit FAULHABER error register (contents of object 0x2320, see Tab. 8, p. 28)
- 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 allows interrogation of 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 detection of errors that lead to subsequent errors.

Tab. 5, p. 22 lists all the errors that have been reported by emergency messages, providing the respective error is included in the emergency mask for the FAULHABER error register (Tab. 9, p. 28).

Tab. 5: 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	Overvoltage error	2	Voltage error
0x3220	Undervoltage	0x0008	3	Undervoltage error	2	Voltage error
0x43F0	Temperature warning	0x0010	4	Temperature warning	1	Current error*
0x4310	Temperature error	0x0020	5	Temperature error	3	Temperature error
0x5410	Output stages	0x0080	7	IntHW error	7	Manufacturer-specific error
0x5530	EEPROM fault	0x0400	10	Memory error	–	–
0x7200	Measurement circuit: Current measurement	0x0200	9	Current measurement error	7	Manufacturer-specific error
0x7300	Sensor fault (encoder)	0x0040	6	Encoder error	7	Manufacturer-specific error
0x7400	Computation circuit: Module fault	0x0100	8	Module error	7	Manufacturer-specific error
0x6100	Software error	0x1000	12	Calculation error	7	Manufacturer-specific error
0x8130 0x8210 0x8310	Process data watchdog expired PDO length RS232 overrun	0x0800	11	Communications error	4	Communications error
0x84F0	Deviation error (velocity controller)	0x0001	0	Speed deviation error	5	Drive-specific error
0x8611	Following error (position controller)	0x0002	1	Following error	5	Drive-specific error

* *The current regulator 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 inTab. 6, p. 23 is sent in the following event:

- In the Error Mask 0x2321, bit 1 (following error) is set under sub-index 1 (emergency mask) (see Tab. 10, p. 29).
- The control deviation value set in object 0x6065.00 for the position regulator corridor has been exceeded for an extended period as defined by the value set for the error delay time in object 0x6066.00 (see the Functional Manual).

Tab. 6: Example of user data assignment to an emergency message

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

3.7 Synchronisation

FAULHABER Motion Controllers support synchronisation by means of distributed clocks and via a SyncManager event. The type of synchronisation 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 synchronisation (FreeRun), the EtherCAT slave operates independently according to its own clock, which is set by the cycle time 0x1C32.02
- 1 or 34: Synchronisation via a SyncManager event (see chap. 3.7.2, p. 25)
- 2: Synchronisation via Distributed Clocks (see chap. 3.7.1, p. 24)

Only the following combinations are permitted for this:

0x1C32.01 SM2		0x1C33.01		Process data receipt
00h	FreeRun	00h	FreeRun	No checking of the cycle time
01h	SM synchronous	22h	SM synchronised with SM2	The cycle time is monitored if it is entered with a value > 0
02h	DC Sync0	02h	DC Sync0	The cycle time is monitored

The cycle time generated by the master must always be a multiple of 500 µs. 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 µs.

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 *Slot* allows the right SM-configuration to be generated easily and without errors.

3.7.1 Synchronisation via distributed clocks (DC-Sync)

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

For synchronisation 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 the 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, by default the first EtherCAT slave after the EtherCAT master is selected as the reference slave.

The reference time read by the EtherCAT participants is always corrected by the respective EtherCAT participants to allow for the travel time taken by the datagram from the reference clock. 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 drives has an internal master clock that is synchronised to the master clock of the reference slave. The synchronisation makes an allowance for the telegram travel time. The internal master clock generates a Sync0 signal which starts the local cycle of the drives.

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

The master should send a telegram within the same cycle as the cycle time of the slaves, so that the data that are processed are always the current data from the slave. If due to a jitter in the cycle of the master a packet is set 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 µs or a multiple thereof.

3.7.2 Synchronisation 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 transmit PDOs are transmitted cyclically, the EtherCAT slave is synchronised at the SyncManager2 event (SM2 event). If only cyclical receipt PDOs are transmitted, the EtherCAT slave is synchronised at the SyncManager3 event (SM3 event).

The parameter for synchronisation via a SyncManager event are set via the objects 0x1C32 (SM2) and 0x1C33 (SM3) in the *Pre-Operational* state (see chap. 6.1, p. 38).

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 arrive at the slave, if 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 can be deactivated.

3.8 Layer management

The Layer Management makes available the following services:

- Controlling the EtherCAT state machine (chap. 3.8.1, p. 26)
- Reading and writing at the Slave Information Interface (see chap. 3.8.2, p. 27)

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

3.8.1 Controlling the EtherCAT state machine

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

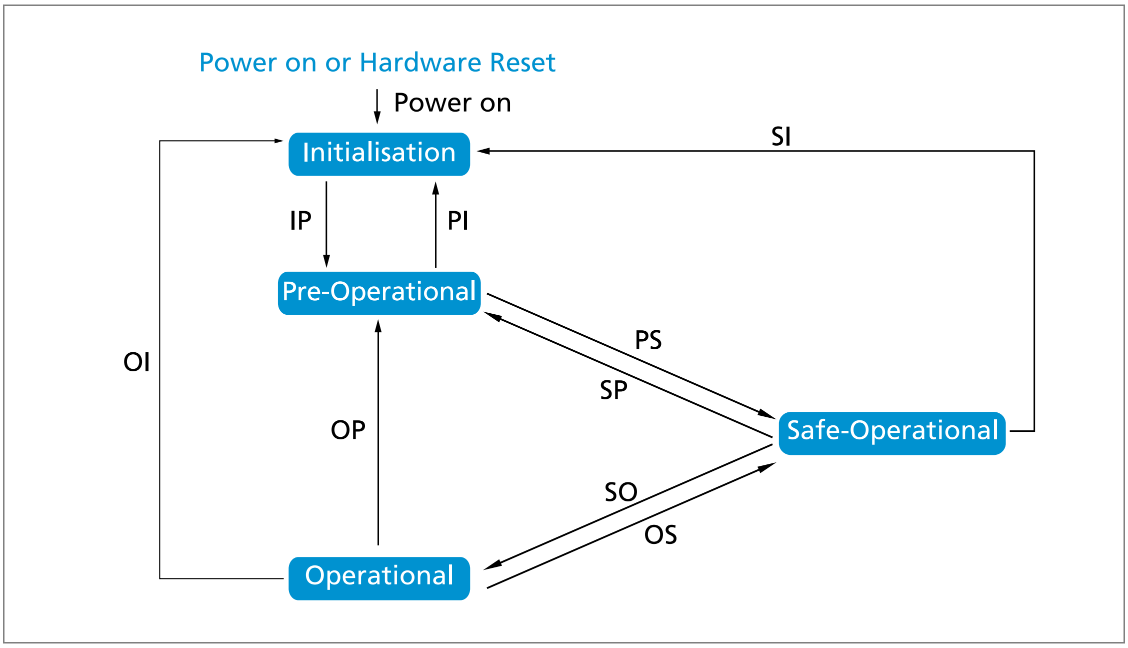

Fig. 6: EtherCAT state machine

Tab. 7: Changes of state

Transition	Actions
Power on	▪ The initialisation state is achieved automatically on switching on. ▪ Neither mailbox communication nor process data communication are available. ▪ The EtherCAT master initialises the SyncManager channels for mailbox communication.
IP	▪ The EtherCAT master synchronises the EtherCAT field bus. ▪ The EtherCAT master initialises the SyncManager channels for process data communication, the FMMU channels and the SyncManager-PDO assignment. ▪ Mailbox communication is established between the EtherCAT master and EtherCAT slaves. ▪ Settings for process data transmission are transmitted.
PS	▪ The EtherCAT slave checks that the SyncManager channel for process data communication and the settings for the Distributed Clocks are correct. ▪ The EtherCAT slave copies the current input data in the memory areas of the ESC. ▪ Mailbox and process data communication are now available. The outputs from the EtherCAT slave remain in a safe state and are not output. The input data are updated cyclically.
SO	▪ The EtherCAT master transmits valid output data to the EtherCAT slave. ▪ The EtherCAT master switches the EtherCAT slave into the *Operational* state. ▪ In the *Operational* state, the EtherCAT slave copies the input data to its outputs. ▪ Mailbox and process data communication are now available.

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 parametrisation is necessary at system start.

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

In the event of a serious communications error the Motion Controller switches by default to the *Pre-Operational* state. Different behaviour can be set using the object 0x1029.

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

The drive is controlled by objects of the drive profile (control word, status word). 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. Prior to this there has been no SDO communication in operation.

3.8.2 Slave Information Interface (SII)

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

These data are saved in the EtherCAT EEPROM which is read 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 sub-index (SDO protocol).

- The communication parameters area (index 0x1000 to 0x1FFF) contains communications objects to CiA 301 (see chap. 6.1, p. 38)
- The manufacturer-specific area (index 0x2000 to 0x5FFF) contains manufacturer-specific objects (see chap. 6.2, p. 46)
- The standardised device profiles area (0x6000 to 0x9FFF) contains objects supported by the Motion Controller (see the documentation of the drive functions)

3.10 Error handling

3.10.1 Equipment faults

Tab. 8: FAULHABER error register (0x2320)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2320	00	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 object (0x2321).

Tab. 9: Error coding

Error bit	Error message	Description
0x0001	Speed deviation error	Speed deviation too great
0x0002	Following error	Following error
0x0004	Overvoltage error	Overvoltage detected
0x0008	Undervoltage error	Undervoltage detected
0x0010	Temperature warning	Temperature exceeds that at which a warning is output
0x0020	Temperature error	Temperature exceeds that at which an error message is output
0x0040	Encoder error	Error detected at the encoder
0x0080	IntHW error	Internal hardware error
0x0100	Module error	Error at the external module
0x0200	Current measurement error	Current measurement error
0x0400	Memory error	Memory error (EEPROM)
0x0800	Communications error	Communication errors
0x1000	Calculation error	Internal software error
0x2000	–	Not used, value = 0
0x4000	–	Not used, value = 0
0x8000	–	Not used, value = 0

All of these errors correspond to the emergency error code (see chap. 3.6, p. 21).

The error mask describes the handling of internal errors depending on the error coding (see Tab. 9, p. 28).

Tab. 10: Error mask (0x2321)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2321	00	Number of entries	U8	ro	6	Number of object entries
	01	Emergency mask	U16	rw	0xFFFF	Errors that trigger an emergency telegram
	02	Fault mask	U16	rw	0x0000	Errors that are treated as DSP402 errors and affect the status of the machine (error condition)
	03	Error Out mask	U16	rw	0x0000	Errors that set the error output
	04	Disable voltage mask	U16	ro	0x0024	–
	05	Disable voltage user mask	U16	rw	0x0000	–
	06	Quick stop mask	U16	rw	0x0000	–

For example:

- When the fault mask (sub-index 2) of object 0x2321 is set to 0x0001 the drive is switched off due to overcurrent are set to an error state.
- When the sub-index 3 of object 0x2321 is set to 0, the error output (fault pin) indicates no error. When the sub-index 3 of object 0x2321 is set to 0xFFFF, the error output (fault pin) indicates all errors.

The error handling object (0x2322) is used for additional settings regarding error handling.

Tab. 11: Error handling (0x2322)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2322	00	Number of entries	U8	ro	2	Number of object entries
	01	Error delay	U16	rw	200	Error delay time in 1/100 s, value = 0 to 65,000

3.10.2 Communication errors

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

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 is 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 following EtherCAT slave 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.

SyncManager watchdog
The EtherCAT slaves cannot enquire on the EtherCAT master for any data or information, and thus they cannot report the absence of an EtherCAT frame.

The SyncManager watchdog monitors the output from the SyncManager and thus the updating of data coming from the EtherCAT master. A time window ±100 µs around the target point in time is specified as the update period. If averaged over 10 frames the data was not updated within this time window, the SyncManager watchdog reports time exceeded and changes the operating state from *Operational* to *Safe-Operational* (see chap. 3.8, p. 25). The response to this timeout is specified in the object 0x6007.

Analysis of the network traffic
The network traffic can be analysed 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 (IP, SPS). 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.

3.11 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 (application EEPROM) (see chap. 6.1, p. 38). When the motor is switched on, the parameters are loaded automatically from the non-volatile memory into the volatile memory (RAM).

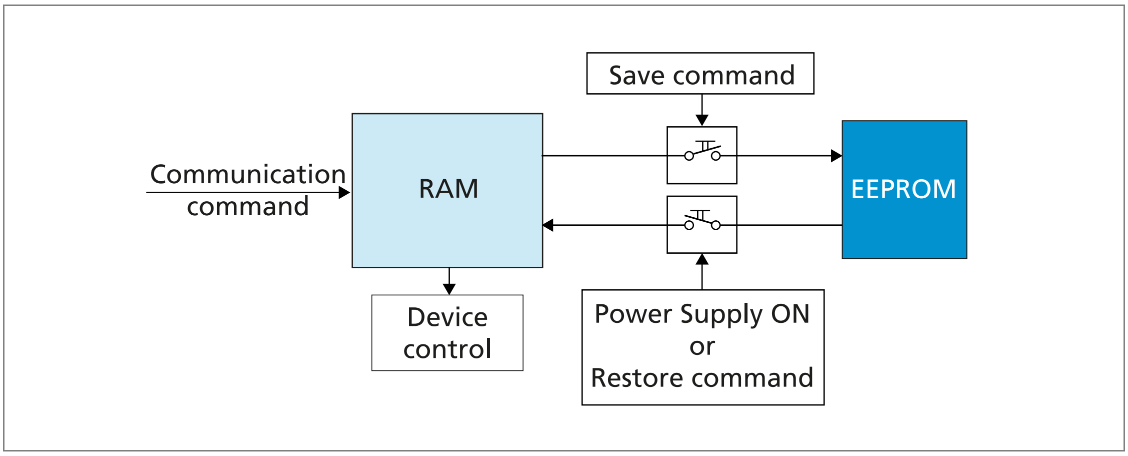

Fig. 7: Saving and restoring parameters

The following parameters can be loaded using the “Restore” command (see chap. 6.1, p. 38):

- Factory settings
- Parameters saved using the “Save” command

3.11.1 Saving parameters

The current parameter settings can be saved in the internal EEPROM (SAVE) (see Tab. 16, p. 39), either completely or for individual ranges.

- Write the “Save” signature to the sub-index 01 to 05 of the object 0x1010 (see Tab. 17, p. 39).

3.11.2 Restoring settings

When the drive is next 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. 18, p. 40).

1. Write the “Load” signature to the sub-index 01 to 06 of the object 0x1011 (see Tab. 19, p. 40).
   - After Restore Factory (01), Restore Communication (02) and Restore Application (03), the parameters are updated only after a reset.
2. Application parameters (04), together with record 1 and record 2 of the special application parameters (05/06) can be updated with the “Restore” command.
   - The “Restore” 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).

4 Trace recorder

The trace recorder allows recording up to 4 parameters of the controller. A trigger source is available for this in the object dictionary. This allows selection of a maximum of 4 signal sources. The parameter values are written to an internal buffer and can then be read (see). The advantage compared to the cyclical transmission of process data is the higher speed. The trace recorder can record controller data at a sampling interval of 100 µs. By comparison, process data can be transmitted only at intervals of 500 µs.

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 the 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. 35).

4.1 Trace settings

The 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. 12: Trace configuration (0x2370)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2370	00	Number of entries	U8	ro	11	Number of object entries
	01	Trigger source	U32	rw	0	Trigger source for the trigger type “Threshold” (see below)
	02	Trigger threshold	S32	rw	0	Trigger threshold
	03	Trigger delay	S32	rw	0	Trigger delay (see below)
	04	Trigger mode	U16	rw	0	Trigger type (see below)
	05	Buffer size	U16	rw	100	Length of the buffer in sampling values.
	06	Sample time	U8	rw	1	Sampling rate of the recording in multiples of the controller sampling time
	07	Source 1	U32	rw	0	1st parameter to be recorded (see below)
	08	Source 2	U32	rw	0	2nd parameter to be recorded (see below)
	09	Source 3	U32	rw	0	3rd parameter to be recorded (see below)
	0A	Source 4	U32	rw	0	4th parameter to be recorded (see below)

Trigger source (0x2370.01), source 1 to 4 (0x2370.07 to 0A)
The parameters to be recorded, source 1 to source 4, must be entered into the objects 0x2370.07 to 0x2370.0A as pointers to a corresponding object entry (index and sub-index

of the desired parameter). The trigger source must be entered into the object 0x2370.01 as a pointer to a corresponding object entry (index and sub-index of the desired parameter).

Example:

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

Trigger threshold (0x2370.02)
The trigger threshold is entered into object 0x2370.02.

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

Trigger delay (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: Recording is started at a time defined as the set multiple times the sample time.
- Delay < 0: Negative delays can be performed up to the length of the buffer. Recording ends at the point in the ring buffer where the recording for the current trigger would have had to start. This ensures that the values recorded before the trigger are retained.

Trigger mode (0x2370.04)
The trigger type and the type of the data sources are determined by the object 0x2370.04. Bit 0 activates the trigger and thus providing the trigger conditions are satisfied starts the recording.

Tab. 13: Trigger mode (0x2370.04)

Bit	Entry	Description
0 (LSB)	EN	▪ 0: No trigger active ▪ 1: Trigger active. Is automatically reset in trigger modes 1 and 3
1 2 3	Edge 0 Edge 1 Edge 2	▪ 0: rising flank or trigger > threshold ▪ 1: falling flank or trigger < threshold
4 to 5	Reserved	–
6 7	Mode 0 Mode 1	▪ 0: No trigger ▪ 1: Single shot ▪ 2: Repeating
8 to 10	Reserved	–
11 12 13 14 15 (MSB)	Source type 1 Source type 2 Source type 3 Source type 4 Trigger type	▪ 0: An object dictionary entry is used as the source ▪ 1: Not currently supported

Buffer size (0x2370.05)
The length of the buffer available for recording is set in object 0x2370.05. The permissible length is dependent on the data type of the parameter to be recorded. A maximum buffer of 2 kB per data source is available.

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

4.2 Reading the trace buffer

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

Tab. 14: Trace buffer (0x2371)

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2371	00	Number of entries	U8	ro	6	Number of object entries
	01	Trigger status	U8	ro	0	Status and index of the first word in the buffer
	02	Value source 1	U8	ro	0	Value of source 1
	03	Value source 2	U8	ro	0	Value of source 2
	04	Value source 3	U8	ro	0	Value of source 3
	05	Value source 4	U8	ro	0	Value of source 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 recorded to the highest resolution of the trace recorder.

Trigger status (0x2371.01)

Tab. 15: Trigger status (0x2371.01)

Bit	Entry	Description
0 (LSB) 1	Status 0 Status 1	▪ 0: No trigger active ▪ 1: Trigger not yet reached ▪ 2: Recording not yet completed ▪ 3: Recording completed, data are available
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 trigger status 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 the segmented SDO upload protocol.

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).
   ↳ This completes the settings for the trace recorder.
7. Test the trigger status (2371.01) at the value **3**.
8. Read the recorded content of the buffer (2371.02 to 05).

5 Troubleshooting

If despite the device being used properly nevertheless unexpected malfunctions occur, please contact your responsible support partner.

6 Parameter description

6.1 Communication objects to CiA 301

Device type

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1000	00	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 = 0x192 (402d)
- Byte LSB (Least Significant Byte): 0x42 (servo drive, type-specific PDO mapping)

Error register

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1001	00	Error register	U8	ro	Yes	Error register

The error register contains the a record of the most recent errors, 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	00	Number of errors	U8	rw	–	Number of errors stored
	01	Standard error field	U32	ro	–	Last error
	02	Standard error field	U32	ro	–	Last error but one

The error log contains the coding for the last error to occur.

- Byte MSB: Error register
- Byte LSB: Error code

The meaning of the error codes is described in chap. 3.6, p. 21.

Writing a 0 to the sub-index 0 clears down the error log.

Manufacturer's device name

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1008	00	Manufacturer's device name	Vis string	const	–	Device name

Manufacturer's hardware version

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1009	00	Manufacturer's hardware version	Vis string	const	–	Hardware version

Manufacturer's software version

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x100A	00	Manufacturer's software version	Vis string	const	–	Software version

Store parameters

Tab. 16: Saving parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1010	00	Number of entries	U8	ro	6	Number of object entries
	01	Save all parameters	U32	rw	1	Saves all parameters
	02	Save communication parameters	U32	rw	1	Save the communication parameters (object dictionary entries 0x0000 to 0x1FFF)
	03	Save application parameters	U32	rw	1	Save the application parameters (object dictionary entries 0x2000 to 0x6FFF)
	04	SAVE application parameters 1	U32	rw	1	Save application parameters for direct changes (set 1)
	05	SAVE application parameters 2	U32	rw	1	Save application parameters for direct changes (set 2)
	06	Save calibration parameters	U32	ro	1	Save calibration parameters

The "Save 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 sub-index initiates the save procedure.

Tab. 17: "Save" signature

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. 18: Restoring the parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1011	00	Number of entries	U8	ro	6	Number of object entries
	01	Restore all factory parameters	U32	rw	1	Restore all factory settings
	02	Restore factory communication	U32	rw	1	Restore all factory settings for communications parameters (0x0000 to 0x1FFF)
	03	Restore factory application	U32	rw	1	Restore all factory settings for application parameters (from 0x2000)
	04	Restore factory application parameters	U32	rw	1	Restore the user's last saved settings for application parameters (from 0x2000)
	05	Restore parameter set	U32	rw	1	Application parameters for direct changes (set 1)
	06	Restore parameter set	U32	rw	1	Application parameters for direct changes (set 2)

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

Tab. 19: “Load” signature

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

The status as delivered 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	00	Number of entries	U8	ro	4	Number of object entries
	01	Vendor ID	U32	ro	327	Manufacturer's code number (FAULHABER: 327)
	02	Product code	U32	ro	–	Product code number
	03	Revision number	U32	ro	–	Version number
	04	Serial number	U32	ro	–	Serial number

Error behaviour

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1029	00	Number of entries	U8	ro	1	Number of object entries
	01	Communication error	U8	rw	0	Behaviour in the event of communication errors 0 = Pre-operational status 1 = No change of status 2 = Stopped status

In the event of a serious communications error the Motion Controller switches to the *Pre-Operational* state. Sub-index 1 allows the behaviour in the event of a serious communications error to be changed.

Receive PDO1 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1600	00	Number of mapped objects	U8	ro	1	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60400010	Pointer to the 16-bit controlword (0x6040)
	02	PDO mapping entry 2	U32	rw	0	
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Receive PDO2 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1601	00	Number of mapped objects	U8	ro	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60400010	Pointer to the 16-bit control word (0x6040)
	02	PDO mapping entry 2	U32	rw	0x607A0020	Pointer to the 32-bit target position (0x607A)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Receive PDO3 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1602	00	Number of mapped objects	U8	ro	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60400010	Pointer to the 16-bit control word (0x6040)
	02	PDO mapping entry 2	U32	rw	0x60FF0020	Pointer to the 32-bit target velocity (0x60FF)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Receive PDO4 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1603	00	Number of mapped objects	U8	ro	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60400010	Pointer to the 16-bit control word (0x6040)
	02	PDO mapping entry 2	U32	rw	0x60710010	Pointer to the 16-bit target torque (0x6071)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Transmit PDO1 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A00	00	Number of mapped objects	U8	rw	1	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60410010	Pointer to the 16-bit statusword (0x6041)
	02	PDO mapping entry 2	U32	rw	0	
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Transmit PDO2 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A01	00	Number of mapped objects	U8	rw	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60410010	Pointer to the 16-bit statusword (0x6041)
	02	PDO mapping entry 2	U32	rw	0x60640020	Pointer to the 32-bit position actual value (0x6064)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Transmit PDO3 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A02	00	Number of mapped objects	U8	rw	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60410010	Pointer to the 16-bit statusword (0x6041)
	02	PDO mapping entry 2	U32	rw	0x606C0020	Pointer to the 32-bit velocity actual value (0x606C)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

Transmit PDO4 mapping parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1A03	00	Number of mapped objects	U8	rw	2	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x60410010	Pointer to the 32-bit position actual value (0x6064)
	02	PDO mapping entry 2	U32	rw	0x60770010	Pointer to the 16-bit torque actual value (0x6077)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

SyncManager communication type

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C00	00	Number of objects	U8	ro	4	Number of objects
	01	SM1 communication type	U8	ro	0	0: SyncManager not in use 1: mailbox receive (master to slave) 2: mailbox send (slave to master) 3: process data output (master to slave) 4: process data input (slave to master)
	02	SM2 communication type	U8	ro	0	
	03	SM3 communication type	U8	ro	0	
	04	SM4 communication type	U8	ro	0	

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

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C12	00	Number of objects	U8	rw	4	Number of objects
	01	SM2: 1st RxPDO assignment	U16	rw	0x1600	Assignment of the SyncManager channel 2 to the receipt PDO 1 Available values: 0x1600...0x1603
	02	SM2: 2nd RxPDO assignment	U16	rw	0x1601	Assignment of the SyncManager channel 2 to the receipt PDO 2 Available values: 0x1600...0x1603
	03	SM2: 3rd RxPDO assignment	U16	rw	0x1602	Assignment of the SyncManager channel 2 to the receipt PDO 3 Available values: 0x1600...0x1603
	04	SM2: 4th RxPDO assignment	U16	rw	0x1603	Assignment of the SyncManager channel 2 to the receipt PDO 4 Available values: 0x1600...0x1603

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

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C13	00	Number of objects	U8	rw	4	Number of objects
	01	SM3: 1st TxPDO assignment	U16	rw	0x1A00	Assignment of the SyncManager channel 3 to the transmit PDO 1 Available values: 0x1A00...0x1A03
	02	SM3: 2nd TxPDO assignment	U16	rw	0x1A01	Assignment of the SyncManager channel 3 to the transmit PDO 2 Available values: 0x1A00...0x1A03
	03	SM3: 3rd TxPDO assignment	U16	rw	0x1A02	Assignment of the SyncManager channel 3 to the transmit PDO 3 Available values: 0x1A00...0x1A03
	04	SM3: 4th TxPDO assignment	U16	rw	0x1A03	Assignment of the SyncManager channel 3 to the transmit PDO 4 Available values: 0x1A00...0x1A03

SyncManager 2 (RxPDO, master to the drive): Parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C32	00	Number of objects	U8	ro	12	Sync-Manager parameters for input PDOs
	01	SM2: Synchronisation type	U16	rw	1	Synchronisation type: ▪ 0: FreeRun ▪ 1: SM-Sync ▪ 2: DC-Sync
	02	SM2: Cycle time	U32	rw	500000	Cycle time (the value must be a multiple of 500000 ns)
	04	SM2: Synchronisation types supported	U16	ro	0	Synchronisation types supported
	05	SM2: Minimum cycle time	U32	ro	0	Minimum cycle time (only in DC-Sync mode)
	06	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)
	09	SM2: Delay time	U32	ro	0	Hardware delay time to outputting the outputs (only in DC-Sync mode)
	11	SM2: SM event missed counter	U16	ro	0	Number of SM events missed (only in DC-Sync mode)
	12	SM2: Cycle time too short counter	U16	ro	0	Error counter that is incremented by 1 when process input data are not updated before the next SM2 event arrives (not in FreeRun mode)

SyncManager 3 (TxPDO, drive to the master): Parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1C33	00	SyncManager 3 (TxPDO): Parameters	U8	ro	12	SyncManager parameters for transmit PDOs
	01	SM3: Synchronisation type	U16	rw	34	Synchronisation type: ▪ 0: FreeRun ▪ 2: SM-Sync ▪ 34: DC-Sync to SM2
	02	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. In the normal case (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 the same periods of time can be used.
	04	SM3: Synchronisation types supported	U16	ro	0	Synchronisation types supported
	05	SM3: Minimum cycle time	U32	ro	0	Minimum cycle time (only in DC-Sync mode)
	06	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)
	11	SM3: SM event missed counter	U16	ro	0	Number of SM events missed (only in DC-Sync mode)
	12	SM3: Cycle time too short counter	U16	ro	0	Error counter that is incremented by 1 when process input data are not updated before the next SM2 event arrives (not in FreeRun mode)

6.2 Manufacturer-specific objects

FAULHABER error register

Index	Subindex	Name	Typ e	Attr.	Default value	Meaning
0x2320	00	Fault register	U16	ro	–	FAULHABER error register

Error mask

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2321	00	Number of entries	U8	ro	6	Number of object entries
	01	Emergency mask	U16	rw	0x00FF	Errors that trigger an emergency telegram
	02	Fault mask	U16	rw	0x0000	Errors that are treated as DSP402 errors and affect the status of the machine (error condition)
	03	Error Out mask	U16	rw	0x00FF	Errors that set the error output
	04	Disable voltage mask	U16	ro	0x0000	
	05	Disable voltage user mask	U16	rw	0x0000	
	06	Quick stop mask	U16	rw	0x0000	

Trace configuration

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2370	00	Number of entries	U8	ro	11	Number of object entries
	01	Trigger source	U32	rw	0	Trigger source for the trigger type "Threshold"
	02	Trigger threshold	S32	rw	0	Trigger threshold
	03	Trigger delay	S32	rw	0	Trigger delay
	04	Trigger mode	U16	rw	0	Type of trigger
	05	Buffer size	U16	rw	100	Length of the buffer in sampling values.
	06	Sample time	U8	rw	1	Sampling rate of the recording in multiples of the controller sampling time
	07	Source 1	U32	rw	0	1st parameter to be recorded
	08	Source 2	U32	rw	0	2nd parameter to be recorded
	09	Source 3	U32	rw	0	3rd parameter to be recorded
	0A	Source 4	U32	rw	0	4th parameter to be recorded

Trace buffer

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2371	00	Number of entries	U8	ro	6	Number of object entries
	01	Trigger status	U8	ro	0	Status and index of the first word in the buffer
	02	Value source 1	U8	ro	0	Value of source 1
	03	Value source 2	U8	ro	0	Value of source 2
	04	Value source 3	U8	ro	0	Value of source 3
	05	Value source 4	U8	ro	0	Value of source 4

RS232 Baud rate index and node number

Tab. 20: RS232 Baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	00	Number of entries	U8	rw	4	Number of object entries
	02	RS232 Baud rate index	U8	ro	9	Baud rate index
	03	Node ID	U8	rw	255	Node number

7000.05051, 15-04-2016

DR. FRITZ FAULHABER GMBH & CO. KG
Antriebssysteme

Daimlerstraße 23 / 25
71101 Schönaich · Germany
Tel. +49(0)7031/638-0
Fax +49(0)7031/638-100
info@faulhaber.de
www.faulhaber.com