FAULHABER

Communications Manual

MC 5010	MCS
MC 5005	MC 3001
MC 5004	MC 3603
MC 5004 P STO	IMC

RS232 / USB

Version:
7th edition, 2.02.2024

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 Communication structure
- Methods for accessing the parameters
- Drive from the viewpoint of the communication system

This document is intended for software developers and for 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 corresponding data sheet.

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

1.2 Associated documents

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

Manual	Description
Motion Manager 6	Operating instructions for FAULHABER Motion Manager PC software
Quick start guide	Description of the first steps for commissioning and operation of FAULHABER Motion Controllers
Drive functions	Description of the operating modes and functions of the drive
Technical manual	Instructions for installation and use of the FAULHABER Motion Controller

These manuals can be downloaded in pdf format from the web 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
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 designations

⚠ **NOTICE!**
Risk of damage.
▶ Measures for avoidance

ℹ Instructions for understanding or optimizing the operational procedures

✓ Pre-requirement for a requested action
1. First step for a requested action
    ↳ Result of a step
2. Second step of a requested action
↳ Result of an action
▶ Request for a single-step action

2 Overview

2.1 Basic structure of the Motion Controller
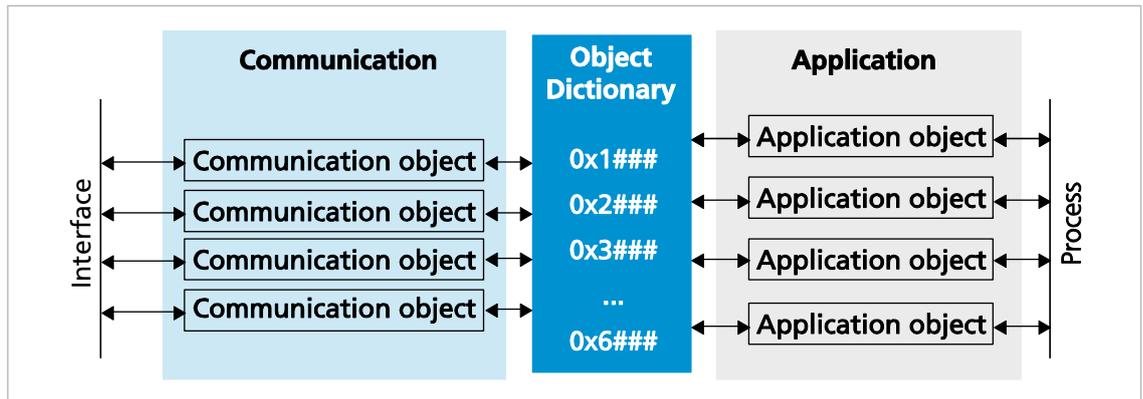
Fig. 1: Basic structure of the Motion Controller

Communication services

The master communicates with the object dictionary via the interface and using the communication services (see chap. 3.2, p. 16). The communication services are based on CAN-open device systems.

Object dictionary

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

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

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

Application part

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

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

2.2 Requirement for communication

FAULHABER drives are supplied with the node number 0xFF (unconfigured) and a RS232 transmission rate of 115 200 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.

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 via 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)

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

2. Configure the host interface to match the drive settings (see chap. 4, p. 33):
   - 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 reports a boot-up message at the last baud rate setting.
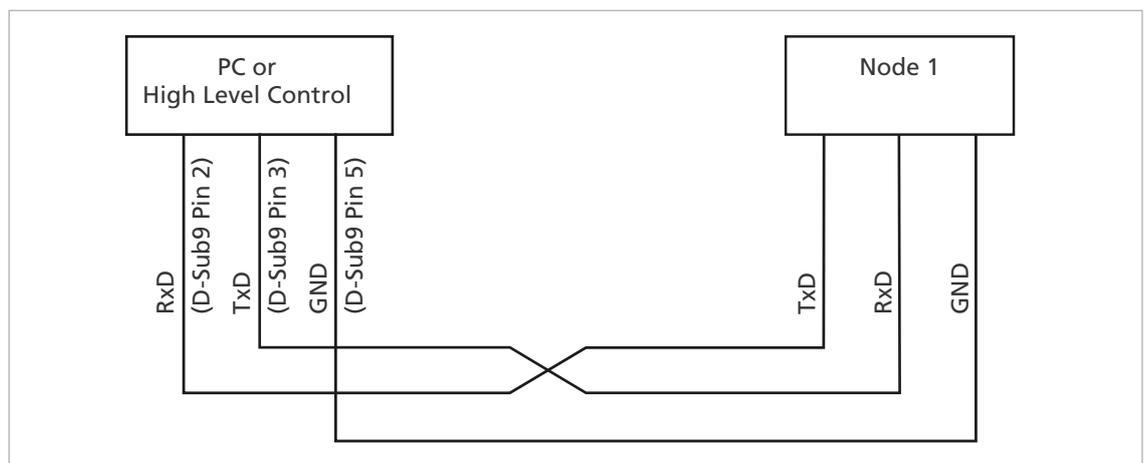
Fig. 2: Wiring between PC/controller and a drive

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.

ℹ️ In network mode, the drives may not send any asynchronous messages because these can interfere with communications with another drive. Asynchronous responses can be deactivated in object 0x2400.04.
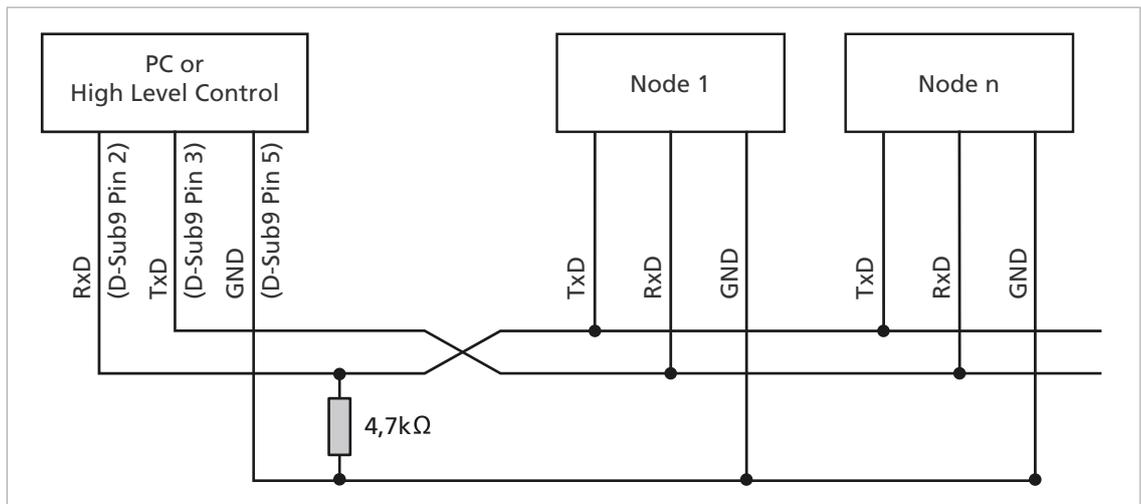
Fig. 3:   Wiring with several Motion Control Systems in RS232 network operation

2.2.2 Operation via the USB interface

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

✓ A USB cable with Mini-USB connector is available

1. Establish a connection with a host interface (typically a PC).

    ↳ Use a USB cable with Mini-USB connector at the device side.

2. Switch on the Motion Controller.

    ↳ Communication will be established.

        The drive reports a boot-up message.

ℹ️ A suitable driver connection is 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 the "FAULHABER Motion Manager" software.

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

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

- Connection wizard: Supports the user in setting up the connection to the connected controller

- Motor wizard: Supports the user in adapting an external controller to the connected motor by selecting the respective FAULHABER motor

- Controller setting wizard: Supports the user in optimizing the controller parameters.

The software can be downloaded free of charge from the FAULHABER website: https://www.faulhaber.com/motionmanager.

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

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

2.4 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 (EEPROM application) (see chap. 5.1, p. 34). When the motor is switched on, the parameters are loaded automatically from the non-volatile memory into RAM.
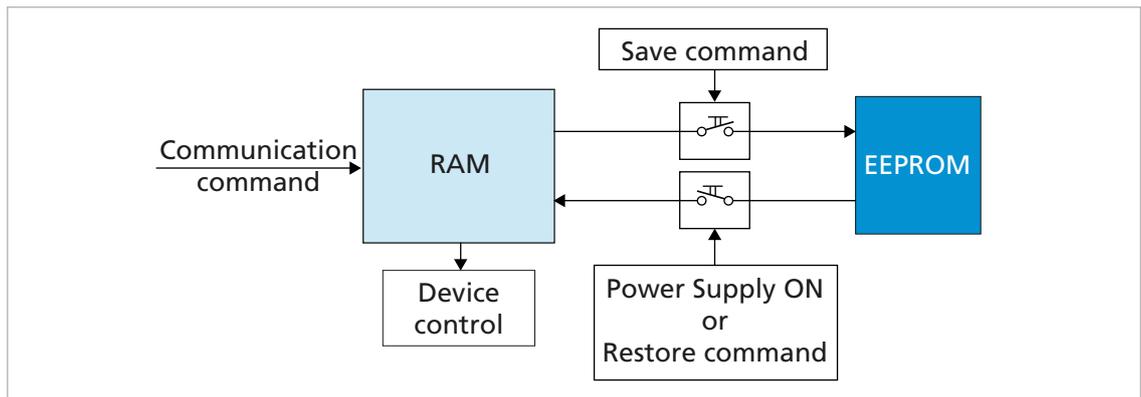
Fig. 4:   Saving and restoring parameters

The following parameters can be loaded using the "Restore" command (see chap. 5.1, p. 34):

■ Factory settings

■ Parameters saved using the "Save" command

2.4.1 Save parameters

The current parameter settings can be saved in the internal EEPROM (SAVE) (see Tab. 38), either completely or for individual ranges.

▸ Write the "save" signature to the subindex 01 to 05 of the object 0x1010 (see Tab. 39).

2.4.2 Restoring settings

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

Factory settings or last saved parameter settings can be loaded from the internal EEPROM at any time, completely or for specific ranges (RESTORE) (see Tab. 40).

1. Write the "Load" signature to the subindex 01 to 06 of the object 0x1011 (see Tab. 41).

    ↳ After Restore Factory (01), Restore Communication (02) and Restore Application (03), the drive must be reset. Only then are the parameters updated.

2. Application parameters (04), together with record 1 and record 2 of the special application parameters (05/06) can be updated with the "Reload" command.

    ↳ The "Reload" command overwrites the values last saved as application parameters.

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

2.4.3 Changing the parameter set

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

Speed controller and filter

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

Position controller

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

Pre-controls

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

Overview

General settings

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

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

Create an application set
▶ Save application parameters 1: Write the "save" signature to subindex 04 of object 0x1010.

   ↳ The current data is saved as the application parameter set 1.

▶ Save application parameters 2: Write the "save" signature to subindex 05 of object 0x1010.

   ↳ The current data is saved as the application parameter set 2.

Activate an application set
▶ Reload application parameters 1: Write the "load" signature to subindex 05 of object 0x1011.

   ↳ Current data from the application parameter set 1 is activated directly.

▶ Reload application parameters 2: Write the "load" signature to subindex 06 of object 0x1011.

   ↳ Current data from the application parameter set 2 is activated directly.

3 Protocol description

3.1 Introduction

Entries in the object dictionary can be written and 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:
- Communication structure
- Control and monitoring functions

CANopen device profiles for drives are defined in CiA 402 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 internally in a common queue and are processed sequentially (FIFO). Acknowledgment by the drive takes place via the same interface.

ℹ️ Each request from the host is concluded with an acknowledgment by the drive. The maximum number of requests buffered in the drive is limited. If no further requests can be added to the 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
1. Byte	SOF	Character (S) as Start of Frame
2. Byte	User data length	Telegram length without SOF/EOF (packet length)
3. Byte	Node number	Node number of the slave (0 = Broadcast)
4. Byte	Command code	See Tab. 2
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. wrong CRC, wrong length, invalid command codes) are not reported back. The node number of the received message, especially in the event of CRC errors, cannot be clearly determined. The telegram remains unanswered and, after the time-out, the master must send the request again.

The characters SOF and EOF are no longer mentioned in the following description. Only the bytes in between are mentioned, and byte 1 corresponds to the 2nd byte 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.

Protocol description

Command codes

Tab. 2: Functions of the command codes

Command code	Name	Function
0x00	Bootup	Boot-up message / Reset Node (receive / request)
0x01	SDORead	Read the object dictionary entry (request / response)
0x02	SDOWrite	Write an object dictionary entry (request / response)
0x03	SDOError	SDO error (abort request / error response)
0x04	Controlword	Writing the controlword (request / response)
0x05	Statusword	Reception of the statusword (receive)
0x06	TraceLog	Trace Request for Trace Logger (request / response)
0x07	Emcy	Reception of an emergency message (receive)
0x08	SDOBlockReadInit	Initialization of the segmented SDO block upload (request / response)
0x09	SDOBlockReadUpload	Execution of the segmented SDO block upload (request / response / acknowledge)
0x0A	SDOBlockReadEnd	End of the segmented SDO block upload (request / acknowledge)
0x0B	SDOBlockWriteInit	Initialization of the segmented SDO block download (request / response)
0x0C	SDOBlockWriteDownload	Execution of the segmented SDO block download (request / response)
0x0D	SDOBlockWriteEnd	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 bytes (without SOF/EOF) of the telegram to be transmitted:

```
#define Polynom 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) ^ Polynom;
        }
        else {
            u8CRC >>= 1;
        }
    }
    return u8CRC;
}
```

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

3.2   Communication services
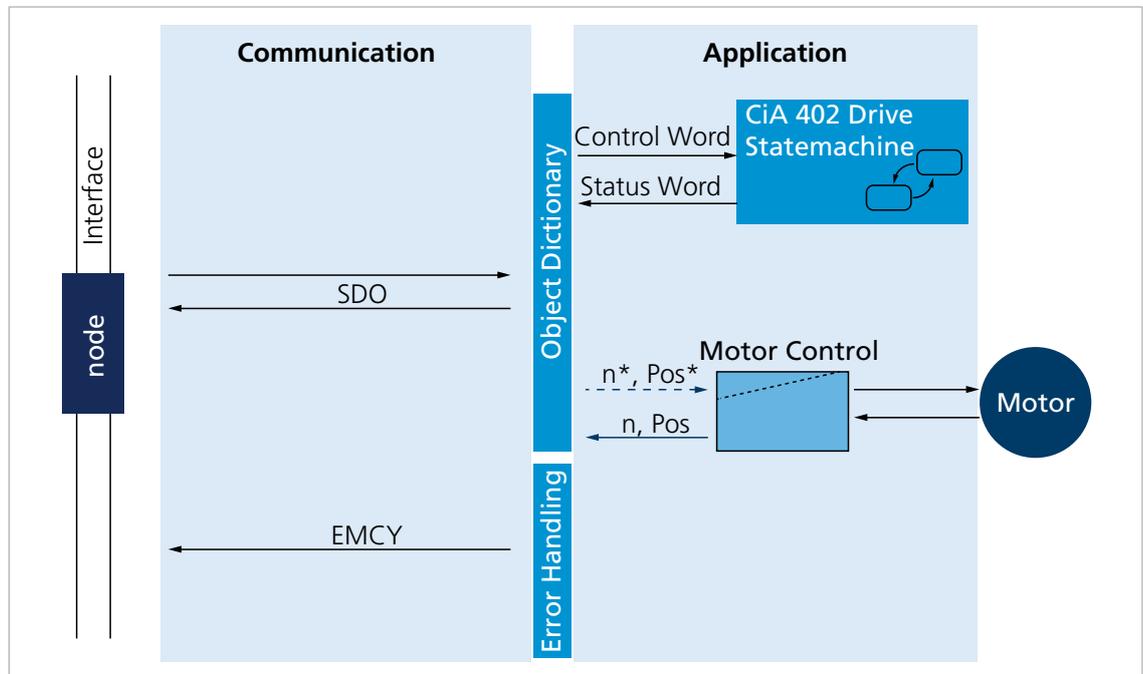
Fig. 5:   Communication services of the Motion Controller

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

Tab. 3:   Distribution of the SDO transfer types

Transfer type	Number of bytes	Purpose
Expedited transfer	Maximum 4 bytes	Read and write individual numeric parameters
Segmented Transfer	More than 58 bytes	Read text parameters (such as device name, firmware version) and transmit data blocks (such as the trace buffer)

3.3.1 Expedited transfer

3.3.1.1 Reading the object dictionary

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

Tab. 4:   Request

Byte	Content	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x01	SDORead command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	CRC	Checksum

Tab. 5:   Response

Byte	Content	Description
1	Length	User data length > 7 bytes
2	Node number	Node number
3	0x01	SDORead command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7–N	Value	Current value of the specified object entry
(N+1)	CRC	Checksum

If the specified object cannot be read out, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 25).

3.3.1.2 Writing to the object dictionary

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

Tab. 6: Request

Byte	Content	Description
1	Length	User data length > 7 bytes
2	Node number	Node number
3	0x02	SDOWrite command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7–N	Value	New value for the specified object entry
(N+1)	CRC	Checksum

Tab. 7: Response

Byte	Content	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x02	SDOWrite command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	CRC	Checksum

If the specified object cannot be written, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 25).

Protocol description

3.3.2 Segmented Transfer

3.3.2.1 SDO Block Upload

The segmented SDO block upload protocol is based on CiA301.

Fig. 6: Sequential diagram for the SDO block upload

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

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

Byte	Content	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x08	SDOBlockReadInit command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	CRC	Checksum

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

Byte	Content	Description
1	Length	User data length > 7 bytes
2	Node number	Node number
3	0x08	SDOBlockReadInit command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex 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	Checksum

If the specified object cannot be read out, the response is an SDO error in accordance with CiA301 (see chap. 3.3.3, p. 25).

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

Byte	Content	Description
1	4	User data length 4 bytes
2	Node number	Node number
3	0x09	SDOBlockReadUpload command
4	CRC	Checksum

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

Byte	Content	Description
1	Length	User data length > 5 bytes
2	Node number	Node number
3	0x09	SDOBlockReadUpload command
4	SeqNr	Sequential number beginning with 1
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Checksum

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

Byte	Content	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x09	SDOBlockReadUpload command
4	AckSeq	Sequential number received
5	CRC	Checksum

If the data length to be transmitted is longer than 110 bytes (max. data length of SDOBlockReadInit + max. data length of SDOBlockReadEnd), the SDOBlockReadUpload 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 "SDOBlockReadEnd".

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

Byte	Content	Description
1	Length	User data length > 5 bytes
2	Node number	Node number
3	0x0A	SDOBlockReadEnd command
4	SeqNr	Sequential number of the last segment (>0)
5–N	Data	Data of the last segment (max. 57 bytes)
(N+1)	CRC	Checksum

Tab. 14: Acknowledge SDO Block Upload End (Master to Slave)

Byte	Content	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x0A	SDOBlockReadEnd command
4	AckSeq	Last sequential number received
5	CRC	Checksum

- If AckSeq = 0, the transmitted segment was not received correctly and the segment must be sent again.

- If a processing error occurs at the controller, an SDO error response acc. to chap. 3.3.3, p. 25 (e.g. timeout) is sent.

- If the block transmission is aborted by the master, an Abort SDO Transfer telegram is signaled (see chap. 3.3.3, p. 25).

3.3.2.2 SDO Block Download

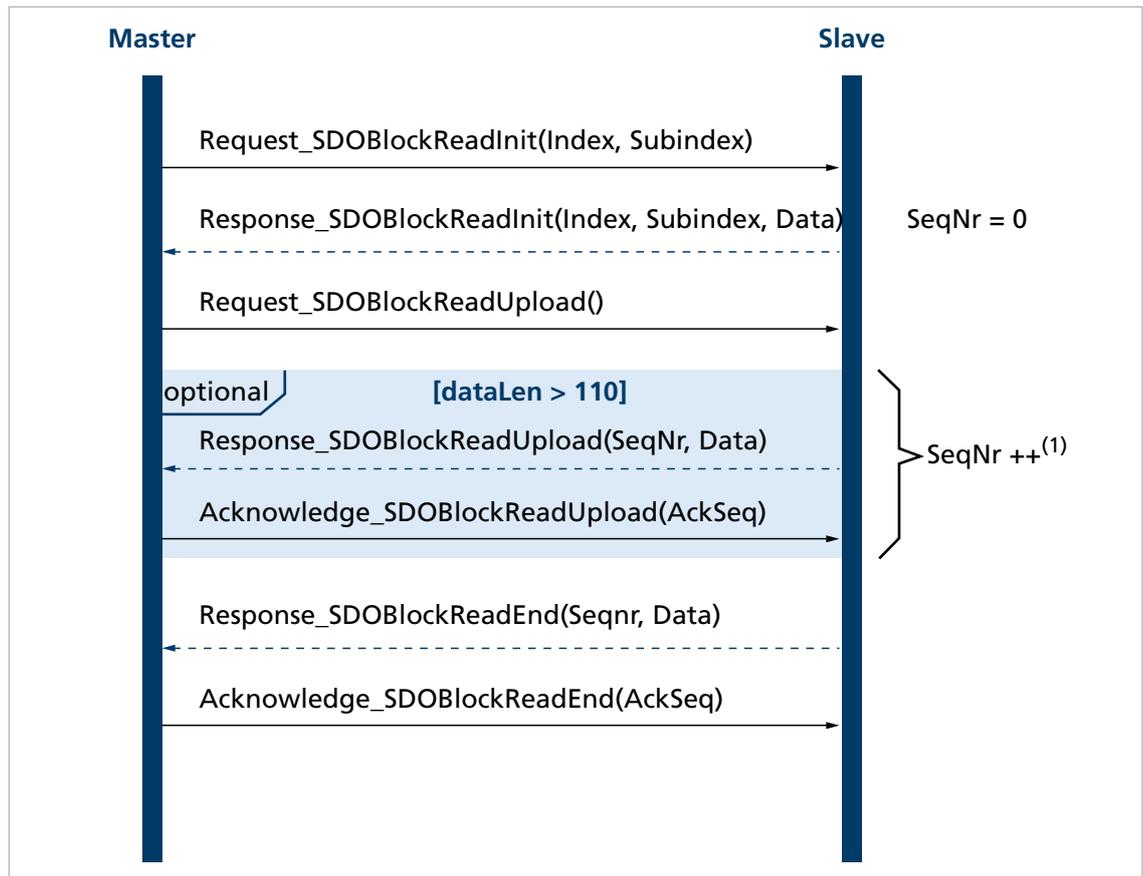The segmented SDO block download protocol is based on CiA301.
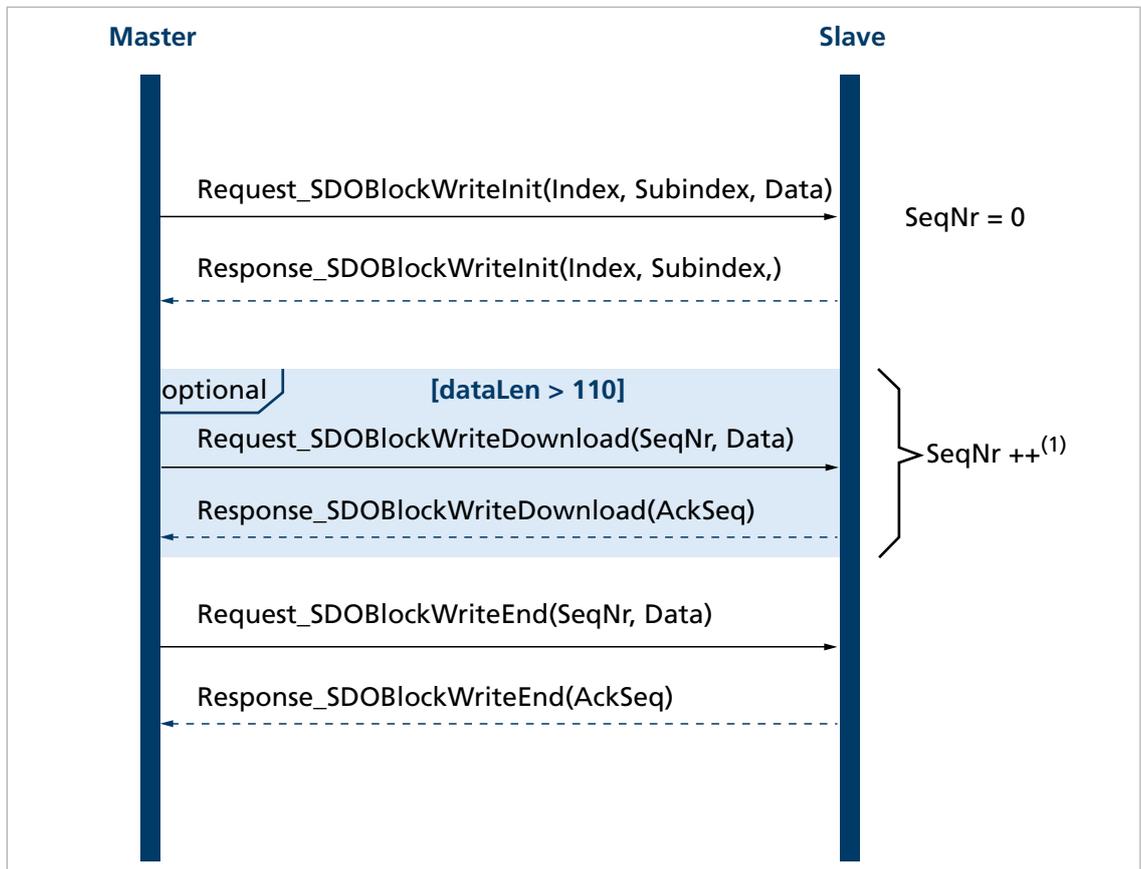
Fig. 7: Sequential diagram for the SDO block download

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

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

Byte	Content	Description
1	Length	User data length > 7 bytes
2	Node number	Node number
3	0x0B	SDOBlockWriteInit command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex 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	Checksum

Protocol description

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

Byte	Content	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x0B	SDOBlockWriteInit command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	CRC	Checksum

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

Byte	Content	Description
1	Length	User data length > 5 bytes
2	Node number	Node number
3	0x0C	SDOBlockWriteDownload command
4	SeqNr	Sequential number beginning with 1
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Checksum

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

Byte	Content	Description
1	Length	User data length > 5 bytes
2	Node number	Node number
3	0x0C	SDOBlockWriteDownload command
4	AckSeq	Sequential number received
5	CRC	Checksum

The SDOBlockWriteDownload 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 "SDOBlock-WriteEnd".

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

Byte	Content	Description
1	Length	User data length > 5 bytes
2	Node number	Node number
3	0x0D	SDOBlockWriteEnd command
4	SeqNr	Sequential number of the last segment
5–N	Data	Data of the respective segment (max. 57 bytes)
(N+1)	CRC	Checksum

Protocol description

Tab. 20: Response SDO Block Download End (Slave to Master)

Byte	Content	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x0D	SDOBlockWriteEnd command
4	AckSeq	Sequential number received
5	CRC	Checksum

- If AckSeq = 0, the transmitted segment was not received correctly and the segment must be sent again.

- If a processing error occurs at the controller, an SDO error response acc. to chap. 3.3.1.2, p. 18 (e.g. timeout) is sent.

- If the block transmission is aborted by the master, an Abort SDO Transfer telegram is signaled (see chap. 3.3.3, p. 25).

Tab. 21: Response

Byte	Content	Description
1	7	User data length 7 bytes
2	Node number	Node number
3	0x03	SDOError command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	CRC	Checksum

3.3.3 SDO error handling

SDO error message

Tab. 22: Error-Response

Byte	Content	Description
1	11	User data length 11 bytes
2	Node number	Node number
3	0x03	SDOError command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	Error0	LB additional error code (see Tab. 24)
8	Error1	HB additional error code (see Tab. 24)
9	Error2	Error code (see Tab. 24)
10	Error3	Error class (see Tab. 24)
11	CRC	Checksum

Block transmission aborted by the master (Abort SDO telegram)

Tab. 23: Request Abort SDO Transfer

Byte	Content	Description
1	11	User data length 11 bytes
2	Node number	Node number
3	0x03	SDOError command
4	Index LB	Index of the LB object entry
5	Index HB	Index of the HB object entry
6	Subindex	Subindex of the object entry
7	Error0	LB additional error code (see Tab. 24)
8	Error1	HB additional error code (see Tab. 24)
9	Error2	Error code (see Tab. 24)
10	Error3	Error class (see Tab. 24)
11	CRC	Checksum

Protocol description

Error byte coding

If the SDO protocol is aborted in the event of an error, the bytes 7 to 10 (error 0 to error 3) are coded.

Tab. 24: 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	Subindex not present
0x06	0x09	0x0030	General value range error
0x06	0x09	0x0031	Value range error: Parameter value too high
0x06	0x09	0x0032	Value range error: Parameter value too low
0x06	0x09	0x0036	Value range error: Maximum value smaller than minimum value
0x08	0x00	0x0000	General SDO error
0x08	0x00	0x0020	Cannot be accessed
0x08	0x00	0x0022	Cannot be accessed at current device status

3.4 Emergency object (error message)

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

Tab. 25: User data assignment of the emergency telegram

Byte	Content	Description
1	12	User data length 12 bytes
2	Node number	Node number
3	0x07	EMCY command
4	Error0	Error code LB
5	Error1	Error code HB
6	Error-Reg	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	Checksum

Assignment of user data:
- Error0(LB)/Error1(HB): 16-bit error code

- Error-Reg: Error register (contents of object 0x1001, see chap. 5.1, p. 34)

- FE0(LB)/FE1(HB): 16-bit FAULHABER error register (contents of object 0x2320, see Tab. 33)

- 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 contains the last value of the error register.

Tab. 26 lists all the errors that have been reported by emergency messages, provided that the respective error is included in the emergency mask for the FAULHABER error register (chap. 3.7.1, p. 31).

Protocol description

Tab. 26: Emergency error codes

Emergency message		FAULHABER error register 0x2320			Error register 0x1001	
Error Code	Designation	Error mask 0x2321	Bit	Designation	Bit	Designation
0x0000	No error (is sent out when an error is no longer present or has been acknowledged)	–	–	–	–	–
–	–	–	–		0	Generic error (is set if one of the error bits 1 to 7 is set)
0x3210	Overvoltage	0x0004	2	OverVoltageError	2	Voltage error
0x3220	Undervoltage	0x0008	3	UnderVoltageError	2	Voltage error
0x43F0	Temperature Warning	0x0010	4	TempWarning	1	Current error [a]
0x4310	Temperature Error	0x0020	5	TempError	3	Temperature error
0x5410	Output stages	0x0080	7	IntHWError	7	Manufacturer-specific error
0x5530	EEPROM fault	0x0400	10	MemError	–	–
0x6100	Software error	0x1000	12	CalcError	7	Manufacturer-specific error
0x7200	Measurement Circuit: Current Measurement	0x0200	9	CurrentMeasError	7	Manufacturer-specific error
0x7300	Sensor Fault (Encoder)	0x0040	6	EncoderError	7	Manufacturer-specific error
0x7400	Computation circuit: module fault	0x0100	8	ModuleError	7	Manufacturer-specific error
0x8110 0x8130 0x8140 0x8310	CAN overrun CAN guarding failed CAN recovered from bus off RS232 overrun	0x0800	11	ComError	4	Communication error
0x84F0	Deviation error (velocity controller)	0x0001	0	SpeedDeviationError	5	Drive-specific error
0x84FF	Max speed error	0x2000	13	DynamicError	7	Manufacturer-specific error
0x8611	Following error (position controller)	0x0002	1	FollowingError	5	Drive-specific error

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

Example

An emergency message with the user data assignment in Tab. 27 is sent in the following event:

- In the Error Mask 0x2321, bit 1 (following error) is set under subindex 1 (emergency mask) (see Tab. 35).

- The control deviation corridor set in object 0x6065.00 for the position controller has been exceeded for an extended period as defined by the value set for the error delay time in object 0x6066.00 (see the documentation of the drive functions).

Tab. 27: Example of user data assignment to an emergency message

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

3.5   Device controller

3.5.1   Boot-up message

Immediately after the initialization phase the Motion Controller sends a boot-up message. A boot-up message signals the end of the initialization phase of a module after it has been switched on.

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

Tab. 28: Structure of a boot-up message

Byte	Content	Description
1	Length	User data length > 4 bytes
2	Node number	Node number
3	0x00	BootUp command
4-N	Device Name	Device name as boot-up message
(N+1)	CRC	Checksum

3.5.2   Reset Node

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

Tab. 29: Structure of a reset node message

Byte	Content	Description
1	4	User data length 4 bytes
2	Node number	Node number
3	0x00	BootUp command
4	CRC	Checksum

FAULHABER

3.5.3 Device Control

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

Tab. 30: "Request Write" controlword (object 0x6040.00 in the object dictionary)

Byte	Content	Description
1	6	User data length 6 bytes
2	Node number	Node number
3	0x04	Controlword command
4	Controlword LB	New controlword value to Cia402
5	Controlword HB	New controlword value to Cia402
6	CRC	Checksum

Tab. 31: Response

Byte	Content	Description
1	5	User data length 5 bytes
2	Node number	Node number
3	0x04	Controlword command
4	Error	Error code: 0 = Ok
5	CRC	Checksum

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

Byte	Content	Description
1	6	User data length 6 bytes
2	Node number	Node number
3	0x05	Statusword command
4	Statusword LB	Current value of the statusword in accordance with CiA402
5	Statusword HB	Current value of the statusword in accordance with CiA402
6	CRC	Checksum

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 subindex (SDO protocol).

- Communication parameters (index 0x1000 to 0x1FFF) contains communications objects to CiA 301, see chap. 5.1, p. 34)

- Manufacturer-specific area (index 0x2000 to 0x5FFF) contains manufacturer-specific objects, see chap. 5.2, p. 37)

- The standardized 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 Device faults

Tab. 33: FAULHABER error register (0x2320)

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

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

Tab. 34: Error coding

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

All of these errors correspond to an Emergency Error Code. (see chap. 3.4, p. 27).

The error mask describes the handling of internal errors depending on the error coding (see Tab. 34).

Protocol description

Tab. 35: Error Mask (0x2321)

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

Examples:

■ When the fault mask (subindex 2) of object 0x2321 is set to 0x0001 the drive is switched off due to overcurrent and its state machine is set to a *Fault Reaction Active* state.

■ When the subindex 3 of object 0x2321 is set to 0, the error output (fault pin) indicates no error. When the subindex 3 of object 0x2321 is set to 0xFFFF, the error output (fault pin) indicates all errors.

4 Communication settings

- The node numbers 1 to 127 can be set.

- An RS232 transfer rate in accordance with Tab. 36 can be set by inputting the index 0 to 3.

- USB does not require specification of the transfer rate

Tab. 36: 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. 37: Baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	0x00	Number of Entries	U8	ro	8	Number of object entries
	0x02	RS232 Rate	U8	rw	3	Index of the baud rate according to Tab. 36
	0x03	Node ID	U8	rw	1	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.

FAULHABER

5 Parameter description

5.1 Communication objects acc. to CiA 301

Device Type

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

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

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

Error Register

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

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

This parameter can be mapped in a PDO.

Predefined Error Field (error log)

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

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

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

The meaning of the error codes is described in chap. 3.4, p. 27.

Writing a 0 to subindex 0 clears the error log (see Tab. 34).

Manufacturer Device Name

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

The segmented SDO protocol must be used to read out the manufacturer device name.

Manufacturer Hardware Version

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

The segmented SDO protocol is used to read out the manufacturer hardware version.

Parameter description

Manufacturer Software Version

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

The segmented SDO protocol must be used to read the Manufacturer Software Version.

Store Parameters

Tab. 38: Save parameters

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

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

Tab. 39: Signature "save"

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

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

▶ Avoid performing frequent saves.

▶ After 10,000 save cycles, replace the device.

Restore Default Parameters

Tab. 40: Restoring parameters

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

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

Tab. 41: "Load" signature

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

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

Identity Object

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

5.2 Manufacturer-specific objects

FAULHABER error register (0x2320)

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

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 Objekt Error Mask (0x2321).

Error Mask (0x2321)

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

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

Trace Configuration

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

Parameter description

Trace Buffer

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

RS232 baud rate index and node number

Index	Subindex	Name	Type	Attr.	Default value	Meaning
0x2400	0x00	Number of Entries	U8	ro	8	Number of object entries
	0x02	RS232 Rate	U8	rw	3	Index of the baud rate according to Tab. 36
	0x03	Node ID	U8	rw	1	Node number
	0x04	Communication Settings	U32	rw	0	Bit mask for communication settings according to Tab. 42
	0x05	RS232 NetMode	U8	rw	0	Settings for RS232 network operation: • 0: Network operation not active • 1: network operation active
	0x06	ComState	U16	ro	0	Bit mask for communication status according to Tab. 43

Tab. 42: Meaning of the bits for 0x2400.04 (Communication Settings)

Bit	Description
0	Transmit EMCY
1	AsyncDriveStatus
2…31	Reserved

Tab. 43: Meaning of the bits for 0x2400.06 (ComState)

Bit	Description
0…6	Reserved
7	Transmit Overflow Signaled
8	BufferOverflow
9…14	Reserved
15	PDOLength