

Communications Manual

MC 5010

MC 5005

MC 5004

MCS

CANopen®

Imprint

Version:
15-04-2016

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

Content

1	About this document.....	5
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 markers.....	7
2	Overview	8
2.1	Basic arrangement of a CANopen device	8
2.2	Pre-conditions for communication	9
2.3	FAULHABER Motion Manager	9
2.4	Saving and restoring parameters	10
2.4.1	Saving parameters.....	10
2.4.2	Restoring settings.....	11
3	CANopen protocol description	12
3.1	Introduction	12
3.2	Communication services.....	13
3.3	Identifier distribution	14
3.4	PDO (Process Data Object)	16
3.4.1	PDO configuration	16
3.4.2	PDO mapping in the standard configuration (status as delivered)	17
3.4.3	Dealing with mapping errors	18
3.5	SDO (Service Data Object).....	19
3.5.1	Expedited transfer.....	19
3.5.2	SDO error description	21
3.6	Emergency object (error message)	22
3.7	SYNC object.....	24
3.7.1	Triggering synchronous PDOs	24
3.8	NMT (Network Management)	25
3.8.1	Boot up	27
3.8.2	Monitoring functions.....	27
3.8.2.1	Node guarding	27
3.8.2.2	Heartbeat	29
3.8.3	Setting the monitoring functions	30
3.9	Entries in the object dictionary.....	30
3.10	Error handling.....	31
3.10.1	CAN error	31
3.10.2	Equipment faults.....	31

Content

4	Trace.....	33
4.1	Trace recorder	33
4.1.1	Trace settings.....	33
4.1.2	Reading the trace buffer	35
4.1.3	Typical execution of the trace function.....	36
4.2	Trace logger	36
5	Communications settings.....	37
5.1	Setting via the CAN network	37
5.1.1	Setting the node number	38
5.1.2	Setting the Baud rate.....	38
5.2	Setting the node number via the object dictionary.....	39
6	Troubleshooting	40
7	Parameter description	41
7.1	Communication objects to CiA 301	41
7.2	Manufacturer-specific objects.....	50

1 About this document

1 About this document

1.1 Validity of this document

This document describes:

- Communication with the drive via CANopen
- 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 CAN-BUS experience, and for CAN-BUS 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 About this document

1.4 List of abbreviations

Abbreviation	Meaning
Attr.	Attribute
CAN	Controller Area Network
CiA	CAN in Automation e.V.
COB ID	Communication Object Identifier
CS	Command Specifier
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMCY	Emergency
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
NMT	Network Management Object
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
PLC	Programmable Logic Controller - PLC
Sxx	Data type signed (negative and positive numbers) with bit size xx
SYNC	Synchronisation Object
TxPDO	Transmit Process Data Object (send a PDO to the drive)
Uxx	Data type unsigned (positive numbers) with bit size xx

1 About this document

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

2 Overview

2.1 Basic arrangement of a CANopen device

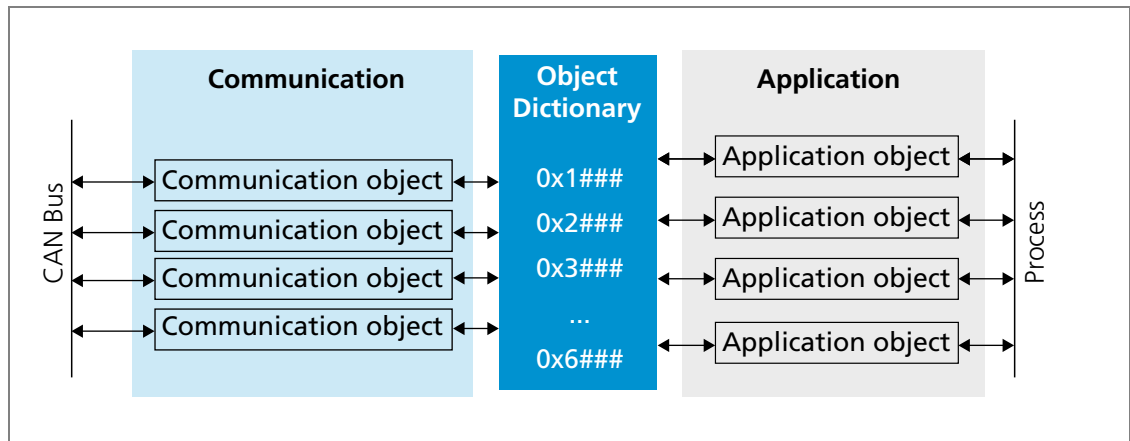


Fig. 1: Basic arrangement of a CANopen device

Communication services

The CANopen master communicates with the object dictionary via the bus system and use of the communication services (see chap. 3.2, p. 13).

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.

Application part

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.

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 Overview

2.2 Pre-conditions for communication

FAULHABER drives are delivered in the unconfigured state. For operation in a CAN network, a unique node number must be assigned and a Baud rate set at commissioning (see chap. 5, p. 37).

After switching on and initialising, the Motion Controller is at first in the *Pre-Operational* state. In order to be able to perform drive functions, the Motion Controller must be brought into the *Operational* state (see chap. 3.8, p. 25).

1. Connect the controller to a power supply (supply at least to the electronics).
2. Connect CAN_H, CAN_L, GND to the respective terminals of a host-side CAN connection.
3. Switch on the power and establish a connection via the configuration application.

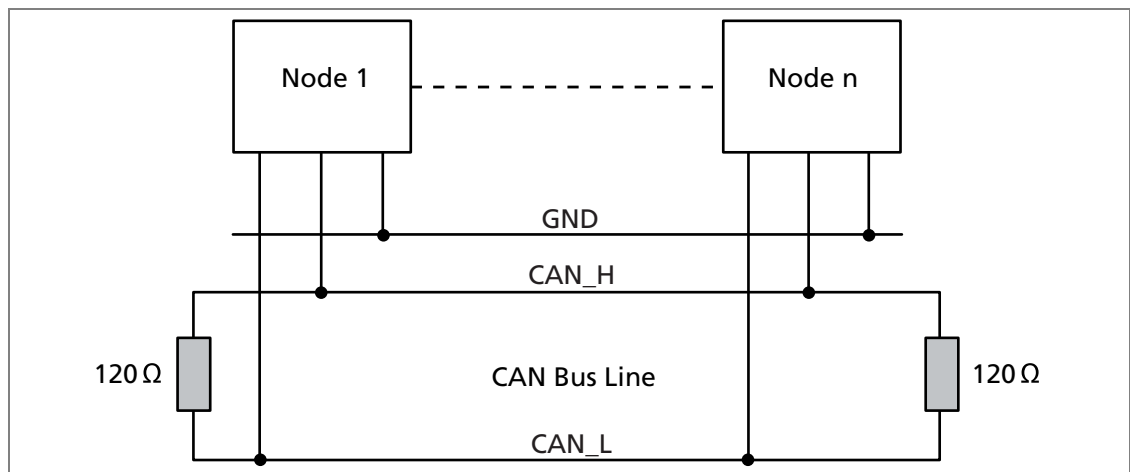


Fig. 2: Connection to the CANopen network

2.3 FAULHABER Motion Manager

We recommend that the first commissioning of a FAULHABER drive is performed using "FAULHABER Motion Manager" software. 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:

- Connection wizard: Supports the user when establishing the connection to the connected controller
- 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.

2 Overview

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.

2.4 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. 7.1, p. 41). When the motor is switched on, the parameters are loaded automatically from the non-volatile memory into the volatile memory (RAM).

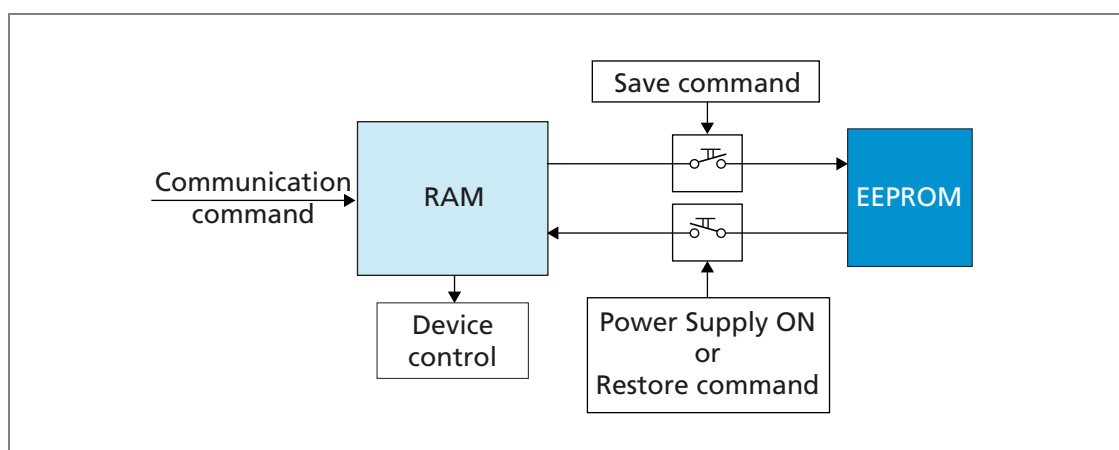


Fig. 3: Saving and restoring parameters

The following parameters can be loaded using the “Restore” command (see chap. 7.1, p. 41):

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


2.4.1 Saving parameters

The current parameter settings can be saved in the internal EEPROM (SAVE) (see Tab. 21, p. 43), either completely or for individual ranges.

- ▶ Write the “Save” signature to the sub-index 01 to 05 of the object 0x1010 (see Tab. 22, p. 43).


2 Overview

2.4.2 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. 23, p. 44).

1. Write the "Load" signature to the sub-index 01 to 06 of the object 0x1011 (see Tab. 24, p. 44).

 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).

3 CANopen protocol description

3 CANopen protocol description

3.1 Introduction

CANopen

CANopen is a standard software protocol. A CAN hardware environment is required for communication using CANopen. Up to 127 nodes can be addressed within a CANopen network. The maximum transmission speed is 1 MBit/s.

CAN standardisation

The CiA defines the following aspects in CiA 301:

- Communications structure
- Control and monitoring functions

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

Structure of a CANopen telegram

A CANopen telegram has an 11-bit identifier and can contain up to 8 bytes of user data.

Tab. 1: Schematic structure of a CANopen telegram

11-bit identifier	up to 8 bytes user data							
11-bit	8-bit	8-bit	8-bit	8-bit	8-bit	8-bit	8-bit	8-bit

3 CANopen protocol description

3.2 Communication services

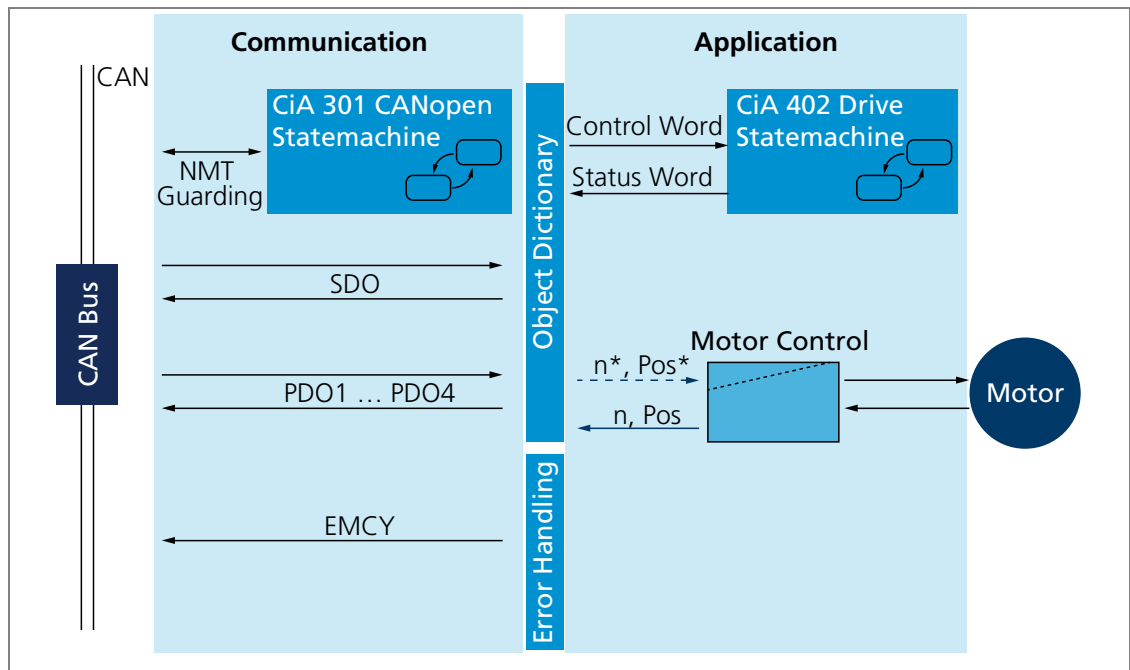


Fig. 4: Communication services of the Motion Controller

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

Tab. 2: Communication services to CiA 301

Communication services	Description
NMT (Network Management)	Activates nodes and monitors the current status of a node (see chap. 3.8, p. 25).
SDO (Service Data Object)	The CANopen master uses the SDO to access parameters within a node. Each SDO access reads or writes exactly one parameter. An SDO can only address one node in a network (see chap. 3.5, p. 19).
PDO (Process Data Object)	The PDO is used to access real-time data. A PDO can use a CAN message to access multiple drive parameters concurrently. The parameters sent or received in a PDO can be freely configured (see chap. 3.4, p. 16).
SYNC object	SYNC objects are used to synchronise different applications on the CAN-BUS (see chap. 3.7, p. 24).
EMCY (Emergency Object)	An emergency message is used to inform the CANopen master about errors. A CAN message conveys the error code asynchronously so that the status of the CANopen slave need not be interrogated after an error (see chap. 3.6, p. 22).

3 CANopen protocol description

Communications profile

FAULHABER Motion Controllers support the CANopen communications profile to CiA 301 V4:

- 4 transmission PDOs
- 4 receipt PDOs
- 1 server SDO
- Emergency object
- NMT with node guarding and heartbeat
- SYNC object

i The data assignment of the PDOs is pre-set to the "PDO set for servo drive" as specified in CiA 402 V3, but can be changed by the user (dynamic PDO mapping).

3.3 Identifier distribution

The Communication Object Identifier (COB-ID) consists of a 7-bit node address (node ID) and a 4-bit function code.

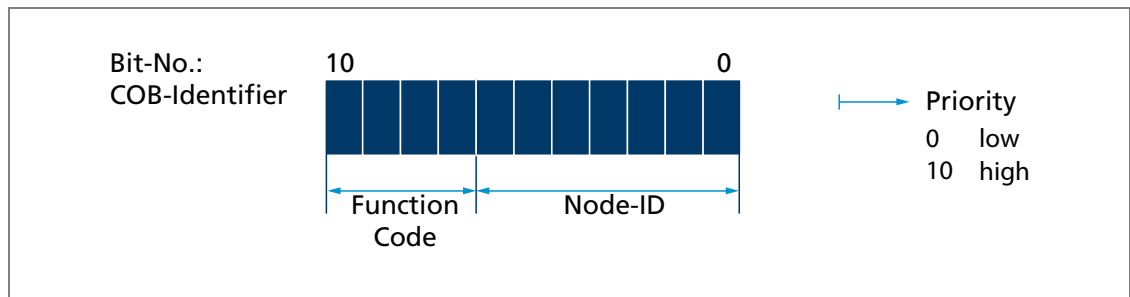


Fig. 5: Identifier distribution

The Predefined-Connection-Set defines the standard identifier for the most important objects.

i After the first assignment, the Motion Controller works with a valid node number with these identifiers.

Tab. 3: Standard identifier

Object	Function code (binary)	Resulting COB-ID	Object index for communication setting
NMT	0000	0	–
SYNC	0001	128 (80h)	1005h
EMERGENCY	0001	129 (81h) to 255 (FFh)	1014h
PDO1 (tx)	0011	385 (181h) to 511 (1FFh)	1800h
PDO1 (rx)	0100	513 (201h) to 639 (27Fh)	1400h
PDO2 (tx)	0101	641 (281h) to 767 (2FFh)	1801h

3 CANopen protocol description

Object	Function code (binary)	Resulting COB-ID	Object index for communication setting
PDO2 (rx)	0110	769 (301h) to 895 (37Fh)	1401h
PDO3 (tx)	0111	897 (381h) to 1023 (3FFh)	1802h
PDO3 (rx)	1000	1025 (401h) to 1151 (47Fh)	1402h
PDO4 (tx)	1001	1153 (481h) to 1279 (4FFh)	1803h
PDO4 (rx)	1010	1281 (501h) to 1407 (57Fh)	1403h
SDO (tx)	1011	1409 (581h) to 1535 (5FFh)	1200h
SDO (rx)	1100	1537 (601h) to 1663 (67Fh)	1200h
NMT error control	1110	1793 (701h) to 1919 (77Fh)	–

The COB-IDs of the PDOs, the SYNC objects and the emergency objects can be changed via the communication parameters in the object dictionary.



If the node number is changed using the LSS protocol, the COB-IDs of the PDOs and the Emergency Object remain unchanged.

If the node number is changed using the Motion Manager, the COB-IDs are automatically set to the predefined connection set.

3 CANopen protocol description

3.4 PDO (Process Data Object)

PDOs are CAN messages with up to 8 bytes user data. 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 NMT "Operational" state see chap. 3.8, p. 25).

The transmission of PDOs can be triggered in various different ways. The behaviour can be set for each PDO via the transmission type parameter of the communication parameters in the object dictionary:

Tab. 4: Types of PDO transmissions

Transmission type	Description
Event-driven	Event-driven RxPDOs are processed immediately on receipt. Event-driven TxPDOs are sent when the statusword of the device is contained and has been changed.
Remote request (RTR)	Data are sent in response to a request message.
Synchronised	Data are sent after receipt of a SYNC object (chap. 3.7, 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 than is used) is necessary for performance of the mapping procedure.
- The transmission types and COB-ID of the PDOs can be changed via the objects 0x1400 to 0x1403 and 0x1800 to 0x1803.
- The transmission type parameters can be used to change the behaviour of a PDO:

Tab. 5: Transmission type of a PDO

Transmission type	Meaning
255	asynchronous (event-driven)
253	asynchronous, on request (RTR)
1 to 240	synchronous, cyclical A PDO is sent after every SYNC object (see chap. 3.7, p. 24). The value is then equal to the number of SYNC objects that must be received before the PDO is sent again (1 = PDO is sent for every SYNC object)
0	synchronous, acyclical A PDO is sent or executed once after a SYNC object when the contents of the PDO have changed (see chap. 3.7, p. 24).

3 CANopen protocol description

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

RxPDO1: Controlword

11-bit identifier	2 bytes user data		
0x200 (512d) + node ID	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

11-bit identifier	2 bytes user data		
0x180 (384d) + node ID	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: Control word, Target Position (pp)

11-bit identifier	6 bytes user data					
0x300 (768d) + node ID	LB	HB	LLB	LHB	HLB	HHB

The RxPDO2 contains the 16-bit controlword and the 32-bit value of the destination position (object 0x607A) for the Profile Position mode (pp).

TxPDO2: Statusword, Position Actual Value

11-bit identifier	6 bytes user data					
0x280 (640d) + node ID	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)

11-bit identifier	6 bytes user data					
0x400 (1024d) + node ID	LB	HB	LLB	LHB	HLB	HHB

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

TxPDO3: Statusword, Velocity Actual Value

11-bit identifier	6 bytes user data					
0x380 (896d) + node ID	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

11-bit identifier	6 bytes user data					
0x400 (1024d) + node ID	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).

3 CANopen protocol description

TxPDO4: Statusword, Torque Actual Value

11-bit identifier	6 bytes user data					
0x380 (896d) + node ID	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. 6: 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.

If the number of mapped objects is 0, a PDO of length 0 will be sent.



Other mapping errors are described in the SDO error table (see chap. 3.5.2, p. 21).

3 CANopen protocol description

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 client (PC, PLC (programmable logic controller)) the Motion Controller makes data available (upload) or receives data from the client (download).

Tab. 7: 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. 8: Distribution of the SDO types of transmission

Type of transmission	Number of bytes	Purpose
Expedited transfer	maximum 4 bytes	Read and write individual numeric parameters
Segmented transfer	more than 4 bytes	Read text parameters (such as device name, firmware version) and transmit data blocks (such as the trace buffer)

Only the expedited transfer is described in this document. The segmented transfer is described in CiA 301.

3.5.1 Expedited transfer

SDO messages are always size 8 bytes.

Read OD entries (Client-to-Server, Upload-Request)

11-bit identifier	8 bytes user data							
0x600 (1536d) + node ID	0x40	Index LB	Index HB	Subindex	0	0	0	0

Server-to-Client, Upload-Response

11-bit identifier	8 bytes user data							
0x580 (1408d) + node ID	CS(0x4x)	Index LB	Index HB	Subindex	LLB (D0)	LHB (D1)	HLB (D2)	HHB (D3)

The command specifier CS(0x4x) specifies the number of valid data bytes in D0 to D3 and the transfer code. The command specifier is coded as follows:

- CS = 0x4F, 1 data byte in D0
- CS = 0x4B, 2 data bytes in D0 to D1
- CS = 0x47, 3 data bytes in D0 to D2
- CS = 0x43, 4 data bytes in D0 to D3

3 CANopen protocol description

Write OD entries (Client-to-Server, Download-Request)

11-bit identifier	8 bytes user data							
0x600 (1536d) + node ID	CS(0x2x)	Index LB	Index HB	Subindex	LLB (D0)	LHB (D1)	HLB (D2)	HHB (D3)

The command specifier CS(0x2x) specifies the number of valid data bytes in D0 to D3 and the transfer code. The command specifier is coded as follows:

- CS = 0x2F, 1 data byte in D0
- CS = 0x2B, 2 data bytes in D0 to D1
- CS = 0x27, 3 data bytes in D0 to D2
- CS = 0x23, 4 data bytes in D0 to D3
- CS = 0x22, no specification of the number of data bytes

Server-to-Client, Download-Response

11-bit identifier	8 bytes user data							
0x580 (1407d) + node ID	0x60	Index LB	Index HB	Subindex	0	0	0	0

Abort in the event of SDO errors

SDO-abort Client-to-Server

11-bit identifier	8 bytes user data							
0x600 (1536d) + node ID	0x80	Index LB	Index HB	Subindex	ERROR 0	ERROR 1	ERROR 2	ERROR 3

SDO-abort Server-to-Client

11-bit identifier	8 bytes user data							
0x580 (1536d) + node ID	0x80	Index LB	Index HB	Subindex	ERROR 0	ERROR 1	ERROR 2	ERROR 3

3 CANopen protocol description

3.5.2 SDO error description

If the SDO protocol on a page cannot be processed further, an SDO-Abort telegram is sent (see chap. 3.5.1, p. 19). 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	Subindex not present
0x06	0x09	0x0030	General value range errors
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 CANopen protocol description

3.6 Emergency object (error message)

The emergency object informs other bus participants of errors asynchronously without requiring interrogation. The emergency object is always size 8 bytes:

11-bit identifier	8 bytes user data							
0x80 (128d) + node ID	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)
- FE0(LB)/FE1(HB): 16-bit FAULHABER error register (contents of object 0x2320, see Tab. 12, p. 31)
- 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.

Tab. 9, 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. 13, p. 32).

Tab. 9: 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

3 CANopen protocol description

Emergency message		FAULHABER error register 0x2320			Error register 0x1001	
Error code	Designation	Error mask 0x2321	Bit	Designation	Bit	Designation
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
0x8110	CAN overrun	0x0800	11	Communications error	4	Communications error
0x8130	CAN guarding failed					
0x8140	CAN recovered from bus stop					
0x8310	RS232 overrun					
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 in Tab. 10, 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. 14, p. 32).
- 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. 10: Example of user data assignment to an emergency message

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

3 CANopen protocol description

3.7 SYNC object

The SYNC object is a message without any user data. The SYNC object is used to trigger synchronous PDOs and at the same time to start processes on various items of equipment.

The identifier of the SYNC objects is set in the object dictionary under the index 0x1005 (by default 0x80).

11-bit identifier	0 bytes of user data
0x80	no user data

i In order that an SYNC object triggers a PDO, the transmission type of the PDO to be triggered must be set accordingly (see Tab. 5, p. 16).

3.7.1 Triggering synchronous PDOs

Synchronous RxPDO: The command transmitted with the PDO is not executed until a SYNC object is received. The transmission types 1 to 240 of an RxPDO are identical to transmission type 0.

Synchronous TxPDO: The PDO with the current data is not sent until a SYNC object is received.

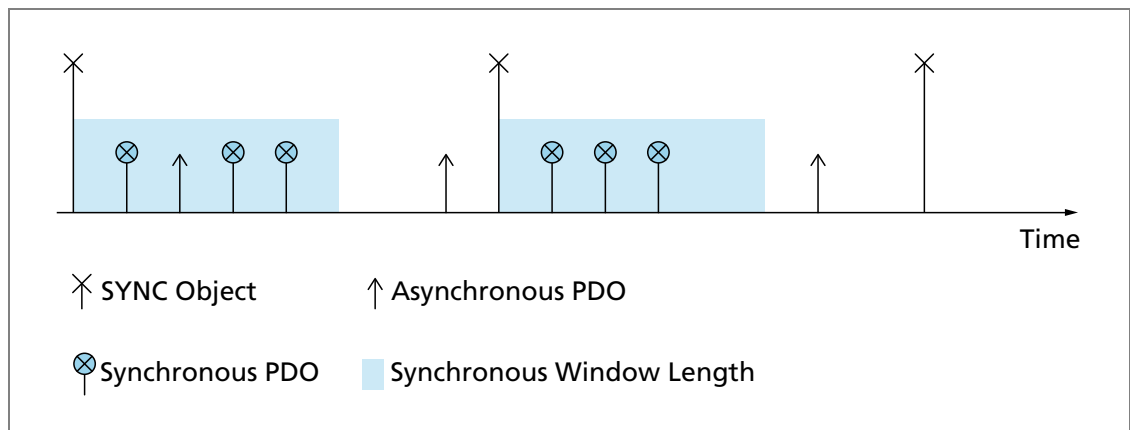


Fig. 6: Window Length chart

i The nodes can also be grouped by transmission types 1-240.

Motion Controllers do not check the synchronous window length of incoming PDOs. They send the available TxPDOs immediately the SYNC object is received.

3 CANopen protocol description

3.8 NMT (Network Management)

The network management object governs the CiA 301 state machine of the CANopen device and monitors the network nodes.

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 only with NMT messages and via SDOs.

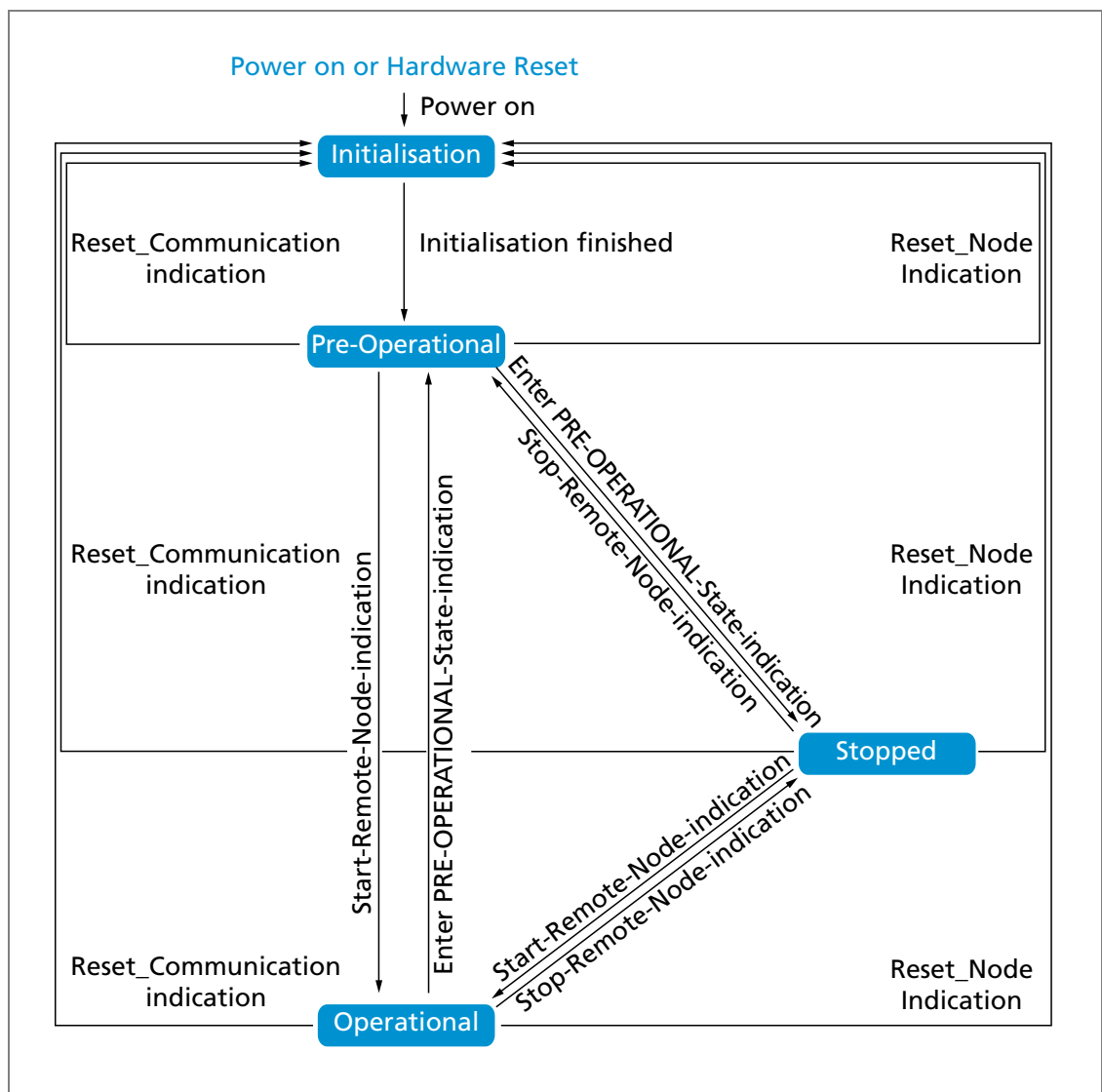


Fig. 7: CiA 301 state machine

Tab. 11: NMT changes of status

Status transition	CS	Meaning
Power on	–	The initialisation state is achieved automatically on switching on.
Initialisation finished	–	After initialisation the device is automatically in the Pre-Operational state, and it sends a boot-up message.
Start remote node indication	0x01 (1d)	This starts the device and enables transmission of PDOs.

3 CANopen protocol description

Status transition	CS	Meaning
Enter Pre-operational state indication	0x80 (128d)	Stops the transmission of PDOs, SDOs are still active.
Stop remote node indication	0x02 (2d)	The drive is set to the stopped status, SDO and PDO are switched off.
Reset node indication	0x81 (129d)	Performs a reset. All objects are reset to Power-On standards.
Reset communication indication	0x82 (130d)	Performs a reset of the communications functions.

i FAULHABER Motion Controllers are equipped with a standard configuration for all objects. In most cases no further parametrisation is necessary at system start.

Any necessary parameter settings can be performed using the FAULHABER Motion Manager and saved permanently in the EEPROM. Settings in the EEPROM are immediately available at system start.

Starting a CANopen node

Start remote node:

11-bit identifier	2 bytes user data	
0x000	0x01	Node ID

An entire network can also be started with a CAN message:

Start all remote nodes:

11-bit identifier	2 bytes user data	
0x000	0x01	0x00

After the node or the entire network is started the device is in the *Operational* state. The device can now be operated using PDOs.

In the *Stopped* state the device is in an error state and can no longer be operated using PDOs. Under these circumstances, communication with the device is available only by NMT messages.

An NMT message always consists of 2 bytes on the identifier 0x000.

NMT message

11-bit identifier	2 bytes user data	
0x000	CS	Node ID

Assignment of user data:

- CS: Command specifier (see Tab. 11, p. 25)
- Node ID: Node address (0 = all nodes)

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


3 CANopen protocol description

3.8.1 Boot up

Immediately after the initialisation phase the Motion Controller sends a boot-up message. A boot-up message signals the end of the initialisation phase of a module after it has been switched on. A boot-up message is a CAN message with one data byte (byte 0 = 0x00) on the identifier of the node guarding message (0x700 + node ID).

11-bit identifier	1 byte of user data
0x700 (1792d) + node ID	0x00

3.8.2 Monitoring functions

 Only one monitoring function; node guarding or heartbeat, can be used at one time.

3.8.2.1 Node guarding

The node guarding object interrogates the instantaneous state of the device. To do this, the master sets a remote frame with a request for the guarding identifier of the node to be monitored. The node to be monitored responds with the guarding message which contains the current status of the node and a toggle bit.

3 CANopen protocol description

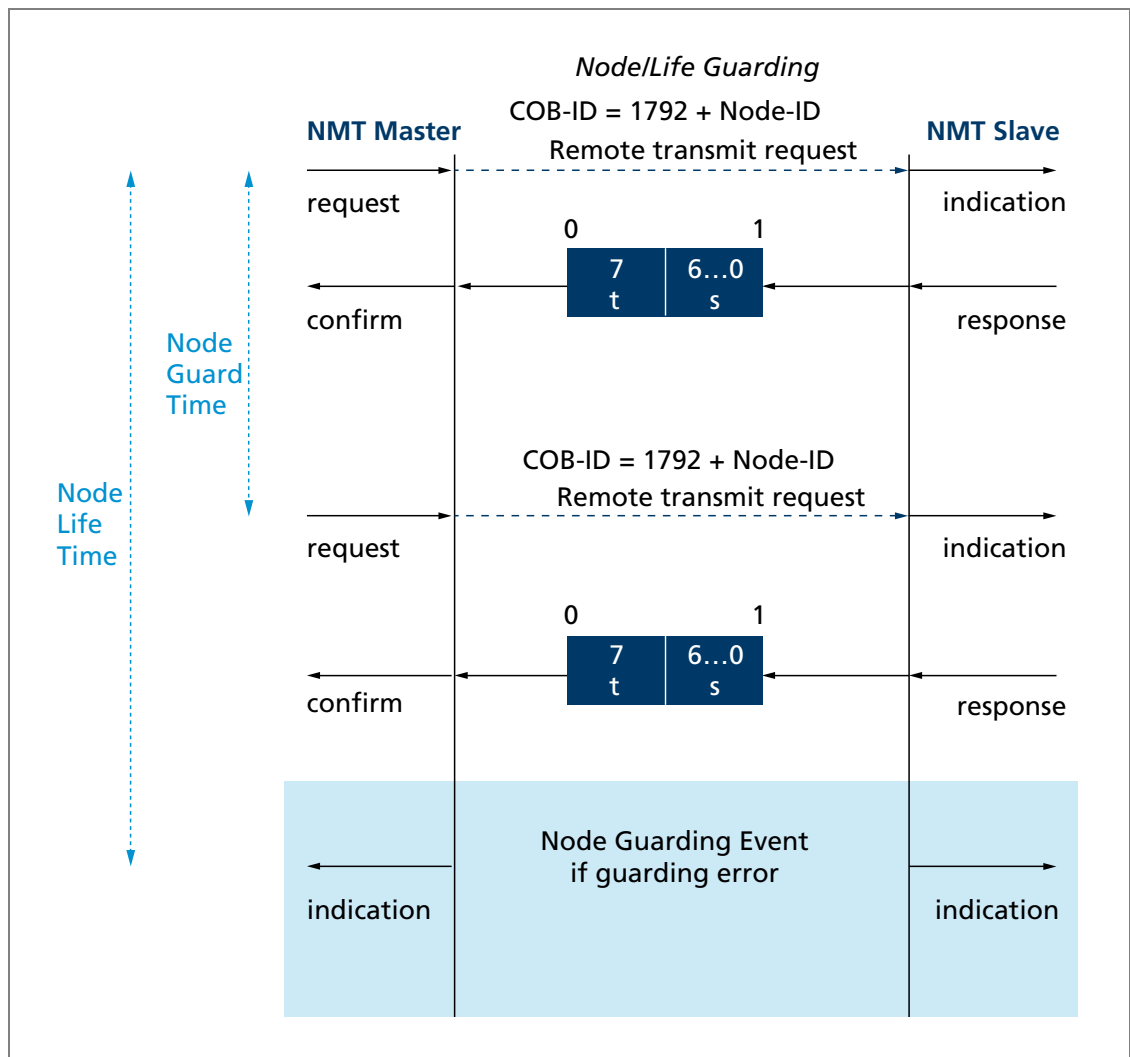


Fig. 8: Diagram of the node guarding protocol

t: Toggle bit

Initially 0, changes its value at each guarding telegram

s: Status

$s = 0x04$ (4d): Stopped

$s = 0x05$ (5d): Operational

$s = 0x7F$ (127d): Pre-operational

If a node life time > 0 is set (objects 0x100C and 0x100D) and no node guarding request is made by the master within the set node life time, a node guarding error is set. The response to a node guarding error is set by the FAULHABER error register (object 0x2321) (see Tab. 14, p. 32). The default is to send the emergency message 0x8130.

3 CANopen protocol description

3.8.2.2 Heartbeat

The Motion Controller can be set to act both as the heartbeat producer and also as the heartbeat consumer.

- **Heartbeat producer:** On a cyclical basis the Motion Controller sends out a message which is received by one or more heartbeat consumers in the network.
- **Heartbeat consumer:** If within the heartbeat consumer time no heartbeat message is received from the heartbeat producers that are being monitored, the Motion Controller responds with the behaviour specified in the FAULHABER error register (object 0x2320) (see Tab. 12, p. 31).

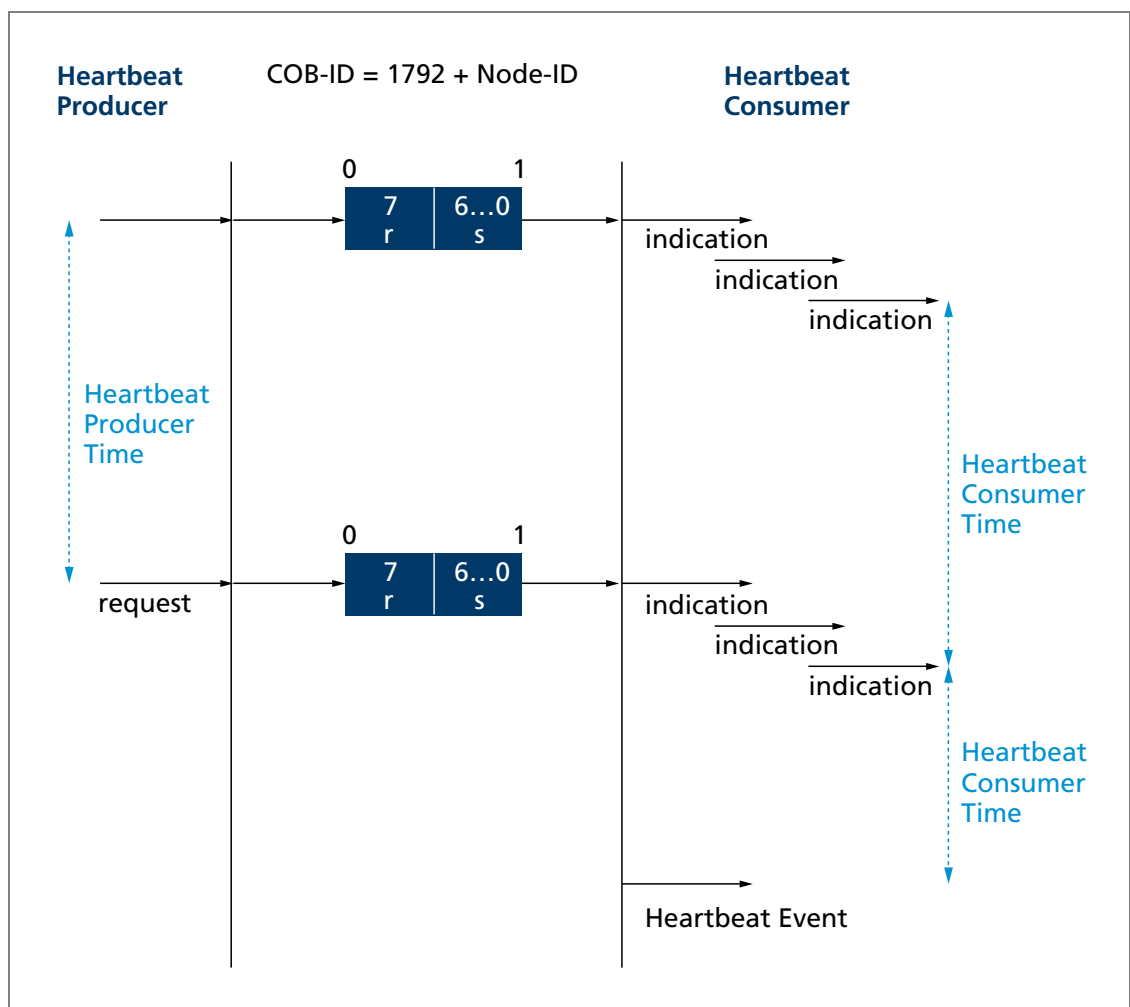


Fig. 9: Chart of the heartbeat protocol

r: Reserved
Always 0

s: Status
s = 0x00 (0d): Boot up
s = 0x04 (4d): Stopped
s = 0x05 (5d): Operational
s = 0x7F (127d): Pre-operational

3 CANopen protocol description

3.8.3 Setting the monitoring functions

- Only one of the two monitoring functions (node guarding, heartbeat) can be activated at one time.
- If the producer heartbeat time is > 0 (object 0x1017) the Motion Controller operates as a heartbeat producer. At intervals of the producer heartbeat time the Motion Controller sends a heartbeat message. The node guarding time is set to 0 (see chap. 3.8.2.1, p. 27).
- If the heartbeat is activated, the boot-up message after the switch-on ranks as the first heartbeat message. Further heartbeats follow at the interval for the producer heartbeat time.
- If in addition to the producer heartbeat time a heartbeat consumer time > 0 is set (object 0x1016.01), the Motion Controller operates as a heartbeat consumer. The settings for the heartbeat producer are inoperative. The node ID of the master to be monitored and the heartbeat consumer time are entered in the object 0x1016.
- The heartbeat consumer time must always be longer than the producer heartbeat time of the master.
- If within the set heartbeat consumer time the Motion Controller receives no heartbeat message from the master, a heartbeat event is triggered. The response to a heartbeat event is determined by the FAULHABER error register error mask (object 0x2321) (see Tab. 14, p. 32). The default is to send the emergency message 0x8130.
- If whilst the heartbeat producer is activated an attempt is made to set a node guarding time, the SDO error 0x08000020 (no access available) is sent.

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. 7.1, p. 41)
- The manufacturer-specific area (index 0x2000 to 0x5FFF) contains manufacturer-specific objects (see chap. 7.2, p. 50)
- The standardised device profiles area (0x6000 to 0x9FFF) contains objects supported by the Motion Controller (see the documentation of the drive functions)

3 CANopen protocol description

3.10 Error handling

3.10.1 CAN error

CAN overrun (object lost)

If messages are lost, the controller sends the emergency message 0x8110. Bit 4 (communication error) is set in the error register and Bit 7 (CAN overrun) is set in the FAULHABER error register. The emergency message is sent out after a delay. Issuing of the emergency message (0x000) does not retract the error. The respective bits in the error register and in the FAULHABER error register are not cleared down.

CAN in error passive mode

If the CAN module of the drive is set to the *Error-Passive* state, the emergency message 0x8120 is sent. Bit 4 (communication error) is set in the error register and Bit 6 (CAN in error passive mode) is set in the FAULHABER error register. The emergency message (0x000) is sent and the error retracted once the drive is restored to the *Error-Active* state.

Recovered from bus stop

If the CAN module of the drive receives a valid message whilst set to the *Bus-Off* state, the emergency message 0x8140 is sent. The emergency message reports that the *Bus-Off* state has been exited. Bit 4 (communication error) is set in the error register and Bit 9 (Recovered from Bus-Off) is set in the FAULHABER error register. This does not retract the error. The respective bits in the error register and in the FAULHABER error register are not cleared down.



"CAN-Overrun" and "Recovered from bus off" are serious communications errors. The respective bits in the error register and in the FAULHABER error register can be cleared down only by restarting the Motion Controller. Other serious communications errors are:

- Node guarding timeouts
- Heartbeat timeouts

3.10.2 Equipment faults

Tab. 12: 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).

3 CANopen protocol description

Tab. 13: 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. 22).

The error mask describes the handling of internal errors depending on the error coding (see Tab. 13, p. 32).

Tab. 14: 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.

4 Trace

4 Trace

The trace function 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. Two different types of recording are available:

- Trace recorder: The parameter values are written to an internal buffer and can then be read (see chap. 4.1, p. 33).
- Trace logger: On request the parameter values are requested and read continuously (see chap. 4.2, p. 36).

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

4.1 Trace recorder

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. 4.1.2, p. 35).

4.1.1 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. 15: 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)

4 Trace

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. 16: Trigger mode (0x2370.04)

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

4 Trace

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.

4.1.2 Reading the trace buffer

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

Tab. 17: 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 be recorded to the highest resolution of the trace recorder.

Trigger status (0x2371.01)


Tab. 18: Trigger status (0x2371.01)

Bit	Entry	Description
0 (LSB)	Status 0	0: No trigger active
1	Status 1	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 Trace

4.1.3 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).

4.2 Trace logger

The trace logger uses the PDO communication service transmit data from the controller. Any TxPDO can be used as the trace PDO. Prior to use, the PDO must be loaded with the desired parameters, using the PDO mapping method (chap. 3.4.2, p. 17). The data can then be requested cyclically using the remote request or SYNC.

 The resolution of the individual data points is dependent on the speed of transmission and processing. The resolution of the data points can be down to 1 ms.

5 Communications settings

5 Communications settings

FAULHABER drives are delivered as standard without any valid node number (node ID = 0xFF) and set to automatic Baud rate detection (AutoBaud). For operation in a CAN network, a unique node number must first be assigned. If necessary the network transmission rate can be set as an alternative to the AutoBaud setting.

5.1 Setting via the CAN network

For setting via the CAN network the FAULHABER Motion Manager or other configuration tool which supports the LSS protocol (Layer Setting Service and Protocol) to CiA 305 is required.



The FAULHABER Motion Manager must be installed on a PC with a supported CAN interface.

There are two ways of setting the communication parameters:

- An individual drive is connected at the CAN interface of the configuration tool:
The "LSS Switch Mode Global" without further data allows the drive to be switched to configuration mode, in order to set the node number and Baud rate.
- The drive to be configured is connected via the CAN interface within a network to the configuration tool:
The "LSS Switch Mode Selective" allows the desired drive to be addressed by inputting the LSS address (Vendor ID, Product code, Revision number, Serial number) and switched to configuration mode, in order to set the node number and Baud rate.
FAULHABER drives of the range MC V3.0 require the following input:

- Vendor ID: 327
- Product code: 48
- Revision number: 1.0
- Serial number: See the product sticker

As well as the setting of the node number and Baud rate, the LSS protocol also supports the reading of the LSS addresses of units that are connected and the reading of the node ID setting.

The identifier 0x7E5 is used (by the master) and 0x7E4 (by the slave) for LSS communication.

After configuration, the Motion Controller saves the set parameters in the EEPROM. They remain available after switching off and on again.

For a detailed description of the LSS protocol please refer to the document CiA 305.

5 Communications settings

5.1.1 Setting the node number

- Node numbers 1 to 127 can be set.
- The node ID 255 (0xFF) marks the node as unconfigured. After it is switched on, the node is in the LSS-Init status, until a valid node number is assigned. After a valid node number has been assigned to the node, the NMT initialisation continues.
- i** When the node ID is changed from the unconfigured state (255) to a valid node number, all the communications settings are set to their default values. The COB-IDs of the PDOs and of the emergency object are then set again in accordance with the predefined connection set (see chap. 3.8, p. 25).

5.1.2 Setting the Baud rate

- If the automatic Baud rate detection (AutoBaud) is active, the drive can be used in a network with any transmission rate in accordance with Tab. 19, p. 38. The Baud rate of the network is detected after no more than 24 telegrams (3 per Baud rate) on the bus cable. The drive then sets itself to match the network Baud rate.
- If the automatic Baud rate detection is active, telegrams cannot be processed until the Baud rate has been detected. If the automatic Baud rate detection is active, it takes correspondingly longer to boot up the system.
- A fixed Baud rate in accordance with Tab. 19, p. 38 can be set by inputting the index 0 to 8.

Tab. 19: Bit timing parameters

Baud rate	Index
1000 kBit/s	00
800 kBit/s	01
500 kBit/s	02
250 kBit/s	03
125 kBit/s	04
50 kBit/s	06
20 kBit/s	07
10 kBit/s	08
AutoBaud	09

5 Communications settings

5.2 Setting the node number via the object dictionary

As an alternative to the LSS method via the CAN network, the node number can also be set via any interface (CAN, USB, RS232) available on the drive.

The setting is performed by writing the object 0x2400.03 in the object dictionary:

Tab. 20: CAN 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
	01	CAN Baud rate index	U8	ro	9	Index of the CAN Baud rate to Tab. 19, p. 38
	03	Node ID	U8	rw	255	Node number

The object 0x2400.01 can be used to read the current setting of the Baud rate (AutoBaud or fixed Baud rate).

A change of the node number via the object 0x2400.03 is acknowledged with the last node number. The changed node number is not loaded until a "Save " command has been executed for the application parameters followed by a "Reset" command.

6 Troubleshooting

6 Troubleshooting

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

7 Parameter description

7 Parameter description

7.1 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. 22.

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

COB-ID SYNC

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1005	00	COB ID SYNC	U32	rw	0x80	CAN-object identifier of the SYNC object

7 Parameter description

Manufacturer's device name

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

The segmented SDO record must be read to determine the manufacturer's 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

The segmented SDO record must be read to determine the manufacturer's hardware version.

Manufacturer's software version

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

The segmented SDO record must be read to determine the manufacturer's software version.

Guard time

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x100C	00	Guard time	U16	rw	0	Monitoring time for node guarding

The guard time is stated in milliseconds. The value 0 switches node guarding off (chap. 3.8.2.1, p. 27).

Life-time factor

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x100D	00	Life time factor	U8	rw	0	Time factor for node guarding

The life-time factor multiplied by the guard time gives the life-time for the node guarding (chap. 3.8, p. 25). The value 0 switches node guarding off.

7 Parameter description

Store parameters

Tab. 21: 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. 22: "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.

7 Parameter description

Restore default parameters

Tab. 23: 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. 24: "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**, a **Reset Node** or **Reset Communication** command must be executed.

COB-ID emergency message

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1014	00	COB-ID EMCY	U32	rw	0x80 + node ID	CAN object identifier of the emergency object

7 Parameter description

Consumer heartbeat time

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1016	00	Number of entries	U8	ro	1	Number of object entries
	01	Consumer heartbeat time	U32	rw	0	Heartbeat monitoring time

- Bits 0 to 15 contain the consumer heartbeat time in milliseconds. If the value is set to 0, the consumer heartbeat function is deactivated (chap. 3.8.2.2, p. 29)
- Bits 16 to 23 contain the node number to which the heartbeat message is sent (master node ID).
- Bits 24 to 31 are not used (spare).

Producer heartbeat time

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1017	00	Producer heartbeat time	U16	rw	0	Heartbeat send time interval

The object producer heartbeat time contains the producer heartbeat time interval in milliseconds. If the value is set to 0, the producer heartbeat function is deactivated (see chap. 3.8.3, p. 30).

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	48	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*/NMT status. Subindex 1 allows the behaviour in the event of a serious communications error to be changed.

7 Parameter description

Server SDO parameters

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1200	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID client to server (rx)	U32	ro	0x600 + node ID	CAN object identifier for the server RxSDO
	02	COB ID server to client (tx)	U32	ro	0x580 + node ID	CAN object identifier for the server TxSDO

Receive PDO1 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1400	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x200 + node ID	CAN object identifier for the server RxPDO1
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Receive PDO2 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1401	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x300 + node ID	CAN object identifier for the server RxPDO2
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Receive PDO3 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1402	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x400 + node ID	CAN object identifier for the server RxPDO3
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Receive PDO4 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1403	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x500 + node ID	CAN object identifier for the server RxPDO1
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

7 Parameter description

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 (0x6071)
	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
0x1604	00	Number of mapped objects	U8	ro	0	Number of mapped objects
	01	PDO mapping entry 1	U32	rw	0x0	Pointer to the 16-bit control word (0x6040)
	02	PDO mapping entry 2	U32	rw	0x0	Pointer to the 32-bit target velocity (0x60FF)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

7 Parameter description

Transmit PDO1 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1800	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x180 + node ID	CAN object identifier of the TxPDO1
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Transmit PDO2 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1801	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x280 + node ID	CAN object identifier of the TxPDO2
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Transmit PDO3 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1802	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x380 + node ID	CAN object identifier of the TxPDO3
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

Transmit PDO4 communication parameter

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x1803	00	Number of entries	U8	ro	2	Number of object entries
	01	COB ID	U32	rw	0x480 + node ID	CAN object identifier of the TxPDO4
	02	Transmission type	U8	rw	255 (asynchr.)	Type of PDO transmission

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	

7 Parameter description

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	0	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 32-bit velocity actual value (0x606C)
	03	PDO mapping entry 3	U32	rw	0	
	04	PDO mapping entry 4	U32	rw	0	

7 Parameter description

7.2 Manufacturer-specific objects

Tab. 25: 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).

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

7 Parameter description

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

CAN Baud rate index and node number

Tab. 26: CAN 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
	01	CAN Baud rate index	U8	ro	9	Index of the CAN Baud rate to Tab. 19, p. 38
	03	Node ID	U8	rw	255	Node number

