FAULHABER

Communications Manual

MC 5010

MC 5005

MC 5004

MCS

RS232 / USB

WE CREATE MOTION

EN

Imprint

Version:
15-04-2016

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

1.1 Validity of this document

This document describes:

- Communication with the drive via RS232
- 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 and for CAN-BUS project engineers with experience of interfaces.

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

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
Attr.	Attribute
CAN	Controller Area Network
CiA	CAN in Automation e.V.
COB ID	Communication Object Identifier
CRC	Cyclic Redundancy Check
CS	Command Specifier
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMCY	Emergency
FIFO	First In – First Out
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
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MSB	Most Significant Byte
OD	Object Dictionary
pp	Profile Position
pv	Profile Velocity
ro	read only
RTR	Remote Request
rw	read-write
SDO	Service Data Object
PLC	Programmable Logic Controller - PLC
Sxx	Data type signed (negative and positive numbers) with bit size xx
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

2.1 Basic structure of the Motion Controller

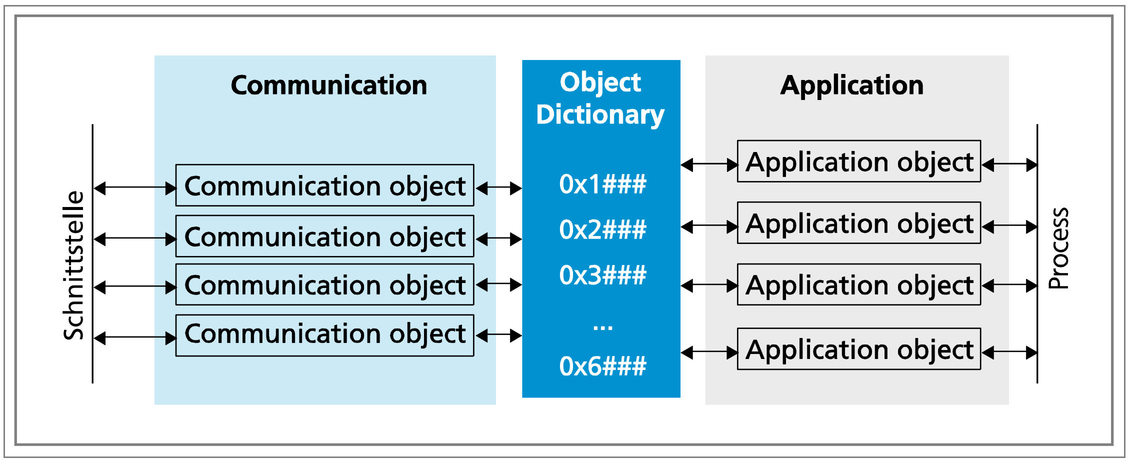

Fig. 1: Basic structure of the Motion Controller

Communication services
The master communicates with the object dictionary via the interface and use of the communication services (see chap. 3.2, p. 15). The communication services are based on CANopen device systems.

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.

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

2.2 Pre-requirement for communication

FAULHABER drives are supplied with the node number 0xFF (unconfigured) and a RS232 transmission rate of 9600 bits/s. For operation over an RS232 or USB interface, a unique node number must be assigned and in addition for RS232 a suitable Baud rate set at commissioning (see).

The Motion Controller uses the same communications protocol for USB and RS232.

If a change is made to the node number or Baud rate, the response must be made from the old node number or at the old Baud rate.

2.2.1 Operation over the RS232 interface

2.2.1.1 Operation of an individual Motion Controller

1. Establish a connection with a host interface (typically a PC or PLC).
   - Connect the Tx data cable on the host side with the Rx pin of the drive
   - Connect the Rx data cable on the host side with the Tx pin of the drive (null modem cable)

Alternatively a USB/RS232 converter can be used at the PC side.

2. Configure the host interface to match the drive settings (see chap. 5, p. 36):
   - The same Baud rate
   - 8 data bits, no parity, 1 stop bit, no flow control
3. Switch on the Motion Controller.

↳ Communication will be established. The drive will report a boot-up message at the last Baud rate setting.

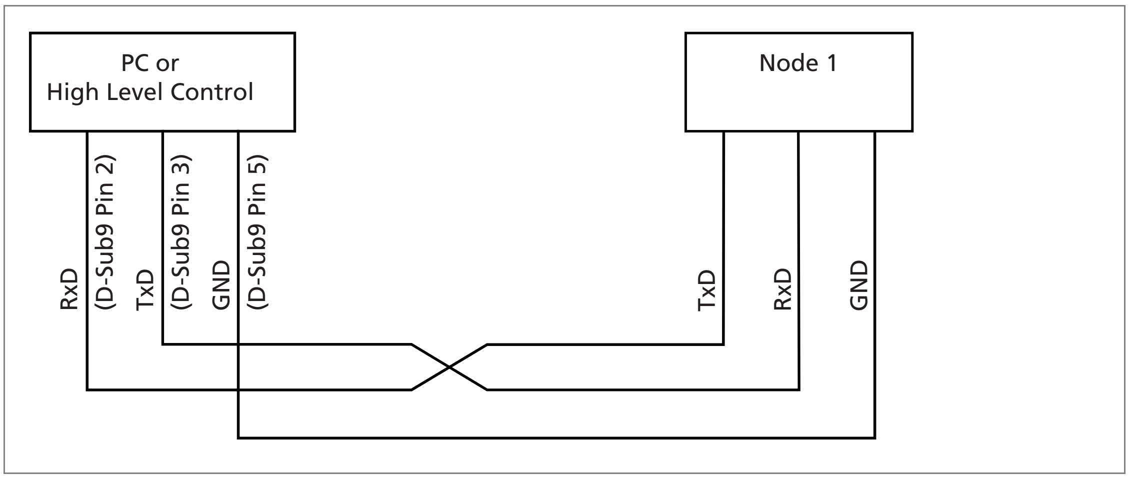

Fig. 2: Wiring between PC/controller and a drive

2.2.1.2 RS232 network operation

Multiple Motion Controllers can be operated on a single RS232 host interface.

- Connect the Tx cables and Rx cables to the controller in parallel.

When they are in network operation the drives may not send any asynchronous messages, because these can interfere with communications with another drive. Asynchronous responses can be deactivated in the object 0x2400.04.

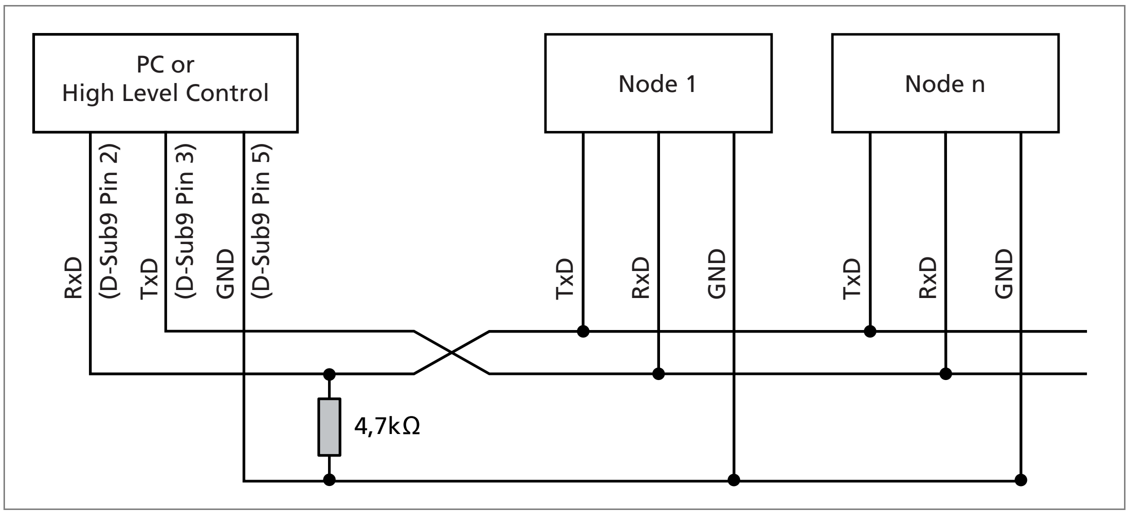

Fig. 3: Wiring with several Motion Control systems in RS232 network operation

2.2.2 Operation via the USB interface

The USB interface can be used for the connection to with the Motion Manager. The appropriate driver is installed automatically with the Motion Manager.

✓ A USB cable with mini-USB plugs must be available

1. Establish a connection with a host interface (typically a PC).
   - Use a USB cable with mini-USB plugs at the device side.
2. Switch on the Motion Controller.
   - Communication will be established.
   - The drive will report a boot-up message.

A suitable driver connection will be required for the use of the USB interface in a specific application. Information about this is available on request.

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.

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

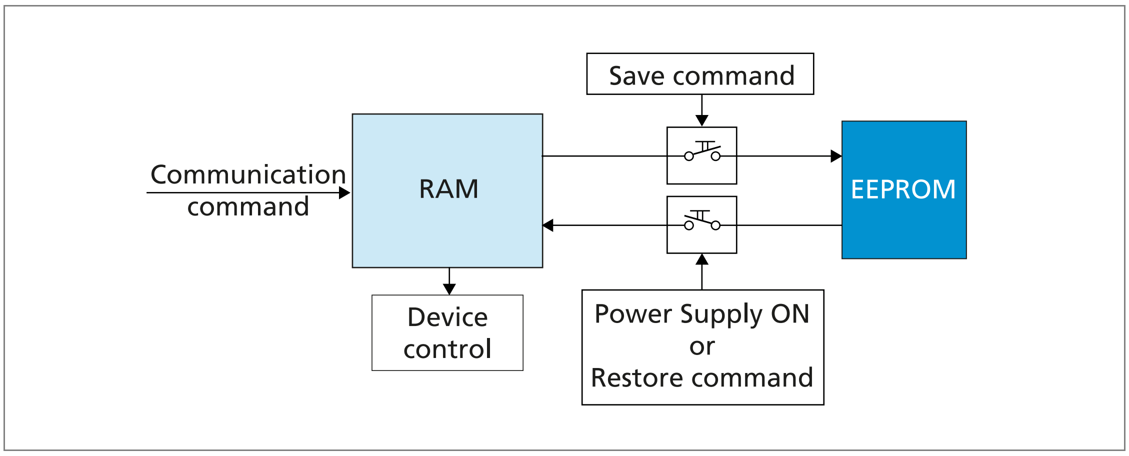

Fig. 4: Saving and restoring parameters

The following parameters can be loaded using the “Restore” command (see chap. 7.1, p. 38):

- 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. 43, 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. 44, p. 39).

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. 45, p. 40).

1. Write the “Load” signature to the sub-index 01 to 06 of the object 0x1011 (see Tab. 46, 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).

3 Protocol description

3.1 Introduction

Entries in the object dictionary can be written or read using the protocol services.

The services defined for the RS232 and USB interfaces are based on the CANopen services, but tailored to the characteristics of the RS232 interface.

In CiA 301, the CiA (CAN in Automation) defines the following aspects:

- Communications structure
- Control and monitoring functions

The CiA 402 CANopen defines the drive device profiles for a range of device classes.

Simultaneous access to the drive both from the RS232 side and also from the USB interface is supported. Messages received via either interface are stored in a common wait queue and are processed sequentially (FIFO). The drive issues the acknowledgement to the same interface.

Each request from the host is concluded with an acknowledgement by the drive. The maximum number of requests buffered in the drive is limited. If no further requests can be added to the buffered queue, an appropriate message is sent and the request is discarded.

Telegram structure
A binary protocol with messages of variable length is used for communication via the USB and RS232 interfaces.

Tab. 1: Schematic structure of a USB/RS232 telegram

Byte	Name	Meaning
1st byte	SOF	Character (S) as Start of Frame
2nd byte	User data length	Telegram length without SOF/EOF (packet length)
3rd byte	Node number	Node number of the slave (0 = Broadcast)
4th: byte	Command code	See
5th – Nth byte	Data	Data area (length = packet length – 4)
(N+1). byte	CRC	CRC8 with polynomial 0xD5 over byte 2–N
(N+2). byte	EOF	Character (E) as End of Frame

Telegram errors (e. g. CRC error, wrong length, invalid command codes) are not reported back. The node number of the message that is received, especially in the event of CRC errors, cannot be determined unambiguously. The telegram remains unanswered and after the time-out the master must send the request again.

The characters SOF and EOF are no longer shown in the following description. Only the byte in between are shown, so byte 1 is actually the 2nd byte. Bytes in the overall telegram frame.

Frame length
The length of the overall frame incl. the SOF and EOF is the user data length + 2 bytes. The user data length is limited to 62 bytes.

3 Protocol description

Command codes

Command code	Name	Function
0x00	`Boot up`	Boot-up message / Reset Node (Receive / Request)
0x01	`SDO Read`	Read the object dictionary entry (Request / Response)
0x02	`SDO Write`	Write an object dictionary entry (Request / Response)
0x03	`SDO Error`	SDO error (abort request / error response)
0x04	`Controlword`	Writing the controlword (request / response)
0x05	`Statusword`	Reception of the statusword (receive)
0x06	`Trace Log`	Trace Request for Trace Logger (Request / Response)
0x07	`EMCY`	Reception of an emergency message (receive)
0x08	`SDO Block Read Init`	Initialisation of the segmented SDO block upload (request / response)
0x09	`SDO Block Read Upload`	Execution of the segmented SDO block upload (request / response / acknowledge)
0x0A	`SDO Block Read End`	End of the segmented SDO block upload (request / acknowledge)
0x0B	`SDO Block Write Init`	Initialisation of the segmented SDO block download (request / response)
0x0C	`SDO Block Write Download`	Execution of the segmented SDO block download (request / response)
0x0D	`SDO Block Write End`	End of the segmented SDO block download (request / response)

Data
The data format is based on the CANopen data format. The data transmission always starts with the lowest value byte.

CRC (Cyclic Redundancy Check)
For the check sum calculation, the following algorithm is applied to all the bytes (except for SOF/EOF) of the telegram to be processed:

```
#define polynomial 0xD5

uint8_t CalcCRCByte(uint8_t u8Byte, uint8_t u8CRC)
{
uint8_t i;
     u8CRC = u8CRC ^ u8Byte;
     for (i = 0; i < 8; i++) {
         if (u8CRC & 0x01) {
             u8CRC = (u8CRC >> 1) ^ polynomial;
         }
         else {
             u8CRC >>= 1;
         }
      }
      return u8CRC;
}
```

0xFF is used as the start value for the CRC8.

3.2 Communication services

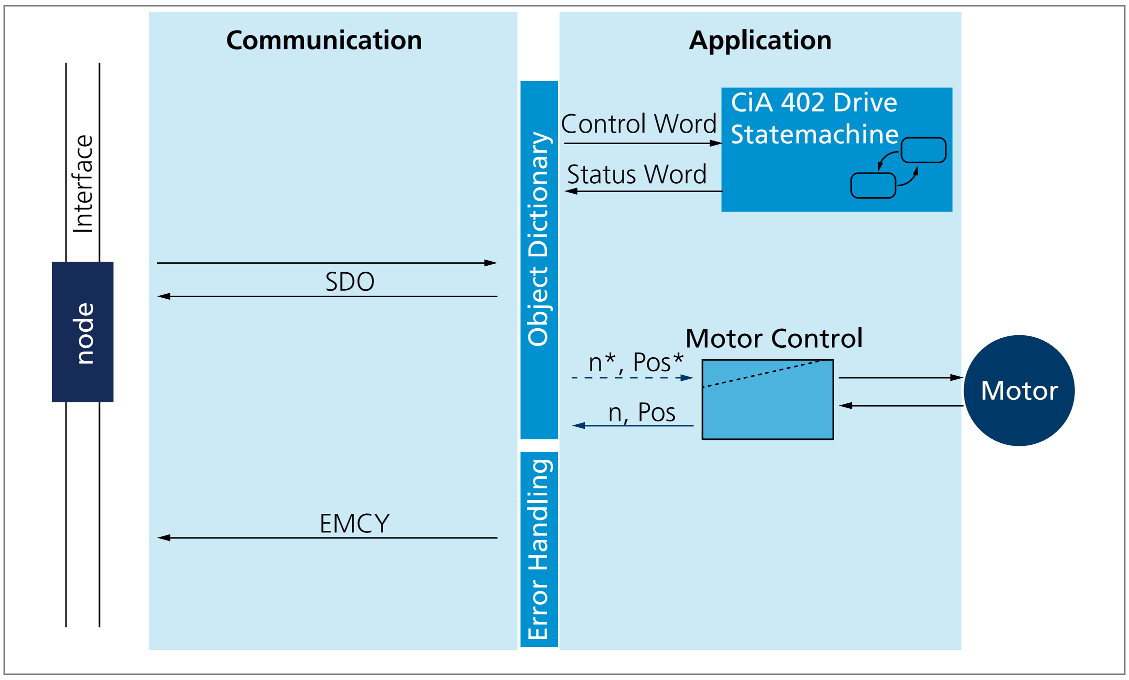

Fig. 5: Communication services of the Motion Controller

The following communication services are available:

- Boot-up message
- Write or read service for each individual parameter (SDO message)
- Direct write access to the controlword of the drive
- Direct read access to the statusword of the drive
- Communication service for signalling error states by a message (EMCY) transmitted by the drive in the event of an error
- Communication service for accessing the values on the data logger (trace)

3.3 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) the Motion Controller makes data available (upload) or receives data from the client (download).

Tab. 2: Distribution of the SDO types of transmission

Type of transmission	Data length	Purpose
Expedited transfer	maximum 58 bytes	Read and write individual numeric parameters
Segmented transfer	any	Transmission of data blocks (such as the trace buffer)

3.3.1 Expedited transfer

3.3.1.1 Read the object dictionary

Entries in the object dictionary can be read using the SDO read. Telegrams are always acknowledged.

Tab. 3: Request

Byte	Contents	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x01	Command SDO Read
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	CRC	Check sum

Tab. 4: Response

Byte	Contents	Description
1	Length	User data length >7 bytes
2	Node number	Node number
3	0x01	Command SDO Read
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7–N	Value	Current value of the specified object entry
(N+1)	CRC	Check sum

If the specified object cannot be read, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 23).

3.3.1.2 Writing to the object dictionary

Entries in the object dictionary can be written using the SDO Write. Telegrams are always acknowledged.

Tab. 5: Request

Byte	Contents	Description
1	Length	User data length >7 bytes
2	Node number	Node number
3	0x02	Command SDO Write
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7–N	Value	New value for the specified object entry
(N+1)	CRC	Check sum

Tab. 6: Response

Byte	Contents	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x02	Command SDO Write
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	CRC	Check sum

If the specified object cannot be read, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 23).

3.3.2 Segmented transfer

3.3.2.1 SDO Block Upload

The segmented SDO block upload protocol is based on CiA301.

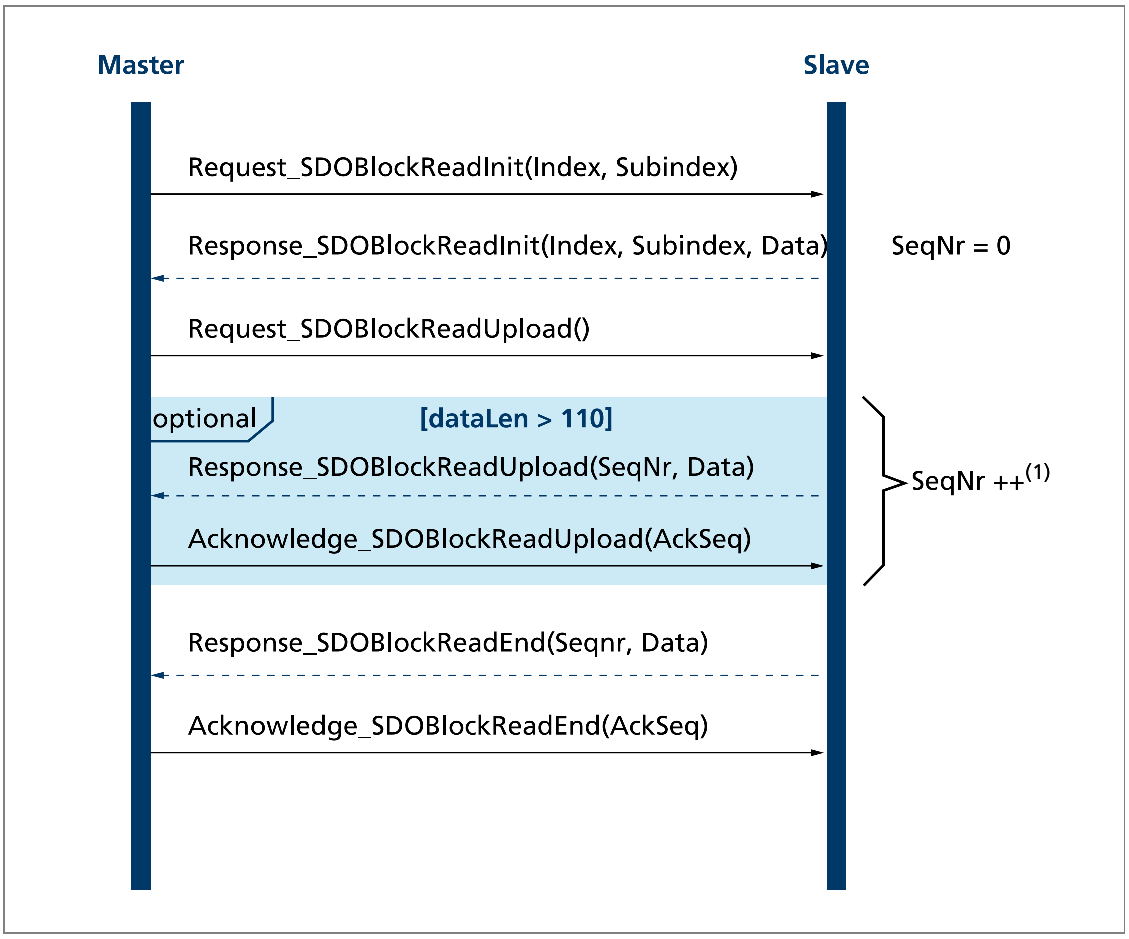
Master
Slave
Request_SDOBlockReadInit(Index, Subindex)
Response_SDOBlockReadInit(Index, Subindex, Data)
SeqNr = 0
Request_SDOBlockReadUpload()
optional
[dataLen > 110]
Response_SDOBlockReadUpload(SeqNr, Data)
Acknowledge_SDOBlockReadUpload(AckSeq)
SeqNr ++(1)
Response_SDOBlockReadEnd(Seqnr, Data)
Acknowledge_SDOBlockReadEnd(AckSeq)

Fig. 6: Sequential diagram for the SDO block upload

1) This is repeated until less than 58 bytes remain to be transmitted

Tab. 7: Request SDO Initiate Block Upload (Master to Slave)

Byte	Contents	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x08	Command SDO Block Read Init
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	CRC	Check sum

Tab. 8: Response SDO Initiate Block Upload (Slave to Master)

Byte	Contents	Description
1	Length	User data length >7 bytes
2	Node number	Node number
3	0x08	Command SDO Block Read Init
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	Data length LB	Overall length of the data to be transmitted in bytes LB
8	Data length HB	Overall length of the data to be transmitted in bytes HB
9–N	Data	Data of the first segment (max. 53 bytes)
(N+1)	CRC	Check sum

If the specified object cannot be read, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 23).

Tab. 9: Request SDO Block Upload (Master to Slave)

Byte	Contents	Description
1	4	User data length 4 bytes
2	Node number	Node number
3	0x09	Command SDO Block Read Upload
4	CRC	Check sum

Tab. 10: Response SDO Block Upload (Slave to Master)

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x09	Command SDO Block Read Upload
4	Sequ no	Sequential number beginning with 1
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Check sum

Tab. 11: Acknowledge SDO Block Upload (Master to Slave)

Byte	Contents	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x09	Command SDO Block Read Upload
4	Ack Sequ	Sequential number received
5	CRC	Check sum

If the data length to be transmitted is longer than 110 bytes (max. data length of an SDO Block Read Init + max. data length of an SDO Block Read End), the SDO Block Read Upload telegram must be sent by the slave in sections one after another with sequential numbers and acknowledged by the master until the complete block has been sent. The last block segment is identified with the command “SDO Block Read End”.

Tab. 12: Response SDO Block Upload End (Slave to Master)

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x0A	Command SDO Block Read End
4	Sequ no	Sequential number of the last segment (>0)
5–N	Data	Data of the last segment (max. 57 bytes)
(N+1)	CRC	Check sum

Tab. 13: Acknowledge SDO Block Upload End (Master to Slave)

Byte	Contents	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x0A	Command SDO Block Read End
4	Ack Sequ	Last sequential number received
5	CRC	Check sum

- If AckSeq = 0, segment transmitted was not received correctly and the segment must be sent again.
- If a processing error occurs at the controller an SDO error response to chap. 3.3.1.1, p. 16 (e.g. timeout) is sent.
- If the block transmission is aborted by the master, an abort SDO transfer telegram is signalled (see chap. 3.3.3, p. 23).

3.3.2.2 SDO Block Download

The segmented SDO block download protocol is based on CiA301.

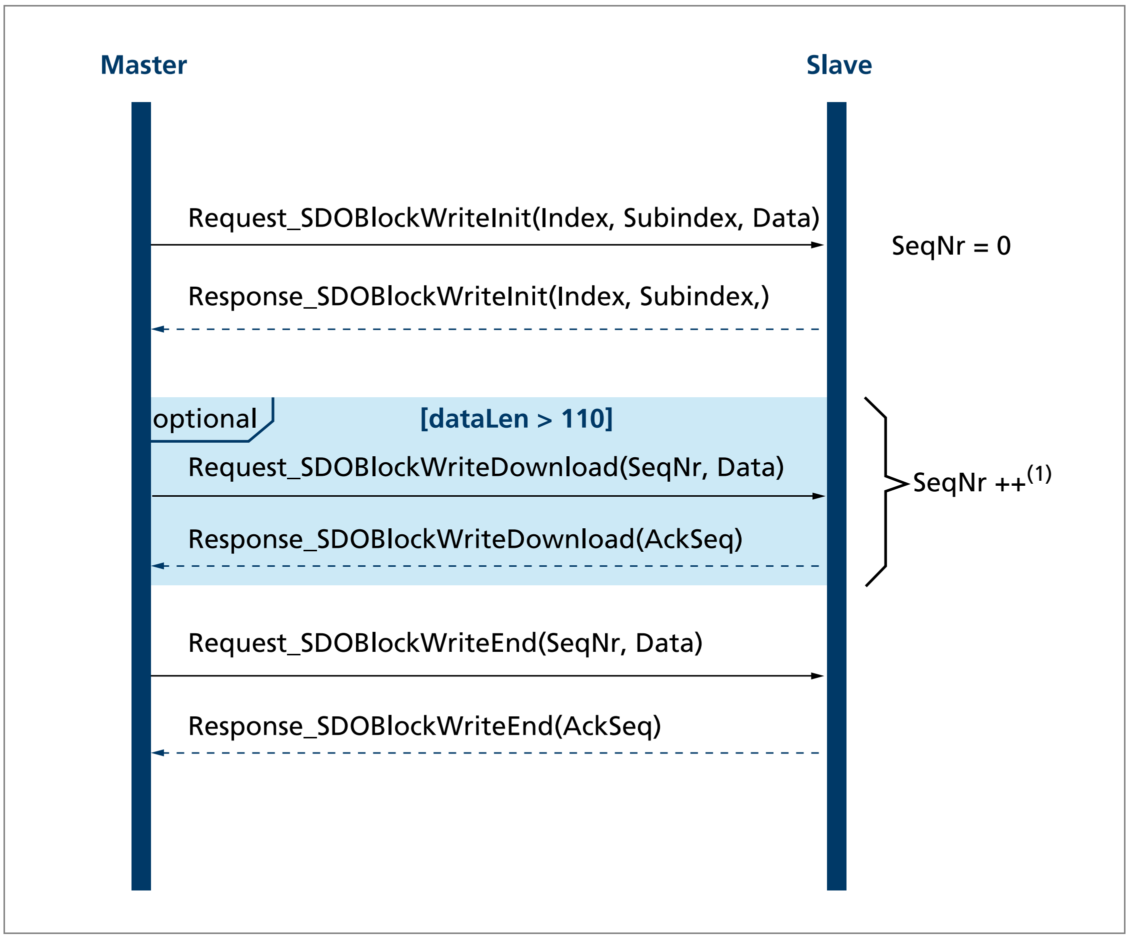
Master
Slave
Request_SDOBlockWriteInit(Index, Subindex, Data)
SeqNr = 0
Response_SDOBlockWriteInit(Index, Subindex,)
optional
[dataLen > 110]
Request_SDOBlockWriteDownload(SeqNr, Data)
SeqNr ++(1)
Response_SDOBlockWriteDownload(AckSeq)
Request_SDOBlockWriteEnd(SeqNr, Data)
Response_SDOBlockWriteEnd(AckSeq)

Fig. 7: Sequential diagram for the SDO block download

1) This is repeated until less than 58 bytes remain to be transmitted

Tab. 14: Request SDO Initiate Block Download (Master to Slave)

Byte	Contents	Description
1	Length	User data length >7 bytes
2	Node number	Node number
3	0x0B	Command SDO Block Write Init
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	Data length LB	Overall length of the data to be transmitted in bytes LB
8	Data length HB	Overall length of the data to be transmitted in bytes HB
9–N	Data	Data of the first segment (max. 53 bytes)
(N+1)	CRC	Check sum

Tab. 15: Response SDO Initiate Block Download (Slave to Master)

Byte	Contents	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x0B	Command SDO Block Write Init
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	CRC	Check sum

Tab. 16: Request SDO Block Download (Master to Slave)

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x0C	Command SDO Block Write Download
4	Sequ no	Sequential number beginning with 1
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Check sum

Tab. 17: Response SDO Block Download (Slave to Master)

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x0C	Command SDO Block Write Download
4	Ack Sequ	Sequential number received
5	CRC	Check sum

The SDO Block Write Download telegram must be sent by the master in sections one after another with sequential numbers and acknowledged by the slave until the complete block has been sent. The last block segment is identified with the command “SDO Block Write End”.

Tab. 18: Request SDO Block Download End (Master to Slave)

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x0D	Command SDO Block Write End
4	Sequ no	Sequential number of the last segment
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Check sum

Tab. 19: *Response SDO Block Download End (Slave to Master)*

Byte	Contents	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x0D	Command SDO Block Write End
4	Ack Sequ	Sequential number received
5	CRC	Check sum

- If AckSeq = 0, segment transmitted was not received correctly and the segment must be sent again.
- If a processing error occurs at the controller an SDO error response to chap. 3.3.1.2, p. 17 (e.g. timeout) is sent.
- If the block transmission is aborted by the master, an abort SDO transfer telegram is signalled (see chap. 3.3.3, p. 23).

Tab. 20: Response

Byte	Contents	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x03	Command SDO Error
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	CRC	Check sum

3.3.3 SDO error handling

SDO error message

Tab. 21: Error response

Byte	Contents	Description
1	11	User data length 11 bytes
2	Node number	Node number
3	0x03	Command SDO Error
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	Error0	Additional error code LB (see Tab. 23, p. 24)
8	Error1	Additional error code HB (see Tab. 23, p. 24)
9	Error2	Error code (see Tab. 23, p. 24)
10	Error3	Error class (see Tab. 23, p. 24)
11	CRC	Check sum

Block transmission aborted by the master (Abort SDO telegram)

Tab. 22: Request Abort SDO Transfer

Byte	Contents	Description
1	11	User data length 11 bytes
2	Node number	Node number
3	0x03	Command SDO Error
4	Index LB	Index of the object entry LB
5	Index HB	Index of the object entry HB
6	Sub-index	Sub-index of the object entry
7	Error0	Additional error code LB (see Tab. 23, p. 24)
8	Error1	Additional error code HB (see Tab. 23, p. 24)
9	Error2	Error code (see Tab. 23, p. 24)
10	Error3	Error class (see Tab. 23, p. 24)
11	CRC	Check sum

Error byte coding

If an abort occurs due to an error in the SDO protocol the bytes 7 to 10 (error 0 to error 3) are coded.

Tab. 23: Error byte coding

Error class	Error code	Additional code	Description
0x05	0x04	0x0001	SDO command 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	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 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

3.4 Emergency object (error message)

The emergency object informs the master of errors asynchronously without requiring interrogation. The emergency object is always size 12 bytes (without SOF and EOF). The emergency message cannot be transmitted in RS232 network operation.

Tab. 24: User data assignment of the emergency telegram

Byte	Contents	Description
1	12	User data length 12 bytes
2	Node number	Node number
3	0x07	Command EMCY
4	Error0	Error code LB
5	Error1	Error code HB
6	Error register	Error register (contents of object 0x1001)
7	Manuf. Spec. Error0	**FAULHABER** error register (contents of object 0x2320)
8	Manuf. Spec. Error1	**Faulhaber** error register HB
9	Manuf. Spec. Error2	Reserved (0)
10	Manuf. Spec. Error3	Reserved (0)
11	Manuf. Spec. Error4	Reserved (0)
12	CRC	Check sum

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. 32, p. 29)
- Bytes 9 to 11: 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. 25, p. 26 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 (chap. 3.7.1, p. 29).

Tab. 25: 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
0x8110 0x8130 0x8140 0x8310	CAN overrun CAN guarding failed CAN recovered from bus stop 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. 26, p. 27 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. 34, p. 30).
- 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. 26: Example of user data assignment to an emergency message

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

3.5 Device control

3.5.1 Boot-up message

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.

i The boot-up message cannot be sent in RS232 network operation.

Tab. 27: Structure of a boot-up message

Byte	Contents	Description
1	Length	User data length >4 bytes
2	Node number	Node number
3	0x00	Command boot-up
4-N	Device Name	Equipment name as boot-up message
(N+1)	CRC	Check sum

3.5.2 Reset Node

A soft reset can be performed by the master by means of the following telegram:

Tab. 28: Structure of a reset node message

Byte	Contents	Description
1	4	User data length 4 bytes
2	Node number	Node number
3	0x00	Command boot-up
4	CRC	Check sum

3.5.3 Device Control

Device Control can be used to perform status changes and to read the current status.

Tab. 29: Request Write controlword (object 0x6040.00 in the object dictionary)

Byte	Contents	Description
1	6	User data length 6 bytes
2	Node number	Node number
3	0x04	Command Controlword
4	Controlword LB	New control word value to Cia402
5	Controlword HB	New control word value to Cia402
6	CRC	Check sum

Tab. 30: Response

Byte	Contents	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x04	Command Controlword
4	Error	Error code: 0 = OK
5	CRC	Check sum

When the status changes, the statusword is sent asynchronously by the drive. This cannot be interrogated directly (the command SDO Read can be used for this).

Tab. 31: Receive statusword (object 0x6041.00 in the object dictionary)

Byte	Contents	Description
1	6	User data length 6 bytes
2	Node number	Node number
3	0x05	Command Statusword
4	Statusword LB	Current value of the status word in accordance with CiA402
5	Statusword HB	Current value of the status word in accordance with CiA402
6	CRC	Check sum

3.6 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. 38)
- The manufacturer-specific area (index 0x2000 to 0x5FFF) contains manufacturer-specific objects (see chap. 7.2, p. 41)
- The standardised device profiles area (0x6000 to 0x9FFF) contains objects supported by the Motion Controller (see the documentation of the drive functions)

3.7 Error handling

3.7.1 Equipment faults

Tab. 32: 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. 33: 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. 3.4, p. 25).

The error mask describes the handling of internal errors depending on the error coding (see Tab. 33, p. 29).

Tab. 34: 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

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. 31).
- Trace logger: On request the parameter values are requested and read continuously (see chap. 4.2, p. 35).

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. 33).

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. 35: 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. 36: 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.

4.1.2 Reading the trace buffer

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

Tab. 37: 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. 38: 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.

4.1.3 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 log service allows individual trace data packets to be requested by request. In this way a continuous recording can be built up over a long period of time. For configuration of the data sources to be transmitted, the same objects are used, source 1 to source 4 of the trace recorders.

The trace request command allows a current data packet with up to 4 data sources can be read in accordance with the set configuration.

Tab. 39: Request

Byte	Contents	Description
1	4	User data length 4 bytes
2	Node number	Node number
3	0x06	Command Trace Log
4	CRC	Check sum

Tab. 40: Response

Byte	Contents	Description
1	Length	User data length >5 bytes
2	Node number	Node number
3	0x06	Command Trace Log
4	Value source 1 LB	Values of the requested data sources, length in accordance with OD
5–N	...	The user data length of the individual data sources is dependent on the number and data length of the values to be transmitted (according to the OD entry).
(N+1)	Time code	Distance from the previous request
(N+2)	CRC	Check sum

i 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

- The node numbers 1 to 127 can be set.
- An RS232 transmission rate in accordance with Tab. 41, p. 36 can be set by inputting the index 0 to 3.
- USB communication does not require specification of the transmission rate

Tab. 41: RS232 bit timing parameter

Baud rate	Index
9600 bit/s	0
19200 bit/s	1
57600 bit/s	2
115200 bit/s	3

The communication parameters are set by writing the following objects in the object dictionary.

Tab. 42: Baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	00	Number of entries	U8	ro	3	Number of object entries
	02	RS232 Baud rate index	U8	rw	0	Index of the Baud rate in accordance with Tab. 41, p. 36
	03	Node ID	U8	rw	255	Node number

A change of the communication parameters is acknowledged with the last setting for the Baud rate and node number. After acknowledgement of the command the new settings are valid. The changed settings are not permanently loaded and available the next time the device is switched on until a “Save” command has been executed for the application parameters.

6 Troubleshooting

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

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 .

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 strin g	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 strin g	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
0x100A	00	Manufacturer’s software version	Vis strin g	const	–	Software version

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

Store parameters

Tab. 43: 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. 44: “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. 45: 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. 46: “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** command must be executed.

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

7.2 Manufacturer-specific objects

Tab. 47: 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	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	

7 Parameter description

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

Baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	00	Number of entries	U8	ro	3	Number of object entries
	02	RS232 Baud rate index	U8	rw	0	Index of the Baud rate in accordance with Tab. 41, p. 36
	03	Node ID	U8	rw	255	Node number

FAULHABER

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