

# CANopen®

## CANopen – SD6 Manual

en-US  
10/2023  
ID 442637.02



**STÖBER**

# Table of contents

<b>1</b>	<b>Foreword .....</b>	<b>5</b>
<b>2</b>	<b>User information.....</b>	<b>6</b>
2.1	Storage and transfer.....	6
2.2	Described product .....	6
2.3	Directives and standards .....	6
2.4	Timeliness.....	6
2.5	Original language.....	6
2.6	Limitation of liability.....	6
2.7	Formatting conventions .....	7
2.7.1	Display of warning messages.....	7
2.7.2	Markup of text elements .....	8
2.7.3	Mathematics and formulas.....	8
2.8	Trademarks.....	9
<b>3</b>	<b>Safety notes .....</b>	<b>10</b>
<b>4</b>	<b>Network structure.....</b>	<b>11</b>
<b>5</b>	<b>CA6 communication module .....</b>	<b>12</b>
5.1	Installation.....	12
<b>6</b>	<b>Connection.....</b>	<b>13</b>
6.1	Selecting suitable cables.....	13
6.2	Terminal resistances.....	13
6.3	Baud rate and cable length.....	14
6.4	X200: Fieldbus connection .....	14
<b>7</b>	<b>What you should know before commissioning .....</b>	<b>15</b>
7.1	DS6 program interface .....	15
7.1.1	Configuring the view.....	17
7.1.2	Navigation using sensitive circuit diagrams.....	18
7.2	Meaning of parameters.....	19
7.2.1	Parameter groups .....	19
7.2.2	Parameter types and data types.....	20
7.2.3	Parameter types .....	21
7.2.4	Parameter structure .....	21
7.2.5	Parameter visibility.....	22
7.3	Signal sources and process data mapping.....	23
7.4	Non-volatile memory.....	23

<b>8</b>	<b>Commissioning.....</b>	<b>24</b>
8.1	DS6: Configuring the drive controller .....	25
8.1.1	Initiating the project .....	25
8.1.2	Parameterizing general CANopen settings .....	28
8.1.3	Configuring PDO transmission .....	29
8.1.4	Transmitting and saving a configuration .....	31
8.2	Putting the CANopen network into operation .....	33
<b>9</b>	<b>Monitoring and diagnostics .....</b>	<b>34</b>
9.1	Connection monitoring.....	34
9.2	LED display.....	35
9.2.1	CANopen state .....	35
9.3	Events .....	36
9.3.1	Event 52: Communication .....	37
9.4	Parameters .....	38
9.4.1	A83   Bus address   G6   V0.....	38
9.4.2	A245   CAN diagnostic   G6   V0.....	38
9.4.3	A246   CANopen NMT-State   G6   V1.....	38
<b>10</b>	<b>More about CANopen?.....</b>	<b>39</b>
10.1	CAN and CANopen.....	39
10.1.1	CANopen – Communication .....	39
10.1.2	Object directory.....	40
10.1.3	Network structure .....	40
10.2	CAN message.....	41
10.2.1	Predefined connection set.....	41
10.2.2	Dynamic distribution .....	43
10.3	Communication objects.....	44
10.3.1	SYNC: Synchronization Objects.....	45
10.3.2	PDO: Process Data Objects .....	45
10.3.3	SDO: Service Data Objects .....	48
10.3.4	NMT: Network Management Objects.....	53
10.3.5	ERROR CONTROL: Error Control Objects .....	55
10.3.6	EMCY: Emergency Objects.....	57
10.4	Cycle times.....	57
10.5	Activating and executing actions .....	58
10.6	Fieldbus scaling.....	60
<b>11</b>	<b>Appendix.....</b>	<b>61</b>
11.1	Supported communication objects .....	61
11.1.1	CiA 301 CANopen: 1000 hex – 1FFFF hex .....	61
11.1.2	Manufacturer-specific parameters: 2000 hex – 53FF hex .....	65

11.2 EMCY message: Device fault error codes ..... 67

11.3 Detailed information ..... 69

11.4 Abbreviations ..... 70

**12 Contact ..... 71**

12.1 Consultation, service and address..... 71

12.2 Your opinion is important to us..... 71

12.3 Close to customers around the world ..... 72

**Glossary ..... 73**

# 1 Foreword

STÖBER drive controllers of the 6th generation offer maximum precision and productivity for automation technology and mechanical engineering despite ever more complex functions.

Drive controllers from the SD6 series can be equipped with the CA6 communication module to provide reliable communication between the drive technology and controller over the CANopen fieldbus system.

This documentation describes a combination of the drive controllers listed with a controller as an CANopen master and the associated automation software.

## 2 User information

This documentation assists you with commissioning the SD6 series of the 6th generation of STÖBER drive controllers in combination with higher level controller systems over an CANopen network.

The SD6 drive controllers provide CANopen functionality over a separate CA6 communication module, which you have to install if it is not pre-installed.

### Technical knowledge

Operating a CANopen network requires having familiarity with the basics of the CANopen network technology.

### Technical requirements

Before you begin operating your CANopen network, you need to wire the drive controllers and initially check that they are functioning correctly. To do this, follow the instructions in the manual for the drive controller.

## 2.1 Storage and transfer

As this documentation contains important information for handling the product safely and efficiently, it must be stored in the immediate vicinity of the product until product disposal and be accessible to qualified personnel at all times.

Also pass on this documentation if the product is transferred or sold to a third party.

## 2.2 Described product

This documentation is binding for:

SD6 series drive controllers in conjunction with the DriveControlSuite software (DS6) in V 6.5-K or later and associated firmware in V 6.5-K or later.

## 2.3 Directives and standards

Refer to the drive controller documentation for the European directives and standards relevant to the drive controller and accessories.

## 2.4 Timeliness

Check whether this document is the latest version of the documentation. We make the latest document versions for our products available for download on our website:

<http://www.stoeber.de/en/downloads/>.

## 2.5 Original language

The original language of this documentation is German; all other language versions are derived from the original language.

## 2.6 Limitation of liability

This documentation was created taking into account the applicable standards and regulations as well as the current state of technology.

No warranty or liability claims for damage shall result from failure to comply with the documentation or from use that deviates from the intended use of the product. This is especially true for damage caused by individual technical modifications to the product or the project configuration and operation of the product by unqualified personnel.

## 2.7 Formatting conventions

Orientation guides in the form of signal words, symbols and special text markups are used to emphasize specific information so that you are able identify it in this documentation quickly.

### 2.7.1 Display of warning messages

Warning messages are identified with symbols. They indicate special risks when handling the product and are accompanied by relevant signal words that express the extent of the risk. Furthermore, useful tips and recommendations for efficient, error-free operation are specially highlighted.

#### ATTENTION!

##### Attention

This indicates that damage to property may occur

- if the stated precautionary measures are not taken.

#### ⚠ CAUTION!

##### Caution

This word with a warning triangle indicates that minor personal injury may occur

- if the stated precautionary measures are not taken.

#### ⚠ WARNING!

##### Warning

This word with a warning triangle means there may be a considerable risk of fatal injury

- if the stated precautionary measures are not taken.

#### ⚠ DANGER!

##### Danger

This word with a warning triangle indicates that there is a considerable risk of fatal injury

- if the stated precautionary measures are not taken.

#### Information

Information indicates important information about the product or serves to emphasize a section in the documentation that deserves special attention from the reader.

## 2.7.2 Markup of text elements

Certain elements of the continuous text are distinguished as follows.

<b>Important information</b>	Words or expressions with a special meaning
Interpolated position mode	Optional: File or product name or other name
<u>Detailed information</u>	Internal cross-reference
<a href="http://www.samplelink.com">http://www.samplelink.com</a>	External cross-reference

### Software and other displays

The following formatting is used to identify the various information content of elements referenced by the software interface or a drive controller display, as well as any user entries.

Main menu Settings	Window names, dialog box names, page names or buttons, combined proper nouns, functions referenced by the interface
Select Referencing method A	Predefined entry
Save your <own IP address>	User-defined entry
EVENT 52: COMMUNICATION	Displays (status, messages, warnings, faults)

Keyboard shortcuts and command sequences or paths are represented as follows.

[Ctrl], [Ctrl] + [S]	Key, key combination
Table > Insert table	Navigation to menus/submenus (path specification)

## 2.7.3 Mathematics and formulas

The following signs are used to represent mathematical relationships and formulas.

–	Subtraction
+	Addition
×	Multiplication
÷	Division
	Absolute value



## 2.8 Trademarks

The following names used in connection with the device, its optional equipment and its accessories are trademarks or registered trademarks of other companies:

CANopen<sup>®</sup>,  
CiA<sup>®</sup>

CANopen<sup>®</sup> and CiA<sup>®</sup> are registered European Union trademarks of CAN in AUTOMATION e.V., Nuremberg, Germany.

Windows<sup>®</sup>,  
Windows<sup>®</sup> 7,  
Windows<sup>®</sup> 10,  
Windows<sup>®</sup> 11

Windows<sup>®</sup>, the Windows<sup>®</sup> logo, Windows<sup>®</sup> XP, Windows<sup>®</sup> 7, Windows<sup>®</sup> 10, and Windows<sup>®</sup> 11 are registered trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks not listed here are the property of their respective owners.

Products that are registered as trademarks are not specially indicated in this documentation. Existing property rights (patents, trademarks, protection of utility models) are to be observed.

## 3 Safety notes

### **WARNING!**

#### **Risk of fatal injury if safety notes and residual risks are not observed!**

Failure to observe the safety notes and residual risks in the drive controller documentation may result in accidents causing serious injury or death.

- Observe the safety notes in the drive controller documentation.
- Consider the residual risks in the risk assessment for the machine or system.

### **WARNING!**

#### **Malfunction of the machine due to incorrect or modified parameterization!**

In the event of incorrect or modified parameterization, malfunctions can occur on machines or systems which can lead to serious injuries or death.

- Observe the security notes in the drive controller documentation.
- Protect the parameterization, e.g. from unauthorized access.
- Take appropriate measures for possible malfunctions (e.g. emergency off or emergency stop).

## 4 Network structure

A CANopen network normally consists of a CANopen master (controller) and CANopen slaves, i.e. drive controllers from the SD6 series.

The CANopen network structure is generally optimized for a line topology. All nodes in the CANopen network (master and slaves) are connected via 3 shielded lines: CAN-High (CAN-H) and CAN-Low (CAN-L) as well as an additional line for the reference potential (GND) between the nodes. To avoid data overlapping due to signal reflections on the two bus ends and thus to reduce the failure proneness significantly, terminating resistors of 120  $\Omega$  are connected between CAN-High and CAN-Low.

You can configure the drive controllers using the STÖBER DriveControlSuite commissioning software and the entire CANopen network using the automation software of the controller.

The following graphic presents an abstract of a CANopen network with a controller as the CANopen master and several SD6 drive controllers as CANopen slaves.

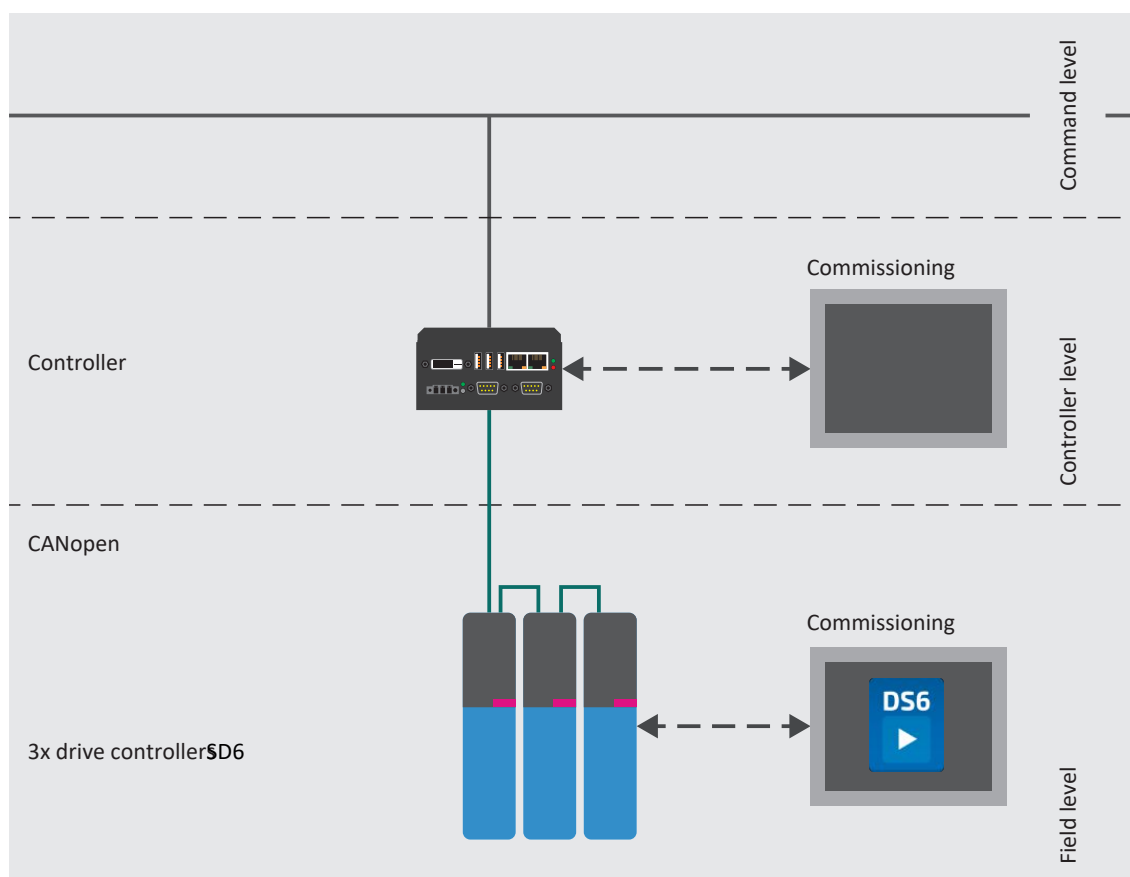


Fig. 1: CANopen: Network structure, using the SD6 series as an example

## 5 CA6 communication module

The drive controllers of the SD6 series are connected to each other and to the CANopen master using the CA6 communication module, which offers the necessary fieldbus interfaces.

CA6 communication modules correspond to the CANopen standard and make it possible for the CANopen master to access all relevant drive parameters and functions cyclically and acyclically.

### 5.1 Installation

Installation work is permitted only when no voltage is present. Observe the 5 safety rules.

Note the minimum clearances listed in the technical data during installation in order to prevent the drive controller from overheating.

Protect the device against falling parts (wire scraps, wires, pieces of metal, etc.) during installation or other work in the control cabinet. Parts with conductive properties may result in a short circuit or failure inside the drive controller.

Remove the additional covers before commissioning so that the drive controller does not overheat.

#### **WARNING!**

##### **Electrical voltage! Risk of fatal injury due to electric shock!**

- Always switch off all power supply voltage before working on the devices!
- Note that the discharge time of the DC link capacitors is up to 6 minutes. You can only determine the absence of voltage after this time period.

#### **ATTENTION!**

##### **Damage to property due to electrostatic discharge!**

Take appropriate measures when handling exposed circuit boards, e.g. wearing ESD-safe clothing.

Do not touch contact surfaces.

#### **Tools and material**

You will need:

- A TORX screwdriver TX10
- The cover and screws included with the communication module

#### **Installation**

1. Unscrew the fastening screw of the dummy cover on top of the drive controller and remove the cover.
2. Slide the communication module on the guide rails into the drive controller.
3. Press on the module in order to push the pin contacts into the box header.
4. Set the tabs of the cover included with the communication module in front in the notch at an angle.
5. Place the cover on the drive controller so that the tabs lie under the edge.
6. Attach the cover using both screws.

## 6 Connection

For connecting individual drive controllers to each other or to the controller, the CA6 communication module provides a fieldbus interface in the form of a 9-pin D-sub connector for connection to the CAN bus system.

### 6.1 Selecting suitable cables

#### CAN bus

There are CAN buses with different sheathing materials for various applications and ambient conditions. The transmission of CAN signals requires at least a 3-pin cable with CAN-High (CAN-H), CAN-Low (CAN-L) and reference potential (GND).

In order to ensure error-free operation, especially at high transmission rates, we recommend using cables that meet the requirements listed in ISO 11898-2, such as the following:

- Characteristic impedance: 95 – 140  $\Omega$
- Maximum operating capacitance: 60 nF/km
- Conductor resistance: 70 m $\Omega$ /m

#### Plug connectors

9-pin D-sub connectors with a metal housing or a housing made of metallized plastic are suitable as plug connectors at the beginning and end of a CAN bus; connector housings with 2 cable entries are advantageous for looping through.

#### Shielding

Suitable shielding is essential, particularly for high transmission rates. The shielding of the CAN bus is mounted under the strain relief of the connector. As a result, the shielding is connected to the drive controller via the connector housing and the D-sub connector. Ensure continuous shielding over the entire cable length.

### 6.2 Terminal resistances

In the CANopen network, terminating resistors must be provided at the first and last nodes, even if the cable length is short. The following diagram abstracts a CANopen network in which the first terminating resistor is connected to the controller (master) and the second to the last participating drive controller SD6 (slave). To activate the terminating resistor on the CA6 communication module, the slider on connection X200 must be set to ON.

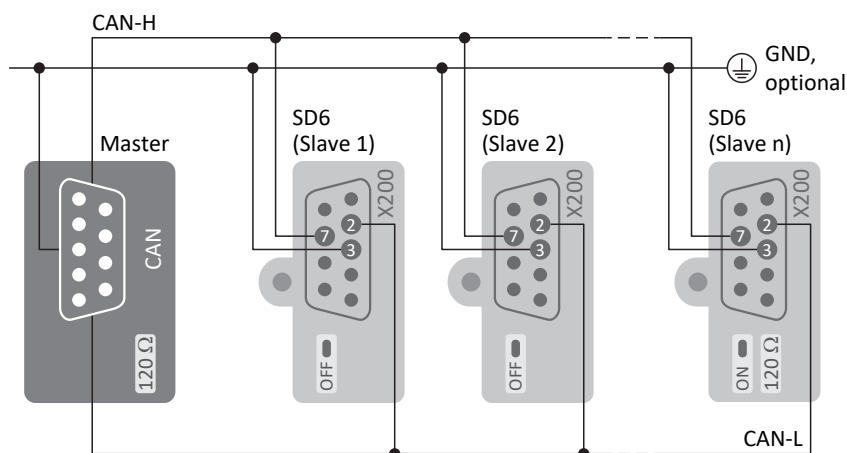


Fig. 2: CANopen network: Terminating resistors

## 6.3 Baud rate and cable length

The maximum permissible length of a CANopen network depends on the transmission rate, i.e. the baud rate. With a shorter cable length, the transmission rate as well as the sensitivity of serial fieldbus systems increase.

The following table shows the maximum permitted cable lengths over the entire length of the CANopen network depending on the respective possible baud rate.

Baud rate (kbps)	Maximum cable length (m)
10	5000
20	2500
50	1000
100	800
125	500
250	250
500	100
800	< 30, only with special cable $\pm 60$ nF/km
1000	< 10, only with special cable $\pm 60$ nF/km

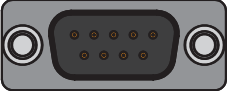
Tab. 1: CAN bus: Transmission rate and max. cable length

### Information

All drive controllers that are connected within a CANopen network via CAN bus must have the same baud rate.

## 6.4 X200: Fieldbus connection

In order to be able to connect the drive controllers to further CANopen nodes, the communication module CA6 provides a 9-pin D-sub connector.

Connector	Pin	Designation	Function
	1	—	—
	2	CAN-L	CAN low wire
	3	GND	Reference potential
	4	—	—
	5	—	—
	6	—	—
	7	CAN-H	CAN high wire
	8	—	—
	9	—	—

Tab. 2: X200 connection description

## 7 What you should know before commissioning

The following chapters provide a quick introduction to the structure of the program interface and accompanying window designations as well as relevant information about parameters and generally saving your project configuration.

### 7.1 DS6 program interface

Using the graphical interface of the DriveControlSuite commissioning software (DS6), you can project, parameterize and commission your drive project quickly and efficiently. In case of service, you can evaluate diagnostic information such as operating states, fault memories and fault counters of your drive project using DriveControlSuite.

#### Information

The program interface of DriveControlSuite is available in German, English and French. To change the language of the program interface, select **Settings > Language**.

#### Information

The DriveControlSuite help in the menu bar can be reached via **Help > Help for DS6** or via the [F1] key on your keyboard. When you press [F1] in an area of the program, the corresponding help topic opens.

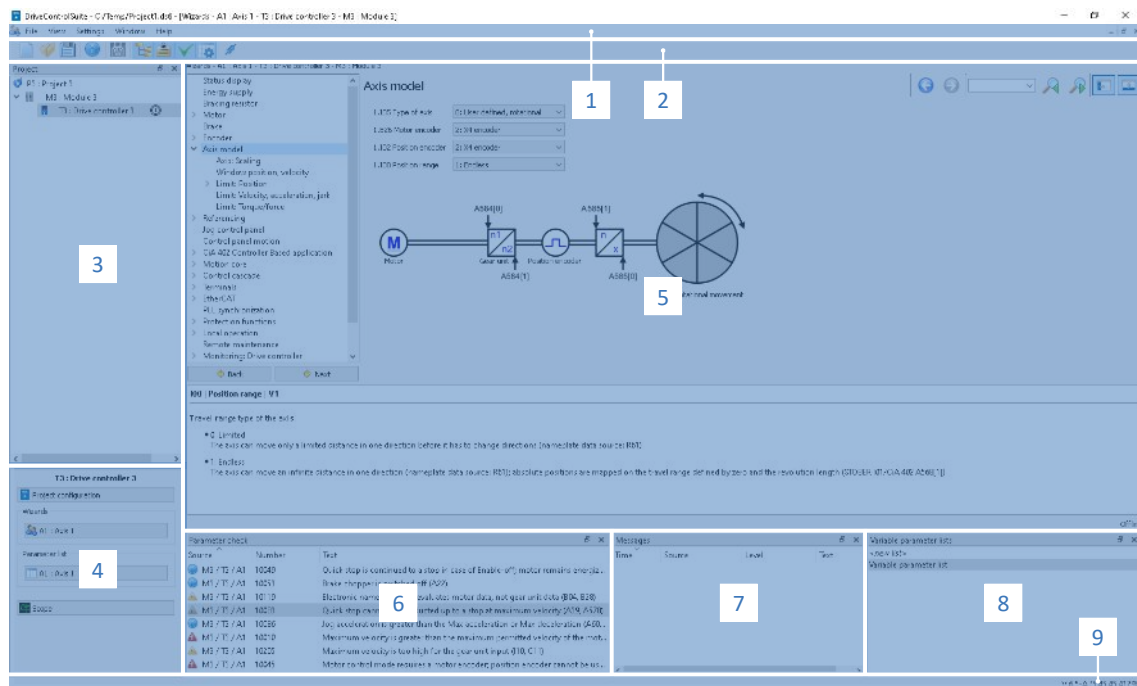


Fig. 3: DS6: Program interface

No.	Area	Description
1	Menu bar	Using the File, View, Settings and Window menus, you can open and save projects, display and hide program windows, select the interface language and access level and change between different windows in the workspace.
2	Toolbar	The toolbar enables quick access to frequently needed functions, like opening and saving projects and hiding and displaying windows in the program interface.
3	Project tree	The project tree forms the structure of your drive project in the form of modules and drive controllers. Select an element using the project tree first in order to edit it using the project menu.
4	Project menu	The project menu offers you various functions for editing the project, module and drive controller. The project menu adapts to the element that you selected in the project tree.
5	Workspace	The different windows which can be used to edit your drive project, such as the configuration dialog, wizards, the parameter list or the scope analysis tool, open in the workspace.
6	Parameter check	The parameter check points out irregularities and inconsistencies that were detected in the plausibility check of calculable parameters.
7	Messages	The entries in the messages log the connection and communication status of the drive controllers, incorrect inputs caught by the system, errors when opening a project or rule violations in the graphical programming.
8	Variable parameter lists	You can use variable parameter lists to compile any parameters in individual parameter lists for a quick overview.
9	Status bar	In the status bar, you can find the specifications of the software version and get additional information about the project file, the devices and the progress of the process during processes such as loading projects.







## 7.1.1 Configuring the view

In DriveControlSuite, you can change the visibility and arrangement of areas and windows, such as to optimize the available space in the workspace when working with smaller screens.

### Showing/hiding areas

Use the icons in the toolbar or the items in the **View** menu to show or hide specific areas in DriveControlSuite as needed.

Icon	Item	Description
—	Reset	Resets the view to factory settings.
	Project	Shows/hides the Project window (project tree, project menu).
	Messages	Shows/hides the Messages window.
	Parameter check	Shows/hides the Parameter check window.
	Variable parameter lists	Shows/hides the Variable parameter lists window.

### Arrange and group areas

You can undock and rearrange the individual areas via drag and drop. If you drag an undocked window to the edge of DriveControlSuite, you can release it there in a color-highlighted area either next to or on top of another window to redock it.

When you release the window onto another window, the two areas are merged into one window where you can use tabs to switch between the areas.

## 7.1.2 Navigation using sensitive circuit diagrams

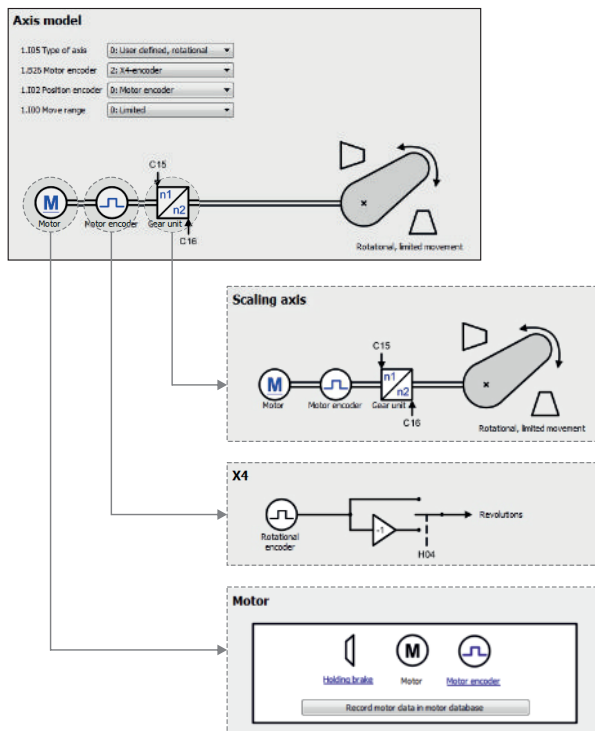


Fig. 4: DriveControlSuite: Navigation using text links and symbols

In order to illustrate graphically the processing sequence of actual and set values, the use of signals or certain drive component arrangements and to make configuring the accompanying parameters easier, they are displayed on the respective wizard pages of the workspace in the form of circuit diagrams.

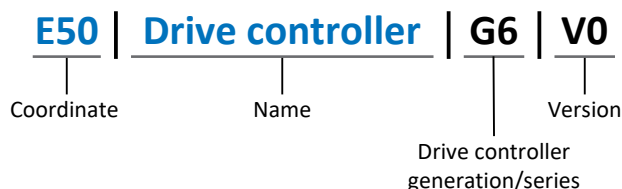
Blue text links or clickable icons indicate links within the program. These refer to the corresponding wizard pages and, as a result, allow you to reach additional helpful detail pages with just a click.

## 7.2 Meaning of parameters

You can use parameters to adapt the function of the drive controller to your individual application. In addition, parameters visualize the current actual values (actual velocity, actual torque, etc.) and trigger actions such as Save values, Test phase, etc.

### Interpretation of parameter identification

Parameter identification consists of the following elements, where short forms are also possible, i.e. only specifying a coordinate or the combination of coordinate and name.



### 7.2.1 Parameter groups

Parameters are assigned to individual groups by topic. The 6th generation of STÖBER drive controllers differentiates between the following parameter groups.

Group	Topic
A	Drive controllers, communication, cycle times
B	Motor
C	Machine, velocity, torque/force, comparators
D	Set value
E	Display
F	Terminals, analog and digital inputs and outputs, brake
G	Technology – Part 1 (application-dependent)
H	Encoder
I	Motion (all motion settings)
J	Motion blocks
K	Control panel
L	Technology – Part 2 (application-dependent)
M	Profiles (application-dependent)
N	Additional functions (application-dependent; e.g. extended cam control unit)
P	Customer-specific parameters (programming)
Q	Customer-specific parameters, instance-dependent (programming)
R	Production data for the drive controller, motor, brakes, motor adapter, gear unit and geared motor
S	Safety (safety technology)
T	Scope
U	Protection functions
Z	Fault counter

Tab. 3: Parameter groups

## 7.2.2 Parameter types and data types

In addition to topic-based sorting in individual groups, all parameters belong to a certain data type and parameter type. The data type of a parameter is displayed in the parameter list, properties table. The connections between parameter types, data types and their value range can be found in the following table.

Data type	Parameter type	Length	Value range (decimal)
INT8	Integer or selection	1 byte (signed)	-128 – 127
INT16	Integer	2 bytes (1 word, signed)	-32768 – 32767
INT32	Integer or position	4 bytes (1 double word, signed)	-2147483648 – 2147483647
BOOL	Binary number	1 bit (internal: LSB in 1 byte)	0, 1
BYTE	Binary number	1 byte (unsigned)	0 – 255
WORD	Binary number	2 bytes (1 word, unsigned)	0 – 65535
DWORD	Binary number or parameter address	4 bytes (1 double word, unsigned)	0 – 4294967295
REAL32 (single type according to IEE754)	Floating-point number	4 bytes (1 double word, signed)	$-3.40282 \times 10^{38} - 3.40282 \times 10^{38}$
STR8	Text	8 characters	—
STR16	Text	16 characters	—
STR80	Text	80 characters	—

Tab. 4: Parameters: data types, parameter types, possible values

### Parameter types: Use

- Integer, floating-point number  
For general computing processes  
Example: Set and actual values
- Selection  
Numeric value to which a direct meaning is assigned  
Example: Sources for signals or set values
- Binary number  
Bit-oriented parameter information that is collected in binary  
Example: Control and status words
- Position  
Integer combined with associated units and decimal places  
Example: Actual and set values of positions
- Velocity, acceleration, deceleration, jerk  
Floating-point number combined with associated units  
Example: Actual and set values for velocity, acceleration, deceleration, jerk
- Parameter address  
Referencing of a parameter  
Example: In F40 AO1 source, for example, E08 n-motor filtered can be parameterized
- Text  
Outputs or messages

## 7.2.3 Parameter types

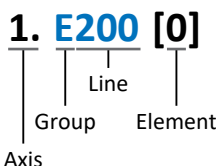
The following types of parameters are differentiated.

Parameter type	Description	Example
Simple parameters	Consist of one group and one line with a defined value.	A21 Brake resistor R: Value = 100 ohms
Array parameters	Consist of a group, a line and multiple sequential (listed) elements, which have the same properties but different values.	A10 Access level <ul style="list-style-type: none"> <li>A10[0] access level: Value = Access level via operating unit</li> <li>A10[2] access level: Value = Access level via CANopen and EtherCAT</li> <li>A10[4] access level: Value = Access level via PROFINET</li> </ul>
Record parameters	Consist of a group, a line and multiple sequential (listed) elements, which can have different properties and different values.	A00 Save values <ul style="list-style-type: none"> <li>A00[0] Start: Value = Start action</li> <li>A00[1] Progress: Value = Display action progress</li> <li>A00[2] Result: Value = Display action result</li> </ul>

Tab. 5: Parameter types

## 7.2.4 Parameter structure

Every parameter has specific coordinates with the following structure.



- Axis (optional)  
Axis to which an axis-specific parameter is assigned; not applicable for global parameters (value range: 1 - 4).
- Group  
The thematic group to which a parameter belongs (value range: A – Z).
- Line  
Distinguishes the parameters within a parameter group (value range: 0 – 999).
- Element (optional)  
Elements of an array or record parameter (value range: 0 – 16000).

## 7.2.5 Parameter visibility

The visibility of a parameter is primarily controlled by the access level you set in DriveControlSuite and by the properties you project for the respective drive controller (e.g. hardware, firmware and application). A parameter can also be shown or hidden depending on other parameters or settings. For example, the parameters of an additional function are only shown as soon as you activate the relevant additional function.

### Access level

The access options for the individual software parameters are ranked hierarchically and divided into individual levels. This means that parameters can be hidden for a specific purpose and, relatedly, their configuration options can be locked starting from a specific level.

Each parameter has one access level for read access (visibility) and one access level for write access (editability). The following levels are present:

- Level 0  
Elementary parameters
- Level 1  
Important parameters of an application
- Level 2  
Important parameters for service with extensive diagnostic options
- Level 3  
All parameters needed for commissioning and optimizing an application

The parameter A10 Access level controls general access to parameters:

- Over the SD6 drive controller display (A10[0])
- Over CANopen or EtherCAT (A10[2])
- Over PROFINET (A10[3])

<b>Information</b>
--------------------

It is not possible to write to or read the parameter hidden in DriveControlSuite during communication via fieldbus.

### Hardware

Which parameters are available to you in DriveControlSuite is determined by which series you select in the configuration dialog for the drive controller, for example, or whether you project an option module. Basically, the only parameters that are displayed are the ones you need to parameterize the configured hardware.

For example, a drive controller can evaluate an encoder using terminal X120, provided that terminal module XI6 has been installed. The accompanying evaluation is activated using parameter H120. However, this parameter is visible only if terminal module XI6 was initially selected during the drive project configuration.

### Firmware

Due to the further development and updating of functions for the 6th generation of STÖBER drive controllers, new parameters and also new versions of existing parameters are continuously being implemented in DriveControlSuite and in the firmware. The parameters are displayed in the software according to the DriveControlSuite version used and the configured firmware version of the respective drive controller.

### Applications

Applications generally differ in terms of functions and their control. For this reason, different parameters are available with each application.

## 7.3 Signal sources and process data mapping

The transmission of control signals and set values in DriveControlSuite meets the following principles.

### Signal sources

Drive controllers are controlled either over a fieldbus, using mixed operation consisting of a fieldbus system and terminals or exclusively using terminals.

You can use the corresponding selection parameters, referred to as signal sources, to configure whether the control signals and set values of the application are obtained over a fieldbus or using terminals.

In case of activation over a fieldbus, parameters that are selected as data sources for control signals or set values must be part of the subsequent process data mapping. In the case of activation using terminals, the respective analog or digital inputs are specified directly.

### Process data mapping

If you are working with a fieldbus system and have selected the source parameters for control signals and set values, configure the fieldbus-specific settings, e.g. the assignment of the process data channels for transmitting receive and transmit process data, as the last step.

## 7.4 Non-volatile memory

All project configurations, parameterizations and related changes to parameter values are in effect after transmission to the drive controller, but are only stored in volatile memory.

### Saving to a drive controller

To save the configuration in non-volatile memory on a drive controller, you have the following options:

- Saving the configuration using the *Save values wizard*:  
Project menu > Wizards area > Projected axis > *Save values wizard*: Select the *Save values* action
- Saving the configuration using the parameter list:  
Project menu > Parameter list area > Projected axis > Group A: Drive controller > A00 *Save values*: Set the parameter A00[0] to the value 1: Active
- Saving the configuration using the operating unit:  
Drive controller with operating unit: Press the save button for 3 seconds

### Saving to all drive controllers within a project

To save the configuration in non-volatile memory on several drive controllers, you have the following options:

- Saving the configuration using the toolbar:  
Toolbar > *Save values* icon: Click the *Save values* icon
- Saving the configuration using the *Online functions* window:  
Project menu > *Online connection* button > *Online functions* window: Click on *Save values (A00)*

#### Information

Do not shut off the drive controller while saving. If the supply voltage to the control unit is interrupted while saving, the drive controller will start without an executable configuration the next time it is switched on. In this case, the configuration must be transferred to the drive controller again and stored in non-volatile memory.

## 8 Commissioning

The following chapters describe the commissioning of a CANopen network consisting of a CANopen master (controller) and several STÖBER drive controllers from SD6, using the STÖBER DriveControlSuite commissioning software.

We assume the following system environment as an **example** so that you can follow the individual commissioning steps exactly:

- Drive controllers of the SD6 series in firmware V 6.5-K or later
- DriveControlSuite commissioning software version 6.5-K or later

in combination with

- Controller with CANopen interface
- Automation software of the controller

**Commissioning is divided into the following steps:**

1. DriveControlSuite  
Project all drive controllers (device control, application, process data, axis model, etc.), parameterize all general CANopen settings as well as the PDO transmission, then transmit your configuration to the drive controllers of your CANopen network.
2. CANopen network  
Afterwards, set up your CANopen network and put it into operation using the automation software of the controller.



## 8.1 DS6: Configuring the drive controller

Project and configure all drive controllers for your drive system in DriveControlSuite (see also [DS6 program interface](#) [► 15]).

### Information

Always perform the steps described below in the specified order!

Some parameters are interdependent and do not become accessible to you until you have first configured certain settings. Follow the steps in the specified sequence so that you can finish the parameterization completely.

### 8.1.1 Initiating the project

In order to be able to configure all drive controllers and axes of your drive system using DriveControlSuite, you must record them as part of a project.

#### 8.1.1.1 Projecting the drive controller and axis

Create a new project and project the first drive controller along with the accompanying axis.

#### Creating a new project

1. Start DriveControlSuite.
2. On the start screen, click **Create new project**.
  - ⇒ The new project is created and the configuration dialog for the first drive controller opens.
  - ⇒ The **Drive controller** button is active.

## Projecting the drive controller

1. **Properties tab:**  
 Establish the relationship between your circuit diagram and the drive controller to be projected in DriveControlSuite.  
 Reference: Specify the reference code (equipment code) of the drive controller.  
 Designation: Give the drive controller a unique name.  
 Version: Version your project configuration.  
 Description: If necessary, specify additional supporting information, such as the change history of the project configuration.
2. **Drive controller tab:**  
 Select the series and device type of the drive controller.
3. **Option modules tab:**  
 Communication module: Select the CA6 communication module.  
 Terminal module: If you are controlling the drive controller in mixed operation, i.e. via analog and digital inputs as well as the CA6, select the corresponding terminal module.  
 Safety module: If the drive controller is part of a safety circuit, select the corresponding safety module.
4. **Device control tab:**  
 Device control: Select CiA 402.  
 Rx process data: Select CANopen Rx+Tx for transmitting the CANopen process data.  
 Tx process data: Select No transmission.

### Information

Make sure that you project the correct series in the **Drive controller** tab. The projected series cannot be changed afterwards.

## Projecting the axis

1. Click on **Axis 1**.
2. **Properties tab:**  
 Establish the connection between your circuit diagram and the axis to be projected in DriveControlSuite.  
 Reference: Specify the reference code (equipment code) of the axis.  
 Designation: Give the axis a unique name.  
 Version: Version your project configuration.  
 Description: If necessary, specify additional supporting information, such as the change history of the project configuration.
3. **Application tab:**  
 Select CiA 402.
4. **Motor tab:**  
 Select the type of motor operated using this axis. If you are working with motors from third-party suppliers, enter the accompanying motor data at a later time.
5. Confirm with **OK**.

### 8.1.1.2 Configuring safety technology

If the drive controller is part of a safety circuit, you must configure the safety technology in accordance with the commissioning steps outlined in the corresponding manual in the next step (see [Detailed information \[► 69\]](#)).

### 8.1.1.3 Creating other drive controllers and modules

In DriveControlSuite, all drive controllers within a project are grouped using modules. If you add a new drive controller to your project, be sure to always assign it to an existing module. Group drive controllers in a module if, for example, they are located in the same control cabinet or work together to operate the same machine part.

#### Creating a drive controller

1. In the project tree, select your project P1 > module M1 > context menu **Create new drive controller**.  
⇒ The drive controller is created in the project tree and the configuration dialog opens.
2. Project the drive controller as described in [Projecting the drive controller and axis](#).
3. Repeat the steps for all other drive controllers that you want to project.

#### Creating a module

1. In the project tree, select your project P1 > context menu **Create new module**.  
⇒ The module is created in the project tree.
2. Project the module as described in [Projecting the module](#) [► 27].
3. Repeat the steps for all other modules that you want to project.

### 8.1.1.4 Projecting the module

Give your module a unique name, enter the reference code and, as an option, store additional information like the version and change history of the module.

1. Select the module in the project tree and click on **Project configuration** in the project menu.  
⇒ The configuration dialog for the module opens.
2. Establish the relationship between your circuit diagram and the module in DriveControlSuite.  
**Reference:** Specify the reference code (equipment code) of the module.  
**Designation:** Give the module a unique name.  
**Version:** Version the module.  
**Description:** If necessary, specify additional supporting information, such as the change history of the module.
3. Confirm with OK.

### 8.1.1.5 Projecting the project

Give your project a unique name, enter the reference code and, as an option, store additional information like the version and change history of the project.

1. Mark the project in the project tree and click on **Project configuration** in the project menu.  
⇒ The configuration dialog for the project opens.
2. Establish the relationship between your circuit diagram and the project in DriveControlSuite.  
**Reference:** Specify the reference code (equipment code) of the project.  
**Designation:** Give the project a unique name.  
**Version:** Version the project.  
**Description:** If necessary, specify additional supporting information, such as the change history of the project.
3. Confirm with OK.

## 8.1.2 Parameterizing general CANopen settings

Parameterize the fieldbus scaling and baud rate of your CANopen network and define the node ID for the respective drive controller. To monitor the communication between the controller and drive controller in the CANopen network, parameterize either the guarding or heartbeat function.

- ✓ You have configured the CA6 communication module with the CANopen Rx + Tx process data.
- 1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
- 2. Select the CANopen wizard.
- 3. A213 Fieldbus scaling:  
Leave the default value at 1: Native (values are passed unchanged).
- 4. A82 CAN baud rate:  
Select the baud rate depending on the cable length (see [Baud rate and cable length](#) [► 14]).
- 5. A83 Bus address:  
Define the bus address (node ID) of the drive controller (address range: 1 – 127).
- 6. A203 Guard time, A204 Life time factor:  
To monitor the communication between the controller and drive controller using the guarding function (node guarding/life guarding), parameterize the guard time and life time factor.
  - 6.1. A203 Guard time:  
Define the time span between 2 consecutive RTRs of the controller (value range: 1 – 4000 ms; 0 = inactive).
  - 6.2. A204 Life time factor:  
Define the number of missing RTRs of the controller after which the drive controller changes to the fault state (value range: 1 – 255; 0 = inactive).
- 7. A210 Producer heartbeat time:  
To monitor the communication between the controller and drive controller using the heartbeat function, define the time span between 2 consecutive heartbeat messages of the drive controller (value range: 1 – 65535 ms; 0 = inactive).

### Information

To monitor the communication between the controller and drive controller in the CANopen network, you can use either the guarding function or the heartbeat function. Since guarding and heartbeat have the same COB-ID and thus the same priority, only one of the two monitoring functions can be used at a time in the CANopen network, i.e. the other function must be deactivated.

For more information on the monitoring functions, see [ERROR CONTROL: Error Control Objects](#) [► 55].

## 8.1.3 Configuring PDO transmission

PDO communication enables simultaneous operation of up to 4 independent PDO channels per transmission direction (RxPDO, TxPDO), each of which can transmit 1 PDO with up to 6 communication parameters. The assignment of the parameters to the PDO channels is freely configurable as long as the resulting data length per channel does not exceed 8 bytes.

For runtime optimization, STÖBER recommends using the default settings: By default, 2 PDO channels are active per transmission direction, as this covers most use cases and reduces the runtime utilization.

### 8.1.3.1 Adapting the receive PDO

✓ You have configured the general CANopen settings.

1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
2. Select the **CANopen wizard** > Received process data RxPDO.
3. Check the presets and/or configure the process data according to your requirements.
  - 3.1. A221[0] COB-ID:  
COB-ID of the 1st RxPDO according to the predefined connection set. Only change the COB-ID if you want to deviate from the standard (see [Dynamic distribution](#) [► 43]).
  - 3.2. A221[1] Transmission type:  
Define the transmission type for the 1st RxPDO.
  - 3.3. A225[0] 1. mapped Parameter – A225[5] 6. mapped Parameter:  
Define the target parameters whose values the drive controller receives from the controller by means of the 1st RxPDO. The respective position (1st – 6th) provides information about the receiving sequence.
  - 3.4. Resulting data length:  
Overall length of the parameters of the first RxPDO to be transmitted. The value must not exceed 8 bytes. To maintain this value, you may need to change the type or number of parameters to be transmitted in this channel.
4. If you want to use other RxPDO channels, repeat the procedure for the respective RxPDO channel.

### 8.1.3.2 Adapting the transmit PDO

- ✓ You have configured the general CANopen settings.
- 1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
- 2. Select the CANopen wizard > Transmitted process data TxPDO.
- 3. Check the presets and/or configure the process data according to your requirements.
  - 3.1. A229[0] COB-ID:  
COB-ID of the 1st TxPDO according to the predefined connection set. Only change the COB-ID if you want to deviate from the standard (see [Dynamic distribution](#) [► 43]).
  - 3.2. A229[1] Transmission type:  
Define the transmission type for the 1st TxPDO.
  - 3.3. A229[2] Inhibit time:  
Define the minimum time interval for sending 2 consecutive TxPDOs.
  - 3.4. A229[3] Event timer  
If you have selected the asynchronous transmission type 254, define the time after which the 1st TxPDO is sent, even if no RxPDO has been received.
  - 3.5. A233[0] – A233[5] 1. mapped Parameter – 6. mapped Parameter:  
Define the source parameters whose values the drive controller sends to the controller by means of the 1st TxPDO. The respective position (1st – 6th) provides information about the transmission sequence.
  - 3.6. Resulting data length:  
Overall length of the elements of the first TxPDO to be transmitted. The value must not exceed 8 bytes. To maintain this value, you may need to change the type or number of elements to be transmitted in this channel.
- 4. If you want to use other TxPDO channels, repeat the procedure for the respective TxPDO channel.

## 8.1.4 Transmitting and saving a configuration

In order to transmit and save the configuration to one or more drive controllers, you must connect your PC and the drive controllers over the network.



### WARNING!

#### Injury to persons and material damage due to axis movement!

If there is an online connection between DriveControlSuite and the drive controller, changes to the configuration can lead to unexpected axis movements.

- Only change the configuration if you have visual contact with the axis.
- Make sure that no people or objects are within the travel range.
- For access via remote maintenance, there must be a communication link between you and a person on site with eye contact to the axis.

### Information

During the search, all drive controllers within the broadcast domain are found via IPv4 limited broadcast.

Requirements for finding a drive controller in the network:

- Network supports IPv4 limited broadcast
- All drive controllers and the PC are in the same subnet (broadcast domain)

### 8.1.4.1 Transmitting the configuration

The steps for transmitting the configuration vary depending on the safety technology.

#### Drive controller without SE6 safety module

- ✓ You have verified the plausibility of the predefined test motion variables.
  - ✓ The drive controllers are switched on.
1. In the project tree, select the module under which you have recorded your drive controller and click **Online connection** in the project menu.
    - ⇒ The **Add connection** dialog box opens. All drive controllers found via IPv4 limited broadcast are displayed.
  2. **Direct connection** tab > **IP address** column:  
 Activate the IP addresses in question and confirm your selection with **OK**.
    - ⇒ The **Online functions** window opens. All drive controllers connected through the previously selected IP addresses are displayed.
  3. Select the drive controller to which you want to transmit a configuration and change the selection of the transmission type from **Read** to **Send**.
  4. Change the selection **Create new drive controller**:  
 Select the configuration that you would like to transfer to the drive controller.
  5. Repeat steps 3 and 4 for all other drive controllers to which you would like to transfer your configuration.
  6. **Online** tab:  
 Click **Establish online connection**.
    - ⇒ The configurations are transferred to the drive controllers.

### Drive controller with SE6 safety module

- ✓ You have verified the plausibility of the predefined test motion variables.
- ✓ The drive controllers are switched on.
- 1. In the project tree, select the module under which you have recorded your drive controller and click **Online connection** in the project menu.
  - ⇒ The **Add connection** dialog box opens. All drive controllers found via IPv4 limited broadcast are displayed.
- 2. **Direct connection tab > IP address column:**  
 Activate the IP addresses in question and confirm your selection with **OK**.
  - ⇒ The **Online functions** window opens. All drive controllers connected through the previously selected IP addresses are displayed.
- 3. Select the drive controller to which you want to transmit a configuration and change the selection of the transmission type from **Read** to **Send**.
- 4. Change the selection **Create new drive controller:**  
 Select the configuration that you would like to transfer to the drive controller.
- 5. Repeat steps 3 and 4 for all other drive controllers to which you would like to transfer your configuration.
- 6. **Online tab:**  
 Click **Establish online connection**.
  - ⇒ The configurations are transferred to the drive controllers.
  - ⇒ A dialog box prompts you to open the **PASmotion** configuration tool.
- 1. Confirm the dialog box with **Yes**.
  - ⇒ **PASmotion** opens.
- 2. In the **PASmotion** project administration, navigate to the safety module for the drive controller and double-click to open it.
  - ⇒ The dialog box for the password prompt opens.
- 3. Enter the password and confirm with **OK**.
  - ⇒ The wizard for device synchronization opens.
  - ⇒ Device configuration and configuration are checked against each other automatically.
- 4. If the configurations match, click on **Done** after device synchronization has finished.
- 5. Optional: If the configurations do not match, click on **Next** after device synchronization has finished.
  - 5.1. Confirm the production number of the safety module and click **Next**.
  - 5.2. Enter the password for the configuration on the safety module and click **Next**.
  - 5.3. Click **Upload** to transfer the device configuration to the project.
  - 5.4. After the successful transfer, click **Done**.
- 6. Exit **PASmotion**.
  - ⇒ The safety configuration is transferred to the selected drive controller.



### 8.1.4.2 Saving the configuration

- ✓ You have successfully transmitted the configuration.
- 1. Online functions window:  
Click Save values (A00).
  - ⇒ The Save values (A00) window opens.
- 2. Click Start action.
  - ⇒ The configuration is stored on the drive controllers in non-volatile memory.
- 3. Close the Save values (A00) window.

#### Information

For the configuration to take effect on the drive controller, a restart is required when the configuration is saved on the drive controller for the first time or when changes are made to the firmware or process data mapping.

#### Restarting a drive controller

- ✓ You have stored the configuration on the drive controller in non-volatile memory.
- 1. Online functions window:  
Click Restart (A09).
  - ⇒ The Restart (A09) window opens.
- 2. Select which of the connected drive controllers you want to restart.
- 3. Click Start action.
- 4. Confirm the safety note with OK.
  - ⇒ The Restart (A09) window closes.
- ⇒ The fieldbus communication and connection between DriveControlSuite and drive controllers are interrupted.
- ⇒ The selected drive controllers restart.

## 8.2 Putting the CANopen network into operation

Afterwards, set up your CANopen network and put it into operation using the automation software of the controller.

## 9 Monitoring and diagnostics

For monitoring purposes and in the event of a fault, the various monitoring and diagnostic options described below are available.

### 9.1 Connection monitoring

To be able to detect a communication failure, activate one of the two monitoring functions for communication in the CANopen network: guarding (node guarding/life guarding) or heartbeat (see [Parameterizing general CANopen settings](#) [► 28]).

When monitoring communication in the CANopen network using guarding, the controller and drive controller monitor each other. When monitoring using heartbeat, the controller monitors the drive controller. For more information on the respective monitoring function, please visit [Guarding](#) [► 55] or [Heartbeat](#) [► 56].

If you use the guarding function and the drive controller does not receive any RTRs from the controller within a certain time, the drive controller evaluates this as a communication error (life guarding event) and triggers the fault 52: Communication with the cause 1: CAN Life Guarding Event in the NMT state Operational. The time interval in which the drive controller expects an RTR from the controller is called the "life time" and is the product of the guard time and life time factor (life time =  $A203 \times A204$ ).

## 9.2 LED display

The drive controllers feature diagnostic LEDs that visualize the state of fieldbus communication and the states of the physical connection.

### 9.2.1 CANopen state

There are 2 LEDs on the top of the drive controller that provide information about the connection between the CANopen master and slave and about the state of the data exchange. This information can also be read out in parameter A245.

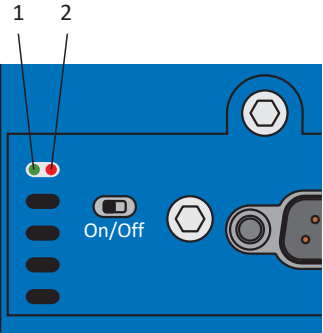











Fig. 5: LEDs for the CAN status

- 1 Green: Run
- 2 Red: Error

Green LED	Conduct	NMT state	Description
	Off	Initialization	No communication between the CANopen master and slave; CAN bus is initialized, the configuration starts and saved values are loaded
	Flashing	Pre-operational	No PDO communication between the CANopen master and slave; CAN bus is active, the drive controller can be parameterized for operation via SDO
	1x flash	Stopped	Communication services are stopped (exception: guarding, heartbeat)
	On	Operational	Normal operation: CAN bus is active, all communication services are in operation

Tab. 6: Meaning of the green LED (Run)

Red LED	Conduct	Error	Description
	Off	No error, no warning	No error
	1x flash	Communication error in the operational state (Warning level)	Check the connection and shielding and correct if necessary
	2x flash	Life guarding event	Check guarding configuration of the controller
	3x flash	SYNC error	Check SYNC configuration of the controller
	On	Bus-off	No network connection

Tab. 7: Meaning of the red LED (error)

## 9.3 Events

The drive controller has a self-monitoring system that uses test rules to protect the drive system from damage. Violating the test rules triggers a corresponding event. There is no possible way for you as the user to intervene in some events, such as the Short/ground event. In others, you can influence the effects and responses.

Possible effects include:

- **Message:** Information that can be evaluated by the controller
- **Warning:** Information that can be evaluated by the controller and becomes a fault after a defined time span has elapsed without the cause being resolved
- **Fault:** Immediate drive controller response; the power unit is disabled and axis movement is no longer controlled by the drive controller or the axis is brought to a standstill by a quick stop or emergency braking

### ATTENTION!

#### Damage to property due to interruption of a quick stop or emergency braking!

If, when executing a quick stop or emergency braking, a fault occurs or STO is active, the quick stop or emergency braking is interrupted. In this case, the machine can be damaged by the uncontrolled axis movement.

Events, their causes and suitable measures are listed below. If the cause of the error is corrected, you can usually acknowledge the error immediately. If the drive controller has to be restarted instead, a corresponding note can be found in the measures.

### Information

To make it easier for control programmers to set up the human-machine interface (HMI), a list of events and their causes can be found in the STÖBER download center at <http://www.stoeber.de/en/downloads/>.

### 9.3.1 Event 52: Communication

The drive controller has a **fault** if:

- A29 = 0: Inactive for Drive Based device controller  
or
- A540 = 0: Disable drive motor coasting for CiA 402 device controller

Response:

- The power unit is disabled and axis movement is no longer controlled by the drive controller
- The brakes are applied in the event of an inactive release override (F06)

The drive controller has a **fault with a quick stop** if:

- A29 = 1: Active for Drive Based device controller  
or
- A540 = 2: Slow down on quick stop ramp for CiA 402 device controller

Response:

- The axis is stopped by a quick stop
- During the quick stop, the brakes remain released
- At the end of the quick stop, the power unit is disabled and axis movement is no longer controlled by the drive controller
- The brakes are applied in the event of an inactive release override (F06)

Cause		Check and action
1: CAN Life Guarding Event	Missing Remote Transmission Request (RTR)	Check the guarding settings for the CANopen master and drive controller and correct them if necessary (A203, A204)
3: CAN Bus Off	Timing error	Check the connection and correct it if necessary; check the cable and replace it if necessary; check the baud rate in the CANopen master and drive controller and correct it if necessary (A82)

Tab. 8: Event 52 – Causes and actions

## 9.4 Parameters

The following diagnostic parameters are available for communication via CANopen.

### 9.4.1 A83 | Bus address | G6 | V0

Bus address of the drive controller in the CANopen network (node ID).

Corresponds to the node ID communication object in accordance with CiA 301; object 100B hex.

### 9.4.2 A245 | CAN diagnostic | G6 | V0

Diagnostic information of the drive controller in the CANopen network.

- Bit [0] – [2]: NMT state of the drive controller in the CANopen network (network management, NMT)  
000 bin = 0: Inactive; 001 bin = 1: Initialisation - Reset App.; 010 bin = 2: Initialisation - Reset Communication; 011 bin = 3: Initialisation - Boot-Up; 100 bin = 4: Pre-operational; 101 bin = 5: Stopped; 110 bin = 6: Operational
- Bit [3]: Warning level  
Communication error in the operational state
- Bit [4]: Bus-off  
Connection error or incorrect baud rate
- Bit [5]: SDO channel 1 toggle bit  
State switches with each frame received on SDO channel 1
- Bit [6]: SDO channel 1 memory utilization  
0 = utilization < 50%; 1 = utilization ≥ 50%
- Bit [7]: PDO channel 1 toggle bit  
State switches with each operational frame received on PDO channel 1
- Bit [8]: PDO channel 1 memory utilization  
0 = utilization < 50%; 1 = utilization ≥ 50%
- Bit [9]: Red LED (error)  
0 = no error, no warning; 1x flash = Warning level; 2x flash = Life guarding event; 3x flash = SYNC error; 1 = Bus-off
- Bit [10]: Green LED (Run)  
0 = Init; flashing = Pre-operational; 1x flash = Stopped; 1 = Operational
- Bit [11]: PDO SYNC behavior error
- Bit [12] – [15]: Reserved

If not otherwise specified: 0 = inactive; 1 = active.

### 9.4.3 A246 | CANopen NMT-State | G6 | V1

NMT state of the drive controller in the CANopen network (network management, NMT).

- 0: Inactive
- 1: Initialisation - Reset App.
- 2: Initialisation - Reset Communication
- 3: Initialisation - Boot-Up
- 4: Pre-operational
- 5: Stopped
- 6: Operational

## 10 More about CANopen?

The following sections summarize the key terms, services and relationships relating to CANopen.

### 10.1 CAN and CANopen

#### CAN

CAN (Controller Area Network) is a serial bus system and industrial standard for real-time requirements in automation technology. Due to their real-time capability, high fault resistance as and good availability, CAN bus systems serve a wide range of application areas and are mainly used where high transmission rates and a simple, cost-effective installation are required.

CAN was originally invented by Robert Bosch GmbH and is still supported by the organization CAN in AUTOMATION e.V. (CiA). CAN is an open technology, standardized in the standard ISO 11898-1 since 1993.

The tasks of a CAN network are defined in what are called the layers. The actual CAN protocol corresponds to layer 2 of the ISO/OSI reference model, the data link layer. In this layer, the CAN protocol manages accesses of nodes that are connected to each other via the CAN bus, thus enabling the interaction of technology from different manufacturers. In the CAN network, messages are transmitted with a certain priority according to their CAN identifier (CAN ID).

#### CANopen

CANopen is a CAN-based communication protocol for networking in automation technology. It defines the basic communication mechanisms and the functionality of nodes in the CANopen network.

CANopen expands upon CAN with the application layer, which corresponds to layer 7 of the ISO/OSI reference model and is also referred to as CAL (CAN Application Layer) in the context of CAN.

CANopen, also maintained by CiA, was first standardized in the standard EN 50325-4 in 1996 and later transferred to the standards IEC 61800-7-201 and IEC 61800-7-301.

#### 10.1.1 CANopen – Communication

CANopen groups nodes in the CANopen network into device classes based on their properties and provides a separate device profile for each device class, e.g. the CiA 402 device profile for controlling electric drives. The device profile defines the functionality and structure of the object dictionary and thus enables standardized access to the communication objects (COB) or parameters of the device class, for example.

The device profile is based on a communication profile, e.g. the CiA 301 communication profile, which defines the contents of the communication objects and thus specifies the basic communication mechanisms between the nodes in the CANopen network.

## 10.1.2 Object directory

The object directory describes the complete range of functions of a node in the CANopen network according to its device profile. For example, the object directory lists standardized data types, communication objects and device profile objects. In addition to the standardized objects, there is also an area for manufacturer-specific objects or parameters.

Service data objects (SDO) can be used to access the entries in the object directory in order to configure the device properties of the node. The entries in the object directory can be addressed via the index (row address, 16 bit) and, for array or record parameters, via an additional subindex (column address, 8 bit).

Index (hex)	Object
0000	Not used
0001 – 001F	Static data types
0020 – 003F	Complex data types
0040 – 005F	Manufacturer-specific data types
0060 – 007F	Profile-specific static data types
0080 – 009F	Profile-specific complex data types
00A0 – 0FFF	Reserved
1000 – 1FFF	Communication profile (CiA 301 and 302)
2000 – 5FFF	Manufacturer-specific parameters
6000 – 9FFF	Parameters from standardized profiles (CiA 4xx)
A000 – AFFF	Network variables
B000 – FFFF	Reserved

Tab. 9: Object directory – Structure

## 10.1.3 Network structure

The topology of a CANopen network is usually linear. Theoretically, up to 127 nodes can be addressed; physically, 64 nodes are possible in the CANopen network. At a baud rate of 50 kBaud, for example, a cable length of 1 km is possible; at a baud rate of 1 MBaud, a cable length of 25 m is realistic (see also [Baud rate and cable length](#) [► 14]). In general, 9 baud rates (10 kBaud – 1 MBaud) are available (parameter: A82).



# 10.2 CAN message

Communication via the CAN protocol takes place in the form of messages, the standardized structure of which is called a frame. The CAN bus distinguishes between 4 types of frames:

- Data frame; for transporting data to another node
- Remote frame; for requesting data from another node
- Error frame; for a known transmission error
- Overload frame; as a forced break between the sending of data and remote frames

## Structure of a CAN message

The following graphic shows the structure of a CAN message.

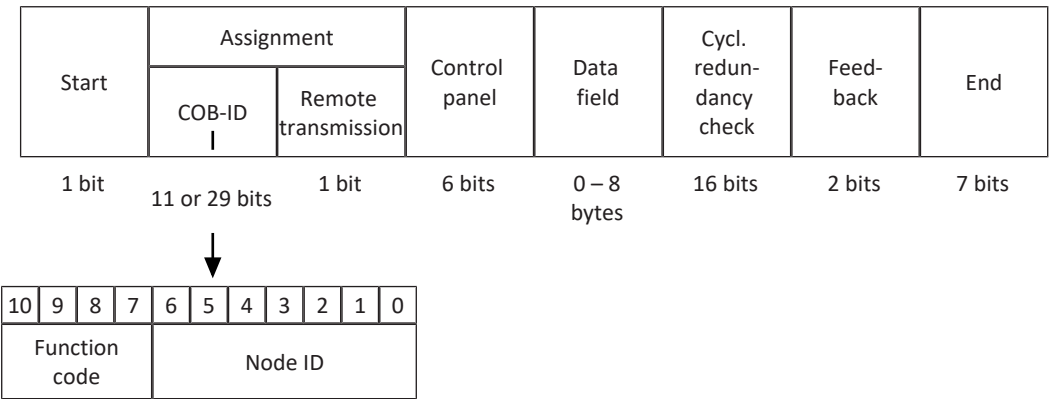


Fig. 6: CAN message: Structure

## COB-ID

The COB-ID (Communication Object Identifier) determines the priority of a message in the CANopen network (low COB-ID = high priority). The COB-ID is composed of the function code (function of the communication object) and the node ID (bus address of the node) and can be changed via SDO accesses. By default, a COB-ID consists of 11 bits (base frame format), of which bits 7 – 10 are occupied by the function code and bits 0 – 6 by the node ID, allowing 2032 different logical addresses to be encoded and 2048 different pieces of information to be sent.

There are 2 mechanisms for assigning the COB-ID: With the predefined connection set, all communication objects are preassigned with standardized function codes; with dynamic distribution, the preassigned function codes can be overwritten for certain applications.

### 10.2.1 Predefined connection set

When COB-IDs are assigned via the predefined connection set, all communication objects are preassigned with standardized function codes. The node ID of the respective drive controller is subsequently added, resulting in a unique COB-ID. This pre-assignment allows for easy commissioning of a CANopen network with one controller and up to 127 drive controllers (or, with active SDO channels 2 – 4, with up to 31 drive controllers), thus covering most applications.

The predefined connection set is active by default for all communication objects (broadcast objects and peer-to-peer objects), unless a manual COB-ID has been assigned for the respective communication object.

The following tables show the COB-IDs of the communication objects corresponding to the predefined connection set.

**Broadcast objects**

Object	Function code	COB-ID		Communication parameter (index)	Priority
NMT	0000	0 hex	0	—	Highest
SYNC	0001	80 hex	128	1005	
TIME	0010	100 hex	256	—	

**Peer-to-peer objects**

When using the predefined connection set, the node ID of the relevant node must be between 1 and 127 or, with active SDO channels 2 – 4, between 1 and 31 (node ID: A83).

Object	Function code	COB-ID			Communication parameter		Priority
					Index	DS6	
EMCY	0001	80 hex + node ID	81 hex – FF hex	129 – 255	1014 hex, 1015 hex	A207	High
TxPDO1	0011	180 hex + node ID	181 hex – 1FF hex	385 – 511	1800 hex	A229[0]	
RxPDO1	0100	200 hex + node ID	201 hex – 27F hex	513 – 639	1400 hex	A221[0]	
TxPDO2	0101	280 hex + node ID	281 hex – 2FF hex	641 – 767	1801 hex	A230[0]	
RxPDO2	0110	300 hex + node ID	301 hex – 37F hex	769 – 895	1401 hex	A222[0]	
TxPDO3	0111	380 hex + node ID	381 hex – 3FF hex	897 – 1023	1802 hex	A231[0]	
RxPDO3	1000	400 hex + node ID	401 hex – 47F hex	1025 – 1151	1402 hex	A223[0]	
TxPDO4	1001	480 hex + node ID	481 hex – 4FF hex	1153 – 1279	1803 hex	A232[0]	
RxPDO4	1010	500 hex + node ID	501 hex – 5FF hex	1281 – 1407	1403 hex	A224[0]	
TxSDO1	1011	580 hex + node ID	581 hex – 59F hex	1409 – 1439	1200 hex	—	
RxSDO1	1100	600 hex + node ID	601 hex – 61F hex	1537 – 1567	1200 hex	—	
TxSDO2	1011	5A0 hex + node ID	5A1 hex – 5BF hex	1441 – 1471	1201 hex	A218[1]	
RxSDO2	1100	620 hex + node ID	621 hex – 63F hex	1539 – 1599	1201 hex	A218[0]	
TxSDO3	1011	5C0 hex + node ID	5C1 hex – 5DF hex	1473 – 1503	1202 hex	A219[1]	
RxSDO3	1100	640 hex + node ID	641 hex – 65F hex	1601 – 1631	1202 hex	A219[0]	
TxSDO4	1011	5E0 hex + node ID	5E1 hex – 5FF hex	1505 – 1535	1203 hex	A220[1]	
RxSDO4	1100	660 hex + node ID	661 hex – 67F hex	1633 – 1663	1203 hex	A220[0]	Low
ERROR CONTROL	1110	700 hex + node ID	701 hex – 77F hex	1793 – 1919	1016 hex, 1017 hex	—	

## 10.2.2 Dynamic distribution

When assigning COB-IDs via dynamic distribution, you can assign the COB-IDs for specific communication objects manually, either directly via DriveControlSuite or via a controller that acts as a distributor (DBT) to dynamically assign the COB-IDs. For this purpose, a process data image of all drive controllers in the CANopen network must be available to the controller at all times.

By manually assigning COB-IDs via dynamic distribution, you can influence the priority of communication objects and thus cover special use cases, such as to optimize complex CANopen networks with different nodes and tasks. Because COB-IDs must be unique, dynamic distribution requires more planning than using the predefined connection set.

### Enabling dynamic distribution

As a rule, the COB-ID of the following communication objects can be overwritten via dynamic distribution:

- PDO 1 – 4 (RxPDO, TxPDO) (COB-ID: A221[0] – A224[0], A229[0] – A232[0])
- SDO 2 – 4 (RxSDO, TxSDO) (COB-ID: A218[0] – A220[0], A218[1] – A220[1])
- SYNC (COB-ID: A200)
- EMCY (COB-ID: A207)

To overwrite a COB-ID using DriveControlSuite, define a unique COB-ID for the desired communication object of the respective drive controller via the corresponding parameter. Then transmit the configuration to the respective drive controller and save it in non-volatile memory on the device (parameter: A00).

To overwrite a COB-ID using the controller, the NMT state machine of the respective drive controller must be in the NMT state Pre-operational. The changed COB-IDs are initialized for PDO by a change to the NMT state Initializing and for SDO by a change to the NMT state Reset Communication. To ensure that the changes remain effective when the drive controllers are restarted, they must be stored in non-volatile memory (parameter: A00).

## 10.3 Communication objects

In general, CANopen provides for communication objects to read or write the values of communication parameters. A write service is designated as "Domain Download" and a read service as "Domain Upload".

In the CANopen network, the following communication objects are essential for data transmission:

- Synchronization Objects (SYNC)  
... for the time-based sorting and synchronization of the nodes
- Process Data Objects (PDO)  
... for the transmission of real-time data of the nodes (actual and set values)
- Service Data Objects (SDOs)  
... for access to the object directory of the nodes for device configuration
- Network Management Objects (NMT)  
... for the control and monitoring of the NMT states of the nodes
- Error Control Objects (monitoring objects, ERROR CONTROL)  
... for monitoring the communication of the nodes (guarding, heartbeat)
- Emergency Objects (EMCY)  
... for monitoring the device states of the nodes
- Time Stamp Objects (TIME)  
... for transmitting the current time (local time, not any defined time zone)

### Information

The CA6 communication module does not support Time Stamp Objects (TIME).

For communication objects, a distinction is made between broadcast objects and peer-to-peer objects. Broadcast objects are sent by the controller to all nodes simultaneously (NMT, SYNC, TIME). Peer-to-peer objects can be exchanged in both send directions (SDO, PDO, EMCY, ERROR CONTROL).

### Information

It is not possible to write to or read the parameter hidden in DriveControlSuite during communication via fieldbus.

### 10.3.1 SYNC: Synchronization Objects

SYNC objects are broadcast objects used for the time-based sorting and synchronization of the nodes in the CANopen network. SYNC objects enable synchronous transmission types for PDO messages by allowing PDOs to be received or sent with reference to SYNC objects; for example, several inputs can be read in parallel or axes can be synchronized. SYNC messages generally do not contain any data.

#### Information

If a CAN bus and IGB motion bus are used simultaneously, CANopen communication cannot be synchronized with the controller.

### 10.3.2 PDO: Process Data Objects

Process Data Objects are peer-to-peer objects that are used to transmit time-critical real-time data of the nodes, such as set and actual values or control and status information such as set positions, travel velocities, or acceleration specifications.

PDOs allow simultaneous access to several communication parameters defined via the object directory of the respective node. Objects are not addressed with PDO transmission. Instead, the values of the communication parameters are transmitted directly to the respective node.

The process data mapping (PDO mapping) defines which communication parameters are sent and received. With process data mapping, which communication parameters are sent or received in which PDO can be freely selected.

PDOs are generally transmitted via process data channels (PDO channels) with high priority. From the perspective of the respective node, receive PDOs (RxPDO) are differentiated from transmit PDOs (TxPDO).

The transmission type defines whether the PDOs are sent event-controlled or time-controlled and cyclically, acyclically or only upon request by another node (Remote Transmission Request, RTR). PDOs are only transmitted if the node is in the NMT state Operational. The assignment and priority of PDO messages are defined by the COB-ID.

For information on scaling, see [Fieldbus scaling](#) [► 60].

#### 10.3.2.1 PDO mapping

The process data mapping (PDO mapping) defines which communication parameters are sent and received. The communication parameters from the object directory of a node are mapped to the respective PDO channels for this purpose.

PDO communication enables simultaneous operation of up to 4 independent PDO channels per transmission direction (RxPDO, TxPDO), each of which can transmit 1 PDO with up to 6 communication parameters. The assignment of the parameters to the PDO channels is freely configurable as long as the resulting data length per channel does not exceed 8 bytes.

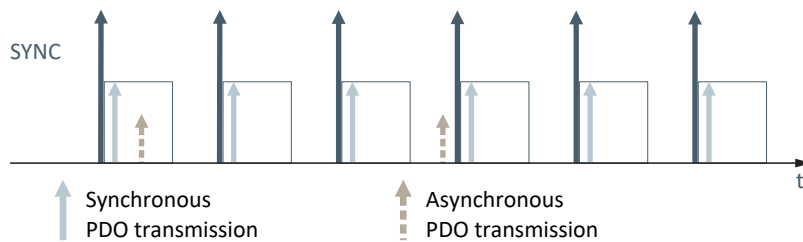
For runtime optimization, STÖBER recommends using the default settings: By default, 2 PDO channels are active per transmission direction, as this covers most use cases and reduces the runtime utilization.

### 10.3.2.2 Transmission type

The transmission type defines whether PDOs are transmitted time-controlled (synchronously) with respect to SYNC objects or event-controlled (asynchronously). Synchronous PDO transmission can be cyclical or acyclical, resulting in the following transmission types:

- Synchronous (cyclical, acyclical)
- Asynchronous

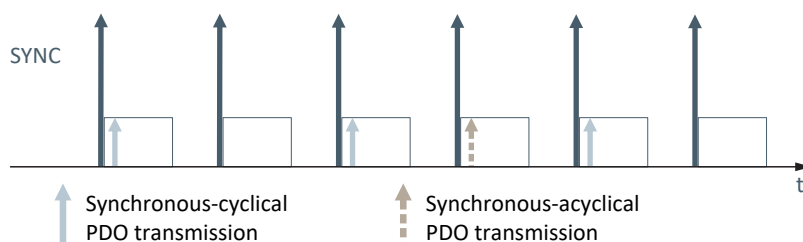
#### Synchronous and asynchronous PDO transmission



With synchronous PDO transmission, the PDO messages are processed based on time, i.e. depending on SYNC objects. For example, an RxPDO can be received with every SYNC object or with every fifth SYNC object.

With asynchronous PDO transmission, the PDO messages are processed based on events, i.e. independent of SYNC objects. For example, an RxPDO can be received immediately and a TxPDO sent immediately, or transmission can be triggered by an event timer or by an RTR.

#### Synchronous-cyclical and synchronous-acyclical PDO transmission



With synchronous PDO transmission, the PDO messages can be transmitted either cyclically or acyclically. Cyclical PDO messages are transmitted in sync with a defined number of SYNC objects. Acyclical PDO messages are transmitted in sync with a SYNC object and based on the occurrence of an internal device event, e.g. the reception of RxPDO.

#### Remote Transmission Requests (RTR)

With event-driven (asynchronous) PDO transmission, the sending of a PDO can be triggered by the request by another node in the CANopen network. For example, when monitoring the nodes in the CANopen network via guarding (node guarding/life guarding), the controller sends Remote Transmission Requests (RTRs) to the nodes in the CANopen network at regular intervals to request the NMT state of the respective drive controller (see [Guarding \[55\]](#)).

#### TxPDO: Inhibit time and event timer

Each TxPDO channel has a inhibit time, which defines the minimum time interval for the transmission of 2 successive TxPDOs, independent of the transmission type.

For asynchronous PDO transmission, a timer (event timer) can be used to define the time per TxPDO channel after which TxPDOs are sent if an event has not otherwise occurred for event-controlled PDO transmission. For example, with transmission type 254, the transmission of TxPDOs is triggered by the receipt of RxPDOs, but at the latest after the event timer has expired if a new RxPDO has not been received beforehand.

### Defining the transmission type

The transmission type can be defined for each channel and transmission direction via the respective parameter (RxPDO: A221[1] – A224[1]; TxPDO: A229[1] – A232[1]).

Value	Cyclical	Acyclical	Synchronou s	Asynchrono us	Description
0	–	✓	✓	–	Receive PDOs are applied with the next SYNC object; transmit PDOs are sent with the next SYNC object
1 – 240	✓	–	✓	–	Value = the number of SYNC objects before a receive PDO is received and applied and before a transmit PDO is sent
241 – 253					Reserved
254	–	–	–	✓	Receive PDOs are applied when received; transmit PDOs are sent immediately afterwards
255	–	–	–	✓	Reserved

Tab. 10: Transmission type

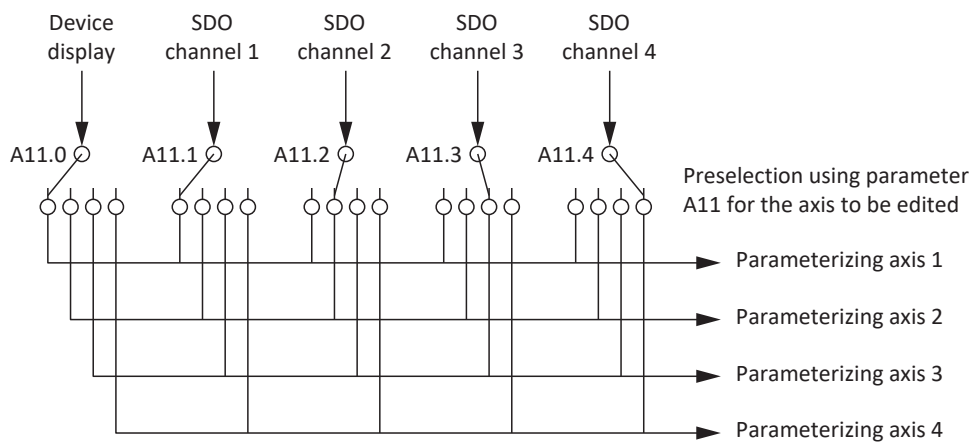
### 10.3.3 SDO: Service Data Objects

Service Data Objects are peer-to-peer objects that are used to transfer non-time-critical data and provide access to entries in a node's object directory in order to configure its device properties.

From the perspective of the drive controller, an SDO transmission always consists of at least one RxSDO message and one TxSDO message. In the RxSDO message, the controller selects an entry from the object directory of the drive controller via index and subindex in order to configure the device properties. The drive controller then acknowledges access to the object directory with a TxSDO message.

There are 4 independent channels available for SDO transmission. SDO channel 1 is always active; the associated COB-IDs correspond to the predefined connection set and cannot be changed (RxSDO1 = 600 hex + node ID; TxSDO1 = 580 hex + node ID). SDO channels 2 – 4 are disabled by default; the associated COB-IDs can be changed if required.

The SD6 drive controller enables the operation of up to 4 logical axes; exactly one SDO channel can be used per axis. The axes are not addressed by the index and subindex, but by parameter A11 for each SDO channel.



Depending on the transmission type, SDOs can generally be used for transmitting data of any length:

- Expedited transfer  
... for transmission of up to 4 bytes in a single message
- Segmented transfer  
... for transmission of more than 4 bytes distributed over several messages

For information on scaling, see [Fieldbus scaling](#) [► 60].

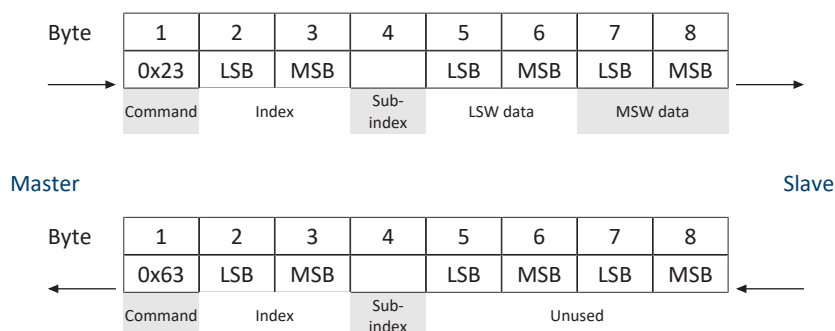


### 10.3.3.1 Expedited transfer

SDO transmission via expedited transfer enables up to 4 bytes of data to be transmitted in a single message. The data is arranged in accordance with the Intel format (little-endian), meaning that the byte with the smallest value is saved at the starting address and transmitted first (compare with big-endian or Motorola format, where the highest-value component is sent first).

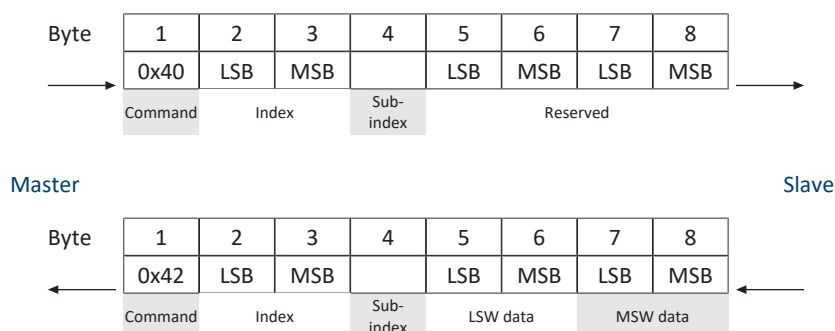
#### Writing parameters (Initiate Domain Download Request)

The controller (master) uses an Initiate Domain Download Request to initiate a write process for a communication parameter. The request receives a positive acknowledgement from an Initiate Domain Download Response of the drive controller (slave).



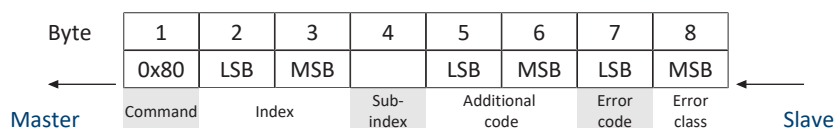
#### Reading parameters (Initiate Domain Upload Request)

The controller (master) uses an Initiate Domain Upload Request to initiate a read process for a communication parameter. The request receives a positive acknowledgement from an Initiate Domain Upload Response of the drive controller (slave).



#### Error message (Abort Domain Transfer)

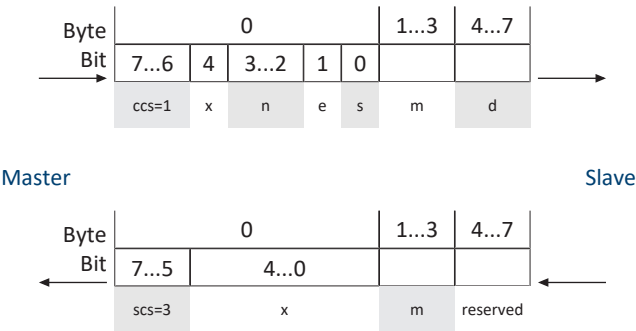
A drive controller (slave) provides a negative response to the write parameter or read parameter requests using an Abort Domain Transfer (see SDO transmission: Error codes).



10.3.3.2 Segmented transfer

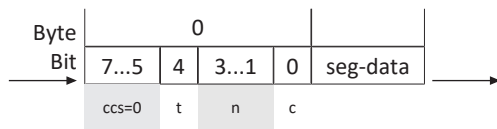
With SDO transmission via segmented transfer, more than 4 bytes of data can be transmitted distributed over several messages. The total number of bytes to be transmitted are sent in a first initiate message (Initiate SDO Download); this is followed by the segments (Download SDO Segment), each with 1 byte of control and protocol information and up to 7 bytes of payload.

Initiate SDO Download Protocol



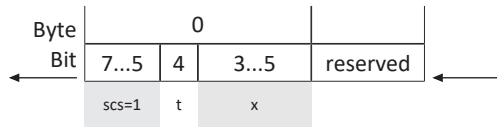
ccs	Client command specifier	1 = Initiate download request
scs	Server command specifier	3 = Initiate download response
n	Number of bytes	Number of bytes in "Data" that contain no usable data. If e = 0 , s = 1, then n = valid, otherwise n = 0
e	Transfer type	<div><div>0 = Normal transfer</div><div>1 = Expedited transfer</div></div>
s	Size indicator	<div><div>0 = Not displayed</div><div>1 = Displayed</div></div>
m	Multiplexor	= Index + subindex
d	Data	<div><div>If e = 0, s = 0, then d = reserved</div><div>If e = 0, s = 1, then d = number of bytes to be transmitted</div><div>If e = 1, s = 1, then d = 4-n</div></div>
x	Unused	x = 0

### Download SDO Segment Protocol



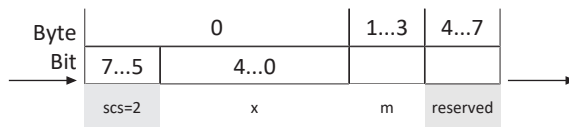
Master

Slave



ccs	Client command specifier	0 = Download segment request
scs	Server command specifier	1 = Download segment response
n	Number of bytes	Number of bytes in "Segment data" that contain no usable data. n = 0: No information about unused data
seg-data	Segment data	7 bytes of usable data
c	Continue	<ul style="list-style-type: none"> <li>0 = More segments follow</li> <li>1 = Last segment</li> </ul>
t	Toggle bit	t = 0 for segment 1; must change for each segment. Identical values for request and response.
x	Unused	x = 0

### Initiate SDO Upload Protocol



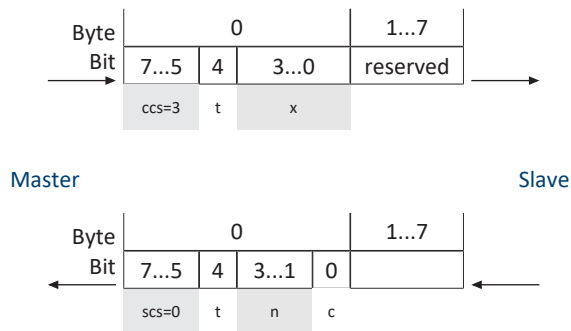
Master

Slave



ccs	Client command specifier	2 = Initiate upload request
scs	Server command specifier	2 = Initiate upload response
n	Number of bytes	Number of bytes in "Data" that contain no usable data. If e = 0, s = 1, then n = valid, otherwise n = 0
e	Transfer type	<ul style="list-style-type: none"> <li>0 = Normal transfer</li> <li>1 = Expedited transfer</li> </ul>
s	Size indicator	<ul style="list-style-type: none"> <li>0 = Not displayed</li> <li>1 = Displayed</li> </ul>
m	Multiplexor	= Index + subindex
d	Data	<ul style="list-style-type: none"> <li>If e = 0, s = 0, then d = reserved</li> <li>If e = 0, s = 1, then d = number of bytes to be transmitted</li> <li>If e = 1, s = 1, then d = 4-n</li> </ul>
x	Unused	x = 0

### Upload SDO Segment Protocol



ccs	Client command specifier	3 = Upload segment request
scs	Server command specifier	0 = Upload segment response
n	Number of bytes	Number of bytes in "Segment data" that contain no usable data. n = 0: No information about unused data
seg-data	Segment data	7 bytes of usable data
c	Continue	<ul style="list-style-type: none"> <li>0 = More segments follow</li> <li>1 = Last segment</li> </ul>
t	Toggle bit	t = 0 for segment 1; must change for each segment. Identical values for request and response.
x	Unused	x = 0

### Examples

Segment download with 16 bytes of data; contents: 01, 02, 03 ... 10 hex

Client: IDReq:	21	idx	x	10 00 00 00	(ccs = 1, e = 0 = normal, s = 1 -> data = no. of bytes)
Server: IDRes:	60	idx	x	00 00 00 00	
Client: DSegReq:	00	01 02 03 04 05 06 07			(ccs = 0, t = 0, n = 0, c = 0 -> all data bytes are used)
Server: DSegRes:	20	00 00 00 00 00 00 00			
Client: DSegReq:	10	08 09 0A 0B 0C 0D 0E			(ccs = 0, t = 1, n = 0, c = 0 -> all data bytes are used)
Server: DSegRes:	30	00 00 00 00 00 00 00			
Client: DSegReq:	0b	0F 10 00 00 00 00 00			(ccs = 0, t = 0, n = 5, c = 1 -> 5 data bytes are unused)
Server: DSegRes:	20	00 00 00 00 00 00 00			

Segment upload with 16 bytes of data; contents: 01, 02, 03 ... 10 hex

Client: IDUReq:	40	idx	x	00 00 00 00	(ccs = 2, rest = 0)
Server: IDURes:	41	idx	x	10 00 00 00	(scs = 2, x = 0, e = 0, s = 1 -> data contains no. of bytes to be uploaded)
Client: USegReq:	60	00 00 00 00 00 00 00			(ccs = 3, t = 0)
Server: USegRes:	00	01 02 03 04 05 06 07			(scs = 0, t = 0, n = 0, c = 0 -> all data bytes are used)
Client: USegReq:	70	00 00 00 00 00 00 00			(ccs = 3, t = 1)
Server: USegRes:	10	08 09 0A 0B 0C 0D 0E			(scs = 0, t = 1, n = 0, c = 0 -> all data bytes are used)
Client: USegReq:	60	00 00 00 00 00 00 00			(ccs = 3, t = 0)
Server: USegRes:	0b	0F 10 00 00 00 00 00			(scs = 0, t = 0, n = 5, c = 1 -> 5 data bytes are unused)

10.3.4 NMT: Network Management Objects

Network Management Objects (NMT objects) are broadcast objects that are used to control and monitor the NMT states of the nodes in the CANopen network. Each node in the CANopen network has a network management state machine (NMT state machine), which can be in various states.

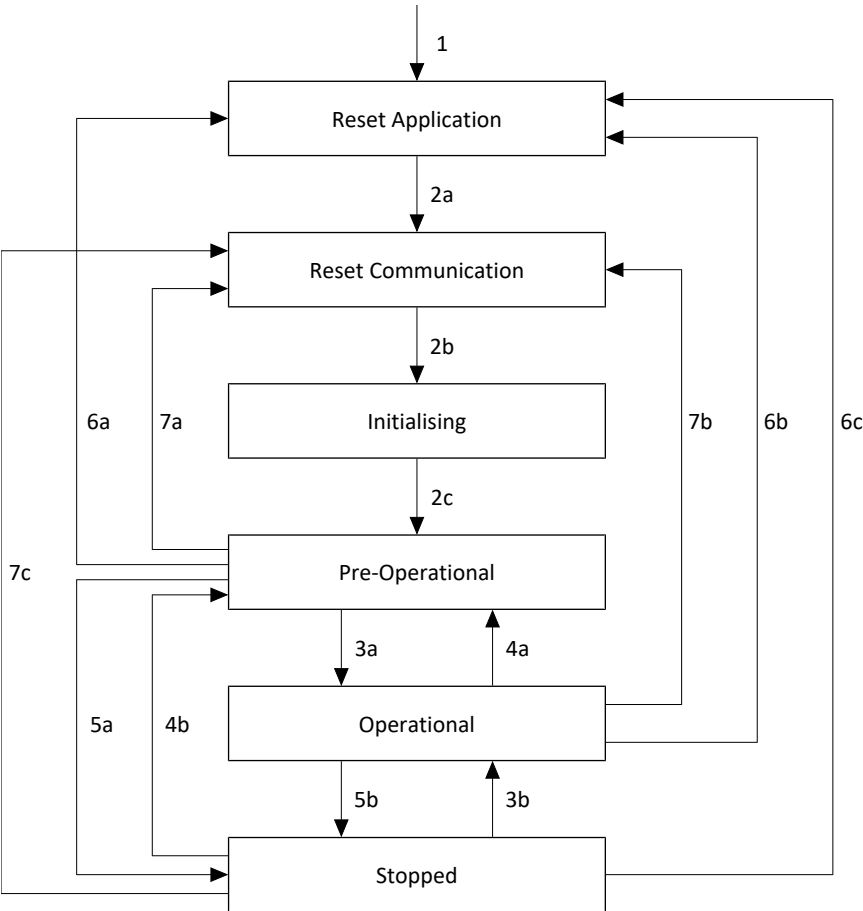


Fig. 7: Network management state machine: States and state changes

States

State		Description
Initialization	Reset application	Parameters of the device profile and manufacturer-specific parameters are loaded
	Reset communication	Parameters of the communication profile are loaded
	Initializing	CAN bus is initialized
Pre-operational		CAN bus is active, drive controller can be parameterized for operation via SDO (PDO: inactive)
Operational		CAN bus is active, all communication services are in operation (PDO: active)
Stopped		Communication services are stopped (exception: guarding, heartbeat)

## Boot-up

Switching on the supply voltage starts the device start-up including initialization of the NMT state machine. Upon the first state change from Initialization to Pre-operational, the drive controller sends a one-time boot-up message to the controller to signal that it is ready to receive NMT commands.

The boot-up message consists of a COB-ID and 1 data byte. The COB-ID of the boot-up message is 1792 or 700 hex + node ID; the content of the data byte is 0. Since the boot-up message is only sent once during device start-up, the COB-ID is then free for other tasks and is used for monitoring the NMT states of the nodes in the CANopen network by means of guarding or heartbeat (see [ERROR CONTROL: Error Control Objects \[► 55\]](#)).

## State change

NMT objects can be used to switch between the states of the NMT state machine. NMT objects consist of a COB-ID and 2 data bytes. The COB-ID for NMT objects is 0. Byte 1 of the NMT object contains the command specifier (cs) and byte 2 contains the node ID of the node; node ID 0 addresses all drive controllers.

COB-ID	Command specifier (cs)	Node ID
0	1	0

No.	Description	Command specifier (cs)
1	Switching on the supply voltage	–
2a, 2b, 2c	Switching on after initialization has been completed	–
3a, 3b	NMT_Start_Remote_Node command received	1
4a, 4b	NMT_Enter_Pre_Operational command received	128
5a, 5b	NMT_Stop_Remote_Node command received	2
6	NMT_Reset_Node command received	129
7	NMT_Reset_Communication command received	130

## Availability of the communication objects

The following table shows the availability of the communication objects depending on the NMT state.

Object	Initialization	Pre-operational	Operational	Stopped
Boot-up	✓	–	–	–
PDO	–	–	✓	–
SDO	–	✓	✓	–
NMT	–	✓	✓	✓
SYNC	–	✓	✓	–
EMCY	–	✓	✓	–

## 10.3.5 ERROR CONTROL: Error Control Objects

Error Control Objects are broadcast objects that are used to monitor the NMT states of the nodes in the CANopen network. CANopen offers 2 error control services for monitoring NMT states based on the periodic transmission of messages: guarding (node guarding/life guarding) and heartbeat. Since guarding and heartbeat have the same COB-ID (1792 or 700 hex + node ID), only one of the two services can be used at a time.

### 10.3.5.1 Guarding

When monitoring communication in the CANopen network via guarding (node guarding/life guarding), the controller and drive controller monitor each other. To use the guarding function, the heartbeat function must be deactivated (A210 = 0 ms).

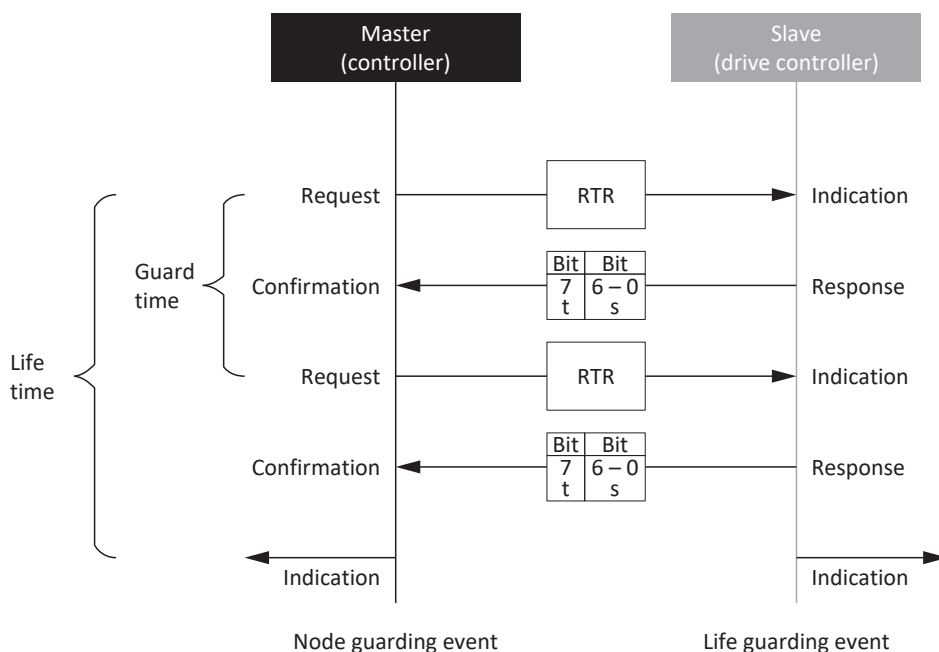


Fig. 8: Guarding protocol (node guarding/life guarding)

s = NMT state

t = Toggle bit

#### Node guarding

When monitoring via node guarding, the controller sends Remote Transmission Requests (RTRs) to all drive controllers in the CANopen network at regular intervals to query the NMT state of the respective node. The time interval in which the controller sends RTRs to the drive controllers is defined via the guard time (parameter: A203).

The drive controller recognizes the request from the controller and responds with a data frame of 1 byte length, of which bits 0 – 6 indicate the NMT state, while bit 7 changes state with each message (toggle bit).

The controller receives the data frame and checks the NMT state of the drive controller (confirmation). If the controller does not receive a data frame from a drive controller in response to the request via RTR, or if the information in the data frame (NMT state, toggle bit) does not match the expected information, the controller evaluates this as a communication error (node guarding event).

Life guarding

When monitoring via life guarding, a drive controller expects RTRs from the controller at regular intervals. If the drive controller does not receive any RTR from the controller within a certain time, the drive controller evaluates this as a communication error (life guarding event) and changes to the Fault device state.

The number of missing RTRs after which the drive controller has a fault is defined via the life time factor (parameter: A204).

The time interval in which the drive controller expects RTRs from the controller is called the "life time" and is the product of the guard time and life time factor (life time = A203 × A204).

10.3.5.2 Heartbeat

When monitoring communication in the CANopen network via heartbeat, the controller monitors the drive controllers. To use the heartbeat function, the guarding function must be deactivated (A203 = 0 ms or A204 = 0).

The advantage of the heartbeat function over the guarding function is that no RTRs are exchanged between the controller and drive controllers, which significantly reduces the utilization of the CANopen network.

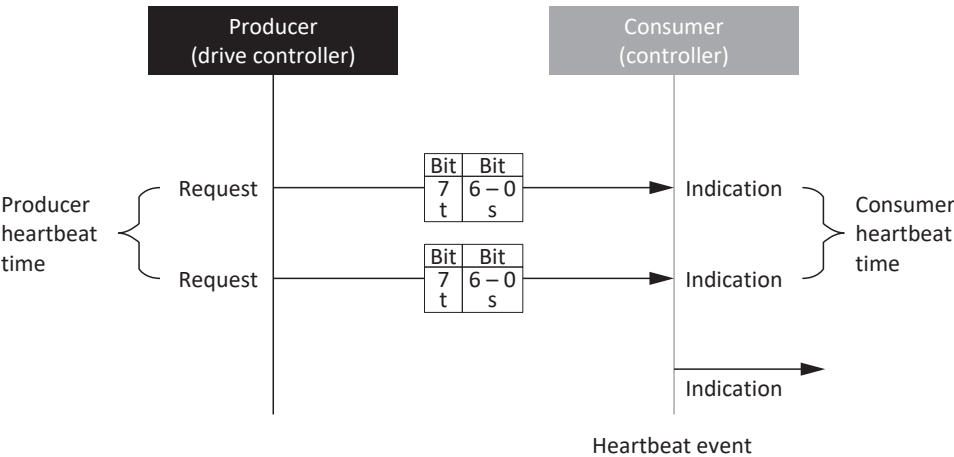


Fig. 9: Heartbeat protocol

s = NMT state

t = Toggle bit

When monitoring via the heartbeat function, a drive controller (heartbeat producer) sends heartbeat messages at regular intervals indicating its NMT state. The time interval in which the drive controller sends the heartbeat messages is defined via the producer heartbeat time (parameter: A210).

The controller (heartbeat consumer) expects the heartbeat messages at regular intervals (consumer heartbeat time). If the controller does not receive a heartbeat message from the drive controller within the defined consumer heartbeat time, this is considered a communication failure (heartbeat event).



Emergency Objects are peer-to-peer objects that are used to monitor the device states of the nodes in the network and are triggered in the event of internal device errors or faults.

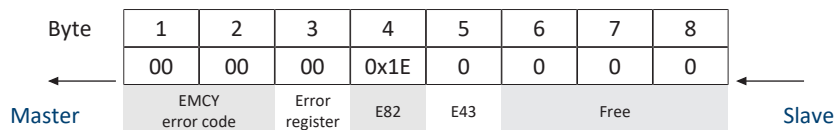
This mechanism automatically notifies the controller of when a drive controller enters and leaves the fault state and of the associated cause for the fault.

**EMCY message: Switch to the fault state**

Byte	1	2	3	4	5	6	7	8
	43	0x10	0x01	0x29	0	0	0	0
	EMCY error code		Error register	E82	E43	Free		

A table with the possible error codes of an EMCY message can be found at [EMCY message: Device fault error codes](#) [► 67].

The following graphic shows an example of the structure of an EMCY message when leaving the Fault device state.



## 10.4 Cycle times

Possible cycle times can be found in the following table.

Type	Cycle times	Relevant parameters
CANopen fieldbus, cyclical communication	1 ms, 2 ms, 4 ms, 8 ms	Adjustable in A150

Tab. 11: Cycle times

## 10.5 Activating and executing actions

To be able to activate and execute actions via fieldbus, you must first enable action activation in DriveControlSuite and extend the process data by the control byte and status word for actions.

### Enabling action activation

1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
2. Select the CiA 402 application wizard > Additional functions.
3. Enable the Action activation option.

### Adjusting receive process data

1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
2. Select the CANopen wizard > Received process data RxPDO.
3. A225[0] – A225[5], A226[0] – A226[5]:  
Add A75, the control byte for activating actions, to the receive process data.

### Adjusting transmit process data

1. Select the relevant drive controller in the project tree and click on the desired projected axis in the Project menu > Wizard area.
2. Select the CANopen wizard > Transmitted process data TxPDO.
3. A233[0] – A233[5], A234[0] – A234[5]:  
Add A69, the status word for activating actions, to the transmit process data.

## Executing an action

Then execute the desired action. Here, take into account any prerequisites with regard to the device state as well as any further measures required after the start of the action. All prerequisites as well as more detailed information on the individual actions can be found in the corresponding parameter descriptions in DriveControlSuite.

Selecting an action	Establishing the device state	Starting an action	Executing the next step	Completing an action (by progress = 100%)
0001 bin = Save values (A00)	—	Execute (A75, bit 0 =1)	—	Undo execute (A75, bit 0 = 0)
0011 bin = Reset memorized values (A37)				
0111 bin = Clear reference (I38)				
1000 bin = Delete limit switch memory (I52)				
0010 bin = Restart (A09)	E48 ≠ 4: Enabled + E48 ≠ 7: Quick stop	Execute (A75, bit 0 =1)	—	Undo execute (A75, bit 0 = 0)
1101 bin = Test winding (B43)	E48 = 2: Ready for switch-on	Execute (A75, bit 0 =1)	—	Undo execute (A75, bit 0 = 0)
1010 bin = Test phase (B40)	E48 = 2: Ready for switch-on	Execute (A75, bit 0 =1)	Enable drive controller (E48 = 4: Enabled)	Undo execute (A75, bit 0 = 0) + Undo enable
1011 bin = Calibrate motor (B41)				
1100 bin = Optimize current controller (B42)				
1110 bin = Optimize current controller (standstill) (B49)				
0100 bin = Test brake (B300)				
0101 bin = Grind brake (B301)				
0110 bin = Brake 2 grind (B302)				
1001 bin = Test brake (S18)				

Tab. 12: Selecting and executing an action

# 10.6     Fieldbus scaling

Using parameter A213, you define the scaling for both cyclical transmission of process data objects as well as acyclical transmission of service data objects in the network in the DriveControlSuite commissioning software. The values are either converted and represented as an integer or transmitted as a raw value without scaling according to their data types.

Regardless of the settings selected in parameter A213, the configuration as well as the firmware both work exclusively with raw values. The following graphic shows an overview of fieldbus scaling.

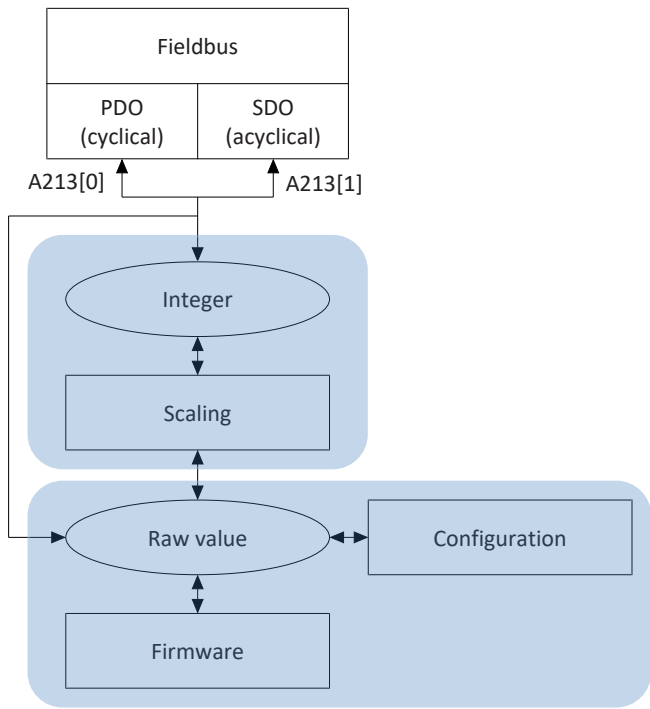


Fig. 10: Overview of fieldbus scaling

For transmission as an integer, the number of decimal places can be defined for all parameters that affect positions, velocities, accelerations, decelerations and jerk. For all other parameters, the number of decimal places is fixed. The values for scaling are output in DriveControlSuite with the properties of a parameter. The following table lists the parameters which you can use to define the number of decimal places for scaled transmission.

Scaling	Axis model	Master axis model
Position	I06	G46
Velocity (DB)	I66	G66
Velocity (CiA)	A310	—
Acceleration, deceleration, jerk (DB)	I67	G67
Acceleration, deceleration, jerk (CiA)	A311	—

Tab. 13: Fieldbus scaling for integer: Parameters for defining the decimal places

# 11 Appendix

## 11.1 Supported communication objects

The following chapters provide an overview of the supported communication objects of the standardized profile CiA 301 CANopen application layer and communication profile for CANopen as well as their mapping to the parameters of STÖBER.

Information on the supported communication objects of the CiA 402 profile can be found in the corresponding application manual.

### 11.1.1 CiA 301 CANopen: 1000 hex – 1FFFF hex

The following table includes the supported communication objects for the standardized profile CiA 301 CANopen application layer and communication profile for CANopen as well as how the objects are mapped to the corresponding STÖBER parameters.

Index	Subindex	TxPDO	RxPDO	Name	Comment
1000 hex	0 hex	—	—	Device type	Constant value 20192 hex Bit 0 – 15: Device profile number, 192 hex = 402 Bit 16 – 23: Type, 2 hex = Servo drive Bit 24 – 31: Reserved
1001 hex	0 hex	—	—	Error register	
1002 hex	0 hex	—	—	Manufacturer status register	E48
1003 hex				Pre-defined error field	
1003 hex	0 hex	—	—	Number of errors	
1003 hex	1 hex – A hex	—	—	Standard error field	
1005 hex	0 hex	—	✓	COB-ID SYNC message	A200; value range 1 – 2047
1006 hex	0 hex	—	✓	Communication cycle period	A201; value range 0 – 32 000 000
1008 hex	0 hex	—	✓	Manufacturer device name	E50
1009 hex	0 hex	—	✓	Manufacturer hardware version	E52[1]
100A hex	0 hex	—	✓	Manufacturer software version	E53[3]
100C hex	0 hex	—	✓	Guard time	A203; life time in ms = 100C hex * 100D hex
100D hex	0 hex	—	✓	Life time factor	A204; life time in ms = 100C hex * 100D hex
1010 hex				Store parameters supported	Record with 1 element
1010 hex	0 hex	—	—	Highest subindex supported	Constant value of 1 hex
1010 hex	1 hex	—	✓	Save all parameters	A00[0]; ASCII <i>save</i> starts Save values
1014 hex	0 hex	—	✓	COB-ID EMCY	A207
1015 hex	0 hex	—	✓	Inhibit time EMCY	A208
1017 hex	0 hex	—	✓	Producer heartbeat time	A210
1018 hex				Identity object	Record with 4 elements
1018 hex	0 hex	—	—	Highest subindex supported	Constant value of 4 hex

Index	Subindex	TxPDO	RxPDO	Name	Comment
1018 hex	1 hex	—	—	Vendor ID	manufacturer ID: B9 hex
1018 hex	2 hex	—	—	Product code	Nominal power in unit 0.1 kW
1018 hex	3 hex	—	—	Revision number	Software build number
1018 hex	4 hex	—	—	Serial number	E52[2]
1020 hex				Verify configuration	Record with 2 elements
1020 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1020 hex	1 hex	—	—	Configuration data	A211
1020 hex	2 hex	—	—	Configuration time	A212
1200 hex				1 <sup>st</sup> SDO server parameter	Record with 2 elements
1200 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1200 hex	1 hex	—	✓	COB-ID Client -> Server (rx)	
1200 hex	2 hex	—	✓	COB-ID Server -> Client (tx)	
1201 hex				2 <sup>nd</sup> SDO server parameter	Record with 3 elements
1201 hex	0 hex	—	—	Highest subindex supported	Number of elements: 3
1201 hex	1 hex	—	✓	COB-ID Client -> Server (rx)	A218[0]
1201 hex	2 hex	—	✓	COB-ID Server -> Client (tx)	A218[1]
1201 hex	3 hex	—	✓	Node ID of the SDO client	A218[2]
1202 hex				3 <sup>rd</sup> SDO server parameter	Record with 3 elements
1202 hex	0 hex	—	—	Highest subindex supported	Constant value of 3 hex
1202 hex	1 hex	—	✓	COB-ID Client -> Server (rx)	A219[0]
1202 hex	2 hex	—	✓	COB-ID Server -> Client (tx)	A219[1]
1202 hex	3 hex	—	✓	Node ID of the SDO client	A219[2]
1203 hex				4 <sup>th</sup> SDO server parameter	Record with 3 elements
1203 hex	0 hex	—	—	Highest subindex supported	Constant value of 3 hex
1203 hex	1 hex	—	✓	COB-ID Client -> Server (rx)	A220[0]
1203 hex	2 hex	—	✓	COB-ID Server -> Client (tx)	A220[1]
1203 hex	3 hex	—	✓	Node ID of the SDO client	A220[2]
1400 hex				1 <sup>st</sup> receive PDO communication parameter	Record with 2 elements
1400 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1400 hex	1 hex	—	✓	COB-ID used by RxPDO	A221[0]
1400 hex	2 hex	—	✓	Transmission type	A221[1]; value range 1 – 240, 254
1401 hex				2 <sup>nd</sup> RxPDO communication parameter	Record with 2 elements
1401 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1401 hex	1 hex	—	✓	COB-ID used by RxPDO	A222[0]
1401 hex	2 hex	—	✓	Transmission type	A222[1]; value range 1 – 240, 254
1402 hex				3 <sup>rd</sup> RxPDO communication parameter	Record with 2 elements
1402 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1402 hex	1 hex	—	✓	COB-ID used by RxPDO	A223[0]
1402 hex	2 hex	—	✓	Transmission type	A223[1]; value range 1 – 240, 254

Index	Subindex	TxPDO	RxPDO	Name	Comment
1403 hex				4 <sup>th</sup> RxPDO communication parameter	Record with 2 elements
1403 hex	0 hex	—	—	Highest subindex supported	Constant value of 2 hex
1403 hex	1 hex	—	✓	COB-ID used by RxPDO	A224[0]
1403 hex	2 hex	—	✓	Transmission type	A224[1]; value range 1 – 240, 254
1600 hex				1st RxPDO mapping parameter	Record with 6 elements
1600 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6 hex
1600 hex	1 hex – 6 hex	—	✓	Application objects	A225[0] – A225[5]
1601 hex				2nd RxPDO mapping parameter	Record with 6 elements
1601 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6 hex
1601 hex	1 hex – 6 hex	—	✓	Application objects	A226[0] – A226[5]
1602 hex				3rd RxPDO mapping parameter	Record with 6 elements
1602 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6 hex
1602 hex	1 hex – 6 hex	—	✓	Application objects	A227[0] – A227[5]
1603 hex				4th RxPDO mapping parameter	Record with 6 elements
1603 hex	0 hex	—	—	Number of mapped application objects in RxPDO	Constant value of 6 hex
1603 hex	1 hex – 6 hex	—	—	Application objects	A228[0] – A228[5]
1800 hex				1 <sup>st</sup> TxPDO communication parameter	Record with 4 elements
1800 hex	0 hex	—	—	Highest subindex supported	Constant value of 5 hex
1800 hex	1 hex	—	✓	COB-ID used by TxPDO	A229[0]
1800 hex	2 hex	—	✓	Transmission type	A229[1]; value range 1 – 240, 254
1800 hex	3 hex	—	✓	Inhibit time	A229[2]
1800 hex	5 hex	—	✓	Event timer	A229[3]
1801 hex				2 <sup>nd</sup> TxPDO communication parameter	Record with 4 elements
1801 hex	0 hex	—	—	Highest subindex supported	Constant value of 5 hex
1801 hex	1 hex	—	✓	COB-ID used by TxPDO	A230[0]
1801 hex	2 hex	—	✓	Transmission type	A230[1]; value range 1 – 240, 254
1801 hex	3 hex	—	✓	Inhibit time	A230[2]
1801 hex	5 hex	—	✓	Event timer	A230[3]
1802 hex				3 <sup>rd</sup> TxPDO communication parameter	Record with 4 elements
1802 hex	0 hex	—	—	Highest subindex supported	Constant value of 5 hex
1802 hex	1 hex	—	✓	COB-ID used by TxPDO	A231[0]
1802 hex	2 hex	—	✓	Transmission type	A231[1]; value range 1 – 240, 254
1802 hex	3 hex	—	✓	Inhibit time	A231[2]

Index	Subindex	TxPDO	RxPDO	Name	Comment
1802 hex	5 hex	—	✓	Event timer	A231[3]
1803 hex				4 <sup>th</sup> TxPDO communication parameter	Record with 4 elements
1803 hex	0 hex	—	—	Highest subindex supported	Constant value of 5 hex
1803 hex	1 hex	—	✓	COB-ID used by TxPDO	A232[0]
1803 hex	2 hex	—	✓	Transmission type	A232[1]; value range 1 – 240, 254
1803 hex	3 hex	—	✓	Inhibit time	A232[2]
1803 hex	5 hex	—	✓	Event timer	A232[3]
1A00 hex				1st TxPDO mapping parameter	Record with 6 elements
1A00 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	Constant value of 6 hex
1A00 hex	1 hex – 6 hex	—	✓	Application objects	A233[0] - A233[5]
1A01 hex				2nd TxPDO mapping parameter	Record with 6 elements
1A01 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	Constant value of 6 hex
1A01 hex	1 hex – 6 hex	—	✓	Application objects	A234[0] - A234[5]
1A02 hex				3rd TxPDO mapping parameter	Record with 6 elements
1A02 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	Constant value of 6 hex
1A02 hex	1 hex – 6 hex	—	✓	Application objects	A235[0] - A235[5]
1A03 hex				4th TxPDO mapping parameter	Array with 6 elements
1A03 hex	0 hex	—	—	Number of mapped application objects in TxPDO	Constant value of 6 hex
1A03 hex	1 hex – 6 hex	—	—	Application objects	A236[0] - A236[5]

Tab. 14: CiA 301 communication objects: 1000 hex – 1FFF hex



11.1.2      Manufacturer-specific parameters: 2000 hex – 53FF hex

Index, subindex and calculation example

Information	
-------------	--

Index and Subindex must be specified in the format required by the controller.

Information	
-------------	--

The calculation described below is only valid for converting the manufacturer-specific parameters.

The index is calculated from the group and line of the parameter according to the following formula:  
Index = 8192 + (number of the group × 512) + number of the line

The subindex for simple parameters is always 0; for array and record parameters, it corresponds to the subindex of the element number of the parameter.

	Simple parameters	Array or record parameter
Index	8192 + (number of the group × 512) + number of the line	
Subindex	0	Element number

Tab. 15: Index and subindex for manufacturer-specific parameters

Calculation example

Calculation for parameter E200[0]:

Number of the group = 4

Number of the line = 200

Index = 8192 + (4 × 512) + 200 = 10440 = 28C8 hex

Subindex = 0 = 0 hex

### Communication objects

The following table includes the supported communication objects and how they are mapped to the corresponding parameters of STÖBER.

Index	Group	Number	Parameters
2000 hex – 21FF hex	A: Drive controller	0	A00 – A511
2200 hex – 23FF hex	B: Motor	1	B00 – B511
2400 hex – 25FF hex	C: Machine	2	C00 – C511
2600 hex – 27FF hex	D: Set value	3	D00 – D511
2800 hex – 29FF hex	E: Show	4	E00 – E511
2A00 hex – 2BFF hex	F: Terminals	5	F00 – F511
2C00 hex – 2DFF hex	G: Technology	6	G00 – G511
2E00 hex – 2FFF hex	H: Encoders	7	H00 – H511
3000 hex – 31FF hex	I: Motion	8	I00 – I511
3200 hex – 33FF hex	J: Motion blocks	9	J00 – J511
3400 hex – 35FF hex	K: Control panel	10	K00 – K511
3600 hex – 37FF hex	M: Profile	12	M00 – M511
3E00 hex – 3FFF hex	P: Customer-specific parameters	15	P00 – P511
4000 hex – 41FF hex	Q: Customer-specific parameters, instance-dependent	16	Q00 – Q511
4200 hex – 43FF hex	R: Production data	17	R00 – R511
4400 hex – 45FF hex	S: Safety	18	S00 – S511
4600 hex – 47FF hex	T: Scope	19	T00 – T511
4800 hex – 49FF hex	U: Protection functions	20	U00 – U511
5200 hex – 53FF hex	Z: Fault counter	25	Z00 – Z511

Tab. 16: Manufacturer-specific communication objects: 2000 hex – 53FF hex

## 11.2 EMCY message: Device fault error codes

Error code	Error register	Event (E82)
0 hex: No error	0 hex: No error	30: Inactive
1000 hex: Generic error	1 hex: Generic error	48: Brake release monitoring, 49: Brake, 50: Safety module, 71: Firmware or 80: Illegal action
2110 hex: Short circuit earth	2 hex: Current	31: Short/ground
2230 hex: Intern short circuit earth	2 hex: Current	32: Short/ground internal
2310 hex: Continous overcurrent	2 hex: Current	33: Overcurrent
3110 hex: Mains overvoltage	4 hex: Voltage	36: High voltage
3120 hex: Mains undervoltage	4 hex: Voltage	46: Low voltage
3130 hex: Phase failure	1 hex: Generic error	83: Failure of one/ all phases (mains)
3180 hex: Mains failure	1 hex: Generic error	84: Drop in network voltage when power section active
4210 hex: Temperature	8 hex: Temperature	38: Temperature drive controller sensor
4280 hex: Temperature device I <sup>2</sup> t	8 hex: Temperature	39: Overtemperature drive controller i <sup>2</sup> t or 59: Overtemperature drive controller i <sup>2</sup> t
4310 hex: Temperature drive	8 hex: Temperature	41: Temp.MotorTMP
4380 hex: Temperature drive I <sup>2</sup> t	8 hex: Temperature	45: Overtemp.motor i <sup>2</sup> t
5200 hex: Device hardware	1 hex: Generic error	34: Hardware fault or 55: Option module
5440 hex: Contacts	1 hex: Generic error	43: AI1 wire break
6010 hex: Internal software	1 hex: Generic error	35: Watchdog or 57: Runtime requirement
6320 hex: Loss of parameters	1 hex: Generic error	40: Invalid data or 70: Parameter consistency
7110 hex: Brake chopper	1 hex: Generic error	72: Brake test timeout, 73: Axis 2 brake test timeout, 74: Axis 3 brake test timeout or 75: Axis 4 brake test timeout
	8 hex: Temperature	42: TempBrakeRes
7120 hex: Motor	1 hex: Generic error	69: Motor connection or 81: Motor allocation
7303 hex: Resolver 1 fault	1 hex: Generic error	37: Motor encoder
7304 hex: Resolver 2 fault	1 hex: Generic error	58: Encoder simulation, 76: Position encoder, 77: Master encoder or 79: Motor/position encoder plausibility
7321 hex: Hall sensor failure	1 hex: Generic error	82: Hall sensor
7500 hex: Communication	10 hex: Communication	52: Communication
7580 hex: Communication control panel	1 hex: Generic error	88: Control panel
8311 hex: Excess torque	1 hex: Generic error	47: Torque/force-max. limit

Error code	Error register	Event (E82)
8400 hex: Velocity speed control	1 hex: Generic error	56: Overspeed
8500 hex: Position control	1 hex: Generic error	53: Limit switch
8510 hex: Excessive reference position jump	1 hex: Generic error	85: Excessive jump in reference value
8600 hex: Positioning controller	1 hex: Generic error	51: Virtual master software limit switch
8611 hex: Following error	1 hex: Generic error	54: Following error
8612 hex: Reference limit	1 hex: Generic error	78: Position limit cyclic
FF00 – FF07 hex: Manufacturer specific error	1 hex: Generic error	60: Application event 0 – 67: Application event 7
FF09 hex: Manufacturer specific error	1 hex: Generic error	44: External fault 1
FF0A hex: Manufacturer specific error	1 hex: Generic error	68: External fault 2

Tab. 17: EMCY: Device fault error codes

## 11.3 Detailed information

The documentation listed below provides you with further relevant information on the 6th STÖBER drive controller generation. You can find the current status of the documentation in the STÖBER download center at <http://www.stoeber.de/en/downloads/>, if you enter the ID of the documentation in the search.

Title	Documentation	Contents	ID
SD6 drive controller	Manual	System design, technical data, project configuration, storage, installation, connection, commissioning, operation, service, diagnostics	442426
CiA 402 application – SD6	Manual	Project planning, configuration, parameterization, function test, detailed information	443077

Additional information and sources that form the basis of this documentation or are referenced by the documentation:

CiA DS 301 V4.02 – CANopen application layer and communication profile

CANopen communication profile; describes the key services and protocols under CANopen.

CiA DSP 302 V3.0 – CANopen application layer and communication profile

CANopen framework for programmable devices

CiA DS 402 V2.0 – CANopen device profile drives and motion control

CANopen device profiles; describe the behavior of numerous device classes.

CiA DRP 303-1, ISO 11898-2

Recommendations for cables and plug connectors.

EN 50325-4 2002 : *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 4: CANopen*. Specification. December 2002.

IEC 61800-7-201:2015 : *Adjustable speed electrical power drive systems - Part 7-201: Generic interface and use of profiles for power drive systems - Profile type 1 specification*. Specification. Version 2.0, November 2015.

IEC 61800-7-301:2015 : *Adjustable speed electrical power drive systems - Part 7-301: Generic interface and use of profiles for power drive systems - Mapping of profile type 1 to network technologies*. Specification. Version 2.0, November 2015.

### Information concerning CANopen

You can obtain general information about CAN and CANopen at the CAN in Automation (CiA) website <https://www.can-cia.org/>.

## 11.4 Abbreviations

Abbreviation	Meaning
CAN	Controller Area Network
CAN-H	CAN-High
CAN-L	CAN-Low
CAL	CAN Application Layer
CiA	CAN in Automation
COB-ID	Communication Object Identifier
DBT	Distributor
DC	Direct Current
EMCY	Emergency
GND	Ground
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IGB	Integrated Bus
IP	Internet Protocol
LSB	Least Significant Bit
LSW	Least Significant Word
MSB	Most Significant Bit
MSW	Most Significant Word
NMT	Network Management
PDO	Process Data Objects
RTR	Remote Transmission Request
RxPDO	Receive PDO (receive process data)
RxSDO	Receive SDO (receive service data)
SDO	Service Data Objects
SYNC	Synchronization
TxPDO	Transmit PDO (transmit process data)
TxSDO	Transmit SDO (transmit service data)

## 12 Contact

### 12.1 Consultation, service and address

We would be happy to help you!

We offer a wealth of information and services to go with our products on our website:

<http://www.stoeber.de/en/service>

For additional or personalized information, contact our consultation and support service:

<http://www.stoeber.de/en/support>

If you need our system support:

Phone +49 7231 582-3060

[systemsupport@stoeber.de](mailto:systemsupport@stoeber.de)

If you need a replacement device:

Phone +49 7231 582-1128

[replace@stoeber.de](mailto:replace@stoeber.de)

Call our 24-hour service hotline:

Phone +49 7231 582-3000

Our address:

STÖBER Antriebstechnik GmbH + Co. KG

Kieselbronner Strasse 12

75177 Pforzheim, Germany

### 12.2 Your opinion is important to us

We created this documentation to the best of our knowledge with the goal of helping you build and expand your expertise productively and efficiently with our products.

Your suggestions, opinions, wishes and constructive criticism help us to ensure and further develop the quality of our documentation.

If you want to contact us for a specific reason, we would be happy to receive an e-mail from you at:

[documentation@stoeber.de](mailto:documentation@stoeber.de)

Thank you for your interest.

Your STÖBER editorial team

## 12.3 Close to customers around the world

We offer you committed, expert advice and support in over 40 countries worldwide:

**STOBER AUSTRIA**

www.stoeber.at  
+43 7613 7600-0  
sales@stoeber.at

**STOBER FRANCE**

www.stober.fr  
+33 478 98 91 80  
sales@stober.fr

**STOBER HUNGARY**

www.stoeber.de  
+36 53 5011140  
info@emtc.hu

**STOBER JAPAN**

www.stober.co.jp  
+81-3-5875-7583  
sales@stober.co.jp

**STOBER TAIWAN**

www.stober.tw  
+886 4 2358 6089  
sales@stober.tw

**STOBER UK**

www.stober.co.uk  
+44 1543 458 858  
sales@stober.co.uk

**STOBER CHINA**

www.stoeber.cn  
+86 512 5320 8850  
sales@stoeber.cn

**STOBER Germany**

www.stoeber.de  
+49 4 7231 582-0  
sales@stoeber.de

**STOBER ITALY**

www.stober.it  
+39 02 93909570  
sales@stober.it

**STOBER SWITZERLAND**

www.stoeber.ch  
+41 56 496 96 50  
sales@stoeber.ch

**STOBER TURKEY**

www.stober.com  
+90 216 510 2290  
sales-turkey@stober.com

**STOBER USA**

www.stober.com  
+1 606 759 5090  
sales@stober.com



# Glossary

## Broadcast domain

---

Logical grouping of network devices within a local network that reaches all nodes via broadcast.

## COB-ID

---

Each message sent using a CAN bus has a COB-ID, which determines the priority of this message (low COB-ID = high priority). The COB-ID is composed of the function code and the node ID ( $\text{COB-ID} = \text{FC} + \text{NID}$ ).

## EMCY

---

Communication objects in a CANopen or EtherCAT network that, in the event of incorrect state transitions or device-internal errors, transmit the associated error codes and causes.

## Function code

---

ID for distinguishing the type of CAN message and its priority. The function code and node ID form the COB-ID.

## IPv4 limited broadcast

---

Type of broadcast in a network with IPv4 (Internet Protocol version 4). The IP address 255.255.255.255 is entered as the destination. The content of the broadcast is not forwarded by a router, which limits it to the local network.

## Node ID

---

Bus address of a device (master, slave) in a CANopen network. The node ID and function code form the COB-ID.

## Process Data Objects (PDO)

---

Communication objects in a CANopen or EtherCAT network that transmit data such as set and actual values, control commands or status information based on events or objectives, in cycles or in real time on request. PDOs are generally exchanged over the process data channel with high priority. Depending on the view of the respective node, a distinction is made between receive PDOs (RxPDO) and transmit PDOs (TxPDO).

## SDO

---

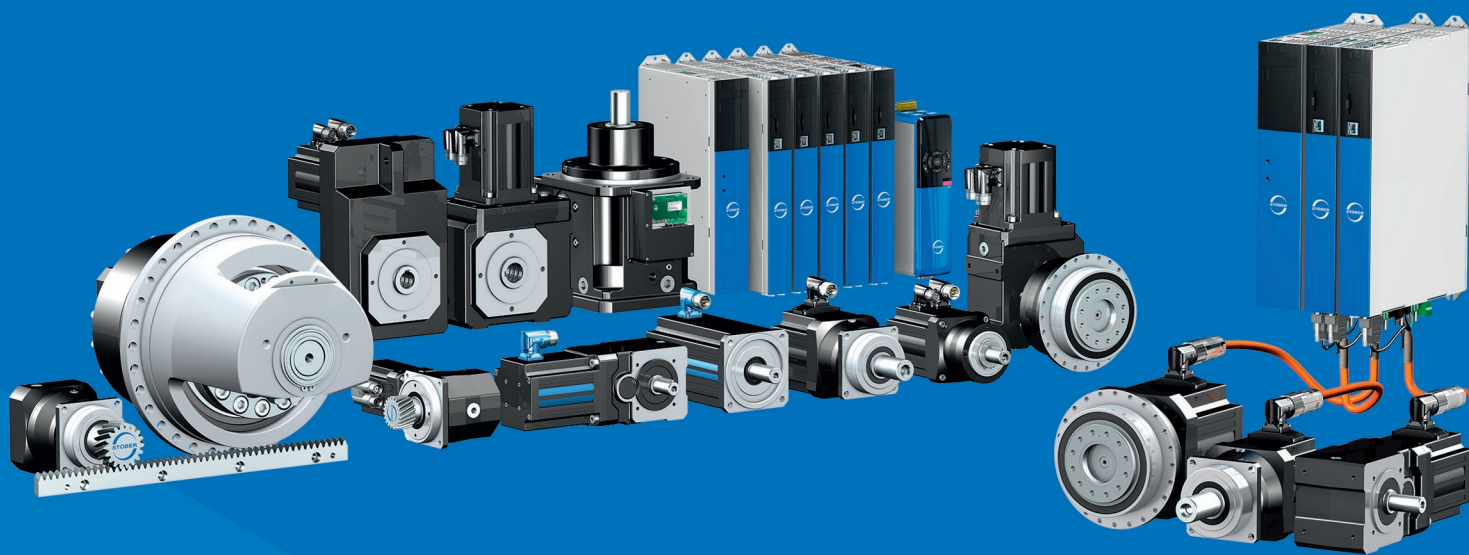
Communication objects in a CANopen or EtherCAT network that grant access to the object directory and enable device configuration. SDOs are transmitted over the mailbox channel acyclically during ongoing cyclical CANopen or EtherCAT operation.

## List of figures

Fig. 1	CANopen: Network structure, using the SD6 series as an example .....	11
Fig. 2	CANopen network: Terminating resistors .....	13
Fig. 3	DS6: Program interface .....	16
Fig. 4	DriveControlSuite: Navigation using text links and symbols .....	18
Fig. 5	LEDs for the CAN status .....	35
Fig. 6	CAN message: Structure .....	41
Fig. 7	Network management state machine: States and state changes .....	53
Fig. 8	Guarding protocol (node guarding/life guarding) .....	55
Fig. 9	Heartbeat protocol .....	56
Fig. 10	Overview of fieldbus scaling .....	60

## List of tables

Tab. 1	CAN bus: Transmission rate and max. cable length .....	14
Tab. 2	X200 connection description .....	14
Tab. 3	Parameter groups .....	19
Tab. 4	Parameters: data types, parameter types, possible values .....	20
Tab. 5	Parameter types .....	21
Tab. 6	Meaning of the green LED (Run) .....	35
Tab. 7	Meaning of the red LED (error) .....	35
Tab. 8	Event 52 – Causes and actions .....	37
Tab. 9	Object directory – Structure .....	40
Tab. 10	Transmission type .....	47
Tab. 11	Cycle times .....	57
Tab. 12	Selecting and executing an action .....	59
Tab. 13	Fieldbus scaling for integer: Parameters for defining the decimal places .....	60
Tab. 14	CiA 301 communication objects: 1000 hex – 1FFFF hex .....	61
Tab. 15	Index and subindex for manufacturer-specific parameters .....	65
Tab. 16	Manufacturer-specific communication objects: 2000 hex – 53FF hex .....	66
Tab. 17	EMCY: Device fault error codes .....	67



4 4 2 6 3 7 . 0 2

10/2023

STÖBER Antriebstechnik GmbH + Co. KG  
Kieselbronner Str. 12  
75177 Pforzheim  
Germany  
Tel. +49 7231 582-0  
mail@stoeber.de  
www.stober.com

24 h Service Hotline  
+49 7231 582-3000

[www.stober.com](http://www.stober.com)