CANopen – SD6
Operating manual

en-US
09/2019
ID 442637.01

STÖBER

Table of contents

1 Foreword 4

2 User information 5

2.1 Storage and transfer 5

2.2 Described product 5

2.3 Timeliness 5

2.4 Original language 5

2.5 Limitation of liability 5

2.6 Formatting conventions 6

2.6.1 Use of symbols 6

2.6.2 Markup of text elements 7

2.6.3 Mathematics and formulas 7

2.7 Trademarks 8

3 General safety instructions 9

3.1 Directives and standards 9

3.2 Qualified personnel 9

3.3 Intended use 9

3.4 Operational environment and operation 9

3.5 Disposal 10

4 Network structure 11

5 CA6 communication module 12

5.1 Installation 12

6 Connection 13

6.1 Selecting suitable lines 13

6.2 Terminal resistances 14

6.3 Transmission rate and line length 14

6.4 X200: CANopen 15

6.5 LED display 16

6.5.1 CAN status 16

7 What you should know before commissioning 17

7.1 Program interfaces 17

7.1.1 DS6: Structure of the program interface 17

7.2 Meaning of parameters 18

7.2.1 Parameter groups 19

7.2.2 Parameter types and data types 20

7.2.3 Parameter types 21

09/2019 | ID 442637.01

7.2.4 Parameter structure ... 21
7.2.5 Parameter visibility ... 22
7.3 Power-loss protected storage ... 22
8 Commissioning ... 23
8.1 DS6: Configuring the drive controller ... 24
8.1.1 Initiating the project ... 24
8.1.2 Parameterizing global CANopen settings ... 25
8.1.3 Configuring PDO transmission ... 26
8.1.4 Transmitting and saving a project configuration ... 28
8.2 Putting the CANopen system into operation ... 28
9 More about CANopen? ... 29
9.1 CAN bus and CANopen ... 29
9.1.1 CANopen – Communication ... 29
9.1.2 Object directory ... 30
9.1.3 Network structure ... 30
9.2 CAN message ... 31
9.2.1 COB-ID ... 32
9.3 Communication objects ... 35
9.3.1 Process data object – PDO ... 35
9.3.2 Service data object – SDO ... 38
9.3.3 Network management object – NMT ... 44
9.3.4 Error Control Objects – ERROR ... 46
9.3.5 Synchronization object – SYNC ... 49
9.3.6 Emergency objects – EMCY ... 49
10 Appendix ... 50
10.1 Supported communication objects ... 50
10.1.1 CiA 301 CANopen: 1000 hex – 1FFFF hex ... 50
10.1.2 Manufacturer-specific parameters: 2000 hex – 53FF hex ... 54
10.1.3 CiA 402 Drives and motion control: 6000 hex – 65FF hex ... 55
10.2 Detailed information ... 58
10.3 Abbreviations ... 59
11 Contact ... 60
11.1 Consultation, service and address ... 60
11.2 Your opinion is important to us ... 60
11.3 Close to customers around the world ... 61
Glossary ... 62

09/2019 | ID 442637.01

1 Foreword

STOBER drive controllers of the SD6 series 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.

09/2019 | ID 442637.01

2 User information

This documentation assists you with commissioning the STOBER SD6 series of the 6th generation in combination with higher-level controller systems over a CANopen network.

The SD6 drive controllers provide CANopen functionality over a separate CA6 communication module that 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 so, follow the instructions in the commissioning instructions for the SD6 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

Drive controllers of the SD6 series in combination with the DriveControlSuite software (DS6) in V 6.4-D or later and associated firmware in V 6.4-D or later.

2.3 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/.

2.4 Original language

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

2.5 Limitation of liability

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

STOBER shall assume no responsibility for damage resulting 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 projecting and operation of the product by unqualified personnel.

2.6 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.

2.6.1 Use of symbols

Safety instructions are identified with the following 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!

Notice

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.

09/2019 | ID 442637.01

2.6.2 Markup of text elements

Certain elements of the continuous text are distinguished as follows.

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

Software and 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.

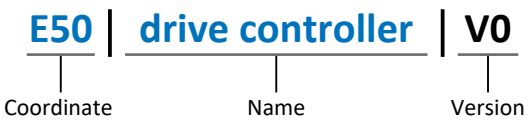
Main menu Settings	Window names, dialog box names, page names or buttons, combined proper nouns, functions referenced by the interface
Select `Referencing method A`	Predefined entry
Save your `<own IP address>`	User-defined entry
EVENT 52: COMMUNICATION	Display indicators (status, messages, warnings,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

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

[CTRL], [CTRL] + [S]	Key, shortcut
Table > Insert table	Navigation to menus/submenus (path specification)

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.



2.6.3 Mathematics and formulas

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

-	Subtraction
+	Addition
×	Multiplication
÷	Division
| |	Amount

2.7 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®, CiA®	CANopen® and CiA® are registered European Union trademarks of CAN in AUTOMATION e.V., Nuremberg, Germany.
Windows®, Windows® XP, Windows® 7, Windows® 10	Windows®, das Windows®-Logo, Windows® XP, Windows® 7 und Windows® 10 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.

09/2019 | ID 442637.01

3 General safety instructions

There are risks associated with the product described in this documentation that can be prevented by complying with the described warning and safety instructions as well as the included technical rules and regulations.

3.1 Directives and standards

The European directives and standards relevant for the product specified in this documentation can be taken from the directives and standards of the corresponding drive controller.

3.2 Qualified personnel

In order to be able to perform the tasks described in this documentation, the persons instructed to perform them must have the appropriate professional qualification and be able to assess the risks and residual hazards when handling the products. For this reason, all work on the products as well as their operation and disposal may be performed only by professionally qualified personnel.

Qualified personal are persons who have acquired authorization to perform these tasks either through training to become a specialist and/or instruction by specialists.

Furthermore, valid regulations, legal requirements, applicable basic rules, this documentation and the safety instructions included in it must be carefully read, understood and observed.

3.3 Intended use

As defined by DIN EN 50178, SD6 drive controllers are electrical devices operating as power electronics to control the flow of energy in high-voltage systems. They are intended solely for the operation of synchronous servo motors, asynchronous motors, linear motors or torque motors.

The connection of other electronic loads or operation outside applicable technical specifications constitutes improper use.

3.4 Operational environment and operation

The products are subject to sales restrictions in accordance with IEC 61800-3.

The products are not designed for use in a public low-voltage network that supplies residential areas. Radio-frequency interference can be expected if the products are used in this type of network.

The products are intended exclusively for installation in control cabinets with at least protection class IP54.

Always operate the products within the limits specified by the technical data.

The following applications are prohibited:

- Use in potentially explosive atmospheres
- Use in environments with harmful substances as specified by EN 60721, such as oils, acids, gases, vapors, dust and radiation

Implementation of the following applications is permitted only after approval from STOBER:

- Use in non-stationary applications
- The use of active components (drive controllers, supply modules, energy recovery units or discharge units) from third-party manufacturers

The products are designed exclusively for operation in TN networks.

3.5 Disposal

Observe the current national and regional regulations when disposing of the product! Dispose of the individual product parts depending on their properties, e.g. as:

- Electronic waste (circuit boards)
- Plastic
- Sheet metal
- Copper
- Aluminum
- Battery

09/2019 | ID 442637.01

4 Network structure

In the standard version, CAN operates with a linear topology in which all nodes (master and slaves) are interconnected by 3 shielded lines: CAN-High and CAN-Low. A line is also provided for the reference ground between the nodes.

To prevent data overlapping through signal reflections at the two bus ends and thus reduce susceptibility to faults substantially, terminal resistances of 120 Ω are connected between CAN-H and CAN-L.

The following graphic abstracts a CANopen network consisting of a controller as well as multiple SD6 drive controllers as bus nodes.

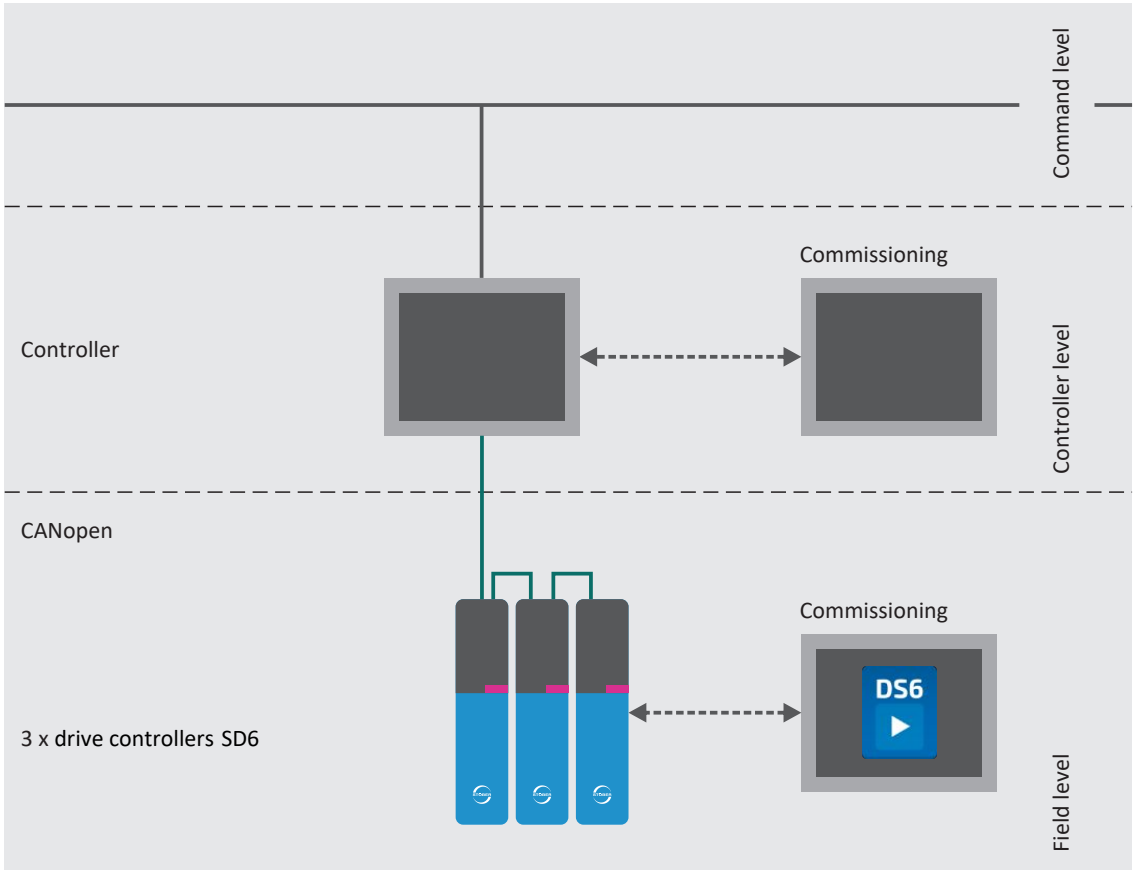


Fig. 1: CANopen – network structure, using the SD6 series as an example

5 CA6 communication module

The drive controllers of the SD6 series are connected to a controller by CANopen using CA6 communication modules, which provide the necessary fieldbus interfaces.

CA6 fully corresponds to the CANopen standard and enables direct access to all relevant drive parameters and functions.

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.

⚠ DANGER!

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

- Always switch off all power supply voltage before working on the devices!
- Note the discharge time of the DC link capacitors in the general technical data. 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 controllers 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.

09/2019 | ID 442637.01

6 Connection

To enable the connection of individual drive controllers with each other or with the controller, the CA6 communication module provides its own interface in the form of a 9-pin D-sub connector for the connection to the CAN bus system.

6.1 Selecting suitable lines

CAN cables

There are CAN bus lines 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-Low and reference ground.

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

- Characteristic impedance: 95 – 140 Ω
- Maximum operating capacitance: 60 nF/km
- Conductor resistance: 70 mΩ/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 line; connector housings with 2 cable entries are advantageous for looping through.

Shielding

A suitable shield is indispensable, particularly for high transmission rates. The shielding of the CAN cable is mounted under the connector's strain relief device; consequently, the shielding is connected to the drive controller by the connector housing and the D-sub connector. Ensure continuous shielding over the entire bus length.

6.2 Terminal resistances

The following graphic abstracts a CANopen network consisting of a controller as master as well as multiple SD6 drive controllers as bus nodes, i.e. slaves. Note that a terminal resistance has to be provided at the first and last node. In the example, one of the two required terminal resistances is connected to the controller and the second is connected to the last participating SD6 (X200: slider at "ON"). Connection of the terminal resistances is required even for very short lines.

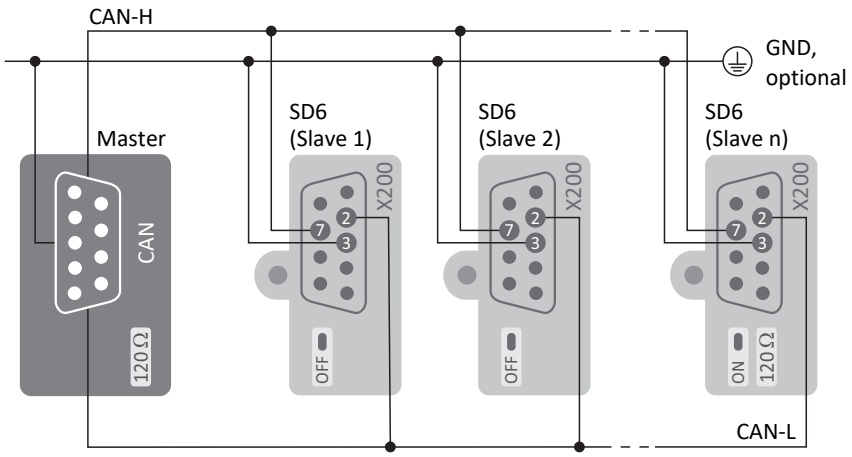


Fig. 2: Connecting the terminal resistances in a CANopen network

6.3 Transmission rate and line length

The maximum length of a CAN bus line depends on the transmission rate, i.e. baud rate.

The following table shows the maximum permitted cable lengths (over the entire length of the bus) in combination with the respective possible baud rates.

Note that the sensitivity of serial bus systems increases as the transmission rate increases.

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 ± 60 nF/km
1000	< 10, only with special cable ± 60 nF/km

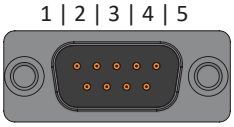
Tab. 1: Transmission rate and max. CAN bus length

Information

All drive controller that are connected over a CAN network must have the same baud rate.

6.4 X200: CANopen

The CA6 communication module provides a 9-pin D-sub connector for connecting the drive controllers to each other.

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

Tab. 2: X200 connection description

6.5 LED display

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

6.5.1 CAN status

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 CAN diagnostic.

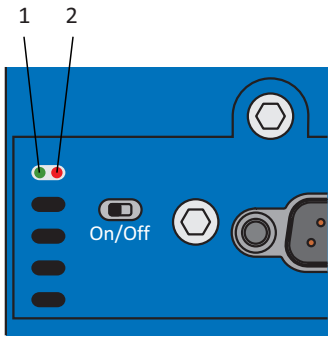


Fig. 3: LEDs for the CAN status

1 Green: Run
2 Red: Error

Green LED	Behavior	NMT state	Description
	Off	Init	No error, no warning
	Flashing	Pre-operational	The drive controller is ready for parameterization for preparing the actual operation.
	1x flash	Stopped	All communication activities have stopped.
	On	Operational	The CAN bus is activated, all services are in operation.

Tab. 3: Meaning of the green LED (run)

Red LED	Behavior	Error	Description
	Off	No error, no warning	No error
	1x flash	Communication error in the operational state (Warning level)	Check the bus cables, shielding and compliance with the CAN specifications regarding this.
	2x flash	Node guard event	Check the node guard function of the controller.
	3x flash	SYNC error	Check the SYNC configuration of the controller.
	On	Bus off	The SD6 in question is no longer participating in the CAN communication. Check the baud rate and bus cables and switch the drive controller off and on again.

Tab. 4: Meaning of the red LED (error)

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 Program interfaces

The following chapters include an overview of the program interfaces for the described software components.

7.1.1 DS6: Structure of the program interface

The DriveControlSuite commissioning software (DS6) offers a graphic interface that you can use to project, parameterize and start up your drive controller quickly and efficiently.

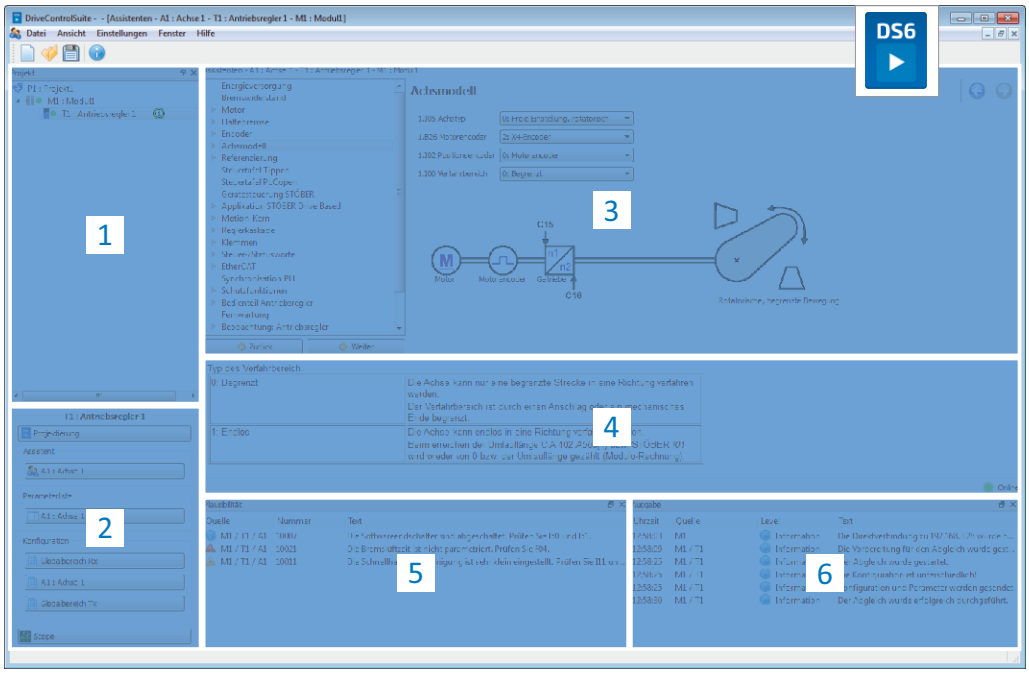


Fig. 4: DS6 – Program interface

1 Project tree
2 Project menu
3 Workspace
4 Parameter description
5 Parameter check
6 Messages

7.1.1.1 Individualized workspace

The project tree (1) and project menu (2) are connected and, like the parameter check and messages (5, 6), can also be docked at the left, right or bottom edge of the screen. This program window can also be displayed or hidden using the View menu.

The workspace (3) and parameter description (4) are also connected to each other and always positioned in the middle. Both areas can be minimized or maximized.

7.1.1.2 Navigation using sensitive circuit diagrams

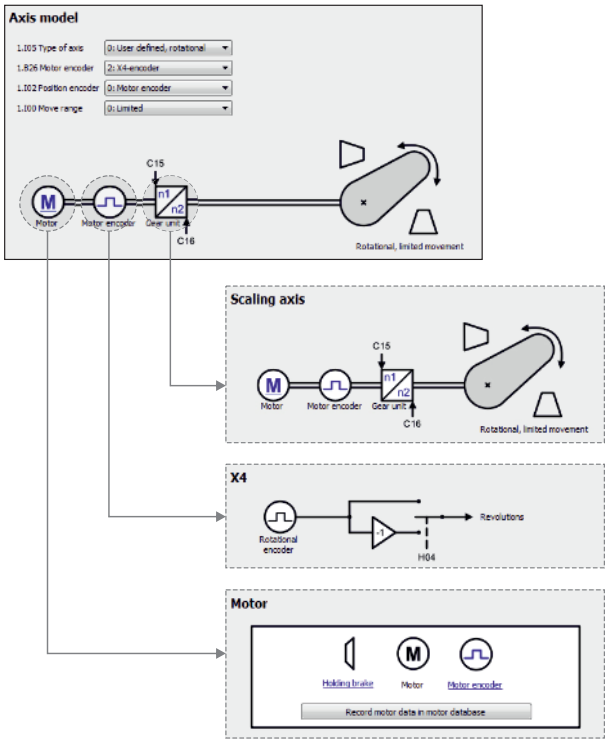


Fig. 5: DriveControlSuite: Navigation using text links and symbols

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 a drive controller to your specific drive model and your work environment. In addition, parameters visualize the current actual values (actual velocity, actual torque, etc.) and trigger general actions like Save values, Test phase etc.

09/2019 | ID 442637.01

7.2.1 Parameter groups

Parameters are assigned to individual groups by topic. The 6th generation of STOBER 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 (depending on the respective application)
H	Encoders
I	Motion (all motion settings)
J	Motion blocks
K	Control panel
L	Technology – Part 2 (depending on the respective application)
M	Profile
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. 5: 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.

Type	Style		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 word, unsigned)	-21474836480 – 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 word, unsigned)	0 – 4294967295
REAL32	Floating-point number	Floating point (1 word, unsigned)	In accordance with ANSI/ IEEE 754
STR8	Text	Text (8 characters)	–
STR16	Text	Text (16 characters)	–
STR80	Text	Text (80 characters)	–

Tab. 6: Parameters – Data types, styles, 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 the associated units and decimal places
  Example: Actual and set values for velocity, acceleration, deceleration, jerk
- Parameter address
  Corresponds to the storage location of another parameter
  Example: Indirect read sources for analog and digital outputs and for fieldbus mapping
- Text
  Outputs or messages
  Example: Displays (SD6)

09/2019 | ID 442637.01

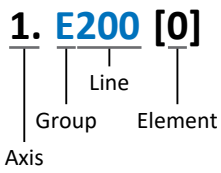
7.2.3 Parameter types

The following types of parameters are differentiated.

- **Simple parameters**
  Consist of one group and one line with a defined value.
  Example:
  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.
  Example:
  A10 Access level
  A10[0] access level: Value = Access level via operating unit
  A10[2] access level: Value = Access level via CANopen and EtherCAT
  A10[4] access level: Value = Access level via PROFINET
- **Structure parameters**
  Consist of a group, a line and multiple sequential (listed) elements, which can have different properties and different values.
  Example:
  A00 Save values
  A00[0] Start: Value = Start action
  A00[1] Progress: Value = Display action progress
  A00[2] Result: Value = Display action result

7.2.4 Parameter structure

Every parameter has specific coordinates with the following pattern.



- Axis
  The axis to which a parameter is assigned in the case of multiple axes (optional).
- Group
  The thematic group to which a parameter belongs.
- Line
  Distinguishes the parameters within a parameter group.
- Element
  Elements of an array or structure parameter (optional).

7.2.5 Parameter visibility

The visibility of a parameter depends on the access level defined in the software, the dependency of other parameters, the selected application and the version of the associated firmware.

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. 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

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])

Hiding functions

Hiding functions are used to hide parameters with regard to their logical relationships to other option modules or parameters.

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.

Applications

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

Firmware

A newer version of the firmware may introduce new parameters. Parameters that have been configured for files of an older firmware function may not be visible in newer versions. In such cases, the respective parameter description includes a corresponding note.

7.3 Power-loss protected storage

All project configurations, parameterizations and related changes to parameter values are in effect after the transmission to the drive controller, but are not yet stored in non-volatile memory.

You save the data using the Save values function in parameter A00 (Project menu > Wizards area > Projected axis > Save values wizard).

As an alternative, save the data using the save button on the display.

Only then is the data stored with power-loss protection.

8 Commissioning

Are you looking to operate multiple SD6 drive controllers combined with a controller over a CAN network?

The following sections describe the commissioning process using the DriveControlSuite commissioning software.

We put forward 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 version 6.4-D or later
- DriveControlSuite commissioning software in version 6.4-D or later
- Controller

Commissioning is divided into the following steps:

1. DriveControlSuite
   The drive controllers communication modules combined with the CiA 402 device control are activated separately in DriveControlSuite for each drive controller participating in the bus system. Similarly, associated CANopen-specific settings, such as parameterization and the assignment of process data channels for data transmission, are configured for each drive controller.
2. CAN system
   Then you have to set up your CAN network and put it into operation using the controller.

8.1 DS6: Configuring the drive controller

Project and configure all drive controllers for your drive system in DriveControlSuite (see DS6: Structure of the program interface [▶ 17]).

Information

Always perform the steps included in the following chapters in the specified order!

Some parameters of the DriveControlSuite 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

Creating a new project

1. Start DriveControlSuite.
2. Click Create new project.

⇨ The project configuration window opens and the Drive controller button is active.

Projecting the drive controller

1. Properties tab:
   Establish the connection 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. using 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.
   Process data Rx: Select CANopenRx+Tx for transmitting the CANopen process data.
   Process data Tx: Select No transmission.

09/2019 | ID 442637.01

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 have to configure the safety technology in accordance with the commissioning steps outlined in the corresponding manual in the next step.

8.1.2 Parameterizing global CANopen settings

✓ As part of drive controller and axis project configuration, you have projected the CiA 402 device control with the CANopen Rx and CANopen Tx process data as well as the CiA 402 application.

1. Highlight the relevant drive controller in the project tree and click on the first projected axis in the project menu > Wizard area.
2. Select the CANopen wizard.
3. A213 Fieldbus scaling:
   Leave the default setting at 1: Native (values are passed unchanged).
4. A82 CAN baud rate:
   Define the transmission rate as a function of the bus length. Permitted value range: 10 – 1000 kbps
   Default: 250 kbps
5. A83 Bus address:
   Indicate the bus address (node ID) of the drive controller. Permitted address range: 0 – 125
   Default: 1
6. A203 Guard time:
   Define the time between 2 successive remote requests from the controller. Permitted value range: 0 – 4000 ms (0 = disabled)
   Default: 2 ms.
   If A203 is enabled, A204 must also be configured.
   Note that simultaneous use of the guard time and heartbeat mechanisms is not permitted! You can disable both control mechanisms or enable one of the two.

7. A204 Life time factor:
   Indicate how high the number of missing remote requests from the controller has to be before the drive controller switches to the Fault state. Permitted value range: 0 – 255
   Default: 0 (no guard time monitoring).

8. A210 Producer heartbeat time:
   Define the time between 2 successive heartbeat messages from the drive controller. Permitted value range: 0 – 65535 ms (0 = disabled)
   Default: 0 ms
   Note that simultaneous use of the guard time and heartbeat mechanisms is not permitted! You can disable both control mechanisms or enable one of the two.

8.1.3 Configuring PDO transmission

The CA6 communication module allows for a maximum of 2 independent PDO channels to be operated simultaneously per transmission and reception direction. Each of these PDO channels includes a PDO with a maximum of 6 elements to be transmitted in a defined sequence.
The current default settings are recommendations from STÖBER Antriebstechnik GmbH + Co. KG for ensuring error-free communication between the controller and drive controller.
Assignment of the COB-IDs corresponds to the principle of predefined connection sets. Do not change this unless you are operating outside the standard parameters.
Due to the unlimited transmission rate of the CAN bus, 2 independent channels are enabled in the CANopen PDO default settings.
The STOBER drive controllers support a flexible assignment of the elements to be transmitted to the individual PDOs.

8.1.3.1 Adapting the receive PDO

✓ You have configured the global CANopen settings.

1. Highlight the relevant drive controller in the project tree and click on the first projected axis in the project menu > Wizard area.

2. Select the wizard CANopen > Receive process data RxPDO.

3. Check the default settings and/or configure the process data according to your requirements.
   A221[0] COB-ID:
   COB-ID of the first receive PDO that is transmitted over this channel.
   A221[1] Transmission type:
   Transmission type of the receive PDO.
   A225[0] – A225[5] 1. mapped Parameter – 6. mapped Parameter:
   Configure the parameters that will be transmitted over the first receive PDO of this channel. The respective position (1st – 6th) provides information about the transmission sequence.

4. Resulting data length:
   Overall length of the parameters of the first receive PDO 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.

5. Repeat step 3 for all of the receive PDO channels.

6. Check the activity status of the channels and enable/disable these, where appropriate, by clicking the green status beacon of a channel.

8.1.3.2 Adapting the transmit PDO

✓ You have configured the global CANopen settings.

1. Highlight the relevant drive controller in the project tree and click on the first projected axis in the project menu > Wizard area.
2. Select the wizard CANopen > Transmit process data TxPDO.
3. Check the default settings and/or configure the process data according to your requirements.
   A229[0] COB-ID:
   COB-ID of the first transmit PDO that is transmitted over this channel.
   A229[1] Transmission type:
   Transmission type of the transmit PDO.
   A229[2] Inhibit time:
   Minimum time between the sending of 2 PDO elements.
   A229[3] Event timer
   Event timer of the first receive process data channel receive PDO for defining the time delay after which a bus node sends a transmit PDO for an asynchronous transmission type, even if no new receive PDO is received.
   A233[0] – A233[5] 1. mapped Parameter – 6. mapped Parameter:
   Configure the parameters that will be transmitted over the first transmit PDO of this channel. The respective position (1st – 6th) provides information about the associated transmission sequence.
4. Resulting data length:
   Overall length of the elements of the first transmit PDO 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.
5. Repeat step 3 for all of the transmit PDO channels that keep the "enabled" status.
6. Check the activity status of the channels and enable/disable these, where appropriate, by clicking the green status beacon of a channel.
7. Once you have recorded all drive controllers of your bus system in the current project, proceed by saving the project configuration and transmitting it to the drive controllers of your bus system.

8.1.4 Transmitting and saving a project configuration

Transmitting a project configuration

✓ The drive controllers are ready for operation.

1. In the project tree, highlight the module under which you have recorded your drive controllers and click Assignment and live firmware update in the project menu.
   ⇨ The Adding a connection window opens. All drive controllers found via IPv4 limited broadcast are displayed.
2. Direct connection tab > IP address column:
   Activate the IP address in question or activate all listed using the context menu. Confirm your selection with OK.
   ⇨ The Assignment and live firmware update window opens. All drive controllers connected through the previously selected IP addresses are displayed.
3. Select the drive controller to which you would like to transfer the project configuration. Change the selection of transfer type from Read to Send.
4. Change the selection Create new drive controller:
   Select the project 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 a project configuration.
6. Online tab:
   Click on Establish online connections.

⇨ The project configurations are transferred to the drive controllers.

Information

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

Prerequisites for finding a drive controller in the network:

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

Saving a project configuration

1. Click on Save values (A00) ... in the Assignment and live firmware update window.
   ⇨ The Save values (A00) window opens.
2. Click on Start action.
   ⇨ The project configuration is saved.
3. Since the configuration only takes effect after the drive controller is restarted, click on Restart (A09) ... in the Assignment and live firmware update window.
   ⇨ The fieldbus communication and connection to DriveControlSuite are interrupted and the drive controllers restart.

8.2 Putting the CANopen system into operation

Afterwards, set up your CAN network and put it into operation using the controller.

09/2019 | ID 442637.01

9 More about CANopen?

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

9.1 CAN bus and CANopen

CAN and communication in CAN bus

The "Controller Area Network" (CAN) bus system is a real-time fieldbus for serial data transmission.
The individual tasks of a CAN bus are defined in what are called the layers. The actual CAN protocol corresponds to the "Data Link Layer" (Layer 2) of the ISO/OSI reference model.
At this level, ordinary or manufacturer-specific CAN networks can be built and nodes (slaves) of different manufacturers can be interconnected without any trouble. In the case of a shared line, the data flow itself is controlled by a protocol that controls the access rights of the individual nodes—regardless of the manufacturer.

CAN provides many advantages: it serves a broad user base thanks to its real-time and multi-master capabilities, high fault resistance and good availability. CAN is used primarily where high transmission rates and a simple, cost-effective installation are required.

What is CANopen?

CANopen is the CAN-based communication protocol and the open standard for networking controllers, drives, encoders or sensors, that is, all nodes in a CAN bus system. Both the functions of the intercommunicating nodes and the foundational communication mechanisms are defined in CANopen. CANopen expands upon CAN with what is called the CAN application layer (CAL = layer 7 of the ISO/OSI reference model). CANopen has been maintained by the organization CiA (CAN in Automation e.v.) since 1996 and is codified as European standard EN 50325-4.

9.1.1 CANopen – Communication

CANopen stipulates that all bus nodes are to be bundled into separate classes with the same properties. Each of these classes has its own "device profile," in which specific features, such as standard configurations, runtime behavior or error handling, are defined along with all associated parameters.
All device profiles are organized in what is called an "object directory." All device profiles are based on their own communication profile. This is where it is specified how the devices connected in a CAN-based network are addressed and which mechanisms are used for exchanging process data or large amounts of data.

9.1.2 Object directory

The object directory of each bus node describes its complete range of functions in the form of different data (standardized data types, objects of the CANopen communication profile and device profile, manufacturer-specific objects, etc.).

Service Data Objects (SDOs) can access the entries of the object directory through an index (row address, 16-bit) and an additional subindex (column address, 8-bit).
This index is structured as a table and subdivided into logical segments.

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. 7: Object directory – Structure

9.1.3 Network structure

The topology of a CAN network is usually linear. In a CANopen network, up to 127 nodes can be addressed in theory, while 64 bus nodes are possible physically.
With respect to network expansion, a length of 1 km is possible for a transmission rate of 50 kBaud, for example, and a network expansion of 25 m is realistic for a transmission rate of 1 MBaud. In general, 9 transmission rates are available (10 kBaud – 1 MBaud).

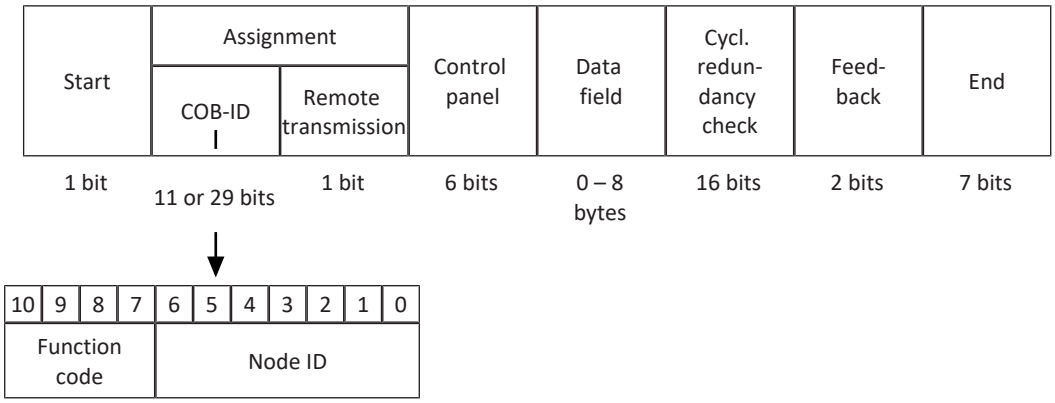
9.2 CAN message

Communication using the CAN protocol is carried out in the form of messages (frames). A frame is the structure of a standardized message "container." The CAN bus system distinguishes between 4 different frames:

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

Structure of a standard CAN message

A standard CAN message is structured as follows.



9.2.1 COB-ID

The COB-ID (Communication Object Identifier) determines the priority of a message in the network (low COB-ID = high priority). COB-IDs can be changed by accessing SDOs.

In a standard message (standard CAN frame), the COB-ID is defined by 11 bits, which allows for 2,032 different logical addresses to be coded and 2,048 different pieces of information to be sent. Since this number proved to be insufficient, the corresponding CAN specification later expanded the identifier to 29 bits, which enabled the transmission of 536,870,912 different pieces of information (extended CAN frame).

Assigning the COB-IDs

There are 2 mechanisms for assigning COB-IDs: predefined connection set and dynamic distribution.

9.2.1.1 Predefined connection set

In factory condition, the communication parameters of the drive controllers are configured according to the principle of the predefined connection set.
This pre-assignment makes it easy to commission a "standard" CAN network with a controller and up to 127 drive controllers.

If the SDO channels 2, 3 and 4 are enabled, a maximum of 31 drive controllers are connected.
This method is recommended for the majority of applications.

A COB-ID consists of 11 bits, where bits 7 – 10 are assigned the "code" function (4 bits) and bits 0 – 6 are assigned the node ID (7 bits).
A special function code is defined for each type of CAN message. In the case of peer-to-peer objects, the function code and node ID are added. This results in the COB-ID for each individual message of a bus node.
The predefined connection set is enabled when the values of the respective function codes are entered in the communication parameters. The associated node IDs are added afterwards.
The following tables contain the COB-IDs of the communication objects based on 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

Note that the node IDs of the node in question must be between 1 and 31, and for the disabled SDO channels SDO 2, 3 and 4 must be no greater than 127.

Object	Function code	COB-ID			Communication parameter (index)		Priority
					CAN	SD6	
EMERGENCY	0001	80 hex + node ID	81 hex – FF hex	129 – 255	1014 hex, 1015 hex	A207	High
TxPDO1	0011	180 hex + node ID	181hex – 1FFhex	385 – 5115	1800 hex	A229[0]	
RxPDO1	0100	200 hex + node ID	201hex – 27Fhex	513 – 639	1400 hex	A221[0]	
TxPDO2	0101	280 hex + node ID	281hex – 2FFhex	641 – 767	1801 hex	A230[0]	
RxPDO2	0110	300 hex + node ID	301hex – 37Fhex	769 – 895	1401 hex	A222[0]	
TxPDO3	0111	380 hex + node ID	381hex – 3FFhex	897 – 1023	1802 hex	A231[0]	
RxPDO3	1000	400 hex + node ID	401hex – 47Fhex	1025 – 1151	1402 hex	A223[0]	
TxPDO4	1001	480 hex + node ID	481hex – 4FFhex	1153 – 1279	1803 hex	A232[0]	
RxPDO4	1010	500 hex + node ID	501hex – 5FFhex	1281 – 1407	1403 hex	A224[0]	
TxSDO1	1011	580 hex + node ID	581hex – 59Fhex	1409 – 1439	1200 hex	—	
RxSDO1	1100	600 hex + node ID	601hex – 61Fhex	1537 – 1567	1200 hex	—	
TxSDO2	1011	5A0 hex + node ID	5A1hex – 5BFhex	1441 – 1471	1201 hex	A218[1]	
RxSDO2	1100	620 hex + node ID	621hex – 63Fhex	1539 – 1599	1201 hex	A218[0]	
TxSDO3	1011	5C0 hex + node ID	5C1hex – 5DFhex	1473 – 1503	1202 hex	A219[1]	
RxSDO3	1100	640 hex + node ID	641hex – 65Fhex	1601 – 1631	1202 hex	A219[0]	
TxSDO4	1011	5E0 hex + node ID	5E1hex – 5FFhex	1505 – 1535	1203 hex	A220[1]	
RxSDO4	1100	660 hex + node ID	661hex – 67Fhex	1633 – 1663	1203 hex	A220[0]	Low
ERROR CONTROL	1110	700 hex + node ID	701hex – 77Fhex	1793 – 1919	1016 hex, 1017 hex	—	

9.2.1.2 Dynamic distribution

In special application cases, it may be necessary to deviate from the predefined assignment of COB-IDs to communication objects and use the dynamic distribution service.
In this process, the COB-IDs are dynamically distributed by a distributor (DBT), for which a process map of all registered bus nodes in the network must be available at all times. The process provides a variety of options for optimizing a more complex CAN network with different nodes and tasks.
In general, the following objects can be changed:

- PDO 1 – 4 (both directions of transmission)
- SDO 2 – 4 (both directions of transmission)
- SYNC
- EMERGENCY

Disabling the predefined connection set

You disable the predefined connection set by changing the preset values of the function codes in the respective communication parameters. This is how you prevent the addition of node IDs to the function codes and, in conjunction, the formation of resulting COB-IDs.

Enabling dynamic distribution

If a drive controller is in pre-operational mode, the controller can overwrite the COB-IDs. To finally enable the COB-IDs, PDOs have to be re-initialized by a change to the "Start" NMT state, and SDOs have to be re-initialized by a change to the "Reset Communication" NMT state. If the changes need to remain in effect even after the supply voltage is shut off, you have to store these in the respective drive controllers; that is, set A00[0] Start to 1: Active (= Save values). At the next start, the drive controller checks whether the values in the associated communication parameters have been set to default. If this is the case, the predefined connection set mechanism takes effect. If no default values have been determined, the DBT principle applies.

Example

You would like to enable the second SDO channel and connect fewer than 64 nodes via the CAN bus.
After the drive controller has been started with bus address 1, only SDO channel 1 is enabled, RxSDO1 has COB-ID 601 hex and TxSDO1 has COB-ID 581 hex.
Use the communication parameter with Index/Subindex 1201/1 to change the 2nd Server SDO Parameter / COB-ID Client->Server to the value 80000641 hex and use the parameter 1201/2 to change the 2nd Server SDO Parameter / COB-ID Server-> to the value 800005C1 hex.
Changing to the "Reset Communication" NMT state and then changing to the "Pre-operational" state enables both the Rx and the Tx direction of the SDO channel. At the same time, the COB-IDs are set to 641 hex for receive SDO (RxSDO) and 5C1 hex for transmit SDO (TxSDO).
By saving these values, this SDO channel is immediately enabled the next time the unit is switched on.
To prevent the IDs from being used for SDO channel 1 by default by another node with node ID 65 (41 hex), no nodes are allowed to get a node ID greater than or equal to 65.

09/2019 | ID 442637.01

9.3 Communication objects

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

The following communication objects are of key importance for data transmission as part of CANopen:

- Service Data Objects (SDOs)
  ... grant access to the object directory, enabling a device configuration.
- Process Data Objects (PDOs)
  ... transmit real-time data such as set/actual values, control commands or status information based on events or time, cyclically or upon request (RTR).
- Network Management Objects (NMT)
  ... control the state machines of the bus nodes and monitor them.
- Synchronization Objects (SYNC)
  ... create typically time-based sorting. Messages can be received or sent with respect to a SYNC object.
- Time Stamp Objects (TIME STAMP)
  ... are for transmitting the current time (local time, not any defined time zone). The CA6 communication module does not support this object.
- Emergency Objects (EMCY)
  ... are triggered by device-internal errors and sent two times, i.e. when the error in question occurs and when it is absent. The messages contain error causes.
- Monitoring objects (ERROR CONTROL)
  ... monitor the CAN network. Some of the monitoring objects are NODE and LIFE GUARDING messages.

NMT, SYNC and TIME STAMP are among the broadcast objects that are sent by the controller to all nodes simultaneously.
SDO, PDO, EMCY and ERROR CONTROL are peer-to-peer objects and can be exchanged in both directions.

9.3.1 Process data object – PDO

Process data objects normally transmit fast, cyclical data such as target positions, travel velocities or acceleration information and are generally used for exchanging data in real time.

They also enable simultaneous access to multiple drive parameters.

The specific communication elements that are sent and received in a specific PDO can be freely selected.
By default, however, these are prepopulated by the predefined connection set.
In a PDO transmission, contents are transmitted directly instead of objects being addressed. The corresponding data content is stored in the object directory of each bus node.

PDOs have a maximum length of 8 bytes.
They are generally transmitted by what are called process data channels (PDO channels) with high priority.

The assignment and priority of PDO messages are defined by the COB-ID.
From the perspective of the respective bus nodes, CANopen differentiates between receive PDO (= RxPDO) and transmit PDO (= TxPDO).

PDOs are sent only when the communication is in the "Operational" NMT state.

9.3.1.1 PDO mapping

In PDO mapping, the communication parameters are mapped from the object directory of a CAN bus node to PDO and associated transmission channels. This determines which communication objects (elements) are transmitted over a PDO channel.

The CA6 communication module allows for a maximum of 2 independent PDO channels to be operated simultaneously in each direction of transmission. Each of these PDO channels includes a PDO with a maximum of 6 elements to be transmitted in a defined sequence.
Assignment of the associated COB-IDs corresponds to the principle of predefined connection sets.

The STOBER drive controllers support a flexible assignment of the elements to be transmitted to the individual PDOs.
If you want to change the configuration of the predefined PDO mapping, note that the overall length of the elements to be transmitted per channel must not exceed 8 bytes.

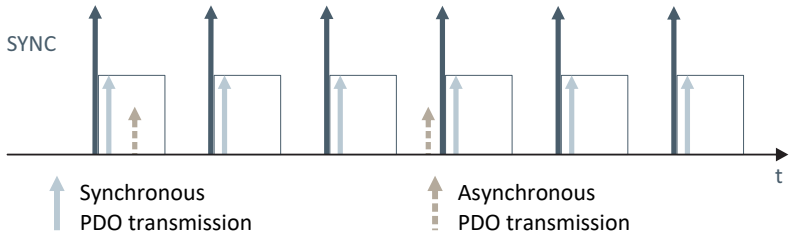
9.3.1.2 Transmission type

The transmission type refers to the PDO transmission type and defines how the sending of a PDO message (in relation to a SYNC object) is triggered or how received PDO messages are handled.

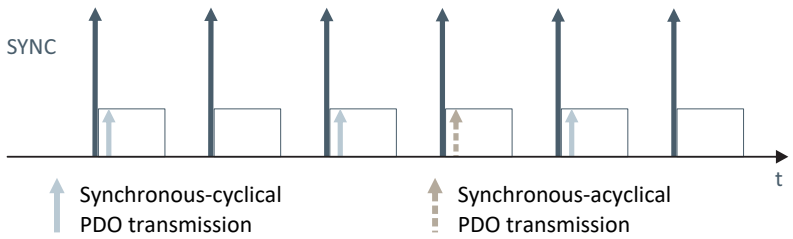
In general, PDOs can be transmitted over 5 different types, including some possible combinations.

- Asynchronous
  Asynchronous PDO messages have no relationship with SYNC objects, i.e. they are not transmitted in sync with a SYNC object. A receive PDO is applied immediately after it is received; similarly, a transmit PDO is sent immediately after it is triggered (by an event timer or a send request).
- Synchronous
  Synchronous PDO messages are processed with chronological relationships with SYNC objects, i.e. periodically or in sync with a SYNC object. A configured transmission type defines the number of SYNC objects after which PDO messages are transmitted. For example, one receive PDO can be applied for every fifth SYNC object.
- Acyclical
  Acyclical PDO messages are transmitted in sync with a SYNC object, but only if an internal event has occurred first.
- Cyclical
  Cyclical PDO messages are transmitted in sync with a defined number of SYNC objects.
- RTR (Remote Transmission Requests)
  For this transmission type, PDOs are sent only upon request from other bus nodes.

Synchronous and asynchronous PDO transmission



Synchronous-cyclical and synchronous-acyclical PDO transmission



The respective PDO transmission type is defined by the entry of a specified value in the corresponding transmission type parameters. Each channel and each direction of transmission has its own transmission type parameter.

Value	Cyclical	Acyclical	Synchronous	Asynchronous	Description
0	–	Yes	Yes	–	Receive PDOs are applied with the next SYNC object, transmit PDOs are sent with the next SYNC object
1 – 240	Yes	–	Yes	–	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	–	–	–	Yes	Receive PDOs are applied upon being received and a transmit PDO is sent immediately afterwards
255	–	–	–	Yes	Reserved

9.3.1.3 Inhibit time

The inhibit time defines the minimum time that must be waited when 2 identical PDO messages will be sent on one channel. It must be defined separately for each channel.

9.3.1.4 Event timer

The event time defines the interval at which a bus node sends a transmit PDO, even if no new data exists.

The event timer applies only to asynchronous transmission types. A separate event timer exists for each transmit PDO channel. The timer is disabled if the entered value is zero.

9.3.2 Service data object – SDO

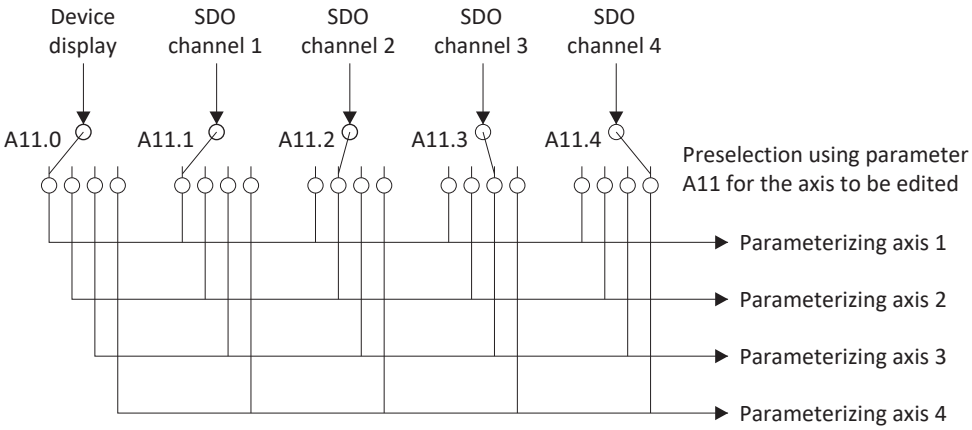
Service data objects are used for transmitting data that is not critical with respect to time and, in general, enables each bus node to be configured.

SDOs directly access the entries in the object list of a node and thus help with the parameterization of object directory entries.

An SDO transmission always consists of at least 2 messages:
A controller usually starts an order with a receive SDO message. Within this order, the controller uses index and subindex to choose a communication object. The object directory of the drive controller contains a list of all achievable parameters that are searched and processed in accordance with the sent order. The drive controller then acknowledges with a corresponding transmit SDO message.

The drive controllers of the 6th generation have 4 independent channels for transmitting service data objects.
The first SDO channel is always enabled. Associated COB-IDs "600 hex + node ID" and "580 hex + node ID" are unchangeable.
The COB-IDs of the remaining channels can be reselected through corresponding parameterization or the channels can be disabled. They are disabled by default.

The drive controllers enable the operation of up to 4 axes. For each axis, exactly one SDO channel can be used.
The axes are not addressed by the index and subindex, but by parameter A11 for each SDO channel.



In general, SDOs can be used for transmitting data of any length. Here, CANopen distinguishes between 2 SDO transmission types:

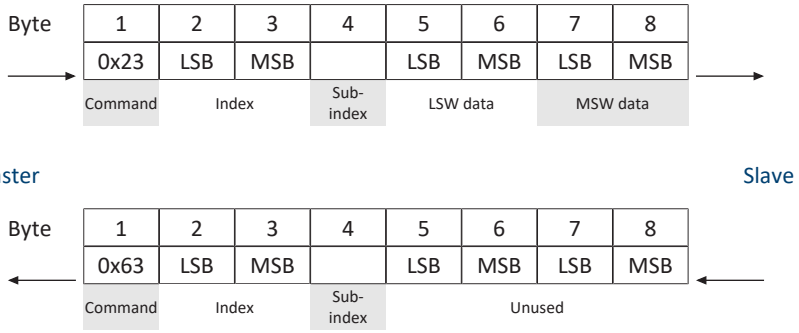
- Expedited transfer
  ... for data transmission of up to 4 bytes in a single message.
- Segmented transfer
  ... for data transmission of over 4 bytes, which are distributed across multiple messages, i.e. segmented.

9.3.2.1 Expedited transfer

The simplified, expedited SDO transfer is used for transferring all parameters with a data type of maximum of 4 bytes. In this type of transmission, 4 data bytes are sent in a single frame. 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).

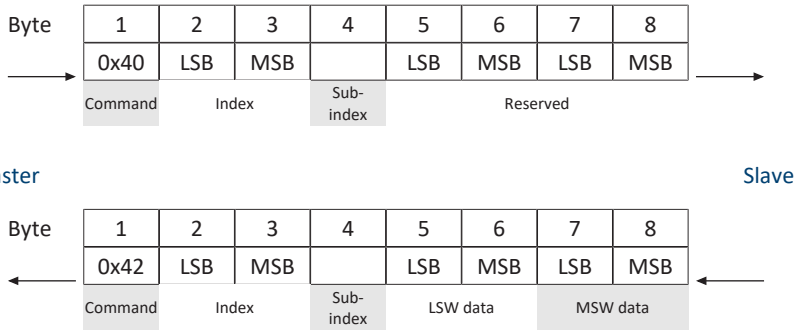
Write parameter (Initiate Domain Download Request)

The 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 slave.



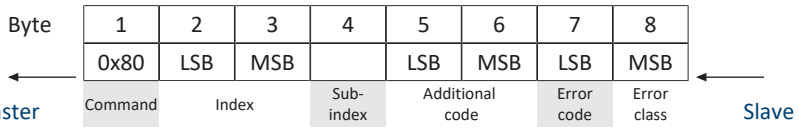
Read parameter (Initiate Domain Upload Request)

The 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 slave.



Error message (Abort Domain Transfer)

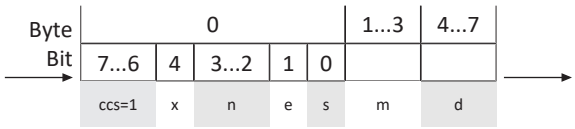
A slave provides a negative response to the write parameter or read parameter requests using an Abort Domain Transfer (see SDO transmission: Error codes).



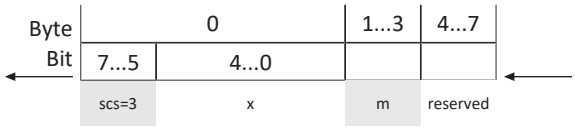
9.3.2.2 Segmented transfer

For a segmented transfer, data is split into segments. The total number of bytes to be transmitted are sent in what is called a first initiate message ("Initiate SDO Download"); this is followed by the remaining segments ("Download SDO Segment"), each with 7 bytes.

Initiate SDO Download Protocol

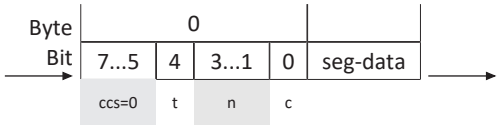


Master Slave

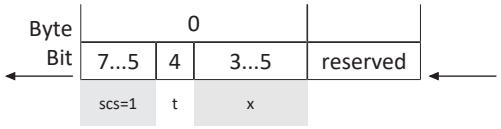


ccs	Client command specifier	1 = Initiate download request
scs	Server command specifier	3 = Initiate download response
n	Number of bytes	Number of bytes in "Data" that contain no usable data. If e = 0 , s = 1, then n = valid, otherwise n = 0
e	Transfer type	▪ 0 = Normal transfer ▪ 1 = Expedited transfer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

Download SDO Segment Protocol

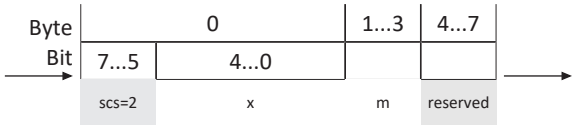


Master Slave

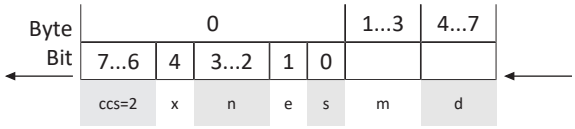


ccs	Client command specifier	0 = Download segment request
scs	Server command specifier	1 = Download segment response
n	Number of bytes	Number of bytes in "Segment data" that contain no usable data. n = 0: No information about unused data
seg-data	Segment data	7 bytes of usable data
c	Continue	▪ 0 = More segments follow ▪ 1 = Last segment
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

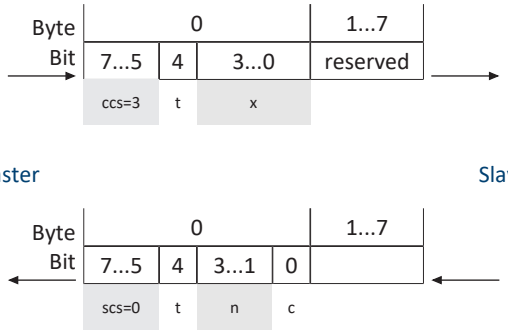


Master Slave



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	▪ 0 = Normal transfer ▪ 1 = Expedited transfer
s	Size indicator	▪ 0 = Not displayed ▪ 1 = Displayed
m	Multiplexor	= Index + subindex
d	Data	▪ If e = 0, s = 0, then d = reserved ▪ If e = 0, s = 1, then d = number of bytes to be transmitted ▪ If e = 1, s = 1, then d = 4-n
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	▪ 0 = More segments follow ▪ 1 = Last segment
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: IDDReq:	21	idx	x	10 00 00 00	(ccs = 1, e = 0 = normal, s = 1 -> data = no. of bytes)
Server: IDDRes:	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)

9.3.3 Network management object – NMT

Network management objects are responsible for initialization, configuration and error handling in the CAN network. In addition to the node ID, each node of a CANopen network has an NMT state machine (Network Management State Machine), which can be in various states.

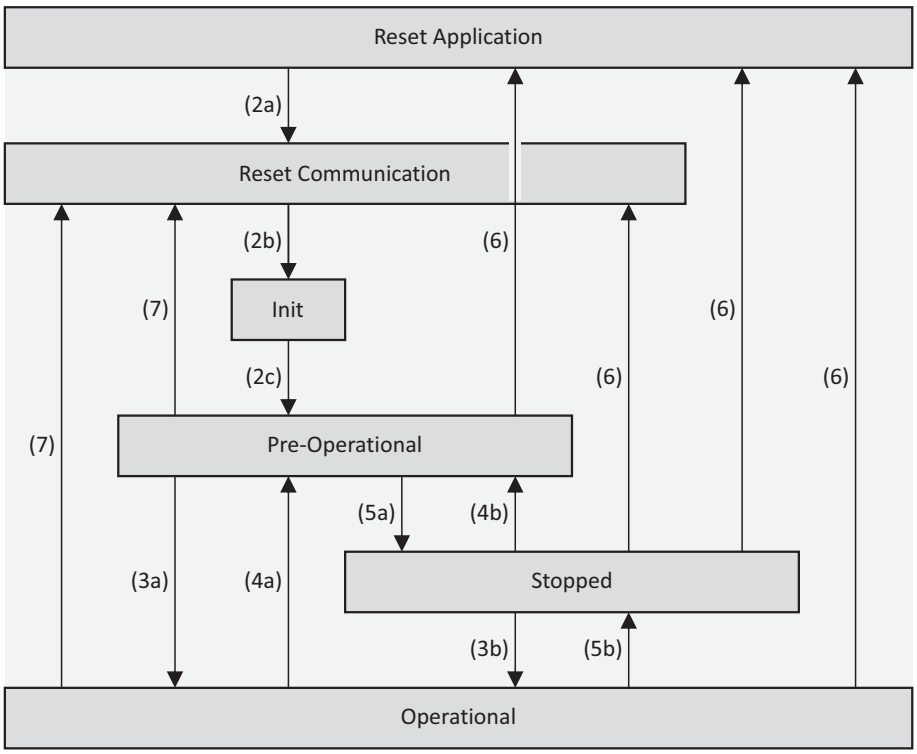


Fig. 6: Network Management Object – NMT – Potential operating states and state changes

Operating states

- Reset application
  Configuration of the drive controller starts; the stored values are loaded.
- Reset communication
  Communication parameters are set to start values.
- Initializing
  CAN bus activation is initialized.
- Pre-operational
  The drive controller is ready for parameterization for preparing the actual operation.
- Operational
  The CAN bus is enabled, all associated services are in operation.
- Stopped
  Nearly all communication activities have stopped.

State transitions

No.		CAN message
(1)	Switch on the supply voltage.	–
(2a), (2b), (2c)	After ending the internal action, keep switching independently.	–
(3a), (3b)	NMT_Start_Remote_Node command received.	ID = 0, byte 1 = 1, byte 2 = 0
(4a), (4b)	NMT_Enter_Pre_Operational command received.	ID = 0, byte 1 = 128, byte 2 = 0
(5a), (5b)	NMT_Stop_Remote_Node command received.	ID = 0, byte 1 = 2, byte 2 = 0
(6)	NMT_Reset_Node command received.	ID = 0, byte 1 = 129, byte 2 = 0
(7)	NMT_Reset_Communication command received.	ID = 0, byte 1 = 130, byte 2 = 0

The following example uses the "NMT_Start_Remote_Node" command to show how an NMT command is composed of the bytes specified in the table:

COB-ID	Command specifier	Node
0	1	0

The COB-ID indicates the NMT command. The respective command is entered in the "Command specifier" byte and which bus node will be addressed is defined in the "Node" byte. If this value is 0, all nodes are addressed. For an individual node to be addressed, its node ID must be entered.

Activity of the communication objects

The following table shows which communication objects are enabled in which NMT state.

Object	Reset App., Reset Com., Initializing	Pre-operational	Operational	Stopped
SDO	–	Active	Active	–
PDO	–	–	Active	–
NMT	–	Active	Active	Active
SYNC	–	Active	Active	–
EMERGENCY	–	Active	Active	Active

9.3.4 Error Control Objects – ERROR

To enable monitoring of the NMT states of the individual bus nodes, CANopen offers 2 ERROR CONTROL services, which are based on periodic message transmission:
node/life guarding and heartbeat. Node/life guarding and heartbeat have the same COB-ID, which means the two control mechanisms cannot be used simultaneously; that is, you can either enable one of the two services or disable both.
ERROR CONTROL messages have the "NMT_Error_Control" identifier together with the associated node ID.

9.3.4.1 Node/life guarding

Node/life guarding provides mutual communication control of the controller and drive controller.
A controller uses remote requests to query the internal operating state (Operational, Pre-operational, etc.) of all drive controller connected by the CAN bus. When a remote request is made, a remote frame is sent that contains no data, only the ID of the desired message. These remote frames are answered from the respective drive controllers by corresponding data frames. This allows the controller to check whether its own state matches that of the drive controllers. In return, the drive controllers detect whether the controller is enabled: If the controller does not send any remote frames within a certain time, the drive controllers assume there is a communication error and trigger corresponding faults.
If the node/life guard mechanism is enabled, the heartbeat service must be disabled.

Guard time

The guard time defines the time interval within which the controller sends remote frames to the drive controllers.

Life time factor

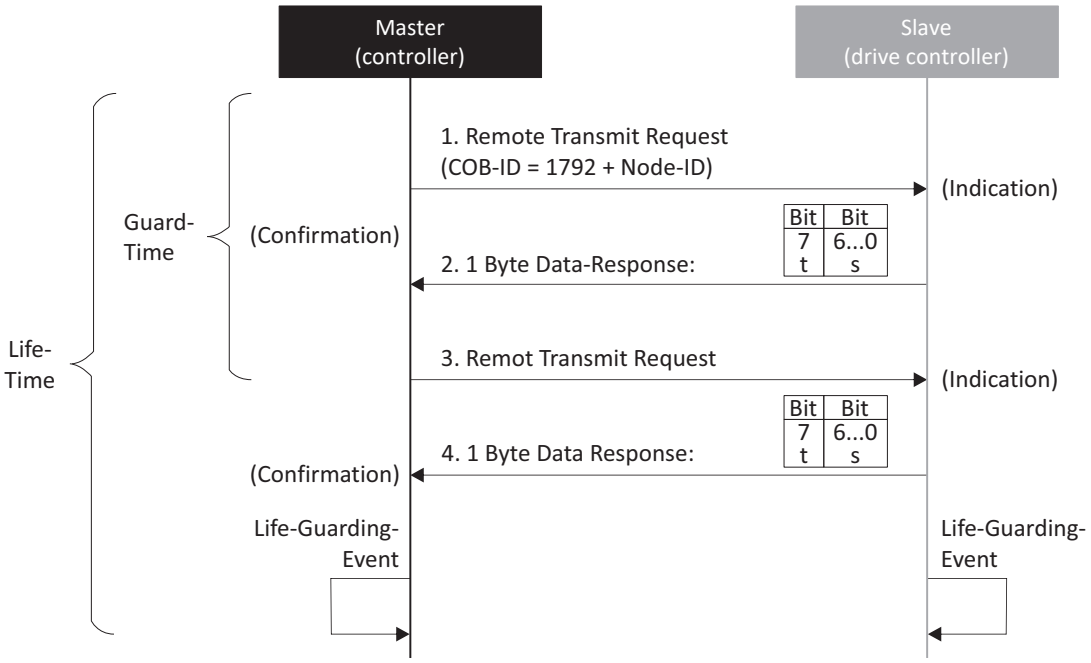
The life time factor indicates how high the number of missing remote frames of the controller has to be before the drive controllers switch to the Fault state.

Life time

The life time is the product of the life time factor and guard time, i.e. the monitoring time within which the controller sends remote frames and the drive controllers answers each of these with a data frame. If either the life time factor or the guard time is set to zero, the life time is automatically disabled.

09/2019 | ID 442637.01

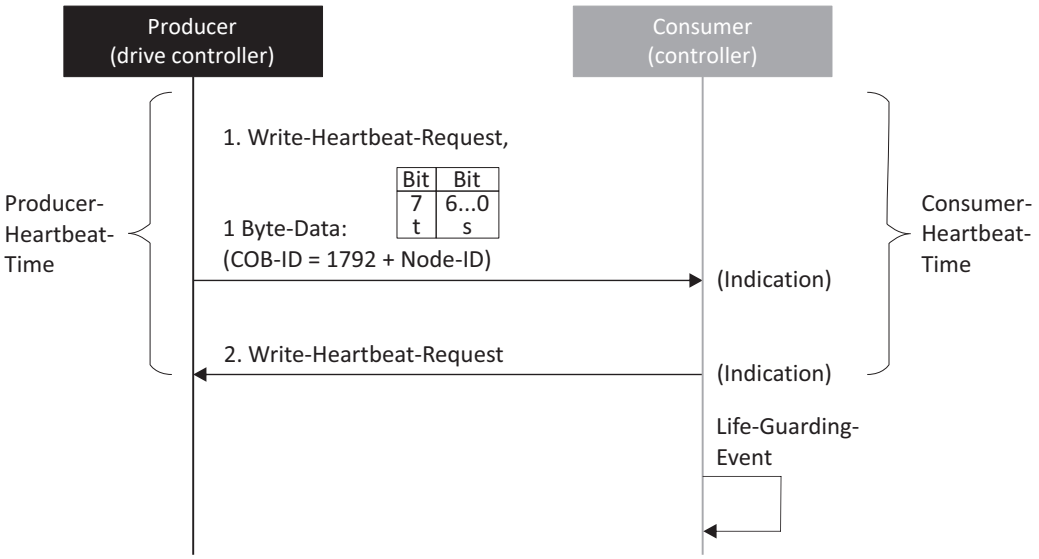
Process – Node/life guarding



As NMT master, the controller uses RTR to send a request to each drive controllers to be monitored (NMT slave) at regular intervals (guard time). Each drive controller detects the request (indication) and sends 1 byte of data as the answer (response). Bit 7 of this byte changes the state with each segment (toggle bit). Bits 0 to 6 contain the state of the NMT state machine. The controller receives the message (confirmation) and uses it to check the drive controller. If the life time fails to appear or is faulty, this condition triggers the node guarding event for the controller.
Conversely, the drive controller monitors the regular requests of the master. If these are absent for this period, the drive controller initiates the life guarding event.

9.3.4.2 Heartbeat

The alternative to the life/node guarding mechanism is the heartbeat service. The advantage of this service over the life/node guarding is that no remote frames are exchanged between the controller and drive controller—which considerably reduces the utilization of the CAN bus.



To enable detection of communication failures, a CAN bus node (usually a drive controller) independently transmits cyclical messages (heartbeat protocol), which display its own operating state. Thus the drive controllers act as producer, while the controller acts as consumer. During the transmission, it is necessary to maintain a defined time interval (producer heartbeat time), within which a message has to be received. The consumer expects regular messages from the producer. If these fail to appear, a heartbeat event is triggered.

9.3.5 Synchronization object – SYNC

Information

Note that if the CAN bus and IGB motion bus are used simultaneously, synchronization of the CANopen communication with the controller is not possible.

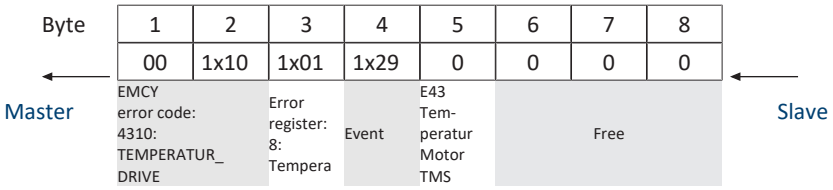
A SYNC object transmits the high-priority SYNC messages, which are received by all nodes simultaneously. SYNC messages are for the synchronization of all bus nodes and cause multiple inputs to be read or axes to be synchronized in parallel, for example. SYNC messages generally do not contain any data.

9.3.6 Emergency objects – EMCY

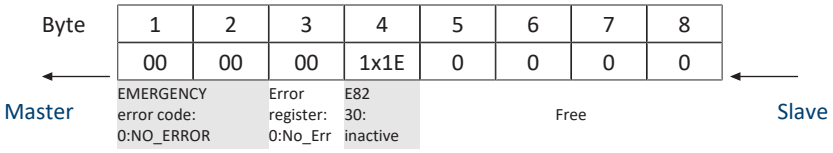
Emergency objects signal device errors, i.e. an EMCY message contains a code that uniquely identifies an error, in accordance with communication profile CiA 301.
If the emergency service is enabled, the state of the bus nodes is permanently monitored. If a node changes to the "Fault" or "Fault reaction active" state, the EMCY message is sent once with one of the error codes described below.
If a node leaves this state, another EMCY message with the content "No error" is transmitted. Thanks to the process described, the controller no longer cyclically queries the device state for a fault, since the controller is automatically informed about each fault case along with its cause and rectification.

EMCY objects – Structure

The following example of a "TMS motor temperature (E43)". fault message shows the structure of an EMCY error message.



If the fault has been rectified, this state is acknowledged with a corresponding message.



The coding of the "Error code" message in the first and second bytes as well as the "Error register" message in the third byte correspond to the information of the CiA/DS-301 and CiA DSP402 profiles. The fourth byte contains the value of the STOBER parameter E82 Event type, the fifth byte contains the value of the parameter E43 Event cause.

Disabling the EMCY service

If the parameter A83 Bus address contains the value 0, then sending EMCY objects is disabled because identifier 128 could interfere with the process data synchronization (SYNC object, also identifier 128).

10 Appendix

10.1 Supported communication objects

10.1.1 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 STOBER-specific 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
1007 hex	0 hex	—	✓	Synchronous window length	A202
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; guard time in ms = 100C hex * 100D hex
100D hex	0 hex	—	✓	Life time factor	A204; guard time in ms = 100C hex * 100D hex
1010 hex				Store parameters supported	Structure
1010 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 1
1010 hex	1 hex	—	✓	Save all parameters	A00[0]; ASCII `save` 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	Structure
1018 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 4
1018 hex	1 hex	—	—	Vendor ID	STOBER 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

Index	Subindex	TxPDO	RxPDO	Name	Comment
1018 hex	4 hex	—	—	Serial number	E52[2]
1200 hex				1st SDO server parameter	Structure
1200 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 2
1200 hex	1 hex	—	✓	COB-ID Client -> Server (rx)	
1200 hex	2 hex	—	✓	COB-ID Server -> Client (tx)	
1201 hex				2nd SDO server parameter	Structure
1201 hex	0 hex	—	—	Highest subindex supported	Number of structure 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				3rd SDO server parameter	Structure
1202 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 3
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				4th SDO server parameter	Structure
1203 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 3
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				1st receive PDO communication parameter	Structure
1400 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 2
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				2nd RxPDO communication parameter	Structure
1401 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 2
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				3rd RxPDO communication parameter	Structure
1402 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 2
1402 hex	1 hex	—	✓	COB-ID used by RxPDO	A223[0]
1402 hex	2 hex	—	✓	Transmission type	A223[1]; value range 1 – 240, 254
1403 hex				4th RxPDO communication parameter	Structure
1403 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 2
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	

Index	Subindex	TxPDO	RxPDO	Name	Comment
1600 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6
1600 hex	1 hex – 6 hex	—	✓	Application objects	A225[0] - A225[5]
1601 hex				2nd RxPDO mapping parameter	
1601 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6
1601 hex	1 hex – 6 hex	—	✓	Application objects	A226[0] - A225[5]
1602 hex				3rd RxPDO mapping parameter	
1602 hex	0 hex	—	✓	Number of mapped application objects in RxPDO	Constant value of 6
1602 hex	1 hex – 6 hex	—	✓	Application objects	A227[0] - A227[5]
1603 hex				4th RxPDO mapping parameter	
1603 hex	0 hex	—	—	Number of mapped application objects in RxPDO	Constant value of 6
1603 hex	1 hex – 6 hex	—	—	Application objects	A228[0] - A228[5]
1800 hex				1st TxPDO communication parameter	Structure
1800 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 5
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	4 hex			Reserved	
1800 hex	5 hex	—	✓	Event timer	A229[3]
1801 hex				2nd TxPDO communication parameter	Structure
1801 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 5
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	4 hex			Reserved	
1801 hex	5 hex	—	✓	Event timer	A230[3]
1802 hex				3rd TxPDO communication parameter	Structure
1802 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 5
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]
1802 hex	4 hex			Reserved	
1802 hex	5 hex	—	✓	Event timer	A231[3]
1803 hex				4th TxPDO communication parameter	Structure

Index	Subindex	TxPDO	RxPDO	Name	Comment
1803 hex	0 hex	—	—	Highest subindex supported	Number of structure elements: 5
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	4 hex			Reserved	
1803 hex	5 hex	—	✓	Event timer	A232[3]
1A00 hex				1st TxPDO mapping parameter	
1A00 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	Constant value of 6
1A00 hex	1 hex – 6 hex	—	✓	Application objects	A233[0] - A233[5]
1A01 hex				2nd TxPDO mapping parameter	
1A01 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	Constant value of 6
1A01 hex	1 hex – 6 hex	—	✓	Application objects	A234[0] - A234[5]
1A02 hex				3rd TxPDO mapping parameter	
1A02 hex	0 hex	—	✓	Number of mapped application objects in TxPDO	
1A02 hex	1 hex – 6 hex	—	✓	Application objects	A235[0] - A235[5]
1A03 hex				4th TxPDO mapping parameter	
1A03 hex	0 hex	—	—	Number of mapped application objects in TxPDO	Constant value of 6
1A03 hex	1 hex – 6 hex	—	—	Application objects	A236[0] - A236[5]

Tab. 8: CiA 301 communication objects: 1000 hex – 1FFFF hex

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

Information

Index, Subindex
Note that the index and subindex must be specified in hexadecimal form in the controller.

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 corresponds to the element number of the parameter, which is always 0 for standard parameters (only significant in case of array and structure parameters).

Calculation example for parameter E200 (number of the group = 4 ,
number of the line = 200):
Index E200 = 8192 + (4 × 512) + 200 = 10440 = 28C8 hex
Subindex E200 = 0 = 0 hex

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

Index area	Group	Number	Addressable 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. 9: Manufacturer-specific communication objects: 2000 hex – 53FF hex

10.1.3 CiA 402 Drives and motion control: 6000 hex – 65FF hex

The following table includes the supported communication objects of the standardized profile CiA 402 Drives and motion control device profile – Part 2: Operation modes and application data for motion control, as well as how they are mapped to the corresponding parameters of STOBER.
The communication objects are used in the applications CiA 402 and CiA 402 HiRes Motion.

Index	Subindex	TxPDO	RxPDO	Name	Comment
603F hex	0 hex	✓	—	Error code	A514
6040 hex	0 hex	✓	✓	Control word	A515
6041 hex	0 hex	✓	—	Statusword	A516
605A hex	0 hex	—	✓	Quick stop option code	A536
605E hex	0 hex	—	✓	Fault reaction option code	A540
6060 hex	0 hex	✓	✓	Modes of operation	A541
6061 hex	0 hex	✓	—	Modes of operation display	A542
6064 hex	0 hex	✓	—	Position actual value	A545
6065 hex	0 hex	✓	✓	Following error window	A546
6066 hex	0 hex	✓	✓	Following error time out	A547
606C hex	0 hex	✓	—	Velocity actual value	A553
6071 hex	0 hex	✓	✓	Target torque	A558
6072 hex	0 hex	✓	✓	Max torque	A559
6076 hex	0 hex	✓	—	Rated torque	A563
6077 hex	0 hex	✓	—	Torque actual value	A564
6078 hex	0 hex	✓	—	Current actual value	A565
6079 hex	0 hex	✓	—	DC link circuit voltage	A566
607A hex	0 hex	✓	✓	Target position	A567
607B hex				Position range limit	
607B hex	0 hex	—	—	Highest subindex supported	Constant value of 2
607B hex	1 hex	✓	✓	Min. position range limit	A568[0]; no function
607B hex	2 hex	✓	✓	Max. position range limit	A568[1]; used as revolution length
607C hex	0 hex	✓	✓	Home offset	A569
607D hex				Software position limit	
607D hex	0 hex	—	—	Highest subindex supported	Constant value of 2
607D hex	1 hex	✓	✓	Min. position range limit	A570[0]
607D hex	2 hex	✓	✓	Max. position range limit	A570[1]
607E hex	0 hex	—	✓	Polarity	A571; only bit 7 has function
607F hex	0 hex	✓	✓	Max profile velocity	A572
6081 hex	0 hex	✓	✓	Profile velocity	A574
6083 hex	0 hex	✓	✓	Profile acceleration	A576
6084 hex	0 hex	✓	✓	Profile deceleration	A577
6085 hex	0 hex	✓	✓	Quick stop deceleration	A578
6091 hex				Gear ratio	

Index	Subindex	TxPDO	RxPDO	Name	Comment
6091 hex	0 hex	—	—	Highest subindex supported	Constant value of 2
6091 hex	1 hex	✓	✓	Motor revolutions	A584[0]
6091 hex	2 hex	✓	✓	Shaft revolutions	A584[1]
6092 hex				Feed constant	
6092 hex	0 hex	—	—	Highest subindex supported	Constant value of 2
6092 hex	1 hex	✓	✓	Feed	A585[0]
6092 hex	2 hex	✓	✓	Shaft revolutions	A585[1]
6098 hex	0 hex	✓	✓	Homing method	A586
6099 hex				Homing speeds	
6099 hex	0 hex	—	—	Highest subindex supported	Constant value of 2
6099 hex	1 hex	✓	✓	Speed during search for switch	A587[0]
6099 hex	2 hex	✓	✓	Speed during search for zero	A587[1]
609A hex	0 hex	✓	✓	Homing acceleration	A588
60A3 hex	0 hex	—	—	Profile jerk use	A589
60A4 hex				Profile jerk	
60A4 hex	0 hex	—	—	Highest subindex supported	Constant value of 1
60A4 hex	1 hex	—	✓	Profile jerk, Profile jerk 1	A590
60B1 hex	0 hex	✓	✓	Velocity offset	A592
60B2 hex	0 hex	✓	✓	Torque offset	A593
60B8 hex	0 hex	✓	✓	Touch probe function	A594
60B9 hex	0 hex	✓	—	Touch probe status	A595
60BA hex	0 hex	✓	—	Touch probe position 1 positive value	A596
60BB hex	0 hex	✓	—	Touch probe position 1 negative value	A597
60BC hex	0 hex	✓	—	Touch probe position 2 positive value	A598
60BD hex	0 hex	✓	—	Touch probe position 2 negative value	A599
60C0 hex	0 hex	—	✓	Interpolation sub mode select	A600
60C1 hex				Interpolation data record	
60C1 hex	0 hex	—	—	Highest subindex supported	Constant value of 1
60C1 hex	1 hex	✓	✓	1st set-point	A601
60C2 hex				Interpolation time period	
60C2 hex	0 hex	—	—	Highest subindex supported	Constant value of 2

09/2019 | ID 442637.01

Index	Subindex	TxPDO	RxPDO	Name	Comment
60C2 hex	1 hex	—	✓	Interpolation time period value	A602[0]
60C2 hex	2 hex	—	✓	Interpolation time index	A602[1]
60C4 hex				Interpolation data configuration	
60C4 hex	0 hex	—	—	Highest subindex supported	Constant value of 5
60C4 hex	1 hex	—	—	Maximum buffer size	A603[0]; no function
60C4 hex	2 hex	—	—	Actual buffer size	A603[1]; no function
60C4 hex	3 hex	—	—	Buffer organisation	A603[2]; no function
60C4 hex	4 hex	—	—	Buffer position	A603[3]; no function
60C4 hex	5 hex	—	—	Size of data record	A603[4]; no function
60C4 hex	6 hex	—	—	Buffer clear	A603[5]; no function
60C5 hex	0 hex	✓	✓	Max acceleration	A604
60C6 hex	0 hex	✓	✓	Max deceleration	A605
60E3 hex				Supported homing methods	
60E3 hex	0 hex	—	—	Highest subindex supported	Constant value of 19
60E3 hex	1 hex – 14 hex	—	—	1st - 19th supported homing method	A619[0] – A619[19]
60E4 hex				Additional position actual value / 1st value	
60E4 hex	0 hex	—	—	Highest subindex supported	Constant value of 1
60E4 hex	1 hex	✓	—	1st additional position actual value	A620
60F2 hex	0 hex	—	✓	Positioning option code	A621
60F4 hex	0 hex	✓	—	Following error actual value	A632
60FD hex	0 hex	✓	—	Digital inputs	A636
60FE hex				Digital outputs	
60FE hex	0 hex			Highest subindex supported	Constant value of 1
60FE hex	1 hex			Physical outputs	A637
60FF hex	0 hex	✓	✓	Target velocity	A638
6502 hex	0 hex	—	—	Supported drive modes	

Tab. 10: CiA 402-2 communication objects: 6000 hex – 65FF hex

10.2 Detailed information

The documentation listed in the following table offers additional relevant information about the associated drive controllers.

Current document versions can be found at http://www.stoeber.de/en/downloads/.

Device/Software	Documentation	Contents	ID
SD6 drive controller	Manual	System setup, technical data, project configuration, storage, installation, connection, commissioning, operation, service, diagnostics	442426
SD6 drive controller	Commissioning instructions	System setup, technical data, storage, installation, connection, commissioning	442537
CiA 402 application – SD6	Manual	Project planning, configuration, parameterization, function test, detailed information	443077

The following CiA specifications are valid within the scope of this 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 Part 4: CANopen
CANopen – standardization as European standard.

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

Information concerning CANopen

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

10.3 Abbreviations

Abbreviation	Meaning
CAN	Controller Area Network
CiA	CAN in Automation
COB-ID	Communication Object Identifier
DBT	Distributor
DC	Direct Current
DIN	Deutsches Institut für Normung e. V. (German Institute for Standardization)
DIN EN	German acceptance of a European standard
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	Received PDO
SDO	Service Data Object
SYNC	Synchronization
TxPDO	Transmitted PDO

11 Contact

11.1 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

Do you need our first-level support?:
Phone +49 7231 582-3060
applications@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

11.2 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

Thank you for your interest.
Your STOBER editorial team

09/2019 | ID 442637.01

11.3 Close to customers around the world

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

STOBER AUSTRIA www.stoeber.at Phone +43 7613 7600-0 sales@stoeber.at	**STOBER SOUTH EAST ASIA** www.stober.sg sales@stober.sg
STOBER CHINA www.stoeber.cn Phone +86 10 6590 7391 sales@stoeber.cn	**STOBER SWITZERLAND** www.stoeber.ch Phone +41 56 496 96 50 sales@stoeber.ch
STOBER FRANCE www.stober.fr Phone +33 4 78.98.91.80 sales@stober.fr	**STOBER TAIWAN** www.stober.tw Phone +886 4 2358 6089 sales@stober.tw
STOBER ITALY www.stober.it Phone +39 02 93909570 sales@stober.it	**STOBER TURKEY** www.stober.com Phone +90 212 338 8014 sales-turkey@stober.com
STOBER JAPAN www.stober.co.jp Phone +81 3 5395 678 8 sales@stober.co.jp	**STOBER UNITED KINGDOM** www.stober.co.uk Phone +44 1543 458 858 sales@stober.co.uk
STOBER USA www.stober.com Phone +1 606 759 5090 sales@stober.com	

09/2019 | ID 442637.01

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 (COB-ID = FC + 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 redirected by a router, which limits it to the local network.

Node ID

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

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.

09/2019 | ID 442637.01

List of figures

Fig. 1 CANopen – network structure, using the SD6 series as an example 11

Fig. 2 Connecting the terminal resistances in a CANopen network 14

Fig. 3 LEDs for the CAN status 16

Fig. 4 DS6 – Program interface 17

Fig. 5 DriveControlSuite: Navigation using text links and symbols 18

Fig. 6 Network Management Object – NMT – Potential operating states and state changes 44

09/2019 | ID 442637.01

List of tables

Tab. 1	Transmission rate and max. CAN bus length	14
Tab. 2	X200 connection description	15
Tab. 3	Meaning of the green LED (run)	16
Tab. 4	Meaning of the red LED (error)	16
Tab. 5	Parameter groups	19
Tab. 6	Parameters – Data types, styles, possible values	20
Tab. 7	Object directory – Structure	30
Tab. 8	CiA 301 communication objects: 1000 hex – 1FFFF hex	50
Tab. 9	Manufacturer-specific communication objects: 2000 hex – 53FF hex	54
Tab. 10	CiA 402-2 communication objects: 6000 hex – 65FF hex	55

09/2019 | ID 442637.01

4 4 2 6 3 7 . 0 1

09/2019

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

STÖBER

www.stober.com